

PART I

Chapter 1 Introduction

1.1	Motivation for this study.....	2
1.2	Problem statement.....	3
1.3	Terminology used.....	5
1.3.1	Web services.....	5
1.3.2	Information security.....	6
1.3.3	Access control service.....	6
1.3.4	Message security.....	6
1.3.5	Trust.....	7
1.3.6	Loosely coupled.....	7
1.3.7	Machine.....	7
1.4	Layout of thesis.....	8

Chapter 2 Access control for web services

2.1	Web services architecture.....	11
2.2	Functional components of web services.....	13
2.2.1	Web services.....	14
2.2.2	Web services operation.....	15
2.2.3	Web services class.....	15
2.2.4	Web services composition.....	15
2.3	Web services technology.....	16
2.3.1	XML (Extensible Markup Language).....	16
2.3.2	WSDL (Web Service Description Language).....	17
2.3.3	SOAP (Simple Object Access Protocol).....	18
2.3.4	UDDI (Universal Discovery Description and Integration).....	20
2.4	Web service security and related standards.....	21
2.4.1	Specifications for XML-based security mechanisms.....	22
2.4.2	Specifications for XML-based security interoperability.....	23
2.5	Environmental access control requirements of web services.....	25
2.5.1	Autonomy.....	25
2.5.2	Loosely coupled.....	26
2.5.3	Policy-based compatibility.....	27
2.5.4	Policy negotiation.....	27
2.5.5	Quality of service.....	28

2.5.6	Standards-based interaction.....	28
2.6	Conclusion.....	28

Chapter 3 Case study

3.1	The virtual application.....	30
3.1.1	eBooks.....	30
3.1.1.1	Minimal inter-dependency.....	31
3.1.1.2	Increased inter-dependency.....	32
3.1.2	eLoans.....	32
3.1.3	The virtual application.....	33
3.2	Access control policy for eBooks.....	35

Chapter 4 Web services access control service

4.1	The scope of the web service access control service.....	37
4.2	Internal access requirements of web services.....	39
4.2.1	Flexibility.....	40
4.2.2	Efficient administration.....	40
4.2.3	Attribute-based access control.....	41
4.2.4	Trust levels.....	41
4.2.5	Exceptions.....	42
4.2.6	Conflict resolution.....	42
4.3	Access control.....	42
4.3.1	Access control models.....	43
4.3.2	Access control mechanisms.....	43
4.3.3	Access control information.....	44
4.4	Access control models.....	44
4.4.1	Discretionary access control (DAC).....	44
4.4.2	Mandatory access control (MAC).....	45
4.4.3	Role-based access control (RBAC).....	47
4.4.4	Chinese Wall access control model.....	48
4.4.5	Credential-based access control.....	49
4.5	Web services access control service.....	50
4.5.1	Dimensions of the web services access control service.....	50
4.5.1.1	Web services access control requirements.....	50
4.5.1.2	Access control mechanisms.....	51
4.5.1.3	Access control information.....	51

4.5.2	Web service access control requirements analysis.....	53
4.5.2.1	Environmental access control requirements.....	53
4.5.2.2	Internal access control requirements.....	53
4.6	Conclusion.....	55

Chapter 5 Web services trust

5.1	The management of trust.....	57
5.1.1	Trust management systems.....	57
5.1.2	The Liberty Alliance trust model.....	58
5.1.3	WS-Trust.....	58
5.1.4	Trust negotiation.....	59
5.1.5	Computational trust.....	59
5.1.6	Summary of approaches to trust management.....	60
5.1.7	Trust management for web services.....	62
5.2	Trust.....	63
5.2.1	Trust for humans.....	63
5.2.1.1	Trust.....	63
5.2.1.2	Properties of trust.....	64
5.2.1.3	Dimensions of trust.....	64
5.2.1.4	Basis of trust.....	64
5.2.2	Trust for organisations.....	66
5.2.3	Trust perspective taken by this research.....	66
5.2.4	Definitions for belief and trust relationships.....	67
5.2.5	Trust for web services.....	68
5.2.5.1	Properties of trust.....	68
5.2.5.2	Dimensions of trust.....	69
5.2.5.3	Basis of trust.....	69
5.3	Conclusion.....	70

Chapter 6 Web services trust formation framework

6.1	Trust formation phases.....	71
6.1.1	Publish trust information.....	72
6.1.2	Discover trust information.....	72
6.1.3	Trust formation.....	72
6.1.4	Trust evolution.....	73
6.2	Trust context.....	73

6.3	Trust level.....	74
6.4	Trust computation.....	75
6.5	Trust assessment for trust concept formation.....	77
	6.5.1 Environmental information.....	78
	6.5.2 References.....	79
	6.5.3 Recommendations.....	79
	6.5.4 Experience.....	79
6.6	Taxonomy of trust concepts.....	80
	6.6.1 Trust in the internal environment.....	81
	6.6.2 Trust in the external environment.....	82
	6.6.3 Trust in the other party.....	82
6.7	Definitions: trust management, trust assessment, trust relationships, trust types and trust concepts.....	84
6.8	Conclusion.....	85

Chapter 7 Access control policy specification

7.1	Purpose of web services access control service policies.....	87
7.2	Policy specification language for access control publication.....	87
	7.2.1 XML.....	88
	7.2.1.1 WS-Policy.....	90
7.3	Policy specification language for access control reasoning.....	92
	7.3.1 First-order logic.....	93
	7.3.2 Examples of policy specification languages for access control reasoning.....	95
	7.3.2.1 Logic-based languages.....	96
	a) ASL.....	96
	b) Ponder.....	96
	c) SPKI/SDSI.....	97
	d) Keynote.....	97
	e) SD3.....	98
	f) Service Access and Information Release.....	98
	g) SECURE.....	99
	7.3.2.2 XML-based languages.....	100
	a) XACML.....	101
	b) X-TNL.....	102
	7.3.2.3 High-level comparison of languages.....	102
	7.3.3 ASL.....	103
7.4	Conclusion.....	106

Chapter 8 Web services access control service architecture

8.1	Architectures for access control and trust.....	107
8.1.1	Architectures for access control.....	107
8.1.1.1	ISO 10181-3 access control framework.....	108
8.1.1.2	IETF policy management architecture.....	109
8.1.1.3	OASIS.....	111
8.1.1.4	Features of access control architectures.....	112
8.1.2	Architectures for trust.....	112
8.1.2.1	Fidelis architecture.....	113
8.1.2.2	Automated trust negotiation architecture.....	114
8.1.2.3	Computational trust architecture.....	115
8.1.2.4	Features of trust architectures.....	115
8.2	Web services access control service architecture.....	116
8.3	Conclusion.....	118

PART II**Chapter 9 The WSACT model – an overview**

9.1	WSACT – Design motivation.....	120
9.1.1	Authorisation interface.....	122
9.1.2	Authorisation manager.....	122
9.1.3	Trust manager.....	123
9.2	Conclusion.....	123

Chapter 10 WSACT - The authorisation interface

10.1	The authorisation interface.....	124
10.1.1	WSACT model specification in Z.....	126
10.1.2	Basic types.....	127
10.1.2.1	Types for initial trust formation.....	127
10.1.2.2	Types for trust evolution.....	128
10.1.2.3	Types for trust in transaction security.....	131
10.1.2.4	Types for access control.....	131
10.1.3	The abstract state space of the authorisation interface.....	132
10.1.4	Authorisation interface operations.....	133
10.1.4.1	Evaluate message.....	134

10.1.4.2	Process policy.....	135
10.1.4.3	Process trust info.....	135
10.1.4.4	AccessAM.....	136
10.2	Conclusion.....	137

Chapter 11 WSACT - The authorisation manager

11.1	The authorisation manager.....	140
11.2	Access control addressing attributes and levels of trust.....	141
11.2.1	Web services operations, attributes and trust levels.....	142
11.2.2	Web services requestors and trust levels.....	142
11.3	WSACT authorisation manager concepts.....	144
11.3.1	Subject attributes.....	144
11.3.2	Roles.....	145
11.3.3	Trust levels.....	146
11.4	WSACT rules and policies.....	147
11.4.1	Sets and relations.....	147
11.4.1.1	Subject attributes.....	147
11.4.1.2	Web services requestors.....	147
11.4.1.3	Web services objects.....	148
11.4.1.4	Actions.....	148
11.4.1.5	Roles.....	148
11.4.1.6	Trust levels associated with a role.....	148
11.4.1.7	Trust levels associated with a web services requestor.....	148
11.4.2	Predicates.....	149
11.4.2.1	Access control predicates.....	149
11.4.2.2	Attribute satisfaction.....	149
11.4.2.3	Role activation predicates.....	149
11.4.2.4	Access derivation.....	150
11.4.2.5	Access request.....	150
11.5	Authorisation manager in Z.....	151
11.5.1	Facts.....	151
11.5.2	The abstract state space of the authorisation manager.....	152
11.6	Conclusion.....	155

Chapter 12 WSACT - The trust manager

12.1	The trust manager.....	158
12.1.1	Trust manager components.....	159

12.2	Fuzzy cognitive map concepts for trust inference.....	160
12.3	Fuzzy cognitive map for web services trust.....	161
12.4	Trust.....	162
12.5	Trust types.....	163
12.5.1	Trust in the internal environment of eBooks.....	163
	C ₅ – Vulnerabilities.....	164
	C ₆ – Successes in dealing with security risks and compromises.....	165
	C ₇ – Complexity.....	166
12.5.2	Trust in the external environment of eBooks.....	167
	C ₈ – Rule of law.....	168
	C ₉ – Assurances.....	169
	C ₁₀ – Compliance.....	170
	C ₁₁ – Implemented security mechanisms.....	171
	C ₁₂ – Identity mechanisms.....	171
	C ₁₃ – Integrity mechanisms.....	172
	C ₁₄ – Confidentiality mechanisms.....	173
	C ₁₅ – Privacy.....	173
12.5.3	Trust in eLoans.....	174
	C ₁₆ – Compliance with agreements.....	175
	C ₁₇ – Competence.....	176
	C ₁₈ – Predictability.....	176
	C ₁₉ – Goodwill.....	177
12.6	Trust inference in WSACT.....	178
12.6.1	Computation.....	178
12.6.2	Threshold.....	179
	12.6.2.1 Binary function.....	179
	12.6.2.2 Logistic function.....	180
12.7	The abstract state space of the trust manager.....	180
12.7.1	Get_Requestor_Trust_Level.....	181
12.7.2	Compute_Trust.....	182
12.7.3	Fuzzify_Trust_Concept.....	183
12.8	Trust level computation.....	184
12.8.1	Ad hoc population of trust concepts.....	185
12.9	Conclusion.....	186

Chapter 13 WSACT – Prototype implementation

13.1	The aim of the prototype.....	187
13.2	Implementation overview.....	187
	13.2.1 Authorisation interface.....	188
	13.2.2 Authorisation manager.....	189
	13.2.3 Trust manager.....	190
13.3	Prototype operation.....	190
	13.3.1 The administrative trust interface of eBooks.....	191
	13.3.2 Trust relationships of ebooks with eLoans and eCompany.....	192
	13.3.3 Trust in the external environment between eLoans and eCompany.....	194
	13.3.4 Trust in the other party – eLoans.....	201
	13.3.4.1 Access based on a Sue’s abilities – search academic operation.	201
	13.3.4.2 Access based on eLoans’ trust level – specials search operation	202
13.4	Conclusion.....	211

Chapter 14 Conclusion

14.1	Revisiting the problem statement.....	212
14.2	Does the model meet desired access control features?.....	215
	14.2.1 Environmental access control requirements.....	215
	14.2.2 Internal access control policy requirements.....	216
14.3	Main contribution.....	217
14.4	Future research.....	217

Papers published in journals.....	219
--	------------

Papers presented at conferences.....	219
---	------------

Bibliography.....	220
--------------------------	------------

Content of CD:

- Papers published and presented
- Source code of prototype

List of figures

Figure 1.1:	Thesis layout.....	10
Figure 2.1:	Web services architecture.....	12
Figure 2.2:	Web services functional components.....	13
Figure 2.3:	WSDL service structure.....	17
Figure 2.4:	SOAP message.....	19
Figure 2.5:	Security standards over SOAP.....	22
Figure 2.6:	Web services security framework.....	24
Figure 3.1:	eBooks web services architecture.....	31
Figure 3.2:	eLoans architecture.....	33
Figure 3.3:	Virtual application architecture.....	33
Figure 3.4:	SOAP request for the search operation.....	34
Figure 3.5:	High-level access control rules for eBooks.....	35
Figure 4.1	eBooks access control service.....	38
Figure 4.2:	Access control entities for web services.....	39
Figure 4.3:	Web services access control service requirements.....	40
Figure 4.4:	Relationships between access control models, mechanisms and information...42	
Figure 4.6:	Web services access control service dimensions.....	52
Figure 4.7:	Web services access control service requirements not to be addressed.....	54
Figure 4.8	Access control service policies.....	54
Figure 5.1:	The trust relationship.....	63
Figure 5.2:	Trust formation.....	65
Figure 6.1	Phases of the formation of trust for web services.....	72
Figure 6.2:	Fuzzy Cognitive Map (FCM).....	77
Figure 6.3:	Taxonomy of trust types and trust concepts.....	81
Figure 7.1:	WS-Policy of eBooks for message security.....	91
Figure 7.2:	Access control policy for the place_order operation of eBooks.....	105
Figure 8.1:	Components of the Access Control Framework.....	108
Figure 8.2:	IETF policy management architecture.....	109
Figure 8.3:	XACML architecture.....	110
Figure 8.4:	OASIS architecture.....	111
Figure 8.5:	TrustBuilder architecture.....	114
Figure 8.6:	SECURE framework architecture.....	115
Figure 9.1:	WSACT components.....	121

Figure 10.1:	WSACT - the authorisation interface and related components.....	125
Figure 10.2:	Authorisation interface operations.....	133
Figure 11.1:	WSACT – the authorisation manager and related components.....	140
Figure 11.2:	Access control entities for web services.....	141
Figure 11.3:	Trust levels of web services requestor.....	143
Figure 12.1:	WSACT - the trust manager and related components.....	158
Figure 12.2:	Trust manager architecture.....	159
Figure 12.3:	Fuzzy cognitive map for eLoans.....	161
Figure 12.4:	Trust in the internal environment of eBooks.....	164
Figure 12.5:	Trust in external environments between eBooks and eLoans.....	168
Figure 12.6:	Trust in eLoans.....	174
Figure 12.7:	Activation by means of the logistic function for $c = 0.8, 0.5, 0.2$ and 0.1	180
Figure 12.8:	Activation of state vector A by C_5, C_7 and C_{17}	185
Figure 12.9:	Activation of state vector A by $C_5, C_7, C_{12}, C_{13}, C_{14}, C_{15}, C_{17}$	185
Figure 13.1:	Web.config file of eBooks.....	189
Figure 13.2:	WSACT prototype.....	191
Figure 13.3:	eBooks Administrative Trust Interface.....	192
Figure 13.4:	Sue Smith as participant in two virtual applications.....	193
Figure 13.5:	Activation of trust concept related to identity (C_{12}).....	194
Figure 13.6:	Inference of trust level based on identity (C_{12}).....	195
Figure 13.7:	Trust concepts set by fuzzifying information from database.....	196
Figure 13.8:	eBooks' trust in external environment with eCompany.....	196
Figure 13.9:	eCompany portal login.....	197
Figure 13.10:	eCompany portal book order functions.....	198
Figure 13.11:	SOAP message for search operation.....	198
Figure 13.12:	Access request for search operation.....	199
Figure 13.13:	Declaration.....	199
Figure 13.14:	A Prolog implementation of the access control policy for search operation.....	199
Figure 13.15:	Log file.....	200
Figure 13.16:	eLoans login.....	201
Figure 13.17:	eLoans student search and order operations.....	202
Figure 13.18:	eLoans search operations.....	202
Figure 13.19:	Error message.....	203
Figure 13. 20:	Trust in the internal environment of eBooks and eLoans.....	203
Figure 13.21:	Tables from the interface database related to trust concepts.....	204
Figure 13.22:	Adding items to the basket.....	206
Figure 13.23:	Finalise an order.....	207
Figure 13.24:	Administrator interface.....	207
Figure 13.25:	Administrator – view order.....	208
Figure 13.26:	Administrator – select order to be paid.....	208

Figure 13.27: Administrator – enter credit card number.....	209
Figure 13.28: Administrator – successful payment.....	209
Figure 13.29: Predictability table.....	209
Figure 13.30: Activation of predictability.....	210
Figure 13.31: Activation of C ₅ and C ₉ by administrator.....	210
Figure 13.32: Order for special books.....	211

List of tables

Table 5.1:	High-level description of approaches to trust and access control.....	60
Table 5.2:	Information layers and sources of trust formation.....	67
Table 6.1:	Trust levels.....	75
Table 7.1:	ASL predicates.....	105
Table 11.1:	Operations, trust levels and subject attributes.....	142
Table 11.2:	Subject attributes per operation.....	144
Table 12.1:	Increase in trust.....	184

PART I

1

Introduction

With the globalisation of the world economy, more and more organisations are becoming aware of the need to move to open standards in order to collaborate with business partners. Web services technology (Gottschalk 2002) represents a response to the need for a flexible and efficient business collaboration environment.

Web services technology extends the Internet from an infrastructure that provides services to humans, to one that provides services to machines that are acting on behalf of humans, organisations or applications. It provides a technical infrastructure to ensure the interoperation of machines that support services and applications from different organisations. Machine interactions automatically perform operations that previously required human interventions, such as searching and buying goods at a pre-determined price, coordinating flight reservations and streamlining invoicing and shipping processes. Applications are created through the use of loosely coupled, reusable software components that are written in diverse programming languages and tools, that reside in different operating systems, and that may be found in different organisational domains. Assembly occurs between applications on a machine-to-machine basis and appears virtual to the human who utilises such functionality. Since the providers of such services concentrate on their particular field of expertise, more sophisticated applications and/or virtual organisations can be created.

The machine-to-machine interactions that support these virtualised environments indirectly constitute a quasi form of social collaboration. For such applications, access to resources under the control of one machine must be granted to large numbers of entities that are known only to another machine in a different domain. Even if the web services access control service knew the identities of requesting entities, this would not provide an adequate solution as they are unknown to the environment of the web service. It is hence not possible to grant them access to specific resources. For this reason, traditional role-based approaches may also prove to be inadequate.

To allow unknown users, machines or organisations to access resources, the trust in the machine that is acting on their behalf can play an important role, as it enables decisions to be made when incomplete information is known. Decisions can be made based on information related to trust, for which an entity can provide evidence such as proof of key attributes, signed statements from a trusted source, and also the recording of experience. In summary – this study is motivated by the need to understand the role of trust in web services access control decisions.

1.1 MOTIVATION FOR THIS STUDY

It is reasonable to suppose that systems created with web services will be of a large scale, highly distributed, pervasive and dynamic, and that they will span organisational boundaries. The rationale behind this research is consequently formulated as follows:

Open nature

Applications that are created with web services are inclined to be of an open nature. For example, a web-based online store is open to anyone with an Internet connection. This implies that web services applications are required to deal with previously unidentified or unfamiliar parties. It would be important for the access control service to be able to grant access to strangers who are authorised, while at the same time preventing access to unauthorised or potentially malicious parties.

Vast numbers of users

Applications created with web services technology are potentially exposed to large numbers of users distributed across the globe. The access control service of a web services provider must be as scalable as the applications themselves.

Virtualised environments

Virtual environments are created as increasing competition motivates organisations to collaborate in new and innovative ways. Virtual environments may span different network, administrative or organisational environments, which leads to the disappearance of set boundaries. The access control service of a web services entity must accommodate cross-domain integration where remote users are authorised not only based on who they are, but also on the environment from where they make a request.

Machines as requestors

When virtual environments are created, there is a growing trend to give machines the responsibility to act on behalf of humans, organisations and other machines in order to simplify collaboration. While this eases the application integration between organisations, the potential for machines to carry out risky transactions or to act maliciously increases. The access control

service needs to be able to determine the trustworthiness of requesting machines before access is granted to sensitive operations or resources.

Autonomous authorities

Because of the above considerations, it is either difficult and costly or impossible to conduct a centralised administration of distributed, cross-domain web services applications that are accessed by large numbers of users. Each administrative domain should rather assume full autonomy of the specification, management and enforcement of access control.

1.2 PROBLEM STATEMENT

This research recognises that reliable access control is a fundamental requirement for applications created with web services technology. A model is proposed for web services access control that uniquely addresses specific requirements of web services environments and consequent cross-domain collaboration. The problem area is addressed by considering the following research questions:

What are the specific access control requirements of web services that need to be addressed?

A unique characteristic of web services is that they allow business partners to communicate through human-legible SOAP messages. A partner's secrets may be exposed in a SOAP response and must be protected from threats such as disclosure to unauthorised parties. Due to the fact that XML is text-based, web services invocations can pass over normal HTTP channels and therefore through firewalls. The applications that are to process the request contained by these XML messages may then be endangered by false claims or malicious information. Access control is thus a critical requirement for web services to be reliable in operation. As there is currently no standard, agreed-upon method for exposing web services operations over the Internet in such a way that only authorised users can call them, access control requirements for web services need to be identified.

How do current access control models meet the requirements of web services?

Various access control models exist, each having its own strengths and weaknesses. Balancing the competing goals of collaboration between web services entities and security is not an easy task. An investigation thus needs to be conducted into the variety of current access models, to determine the extent to which their mechanisms could meet the identified access control requirements of web services.

How can access be granted to a diverse and ever-changing user population, as web services collaborate in virtual environments?

Collaborating web services environments are characterised by the lack of a central control authority. This shifts the responsibility for access control and other decisions to participating machines that support web services. As decisions need to be made based on incomplete information, the management of trust relationships can be used as a possible way to resolve this issue. Even though the concept of trust management has been widely accepted, these systems are inadequate for web services collaboration as they do not allow for any degree of uncertain information, and changing trust relationships are not reflected. It is important to enable a web services entity to collaborate safely, by allowing it to grant others meaningful access. An investigation is thus needed into trust and the role that it can play in access control decisions.

What policies are used by web services entities to communicate access control requirements and enforce access control decisions?

Web services interoperation requires the specification and deployment of various types of policies over and above the interface of a web service. Quality-of-service needs such as security, privacy, performance and availability that are either required or supported by web services entities can be specified. Policies need to be supported by web services access control so that web services entities can be given the ability to interoperate in a platform-independent manner. The access control service thus needs to support different types of policies that can be specified in different languages.

How can trust be incorporated into an access control model for web services?

In traditional access control models, access control decisions are determined according to the authenticated identities of subjects. In open environments such as web services, users belonging to web services requestors are not identified by identities, but rather by signed attributes in order to gain access to resources. Basing access control on attributes of the user or web services requestor provides flexibility and scalability essential to the success of web services environments. To enable autonomous collaboration between web services, the notion of attribute-based access control is not sufficient, as it does not make it possible to discriminate between web services requestors who are trustworthy and those who are not. Subjects may possess valid attributes, but true collaboration ideally requires of a web service provider to grant more and advanced access to subjects of web services requestors who are more trusted. For every decision that is made, the web services access control service thus needs to address the trust in the web services requestor.

How can the model be deployed in the web services architecture?

It is important to consider the architecture of enforcement for web services access control. The access control model to be defined by this research needs to be incorporated into the web services architecture and should support the many different types of policies that are identified, so

that access control is uniformly addressed by multiple enforcement points. It should be able to incorporate current web services security standards and specifications in its architecture. The architecture should address the autonomous way in which access control decisions are made by implementing a decentralised design. The architecture should further address design features such as modularity, extensibility, policy and programming language independency, and should be able to scale with growth. In order to be used in the web services environment, the access control service architecture should not impede on the exchange of SOAP messages.

1.3 TERMINOLOGY USED

In order to avoid any misunderstanding, it is important to correctly interpret the terminology used in this thesis. The researcher now provides a brief definition of what is meant by the terms *web services*, *information security*, *access control service*, *message security*, *loosely coupled*, *trust* and *machine*.

1.3.1 Web services

The definition of web services is still evolving as is manifested by the various definitions found in literature. The W3C (World Wide Web Consortium) (Booth et al. 2004) defines a web service as “a software system designed to support interoperable machine-to-machine interactions over a network. It has an interface described in a machine-processable format. Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards” Another definition states that a web service is a “business function made available via the Internet by a web services provider and accessible by clients that could be human users or software applications” (Casati & Shan 2001). Web services are also defined as “loosely coupled applications using open, cross-platform standards and which interoperate across organisational and trust boundaries” (Abiteboul et al. 2001).

These definitions are complementary where each definition emphasises specific features of web services. The primary goal of web services is to provide a mechanism for application-to-application interaction over the Internet so as to allow software assets to be shared, combined, and reused by machines within or between organisations.

For this research, a web service is defined as a location on a network that exposes an application to a machine. The web service’s behaviour is described by machine-readable descriptions of the messages it receives and optionally returns. A schema for the data contained in the message defines requirements for interactions between a web services requestor and a web services provider. Metadata is used to describe the network address for the web service, the operations it supports, its capabilities, and its requirements for reliability, security and transaction support.

1.3.2 Information security

The focus of this thesis is on access control as security service. It is important to bear in mind that access control does not function alone, but is part of a comprehensive security framework. For this reason, the term *information security* needs to be defined, namely as the measures that are employed to prevent the unauthorised use, misuse, modification or denial of the use of assets (Gollmann 1999). There are five information security services to be considered for this purpose (ISO 7498-2 1989):

- The authentication service deals with processes or methods used to identify and prove the identity of a party who attempts to send a message or access data.
- The access control service deals with limiting users to access only those resources for which they have been granted access rights.
- The confidentiality service deals with the protection of information against disclosure to an unauthorised party.
- The integrity service deals with the protection of information against being changed by an unauthorised party.
- The non-repudiation service deals with proving accountability for actions in the system.

The transport protocol used by web services does not address information security services. In order to create a security context between web services entities, information security services need to be implemented in a platform-independent manner that is understood by both parties.

1.3.3 Access control service

This research deals with the definition of an *access control service* for a web services provider. It is a layer of software that controls every access to the web services provider, by enforcing the access control policy. The access control service is designed to be seamlessly integrated into the web services architecture. When a web services requestor requests access to a web services operation of behalf of an entity from its environment, the access control service intercepts the request in a transparent manner, and the access is either granted or not.

1.3.4 Message security

In the context of this thesis, *message security* refers to the security of SOAP requests and responses that are sent between web services entities. In contrast to transport security, which secures a message between two endpoints, message security applies XML-based security mechanisms to the message itself (Remy & Rosenberg 2004). As a message is passed between different web services entities, it remains secure. Message security describes enhancements to

associate security tokens with message content, and to ensure the integrity and confidentiality of a message.

1.3.5 Trust

Trust is a positive concept that expresses the belief that the other party will behave as expected, where the belief is based on the lack of contrary evidence (Gambetta 1988). Trusting beliefs are based on evidence, recommendations from trusted third parties and simple intuition. For this research, the definition of trust is influenced by the service-oriented view of Dimitrakos (2003). It states that the trust of a web services entity in another party is the measurable belief of the web services entity in the other party for behaving dependably for a specific period within a specified context in relation to the web service that is being used.

1.3.6 Loosely coupled

Coupling is defined as the degree to which components depend on one another (Lyu & Rajaraman 1992). A loosely coupled system is achieved if dependencies among system components have been reduced to a minimum. Techniques promoting loose coupling include the following (Kaye 2003; Orchard 2004):

- *Messages* that are vendor-independent and platform-independent, coarse-grained, self-describing, self-contained and stateless.
- *Interfaces* that are well-defined, extensible, versionable and constrained.
- *Addresses* that are URIs (Uniform Resource Identifiers) as they are well-defined and easily transferred.
- *Exchange patterns* that are asynchronous where possible and appropriate.

Web services are not loosely coupled *per se*. The manner in which the web services community adopts these techniques produces loosely coupled systems. For example, many interfaces are not well defined and rather promote fine-grained messages, and interfaces are often not extensible as they are created by tools to conform to an API (Application Program Interface). By defining an interface that is decoupled from the implementation, and by performing mapping between the abstract interface and the particular implementation, a loosely coupled system can be promoted. For this thesis it is important to consider the design of a loosely coupled access control service.

1.3.7 Machine

The web services community often refers to machine-to-machine interactions. This thesis uses the term machine-based trust. Thus it is important to clarify the term *machine*. A machine is a computer or computational entity consisting of hardware and software. The hardware and

software are themselves not intelligent, but it has been demonstrated that the machine can be programmed to perform tasks that exhibit intelligence (De Silva 2000). A machine is thus a programmed computer or computational entity that may exhibit intelligence.

For web services, a machine is defined as the hardware and software that are programmed to perform the roles of web services requestor and web services provider. For this research, machines exhibit two characteristics to enable machine-to-machine interactions; in other words, web services requestor to web services provider interactions and vice versa:

- Machines interact automatically if they have the ability to process information not only from their own domain, but also from the domains of others. This can occur if information is defined in a standards-based and machine-readable format in XML, with common vocabularies.
- Machines interact autonomously if they are programmed to include incomplete, imprecise and fuzzy information in their decisions.

1.4 LAYOUT OF THESIS

This thesis consists of two parts, with both Part 1 and Part 2 consisting of a number of chapters. Figure 1.1 provides a graphical depiction of the layout of the thesis. **Part I** provides a background and critical overview to the reader, which is essential to the formulation of the model. Important concepts are developed in this section that provide a foundation for the model that is finally presented. The current chapter, **Chapter 1**, provides an introduction to the research.

Chapter 2 provides a background on web services, with the aim of identifying relevant issues to be addressed by a web services access control service. From this discussion, a list of six access control requirements is identified. Current web services security standards are described.

Chapter 3 describes an example of collaboration between web services providers and requestors. The discussion emphasises reasons for, and background to, cross-domain functional integration. A high-level access control policy is presented, highlighting the difficulties faced by administrators of an access control service of a web services provider.

Chapter 4 defines the access control service of a web services provider and defines its scope. Another six access control requirements are identified from the access control considerations highlighted by the case study in Chapter 3. Some background information is given on access control, while access control models are discussed in the light of the contribution that their mechanisms can make towards supporting access control requirements. Two types of policies are identified to address access control requirements.

Chapter 5 discusses trust parallel to access control. It commences with a background to trust management and continues with a discussion on social and organisational forms of trust. Trust between web services is examined in order to determine its alignment with these forms of trust.

Chapter 6 defines a web services trust formation framework. A phased approach to trust formation is presented. Trust context, trust levels, trust computation, trust assessment, a trust taxonomy and formal definitions to be used in the model are described.

Chapter 7 discusses the purpose of the policies identified in Chapter 4. The spotlight subsequently falls on policy specification languages for the purpose of policy publication and access control reasoning.

Chapter 8 identifies components to be included in the web services access control service architecture. Architectures for trust management, automated trust negotiation and computational trust are described. The chapter concludes with those essential components and considerations for the web services access control service architecture that address access control requirements.

Part II of this thesis sets out to develop an access control model to meet access control requirements by incorporating trust with access control – in a single model. To introduce the proposed model, **Chapter 9** gives an overview of the model for **Web Services Access Control** incorporating Trust, or the WSACT model for short.

Chapter 10 considers the functionality of the authorisation interface in collecting information for trust formation and access control. The design of the interface policy is described with components that use and source information and evidence.

Chapter 11 describes the role of trust levels in access control decisions. This is followed by a discussion of important facets of the authorisation manager of the $W_S T_B AC$ model.

Chapter 12 identifies components of the trust manager. Concepts for the fuzzy cognitive map for web services trust are discussed, and the structure of trust, trust types and trust concepts is described. Next follows an illustration of trust inference and the computation of trust levels, and ranges for trust levels are identified.

Chapter 13 describes the deployment of the $W_S T_B AC$ model in a prototype implementation. An overview is given of the implementation of components, while the operation of the prototype is discussed on the basis of the case study example.

Chapter 14 concludes the thesis.

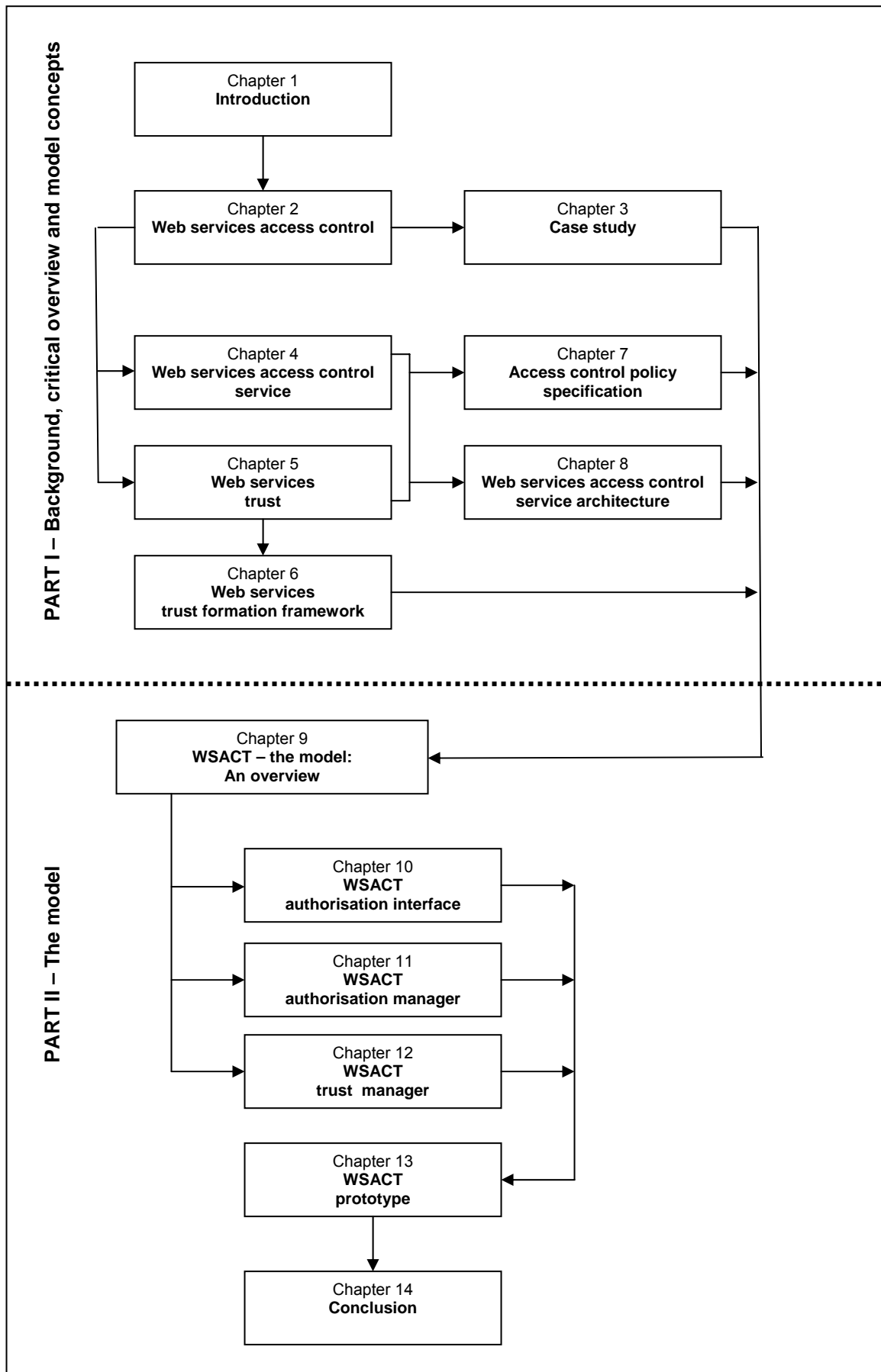


Figure 1.1: Thesis layout

2

Web Services Access Control

Web services are Extensible Markup Language (XML) applications mapped to programs, objects, databases or comprehensive business functions (Newcomer 2002). By using an XML document in the form of a message, a machine sends a request across a network to another machine that hosts a web service. A reply in the form of an XML document may optionally be returned. Web services standards define the format of the message, specify the interface to which the message is sent, define mechanisms to publish and discover web services interfaces, and describe conventions for mapping the contents of the message into and out of the applications that implement web services. The fundamental abstraction is thus one of message passing, as applications access web services operations by sending messages. Exchange of XML messages occur for both synchronous and asynchronous interactions.

The focus of this chapter is to provide a background on web services, with the aim of identifying relevant issues to be addressed by a web services access control service. As the focus of web services integration is machine-to-machine interaction, the web services access control service should be designed so that human intervention is limited. As a web service is essentially just an interface, access control is to be implemented at the perimeter of web services-enabled applications, where interfaces are exposed. Finally, it is important to determine the level of granularity of access control by analysing components of web services that need protection. These issues are addressed by describing the web services architecture, with a high-level account of components and web services technologies. The last paragraph describes current web services security standards.

2.1 WEB SERVICES ARCHITECTURE

The web services architecture is based upon the interactions between three roles (Coyle 2002): web services provider, web services broker and web services requestor, shown in Figure 2.1.

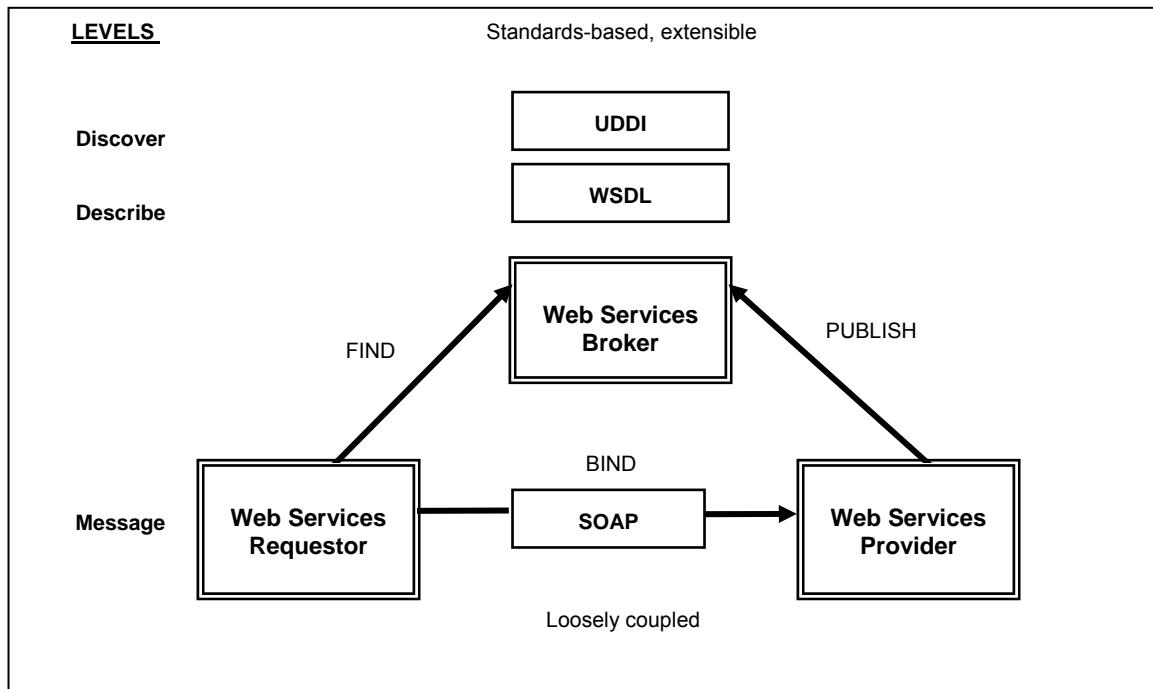


Figure 2.1: Web services architecture

- **Web services provider:** At business level, the web services provider is the owner of application programs exposed as web services. From this viewpoint, the web services provider is referred to as the *web services entity*. From the perspective of the access control service, the *web services provider* is seen as the machine that holds the web services implementation.
- **Web services requestor:** At business level, the web services requestor is a human or organisation that requires a function to be fulfilled. From this viewpoint, the web services requestor is the entity that consumes web services that are available on the Internet, and is considered a *web services entity*. From the perspective of the access control service, a *web services requestor* is a machine that acts on behalf of humans or organisations or other web services or applications.
- **Web services broker:** A web services broker is a searchable registry of service descriptions where web services entities publish their web services, and where web services requestors or web services entities find and obtain binding information for these web services.

Operations that support the interactions between web services providers, web services brokers and web services requestors are publish, find and bind:

- **Publish:** Publication of a web services description is any action by the web services provider that makes its interface available to web services requestors. It may include advertising the interface description through a web services broker, providing a URL (Uniform Resource Locator) pointer, or giving direct access to the interface description.

- **Find:** Finding web services is any action where the web services requestor retrieves a web services description directly or queries a web services broker for the type of web services required. This can happen at either design time for the purposes of application development, and at runtime to retrieve the web services' binding and location description for invocation.
- **Bind:** Binding to web services is when a web services requestor invokes or initiates an interaction with web services at runtime, using the binding details in the web services interface description to locate, contact and invoke the web services.

2.3 FUNCTIONAL COMPONENTS OF WEB SERVICES

In order to determine all aspects of web services that must be considered for access control, a high-level view of the components of web services and their interactions is now defined. Web services entities may have a number of machines that support comprehensive business functionality offered by their organisations. Machines support a variety of software components such as object methods, functions, database queries, legacy applications, and adapters to ERP (Enterprise Resource Planning) packages, which are exposed by web services interfaces. Web services functionality that needs to be protected can be grouped according to web services operations, web services, web services classes and web services compositions, as is depicted in Figure 2.2.

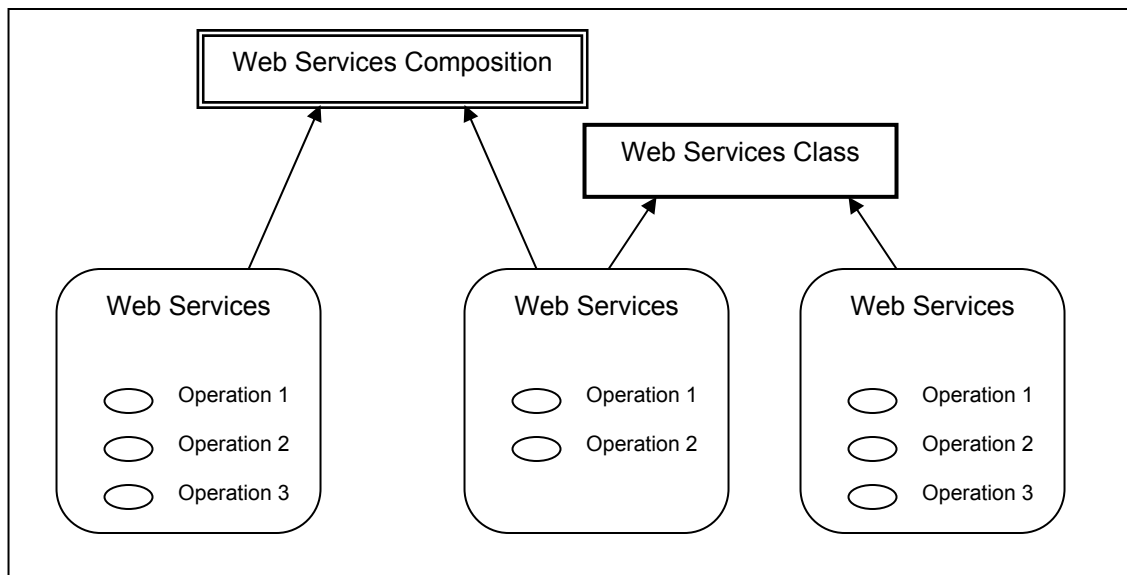


Figure 2.2: Web services functional components

For instance, an operation such as *place_order* is part of the *order* web services, which is grouped in the *process_order* web services class. The *order* web services is part of the *book_store* web services composition. These groupings identify hierarchies over which access

control logic can be defined. The next paragraphs consider each of these groups in order to explain how they are structured.

2.2.1 Web services

The term *web services* is used in many different contexts. In literature, terms such as *XML Web Service* (Microsoft White Paper 2001), *e-service* (Damiani et al. 2001) or just *service* (Christensen et al. 2000) have been used interchangeably, where the type of technology very often determines the terminology. Recently, there has been a general convergence to the term web services (MS Web Services 2005), (Apache Web Services 2005), (IBM Web Services 2005), (Booth et al. 2004). For the purposes of this thesis, the term web services (spelt lower case) will be used.

The W3C (World Wide Web Consortium) (W3C 2005) provides the following definition of web services in the Web Services Architecture document (Booth et al. 2004).

*“A Web Service is a software system designed to support **interoperable machine-to-machine interactions** over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”*

A web service is thus an interface that describes operations that are network accessible to XML-based messages. The interface exposes software components that support interoperable machine-to-machine interactions over a network. Thus, it is mainly used to collect operations that map to software components. The interface hides web services implementation details, allowing it to be independent of the hardware and software platform where it resides, and of the programming language in which it is written. Other systems interact with web services through SOAP (Simple Object Access Protocol) messages, in a manner prescribed by the web services interface description.

The web services interface description includes all information needed by the communication protocol to interact with the web services, such as the URI (Uniform Resource Identifier) that uniquely identifies its location, network protocols by which it may be accessed, the list of operations that it exposes, and metadata such as namespaces and other requirements. This information is described in a machine-readable document in XML notation, and is made available to web services requestors. Additional metadata to describe non-functional aspects of a web services may be made available in XML documents, where the web services description document may maintain a pointer to such documents.

2.2.2 Web services operation

A web services operation is the software or hardware that sends and receives messages (Booth et al. 2004). It is a computational resource that is owned by a web services entity. As web services mainly focus on message exchanges, a web services operation is defined by the interactions between a web services provider and requestor. Receiving SOAP software at the web services hosting machine will decide how to activate the software component, after receiving a request. Essentially the software component processes a message in the form of an XML document. Web services operations are stateless as there is no assumption that subsequent messages received are associated with prior ones, and web services operations are not mutually dependent.

An operation is invoked by a message. For instance, for *request-response* interactions, a request message defines a request for a software component and includes required parameter values. The response message expresses the return value and output parameters. In order to define a *request* for a software component, the following information is required: the name of the web services operation, its location, a set of input parameters, metadata such as namespaces, and non-functional requirements. In order to define a *response* of a software component, the following information is required: a set of output parameters, metadata such as namespaces and non-functional requirements.

2.2.3 Web services class

Web services can be organised into classes of services. Such classes can be used to refer to a set of web services. For instance, a web services *process_order* may group other web services such as *view_order* and *cancel_order*. This concept is not supported by current tools and technologies, but may be abstractly defined for the purposes of access control logic.

2.3.4 Web services composition

Web services that expose basic functionality do not rely on other web services to fulfil requests. Typical examples include stock quotes and credit card verification. In order to carry out a complex business transaction, web services can be used in conjunction with other web services. Such web services compositions provide a value-added service such as integrated travel planning and insurance brokering.

2.3 WEB SERVICES TECHNOLOGY

Web services build on XML-based technologies such as SOAP (Box et al. 2000), WSDL (Christensen et al. 2000) and UDDI (Bellwood et al. 2003), and are discussed in the next paragraphs. The web services environment is rapidly developing, with a large number of interrelated standards and development tools becoming available. Standards are designed to be extensible, so that they can continue to evolve.

2.3.1 XML (Extensible Markup Language)

XML (Bray et al. 2000) provides the mechanism by which web services definitions and messages and related documents are made accessible across platforms. XML is a metalanguage defined by the W3C. It has become a standard for communication between applications. XML is a set of rules and guidelines for describing structured data in plain text rather than through proprietary binary representations. With XML, an application defines markup tags to represent the different elements of data in a text file so that the data can be read and processed by any application that uses XML.

In the context of web services and the security of SOAP messages, XML is not only used as a message format. It is also used to describe web services comprehensively. SOAP messages are protected by security mechanisms that are described and implemented by means of XML. Parties can only understand one another if they share the same definitions of web services and security mechanisms. Two concepts that are important to highlight are namespaces and schemas:

Namespace

A namespace is a unique name that identifies an XML document or application. A namespace is identified via a URI (Uniform Resource Identifier), which is a unique name used to access Internet resources. It is not necessarily a specific file location, and is therefore preferred over a URL (Uniform Resource Locator). Web services specifications that define the structure of XML-based interfaces, documents and messages are globally located via these unique identifiers in order to enable interoperation between independent domains.

Schema

A schema is a formal specification of the grammar for a specific XML vocabulary. When XML data is passed between domains, it is validated against a schema by an XML parser to ensure that the XML data conforms to the grammar expressed by the XML schema. For instance, an XML Schema such as XSD (XML Schema 2004) provides a language for defining data such as elements, attributes, and their types, as well as the ordering of the data within an XML document.

This is important, as it enables the exchange of valid information across application or organisational boundaries that are understood by both the web services requestors and provider.

2.3.2 WSDL (Web Services Description Language)

According to Christensen et al. (2000) WSDL provides the mechanism by which web services definitions are exposed so that web services requestors can send SOAP messages that would be understood by web services providers. WSDL defines the interface and manner of web services interactions. Additional descriptions are necessary to specify the business context, quality of service and relationships between web services. The WSDL document can be complemented by other web services description documents to describe these higher-level aspects of web services.

WSDL refers to web services as services, and describes them as a set of endpoints operating on messages containing either document-oriented or RPC (Remote Procedure Call) messages. The operations and messages are described abstractly, and then bound together in a concrete network protocol and message format to define an endpoint, as shown in Figure 2.3. Related concrete endpoints are combined into abstract endpoints or services. WSDL is extensible to allow the description of endpoints and their messages, regardless of what message formats or network protocols are used to communicate.

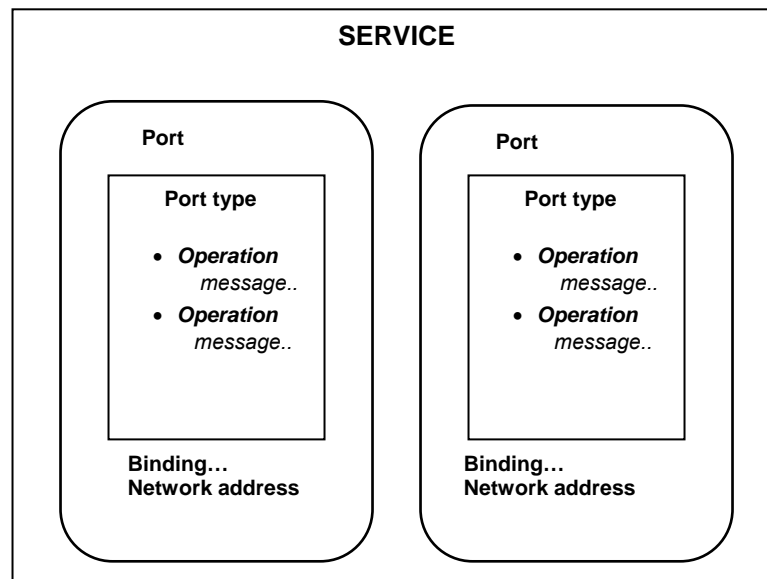


Figure 2.3: WSDL service structure

WSDL documents consist of the following parts:

- **Message:** A message is an abstract definition of the data, in the form of either a document, or parameters used for operation invocation.

- **Operation:** An operation is an abstract definition of the operation for a message such as the name of a method, database query or ERP process that will accept and process the message.
- **PortType:** A port type is an abstract set of operations, mapped to one or more endpoints, defining the collection of operations for the binding. The term portType has been renamed to *interface* in version 2.0 of the WSDL specification.
- **Binding:** A binding is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- **Port:** A port is a combination of a binding and a network address, and it provides the target address for service communication. The term port has been renamed to *endpoint* in version 2.0 of the WSDL specification.
- **Service:** A service is a collection of related endpoints.

The access control service of a web services provider can use WSDL documents to identify resources descriptions that need protection, such as web services and web services operations and parameters. From the viewpoint of a web services requestor, WSDL does not address security, as it specifies functional components only. Security requirements of web services components that publish how authentication, confidentiality and integrity should be enforced in messages may be found in related web services description documents.

2.3.3 SOAP (Simple Object Access Protocol)

SOAP (Box et al. 2000) provides the mechanism by which web services requestors and providers communicate over the Internet. For Internet communication, HTTP (Hyper Text Transfer Protocol) (Fielding et al. 1997) is commonly used because of its stateless nature, ability to pass through firewalls, and pervasive presence between platforms and devices. HTTP has inherent limitations for web services communication, as HTTP cannot specify how the data within the body of a message should be encoded. It also cannot associate metadata in the header with specific transport protocols required when messages are routed between intermediaries to their destinations. To address these limitations, SOAP syntax is defined over HTTP with XML. This means that it is portable without depending on the underlying platform such as byte-ordering or machine-word widths. Different protocols such as HTTP, FTP or SMTP can carry SOAP messages, as a formal set of rules has been defined in the SOAP specification for this purpose.

For the purposes of access control, it is important to understand the semantics of SOAP messages. This includes the type and structure of a SOAP message. SOAP software architecture will also influence the design of an access control service.

SOAP message type

The web services access control service intercepts each SOAP message in order to make an access control decision, based on the content of the message. It therefore needs to know in which XML representation the message is. SOAP message encoding styles that have been defined for this purpose are *Document* and *RPC* (Remote Procedure Call). *Document* uses no SOAP formatting rules for the body of the message. The message contains any agreed upon data between the web services requestor and provider. *RPC* requires the body to contain the name of the operation being invoked, and an element for each parameter of the operation, as shown in Figure 2.4. The access control service is configured according to these semantics.

SOAP message structure

To enable the web services access control service to interpret a SOAP message that is formatted with the SOAP specification, it needs to know what it consists of. Each SOAP message is structured into three parts. At the core is an *envelope* that defines a framework for describing what is in the message, and how to process it. The *envelope* contains an optional *header* and mandatory *body*, as shown in Figure 2.4.

```

<SOAP:Envelope>
  <SOAP:Header>
    <y:XYZHeader xmlns:y="XYZ_URI" SOAP:mustUnderstand="0">
      HeaderInfo
    </y:XYZHeader>
  </SOAP:Header>
  <SOAP:Body>
    <x:XYZMethod xmlns:x="XYZ_URI">
      <XYZArgument>1</XYZArgument>
    </x:XYZMethod>
    <SOAP:Fault>
      <faultcode>Server.InvalidName</faultcode>
      <faultstring>Name is wrong </faultstring>
    </SOAP:Fault>
  </SOAP:Body>
</SOAP:Envelope>

```

Figure 2.4: SOAP message

SOAP message body

The focus of web services access control is the body, which in the case of *RPC*, contains the operation name and associated parameters in either the SOAP request or response, and optional status or error information in the *fault* section. A SOAP fault element contains an application-defined error code, a human-readable message describing the fault and any application-specific error information. The SOAP fault can be used to return an error code to a web services requestor to indicate that an access request is denied.

SOAP message header

The *header* is used to hold metadata that is associated with the SOAP request and plays a very important role in the security of messages. The header may be extended with XML elements not defined by the SOAP specification itself in order to carry authentication and access control information, as well as public keys and information on encryption algorithms that are used to encrypt the message. Header information is not always intended for the ultimate receiver, as a message may partly be processed *en route* to its destination.

SOAP software architecture

The architecture of SOAP client and server software accommodates the pre-and post processing of SOAP messages. SOAP messages can be intercepted before they are deserialised from XML into types that are used by the application. The operation name that is requested, as well as parameters and associated header information can be extracted. This architecture gives developers the ability to build an infrastructure for access control.

2.3.4 UDDI (Universal Discovery Description and Integration)

UDDI (Bellwood et al. 2003) provides the mechanism by which available web services are found. UDDI creates a global, platform-independent, open framework to enable web services providers and requestors to discover each other. UDDI is a technical discovery layer that defines the structure for a registry of web services providers and their web services, and a programming interface that can be used to access registries with this structure. The information provided in a UDDI registration consists of three components: White page, including address, contact and known identifiers; Yellow page, including industrial categorisations based on standard taxonomies; and Green page, for the technical information about web services that are exposed by the business. For the purposes of this thesis it is important to note that UDDI can be used by web services providers as a source of information, in order to establish trust with web services requestors that are unknown.

Well-known vendors such as IBM (IBM UDDI 2004), SAP (SAP UDDI 2004) and Microsoft (Microsoft UDDI 2004) operate public UDDI registries. The operators replicate the registrations across all nodes on a regular basis, thus resulting in a complete set of registered records available to all. The operators all support a common set of SOAP APIs that will ensure the integrity and availability of the information provided. As UDDI registries are not moderated, the information that is held is often unreliable and outdated.

2.4 WEB SERVICES SECURITY AND RELATED STANDARDS

The main concern and obstacle to the adoption of web services as an industry solution is security (Damiani et al. 2002; Schmidt et al. 2005). SOAP messages present new security problems to organisations, as they are text-based and not only machine-readable, but also human-readable. SOAP requires no additional ports or access mechanisms beyond those used in web servers, which are found in almost every organisation. SOAP messages may be treated by firewalls as simple HTTP requests for web pages, resulting in possible unauthorised access to the internal applications behind the firewall. Each SOAP message can be seen as a potential security threat, as it presents itself just as normal web traffic would. Applications that receive SOAP messages may therefore be endangered by false claims or malicious information.

Web services that are exposed to the external world should not only be protected from unauthorised access, but the authentication of web services requestors, message integrity and confidentiality, as well as the non-repudiation of transactions should also be considered. Since web services use message-based communication, web services security must be based on message security. While there are many techniques to secure messages, interoperability between security domains is an important question that needs to be addressed.

The confidentiality and integrity of SOAP messages, and the authentication of web services requestors can be ensured with SSL (Secure Socket Layer) (Frier et al. 1996). This will only be practical in cases where a SOAP message is sent directly between a web services requestor and provider. SSL does not provide end-to-end security when a message passes between intermediaries who may all need to access parts of the message. It follows that authentication, and message integrity and confidentiality cannot be assured when messages pass through intermediaries. Because SOAP can be used over various Internet protocols, an intermediary point may decide to change to, for instance, SMTP (Secure Mail Transfer Protocol). This will cause interoperability problems when securing messages, as SSL cannot be used for SMTP communication. Messages should therefore be secured by mechanisms that are protocol-independent.

Conceptually, information security is enforced by means of information security services such as authentication, authorisation, confidentiality and integrity. A number of specifications have been developed over XML to ensure the security of a message between intermediate and destination security domains. Others have been developed to ensure interoperability. The next paragraphs describe these types of specifications.

2.4.1 Specifications for XML-based security mechanisms

Security specifications have been developed to selectively encrypt or sign parts of XML documents, to manage public keys, to transfer assertions and to define access control. Figure 2.5 shows some of the security standards that have been defined over and above the SOAP protocol.

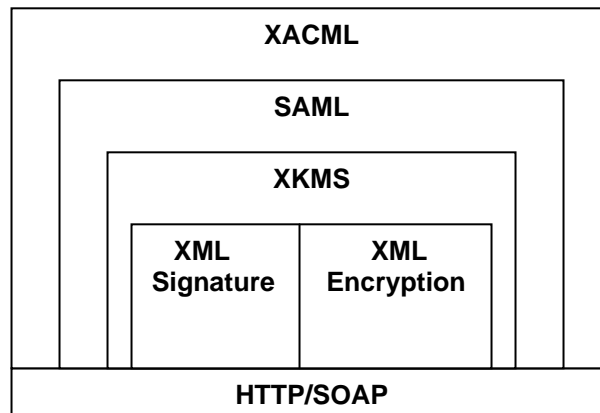


Figure 2.5: Security standards over SOAP

XML Encryption

XML Encryption (Imamura et al. 2002) describes how the integrity and confidentiality of XML documents are enforced. XML Encryption does not introduce new algorithms or techniques, but rather provides a way to format metadata about the algorithm that was used, and when encryption occurred.

XML Signature

XML Signature defines how to sign portions of an XML document, and how to express the digital signature of any data as XML (Bartel et al. 2002).

XKMS

XKMS (XML Key Management Specification) enables PKI (Public Key Infrastructure) services such as trustworthy registering, finding and validating of keys through XML messages (Dillaway 2001). As an XKMS service communicates with XML messages by default, it should be implemented as a web service.

SAML

SAML (Security Assertion Markup Language) (Hallam-Baker et al. 2003) is a standard methodology to represent authentication and authorisation information in XML format so that it can be exchanged across security domains. Information is called an assertion, or declaration of fact, and is made by an issuing authority. The following three types of assertions can be used:

- *Authentication assertion* that declares that a subject was authenticated by the issuing authority at a given time, for example “subject X has been authenticated by means of methodology Y at time Z”.
- *Attribute assertion* that describes specific attributes of a subject as name-value pairs, for example “subject A has been associated with name-value pairs a=b, c=f at the time of this assertion”.
- *Authorisation decision assertion* that declares whether a given subject has been granted specific permissions to access a particular resource, for example “subject A with evidence B has been permitted to access resource C with privilege D at time E”.

SAML assertions can enable a web services requestor and provider to communicate the abilities of users from their domains to each other in a standardised manner – in a SOAP header. This can only be achieved if web services requestors and providers use technologies that support SAML.

XACML

XACML (XML Access Control Markup Language) (Anderson et al. 2003) is an initiative to standardise representation of access control policies in a flexible, extensible XML format. It allows fine-grained access control policies to be expressed in XML to protect files, web services operations, and other objects. It is an OASIS (OASIS 2005) standard that describes both a policy language and an access control decision request/response language in XML. XACML is not only used to protect XML documents, but it can be used to protect any of the resources of an organisation. Web services requestors and providers can use XACML to integrate or share access control policies in order to express cross-domain policies. Policies of both web services requestor and provider should refer to objects and subjects in a similar manner, if policies are to be used by a single decision-making component.

2.4.2 Specifications for XML-based security interoperability

The XML security standards mentioned so far, address mechanisms to implement security services for XML documents that are transported over SOAP. The SOAP header plays an important role in implementing security services, as it is used to carry identities, abilities or roles of users or applications, access control information, public keys, tokens and other information such as signature and algorithms that are used to encrypt and sign a message. Web services requestors, who communicate security information to web services providers, can employ their own schemas to define, for instance, usernames and passwords. This leads to costly, proprietary implementations and prevents interoperability in a large population of web services requestors and providers. To ensure interoperability, SOAP messages need to be standardised.

The web services security framework proposed by Microsoft and IBM (IBM and Microsoft 2002) has the aim of providing interoperability between independent security domains. It is defined by seven security specifications, shown in Figure 2.6. The first specification, WS-Security, was submitted to the Organization for the Advancement of Structured Information Standards (OASIS) in September 2002. By August 2003, WS-Policy, WS-Trust and WS-SecureConversation and WS-Federation were submitted. WS-Authorisation and WS-Privacy have not yet been addressed.

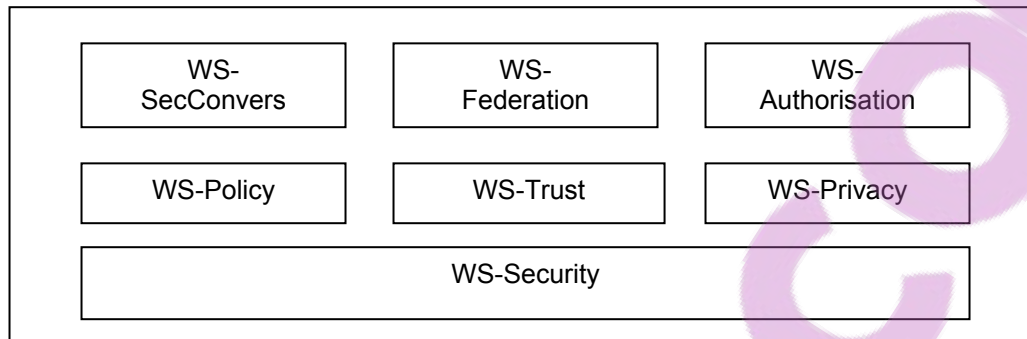


Figure 2.6: Web services security framework

WS-Security

WS-Security (Web Services Security) (Atkinson et al. 2002) is a mechanism for incorporating security information into SOAP messages with SOAP header extensions in a standardised manner. It structures the use of binary tokens for authentication, digital signatures for integrity, and content-level encryption for confidentiality. As the security semantics of SOAP messages are standardised, tool support can be provided. External integration with partners or internal integration between geographically dispersed applications can be implemented.

Six other specifications defined over and above WS-Security address two areas, namely *policy* and *federation*. Policy is defined by WS-Policy, WS-Trust and WS-Privacy and expresses requirements and capabilities that can enable organisations to interoperate. Federation is defined by WS-SecureConversation, WS-Federation and WS-Authorization, and it addresses the manner in which web services create a security context between themselves in order to interoperate.

Two existing specifications that are related to the web services access control service described here is WS-Trust and WS-Policy. For web services, trust between independent domains is important as it forms the basis for accesses granted to sensitive resources.

WS-Trust

In order to address trust for web services, the Web Services Trust Language or WS-Trust (Della-Libera et al. 2003) has been published. It introduces the concept of a Security Token Service (STC). It allows interoperability between a web services requestor and provider that do not know each other, by enabling them to determine whether they can trust each other's asserted credentials. The STC can, for instance, convert one token into another, and thereby create trust

between two domains. WS-Trust poses limitations to the establishment of trust relationships, as it does not enable web services to treat partners and strangers differently. Trust established between web services is of binary format – it either exists or it does not. In this thesis, access control decisions are influenced by a trust relationship between a web services requestor and provider, which exists in a virtual society. Trust relationships should be able to portray elements of human trust in order to be able to take decisions when information is incomplete. WS-Trust therefore provides a solution of limited nature.

WS-Policy

WS-Policy (Box et al. 2003) provides a mechanism for exchanging requirements between web services requestors and providers. WSDL provides a very limited instrument for describing web services, since service requirements such as transactions, reliable messaging, privacy, and security are not addressed. WS-policy is a set of specifications that provide a common language for describing rules that govern the interactions of a web service, or the requirements of web services requestors that interact with a web service. Requirements are written as statements in a formal notation with XML. For instance, a policy may specify that a particular web services operation is only available at a particular time of the day, or that all requests and responses must be encrypted. The ultimate vision for WS-Policy is one of automated machine-to-machine policy negotiation, before web services interaction occurs. This would enable the establishment of a dynamic, mutually agreed-upon policy for both web services requestors and providers. It would be important to make use of the standard manner in which WS-Policy communicates requirements to others, as access control and trust requirements should be made available to others to support interoperability at all levels. WS-Policy is for instance used by WS-Trust to express which security tokens may be used by a particular web service or web services operation.

2.5 ENVIRONMENTAL ACCESS CONTROL REQUIREMENTS OF WEB SERVICES

A web services access control service will be effective only if it considers the specific requirements of the environment in which it is to be deployed. The web services environment refers to an environment supported and delimited by the web services architecture. When access control is defined for web services, *environmental access control requirements* to be addressed include the following:

2.5.1 Autonomy

Web services requestors and providers interact across trust boundaries, with no single entity in charge. It would be unrealistic to expect of owners of web services to modify their access control

policies and enforcement to comply with the requirements of others. Web services are therefore autonomous entities.

For this research, autonomy refers to the degree of compliance of a web services requestor or provider to global access control rules, applicable to all parties participating in web services interactions, which have been defined in a centralised manner. Interacting web services can be autonomous in both their design and communication. In autonomous interactions, each web services entity can be viewed as a black box that is able to exchange information by sending and receive messages. Autonomous decision-making requires that information is acquired from several sources, and is evaluated to determine how to behave, depending on pre-defined goals expressed via policies or rules.

Loosely-coupled design, an access control requirement that is defined next, promotes autonomy, as interactions are defined via well-defined interfaces and standards that allow web services providers to have more local control over implementation. Fully autonomous interactions are difficult to achieve as it requires the use of well-defined vocabularies and translation capabilities by interacting web services entities.

2.5.2 Loosely coupled

A loosely coupled access control service for a web services provider can be achieved if it's dependencies with the application of the web services requestor is reduced to a minimum. In order to achieve this, the access control service needs to support techniques that promote loose coupling described in Chapter 1.

The nature of access control influences the implementation of these techniques. Ideally, an access control service should be designed so that it functions unobtrusively in the background of the application. This means that its messages, interface, addressing, and exchange patterns should be implemented so that the access control service remains unobtrusive. In order to remain unobtrusive, the access control service should preferably use the same messaging infrastructure as the web service, its interface should be part of the functional interface of the web service, it should seamlessly be invoked when web services operations are accessed and its should be use the same exchange patterns as the web service.

To be loosely coupled, and at the same time unobtrusive, the access control services should be integrated with the functional web service, and be deployed as a layer in front of the web service. It needs to be supported by access control policies that decouple access control from platform dependent access control mechanisms. This feature leads to the next environmental access control requirement, namely policy-based compatibility.

2.5.3 Policy-based compatibility

Web services providers interact with others based solely on contracts defined by interface documents that describe message-passing behaviour. To be contracted means that a web service's behavior, as well as how to bind to it, and its input and output parameters, are available to those web services requestors who are able to access it. A web service provider requires various features that can be controlled or described using policy rules. Examples of such features include authentication, access control, reliable messaging, privacy, and application specific service options. Web services requestors and providers can only interoperate by means of policy if both structural and semantic policy compatibility exists where:

- *structural compatibility* is based on schema
- *semantic compatibility* is based on explicit statements of capabilities and requirements defined by vocabularies

To enable a loosely coupled web services access control service, a web services provider thus needs to publish well-defined access control requirements. This will reduce the interdependencies between the web services requestor and provider. A central question is determining what to publish, without compromising the access control policy of the web services provider. In addition, web services requestors and providers need to create a comprehensive contract that will govern all information security related interactions, by a process of policy negotiation, the next environmental access control requirement.

2.5.4 Policy negotiation

Negotiation by means of policy is a technique used by automated trust negotiation systems (Bertino et al. 2004). In these systems, access is granted to a user based on a process of iterative credential disclosure between a client and a server, where the process is governed by policy. For the web services access control service, this concept is extended. Policy negotiation also refers to negotiation over non-functional elements such as authentication, reliable messaging, privacy, and application specific service options, by means of policy. It is the process by which web services requestors and providers come to an agreement with regards to the types of mechanisms or constraints that are acceptable to both parties. This means that policies need to be attached to specific web services, need to be discovered by potential web services requestors or providers, and exchanged. A negotiation protocol is required to ensure that interactions are understood by all parties. When policies are retrieved by either a web services requestor or provider, negotiation can for instance be accomplished by merging the policies of parties to determine their compatibility. The inclusion of policy agreement over non-functional requirements gives comprehensive meaning to the term policy negotiation.

2.5.5 Quality of service

Quality of service (QoS) helps to distinguish between web services providers. The QoS a web services provider delivers is a decisive criterion when web services with the same functionalities are available to a web services requestor. QoS determines the web services usability and utility, both of which influence the popularity of operations exposed by the web service. Important requirements for supporting QoS in web services providers are its availability, accessibility, its transaction integrity, performance, reliability, adherence to correct versions of standards, and information security. Information security is considered very important to QoS because web services operations are invoked over the public Internet. Information security addresses services such as authentication, integrity, confidentiality and access control that are all addressed in a comprehensive solution. Access control would be particularly important to web services requestors who are sensitive about their personal information. Web services requestors may select the “best” service, based on the manner in which information security services are implemented.

2.5.6 Standards-based interaction

Web services entities frequently form new business relationships with others. In order to adopt to a changing environment, a flexible security framework is required that is based on approval and universal acceptance of standards. Security standards typically focus on establishing broad frameworks and avoid specific technologies and configuration models. Business partners are thus able to lessen interoperability problems among their disparate applications. The adoption of web services security and other standards is important, as it would allow interoperation between web services requestors and providers, perhaps with the same success achieved by the adoption of SSL for Web-based transactions. Consequently, the semantics of messages related to access control interactions, and policies used by the access control service would be understood by others.

2.6 CONCLUSION

The concerns about using web services between organisations are an extension of those that currently exist within organisations. Security has always been a difficult problem to solve for cross-organisational interactions. The use of SOAP augments this difficulty, as it poses an additional threat to organisations. As mentioned earlier, current solutions such as SSL can provide a point-to-point solution, but for loosely coupled interactions, end-to-end protection is required. Several specifications have been created in order to transport security information in XML across domains. Unfortunately, none of the current specifications support cross-domain access control integration.

This chapter highlights six *environmental access control requirements* that need to be met when a web services access control service is designed. A complicating factor is the autonomy of web services requestors and providers, and the fact that they cannot adopt similar access control capabilities. The autonomy of web services providers lead to the access control requirement of loosely coupled, which in its turn requires the use of policy and policy-based compatibility. The existence of different policies demands a process of policy negotiation. By implementing information security services that includes access control, others are able to determine the quality of a web service. Finally, the adoption of standards-based solution would ensure that web services requestors and providers can interoperate autonomously.

The mentioned *environmental access control requirements* underscore the fact that an access control service should be designed in conjunction with the loosely coupled web services architecture. To address access control in a loosely coupled manner, the “describe”, “discover”, and “message” levels play an important role. A web services provider can make its access control requirements known to requestors through the “describe” and “discover” levels. Requestors communicate platform-independent access control information such as credentials to service providers at the “message” level with SOAP headers. In order to address the shortcomings of current access control solutions, this thesis sets out to define an access control service that maintains the autonomy of web services providers. To further highlight additional access control requirements for web services providers, the next chapter describes a case study.

3

Case Study

Web services technology enables the integration of business functionality between independent domains, to form more sophisticated applications. Integration occurs on a machine-to-machine basis and appears virtual to the end-user using such functionality. For such applications, the term *virtual application* will be used throughout this thesis. The focus of this chapter is to describe an example of a typical virtual application. The discussion emphasises reasons for, and background to, cross-domain functional integration. The chapter is concluded with a high-level access control policy for eBooks, a web services provider, which highlights difficulties faced by administrators of an access control service.

3.1 THE VIRTUAL APPLICATION

The virtual application described here consists of two independent organisations, eBooks and eLoans. eLoans is a web services requestor that integrates eBooks, a web services provider, into its business functionality to provide a real-life business scenario. The paragraphs below describe eBooks, eLoans and the virtual application in which they collaborate.

3.1.1 eBooks

eBooks is a web services provider supported by Books Inc., a large retailer specialising in academic books. It has a well-established infrastructure for efficient online ordering and payment processing. It is well known for its fast and efficient delivery of books for local and international orders. After a number of years of investing in development and infrastructure, the platform is stable enough to be used by a large numbers of partners and their remote users. eBooks exposed its business functionality by means of web services interfaces, shown in Figure 3.1. Web services entities are given the ability to enhance their applications with the exposed web services operations of eBooks, and at the same time, eBooks is exposed to a much larger user population that could potentially increase its revenue. Web services requestors receive a small percentage of all sales made through its applications as an incentive to participate.

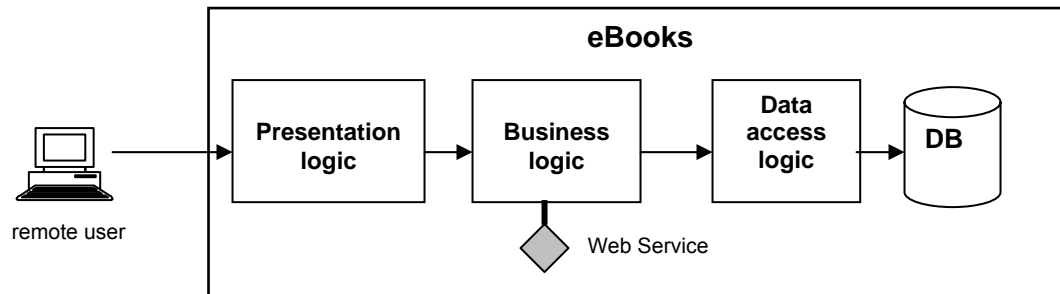


Figure 3.1: eBooks web services architecture

Business logic of eBooks, exposed as web services operations, supports a number of operations that can be invoked. All web services requestors are required to register by using the *Register_Requestor* operation. Details such as the requestor's name, address and contact information are required. In response, successfully registered web services requestors are supplied with URLs of policy documents that are needed in order to be able to further collaborate with eBooks. Other operations available are:

- *Search*: A user of any web services requestor may search all available books by title, author or ISBN number.
- *Search_Academic*: A user of any web services requestor, who is a registered student at a recognised tertiary institution, may search for unusual or unique academic books by title, author or ISBN number. The student must prove that he/she is registered at an institution, by supplying a credential.

Web services operations are then organised according to two levels of service as follows:

- Minimal inter-dependency
- Increased inter-dependency

3.1.1.1 Minimal inter-dependency

Web services requestors act as intermediary points between the eBooks web service and the users of web services requestors such as eLoans. Users are held accountable for their actions as they are registered individually with eBooks, and are liable for orders that they place. Operations include:

- *Register_User*: A user, invoking operations from a web services requestor, first registers by supplying personal information. He/She is provided with a password.
- *User_Order*: A remote user may place an order for books.
- *View_User_Order*: A remote user may be given the option of viewing his/her own, already placed order.

- *User_Payment*: A remote user may be given the option of paying for an already placed order. The order number must be supplied, with the credit card details of the remote user, and his/her identity.

Access to sensitive operations such as the placement of orders is mainly determined by the verification of the credit card details of the individual user. The trustworthiness of the web services requestors has no significant effect on access control decisions.

3.1.1.2 Increased inter-dependency

eBooks exposes additional web services operations, to provide more flexibility to web services requestors. They are given limited control over manner in which interaction occurs. Operations include:

- *Order*: Web service requestors can give their users the ability to buy books, but can control all order details such as limiting categories of books that are ordered, and total amount spent. All charges are to the account of the web services requestor organisation. The web services requestor is thus responsible for all users' orders that are placed.
- *List_Specials*: Special offers may be made available on to-be-released products or other special offers. At the same time, the profit made by a web services requestor decreases. This is thus a benefit that a web services requestor may pass on to some of its user population. Even though this is a choice made by the management of a web services requestor, it is applied by the web services operation at eBooks.
- *View_Order*: An administrator of a web services requestor is given the option of viewing orders placed by their users.
- *Make_Payment*: An administrator of a web services requestor may make a payment for orders placed by their remote users. The order number must be supplied, the credit card details of the web services requestor organisation, and the identity of the administrator.

These operations are the focus of this research. The web services requestor acts on behalf of its users, and is responsible for their actions. Its trustworthiness thus becomes more important to eBooks. Furthermore, it would be advantageous to identify web services requestors who would bring in more revenue.

3.2.1 eLoans

eLoans is an organisation that provides study loans to students. After a recognised tertiary institution accepts a student, she/he may apply for a study loan. eLoans is an online system that allows students to monitor their loan application process, as shown in figure 3.2. The eLoans portal provides options to students such as *List_loan_types*, *Register*, *Submit_application*,

Modify_application, *View_application_status* and *Accept_loan* that is supported by presentation and business logic. Employees of eLoans have a web interface from where they manage loan applications. They will be able to view, accept, modify and reject applications.

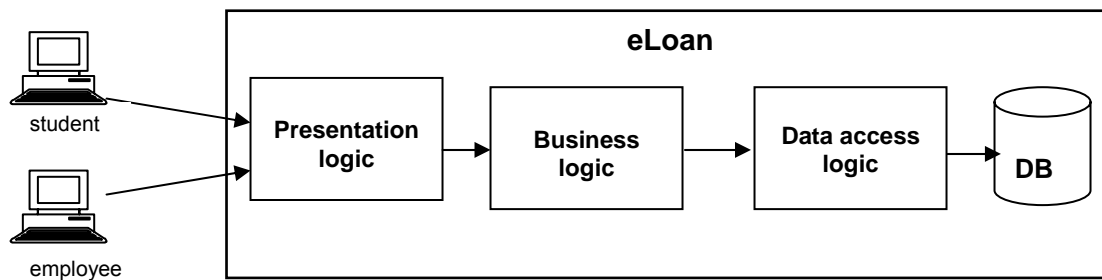


Figure 3.2: eLoans architecture

If a granted study loan is not paid directly to a student, the management of eLoans can control the expenditure of the student against his/her approved loan. To achieve this, eLoans makes use of web services to interoperate with other organisations as follows:

- The study and hostel fees of students are paid directly to the tertiary institution where they are registered.
- Tertiary institutions directly report to eLoans on students' progress during the year.
- Expenditures such as the purchase of academic books are controlled by allowing students to only purchase books through online bookstores such as eBooks.

3.1.3 The virtual application

A part of the loan granted to a student is for the purchase of academic books. To directly manage student purchases for books, eLoans gives students access to eBooks web services operations at its portal. This integration leads to the creation of a virtual application, as shown in Figure 3.3.

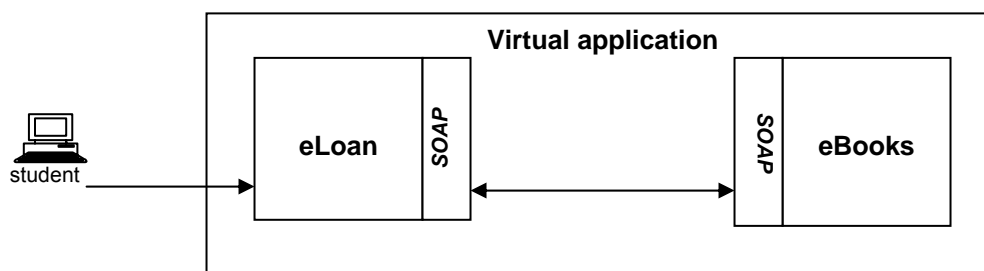


Figure 3.3: Virtual application architecture

A student may interact with the web services operations of eBooks as follows:

He/She logs in to the eLoans portal application where he/she is authenticated. The student selects an option displayed on a page on her/his browser for the execution of a web services

operation that is defined in the eBooks security domain. This operation may for instance be to perform a *search* through the catalogue of books by ISBN number. On behalf of the student, eLoans, the web services requestor, invokes the search operation. It constructs an XML document that includes the required information. The message is wrapped in SOAP, as shown in Figure 3.4, and sent to eBooks. At eBooks, the SOAP message is routed to the appropriate web services operation, and its content is extracted to invoke a software component. Next, a number of machine-to-machine interactions (based on the SOAP protocol) occur on behalf of the student, before an order for books is finalised.

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  <soap:Body xmlns:m="http://www.eBooks/retail">
    <m:search>
      <m:ISBN>99299887651101</m:ISBN>
    </m:search>
  </soap:Body>
</soap:Envelope>
```

Figure 3.4: SOAP request for the search operation

An important benefit that eLoans acquires by collaborating with eBooks is the ability to restrict students. As the eLoans web services requestor determines the content of the SOAP message, students are easily limited to purchase a controlled number of books, at a limited price, from specific academic categories. eLoans thus provides sophisticated yet cost-effective services to students. On the downside, the virtual application becomes more complex to maintain as both the functional and non-functional components, requirements and capabilities of eLoans and eBooks need to interoperate. To the students and employees of eLoans, the virtual application would seem local to their organisation. To be effective, it would require access control to be enforced across domains. Characteristics that increase the complexity of access control for this environment are:

- The remote user population is large, distributed, dynamic and unknown.
- eBooks and eLoans are supported by their own distinctive and highly integrated networks, with unique authentication and access control policies that creates substantial administrative problems.
- The relationship of trust between eLoans and eBooks, created when eLoans registers with eBooks through a *Register_Requestor* operation is very limited, as it does not reflect the real trustworthiness of eLoans.

When eLoans and eBooks interact, access control restrictions determine which actions might be performed. The next paragraph describes the types of access control restrictions that eBooks implements to protect its resources.

3.2 ACCESS CONTROL POLICY FOR eBooks

In order to illustrate access control restrictions required by the web services of eBooks, a fragment of a high-level access control policy is shown in Figure 3.5. The objects, subjects and actions to be addressed in the access control policy are defined as follows:

- *Subjects* execute activities and request access to information. It is important to distinguish between the subject who makes the request, and the active subject that is invoked for this purpose. For web services, the subject can be a human, organisation, or application. In the case of eBooks it can be the student. The web services requestor is delegated the right to make requests on behalf of the subject, and is referred to here as the web services requestor or machine. For eBooks this is the eLoans application. A security context must exist between the subject and web services requestor for delegation to occur.
- *Objects* are the targets of activities that are to be performed. The web service interface description contains information on the operations that can be performed, as well as the input and output parameters. Other objects that could be considered would be the server on which the web services reside, the IP address, or the URI of the web service. Internal data kept in a database and other objects must also be protected. In the example policy in Figure 3.5, the search, order operations, as well as the Book.db file are referred to.
- *Actions* that can be performed differ, depending on the type of subject that issues a request. Requestors would generally be allowed to execute a web services operation or to access a server that hosts a number of web services objects or an application. Developers and administrators may be given administrative rights.

Next, the example access control policy for eBooks is given. There are three rules that illustrate the type of access control restrictions that are required to protect resources.

<p>1. Search Academic operation: Web services requestors may execute the <i>Search Academic</i> operation on behalf of subjects who are registered at a tertiary institution</p> <p>2. Order operation: Web services requestors may execute the <i>Order</i> operation on behalf of subjects, if the subject has been authenticated, and the web services requestor is trusted</p> <p>3. List Specials operation: Web services requestors may execute the <i>List Specials</i> operation on behalf of subjects, if the web services requestor is highly trusted</p>
--

Figure 3.5: High-level access control rules for eBooks

These rules highlight the following considerations:

- For rule 1, the focus of access control is on the subject. Its ability as a registered student is required to be granted access to the *Search_Academic* operation.
- For rule 2, the focus of access control is on both the subject and web services requestor. Both the ability (identity) of the subject and the level of trust in the web services requestor is required to be granted access to the *Order* operation.
- For rule 3, the focus of access control is on the web services requestor. The level of trust of the web services requestor must be high before access is granted to the *List_Specials* operation.

The trustworthiness of the web services requestor is needed when these access control rules are applied. To determine the trustworthiness of eLoans, a number of checks can be made such as the verification of the identity of eLoans, the country that eLoans is located in, as this may ensure legal protection in the face of misconduct, the credit record of eLoans, contracts and agreements that are in place to provide protection, the existence of insurance against loss, and encryption algorithms used when sensitive information is sent across a network. eBooks has little means to determine the trustworthiness of eLoans other than with time-consuming and expensive processes that quickly become outdated.

These rules highlight two important considerations:

- To be able to support the application integration provided by web services technology, mechanisms are required to dynamically assess the trustworthiness of eLoans.
- Access control mechanisms are needed to implement stated access control rules.

4

Web Services Access Control Service

The case study highlighted new considerations for web services access control. Current implementations of access control in Web services environments inevitably lead to tightly coupled access control and fail to provide effective protection. This is due to lack of expressiveness of access control policies, missing information and an unmanageable administrative burden that cannot keep up with constant change. In addition, the web services environmental access control requirements that were identified in Chapter 2 are not addressed.

The focus of this chapter is to address the access control considerations identified by the case study. This chapter commences by discussing the access control service and defining its scope for web services providers. Internal access control requirements are listed, with a view to addressing access control considerations that were identified. This is followed by a background on access control and a discussion of the interrelationship between the access control models, mechanisms, and information. Access control models are then discussed in the light of the contribution that their mechanisms can make towards supporting the environmental and internal access control requirements. An analysis of current access control mechanisms finally leads to the identification of two policy levels at which web services access control needs to be addressed.

4.1 THE SCOPE OF THE WEB SERVICES ACCESS CONTROL SERVICE

For this research, the focus of access control is on the web services provider. The access control service is a component placed before the web services provider, as shown in Figure 4.1. It should be designed in such a way that it does not impede the manner in which SOAP messages are exchanged and complies with the environmental access control requirements identified in Chapter 2. These requirements are shown in Figure 4.1.

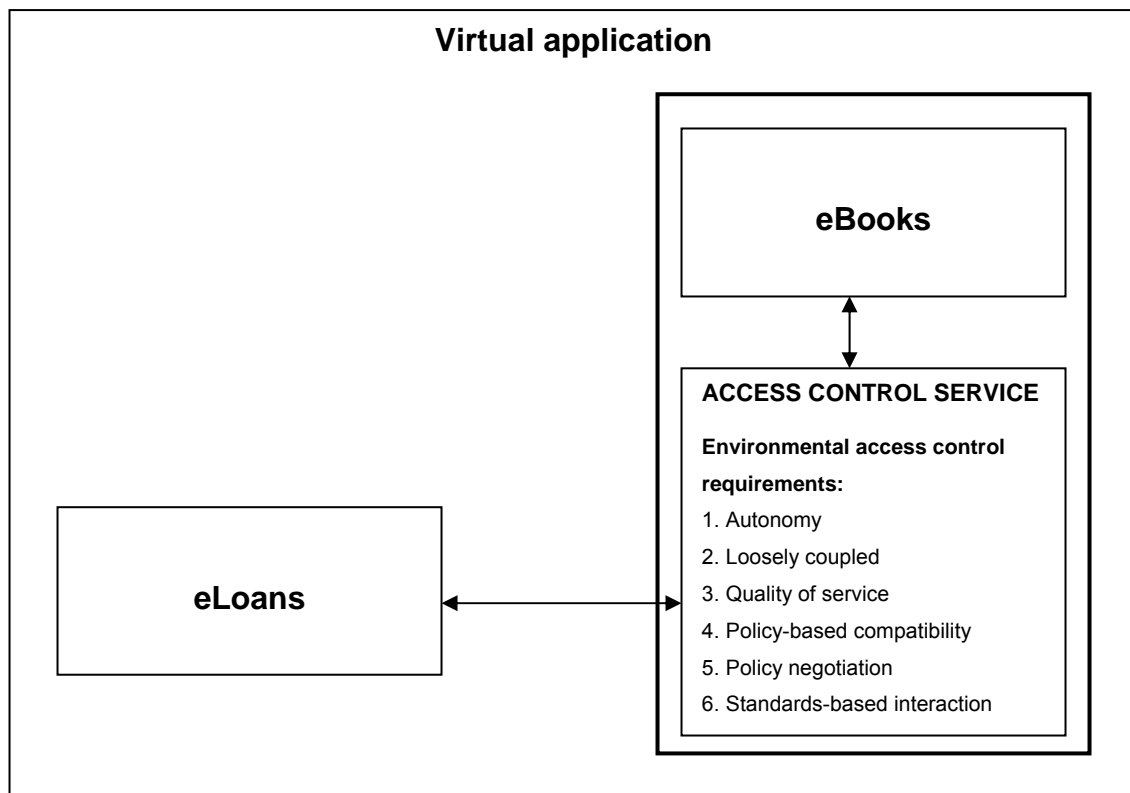


Figure 4.1 eBooks access control service

The scope of the web services access control service needs to be defined for this research, as access control can be addressed on two levels.

- Firstly, access control is autonomously defined for each web services provider that participates in virtual applications.
- Secondly, access control decisions made by web services providers are orchestrated when interactions between autonomous web services occur.

The focus of this research is narrowed down to the first consideration. Web services operations that are accessed by web services requestors must be protected and made secure by their own environment, as it would be unrealistic to expect of web services providers to modify their access control policies and enforcement to comply with the requirements of other web services or web services requestors. The access control service shown in Figure 4.1 is thus limited to independent web services.

In addition to complying with environmental access control requirements, the web services access control service needs to address internal access control requirements that are aimed at protecting internal resources. The next paragraph identifies such internal access control requirements.

4.2 INTERNAL ACCESS CONTROL REQUIREMENTS OF WEB SERVICES

The access control requirements of independent web services such as eBooks need to be defined so that flexible access can be granted to requestors such as eLoans. At the same time, however, access control of an autonomous nature should be ensured. In order to establish such requirements, the entities that participate in internal access control decisions as identified by the case study, are highlighted. Figure 4.2 identifies the entities on which access control decisions are based:

- *Subject (S)* – a user, machine or organisation.
- *Requestor (R)* – the machine making the request on behalf of S.
- *Provider (P)* – the web service that accepts and processes SOAP requests.

There may be intermediary points that pass a SOAP request to its destination. For this research, the focus is on the subject, the web services requestor where the request originates from, and the final destination where the request is processed. Two important considerations are the trust relationship between *P* and *R*, and the properties of *S* that *R* presents to *P*.

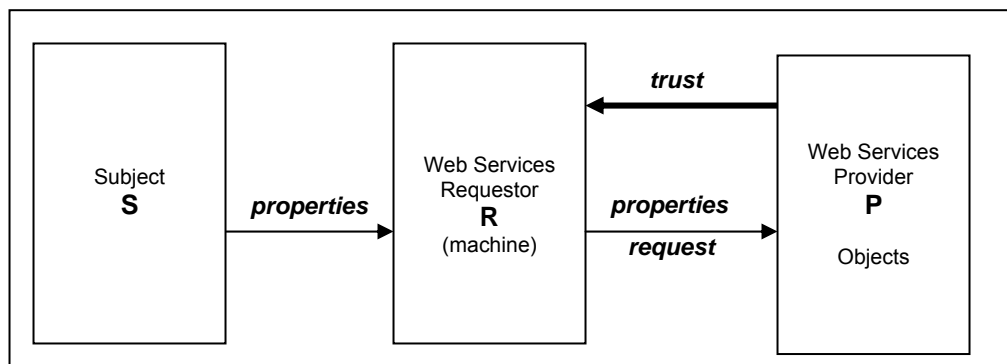


Figure 4.2: Access control entities for web services

From the high-level access control policy rules that were defined in Chapter 3, the following questions emerge:

- *Which objects of P require protection?*
- *On behalf of which subjects (S) can R act, in the domain of P?*
- *Which properties of S, presented by R to P, can give S access to resources?*
- *What level of trust does P have in R?*
- *What level of access does P grant R, acting on behalf of S, if trust in it is low/high?*

These questions help to identify an important focus of the research. An access control decision is based not only on credentials presented by subjects, but also on the trust relationship with the requestor presenting the credentials. The extent to which a trust relationship plays a role in access control decisions is explored by this thesis.

In order to address these questions, six internal access control requirements are listed next and shown in Figure 4.3. For example, requirements of attribute-based access control and differentiated trust are identified as vital to this research. Other requirements such as flexibility, efficient administration, exceptions and conflict resolution, which have been identified by previous research, are included and discussed below for the sake of defining a complete solution in respect of the access control service.

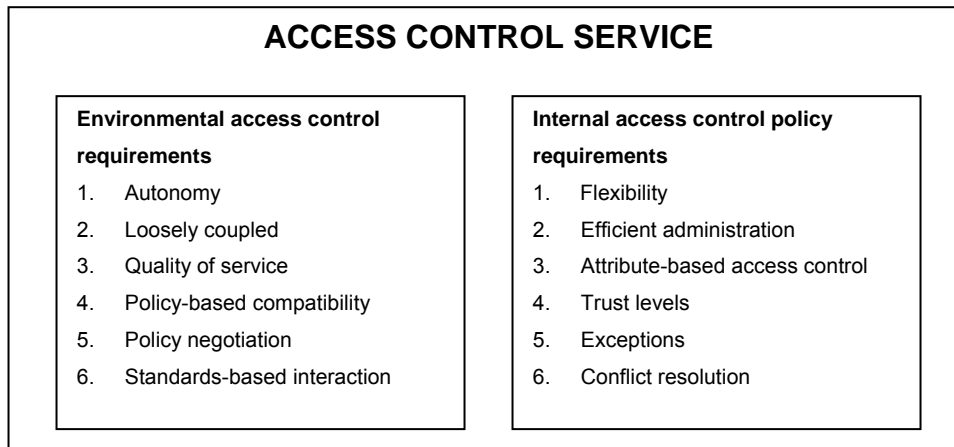


Figure 4.3: Web services access control service requirements

4.2.1 Flexibility

It should be possible to specify access control over every resource, starting from a coarse level such as a collection of web services to a finer level for a particular web services operation (De Capitani Di Vimercati & Samarati 2000). It may even be desirable to specify access control rules below the operation level on parameters. Web services providers should be able to decide about the level of access control rule specification. For instance, in eBooks, all web services operations related to orders can be grouped together and access control can subsequently be specified for this group of objects. On the other hand, if a specific web services operation requires special access control treatment, rules should be formulated to address such an operation specifically.

4.2.2 Efficient administration

Access control rules are specified for large numbers of objects such as web services collections, web services and web services operations, and for other resources such as files and databases. Rules also include subjects such as web services requestors, acting on behalf on users, organisations and other applications. Rule assignment should be administered efficiently (De Capitani Di Vimercati & Samarati 2000). To quote an example from eBooks again – subjects can be organised into groups or roles according to the properties that they possess, and access can be granted accordingly.

These first two requirements are important as web services environments are accessed by large numbers of web service requestors with frequently changing subjects and objects.

4.2.3 Attribute-based access control

The fact that multiple authentication mechanisms exist and are used by web services requestors and providers, creates interoperability, but also administration problems. The web services requestor is an application, and acts on behalf of its subjects. If the web services provider trusts the web services requestor to have performed proper authentication of its subjects, access to resources can be granted without having to re-authenticate subjects. The properties of these subjects (expressed as a list of attributes), rather than their identity, become important to evaluate. For eBooks, subjects can be granted access to web services operations based on properties such as student number, seniority, tertiary institution, library membership and field of study.

4.2.4 Trust levels

In order to process requests from web services requestors, they must be trusted for several reasons. Firstly, they must be trusted to have performed proper authentication of subjects so that access to sensitive resources can be allowed. Secondly, they must be trusted to perform transactions in good faith. Their credibility and reliability will determine the level of trust that the web services provider has in them, and this, in turn, will affect the level of access that their subjects are granted.

To establish different trust levels such as “low” or “high”, a web services provider must be familiar with the various properties that web services requestors possess. Trust statements in the access control policy allow a web services provider to grant better and more advanced access to subjects of trusted web services requestors, rather than to subjects who make requests through a web services requestor with whom there is only a minimal level of trust. Such flexibility gives a web services requestor the ability to foster meaningful business relationships that portray humanistic forms of trust.

For instance, eBooks may have had numerous positive interactions with eLoans over a long period and therefore has high trust in eLoans. If an unknown web services requestor, with whom eBooks has no trust relationship, competes with eLoans for a limited edition of scarce books, eBooks would rather sell these books to subjects from eLoans, simply because it can predict with fair certainty the outcome of these transactions. It also uses this opportunity to strengthen its trust relationship with eBooks by giving preference to the subjects of eLoans.

4.2.5 Exceptions

Both positive and negative access control rules (De Capitani Di Vimercati & Samarati 2000) must be specified so that rights can be revoked dynamically as needed in order to support exceptions that may occur in a dynamic web services environment. For instance, in eBooks access is granted to all web services requestors to execute the *place_order* operation for registered subjects, but such access is not granted for un-registered subjects. An access control rule must therefore be specified for this purpose.

4.2.6 Conflict resolution

Conflict resolution strategies should be supported when conflicting access control rules exist (De Capitani Di Vimercati & Samarati 2000). For instance, in eBooks an administrator may have accidentally granted or denied access to subjects for the *place_order* operation. A conflict resolution rule should exist that would either grant or deny access when such conflicts exist, in order to give the access control service the ability to resolve such inconsistencies.

In order to address the identified access control requirements for the web services access control service, a background investigation on access control follows next. It motivates the access control approach of the web services access control service, and addresses access control models, mechanisms and information.

4.3 ACCESS CONTROL

Access control limits users to obtain access to only those resources for which they have been granted access rights. The development of the web services access control service starts with the definition of high-level access control policy rules, as defined in Chapter 3, and ends with the software implementation described in Chapter 13. This process requires the access control models, access control mechanisms and access control information to be investigated. The relationship between access control models, mechanisms and information is shown in Figure 4.4. An access control model is implemented by means of access control mechanisms, and it is supported by access control information. Access control information, again, is dictated by access control mechanisms.

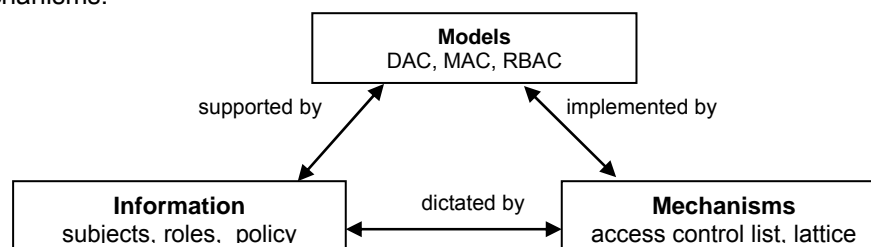


Figure 4.4: Relationships between access control models, mechanisms and information

Access control models concern the formulation of *what* security requirements are to be met, whereas mechanisms focus on *how* these requirements can be met. Information plays a supporting role in both these aspects.

4.3.1 Access control models

Access control models are based on the definition of access control rules, which generally concern the form (subject, object, action). These access control rules specify which subjects can perform which actions on objects. Access control is comprehensively discussed in De Capitani Di Vimercati and Samarati (2000). Models can be grouped into three main classes, namely discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC) models. The discretionary access control model (Lampson 1971) involves the specification of access control rules to govern the access of users to information, whereas the mandatory access control model is mostly concerned with controlling information flow between the objects of a system, as illustrated by the Bell-LaPadula model (Bell & LaPadula 1973). For commercial environments, the Chinese Wall model (Brewer & Nash 1989) addresses the conflict of interests that commonly occur between commercial organisations, and prevents the breach of confidentiality by insider knowledge through considering the history of accesses that have been granted. The role-based access control (Ferrariolo & Kuhn 1992) was devised to address shortcomings of the DAC and MAC models in commercial environments. RBAC is policy-neutral and can be used to implement an access control policy, rather than express a particular access control model (Sandu 1996) (Eloff & Von Solms 1998).

4.3.2 Access control mechanisms

Access control mechanisms are the low-level hardware and software functions that implement the controls imposed by the policy and formally stated in the model. The access matrix model by Lampson (1971) is very often used as a mechanism for reasoning about permitted accesses. The state of the system is defined by a triple (s, o, a) , where s is the set of subjects, o is the set of objects and a is the set of access rights. Either an *access control list* or a *capability list* can be used to implement an access control matrix (Gollmann 1999). An *access control list* stores the subjects that have access rights to an object with the object. Subjects can be organised into groups and roles to make the assignment of access rights more manageable. *Capability lists* store a list of the objects that the subject can access with the subject. It identifies the access rights of each subject. Mechanisms that set access rights per object are usually more prevalent. Therefore, most operating systems protect files by means of access control lists. In distributed systems, however, a combination of both approaches can be used (Gong 1989; Abadi et al. 1999).

4.3.3 Access control information

Access control information is firstly presented at a high level, in an access control policy, to describe the proposed operation of the access control system. Subjects, objects, actions, protection requirements and other related characteristics of access control are identified. Information is next reviewed and analysed to identify the information required by access control mechanisms in order that they can operate (Eloff & Von Solms 1998). Different mechanisms require different information, and this may vary with different platforms and applications. For instance, the access control list (ACL) is defined over subjects, objects and actions. The information required to operate the ACL for the Unix operating system may be very different from the information required for the ACL defined for a database application. The next paragraph describes access control models.

4.4 ACCESS CONTROL MODELS

There is a range of existing access control models and associated mechanisms that can be used by the web services access control service. An access control model for web services needs to be investigated, as the research community has not yet formally established one. The next paragraphs discuss DAC, MAC, RBAC, the Chinese Wall model and credential-based access control. Each model is discussed first, after which its mechanisms and access control information are listed and their relevancy for web services access control is highlighted.

4.4.1 Discretionary access control (DAC)

The discretionary access control model (Lampson 1971) is well established in the research community (Castano et al. 1995; Joshi et al. 2001; De Capitani Di Vimercati & Samarati 2000). It is also commercially used in various types of applications such as operating systems, web servers and relational databases. DAC assumes that the owner of an object should be trusted to manage its security. It therefore permits the granting and revoking of permissions at the discretion of the owner of the object. A limitation of this model is the substantial administration overheads involved in assigning permissions to all individual subjects. By combining subjects into groups, however, the assignment of permissions is made more manageable.

Mechanism: Discretionary access control is implemented with the ACL (access control list) (Gollmann 1999).

Information: ACLs make use of objects, subjects, actions and groups (De Capitani Di Vimercati & Samarati 2000). Rules of the format (object, subject, action) are used to describe which objects may be accessed by which subjects through which actions.

Application to web services: It may be argued that web services have similar protection requirements as web pages. The content of both is dynamic, as it may depend on input parameters. Web pages, exposed by Internet applications, are protected with DAC by means of access control lists (ACLs). Examples of applications that make use of DAC are WebDAV (The Internet Society 2004), LDAPv3 (Blakeley et al. 2001) and Web servers (The Apache Software Foundation 2005). Access control differences between web pages and web services can be summarised as follows:

- Machines acting on behalf of humans or organisations access web services, whereas humans access web pages.
- Web services directly expose applications, whereas web pages are a presentation layer in front of applications.
- Direct access from a human for sensitive information may raise suspicion, but access to an improperly protected SOAP interface can easily go undetected.

From the above discussion, it follows that web services not only require stricter, but also more flexible access control. For DAC, access control rules can only be specified as rules of the form (s, o, a). Such rules address the requirement of flexibility and efficient administration as individuals or groups can be assigned permissions, though not to a satisfactory level for web services environments. Recent work has highlighted the limitations of access control rules of this form (Jajodia et al. 1997; Lischka & Wedde 2003; Kudo et al. 2001; Parisi-Presicce et al. 2004), as requirements presented by complex systems cannot be addressed.

Even though extensive platform support exists for DAC, it is not an optimal model to use. As owners control their objects, a difficulty lies in ensuring that all assigned permissions are valid. DAC controls are thus less strict and provide limited protection because the flow of information cannot be controlled. Illegal copies of information can be made, which can be used unlawfully.

4.4.2 Mandatory access control (MAC)

Mandatory access control (Bell & LaPadula 1973; Biba 1975; Denning 1976) ensures that secrets are kept, as illegal copies of information cannot be made. This is achieved by controlling the flow of information and ensuring that it occurs in one direction only. A mandatory policy for access control applies to information that requires multilevel protection in environments such as a military environment. The security levels are organised in a hierarchy such as top secret, secret, confidential and unclassified, which reflects organisational needs. To control indirect access by processes, users are considered as human, whereas subjects are considered as processes acting on behalf of users.

The seminal attempt to formalise the multi-level security policy was the Bell-LaPadula model (Bell & LaPadula 1973). According to this model, security levels are defined for every subject and

object. The security level for subjects represents its classification, and for objects its category. A security lattice (L, \leq) consists of a set of levels L and a partial ordering \leq so that a subject is associated with a maximum security level and current security level (Gollmann 1999).

Administrators evaluate the trustworthiness of subjects and the sensitivity of objects. An access request is permitted if the trustworthiness level of the subject dominates the sensitivity level of the object to be accessed. This is achieved by a partial-order relation *dominates* (Castano et al. 1995), which is defined between a pair of security levels, x and y , in such way that:

$$\forall x, y \in \text{levels}, \quad x \text{ dominates } y \Leftrightarrow \text{classification}(x) \geq \text{classification}(y) \wedge \text{categories}(x) \supseteq \text{categories}(y)$$

The Bell-LaPadula model imposes two restrictions on all reads and writes of objects:

- A subject is not allowed to *read* any object that is at a higher security level. Therefore, no reads are allowed to a higher level.
- A subject may *write* to objects with the same or higher security level than for which she or he has clearance.

Mechanism: A security lattice supports MAC (Gollmann 1999).

Information: A security lattice is defined over security levels and a set of categories, so that objects are protected from the read and write actions made by subjects.

Application to web services: Web services do not only expose application interfaces, but they are also accessed by machines acting on behalf of their subjects. The operation performed by machines on web services is *execute*, which has the same restrictions as the *read* operation according to the Bell-LaPadula model. As machines are viewed to be not as trustworthy as humans to obey access control rules (De Capitani Di Vimercati & Samarati 2000), mandatory access control becomes important to consider for web services. MAC enables stricter access control and is therefore able to better protect resources from machine requests than DAC. An important instrument in MAC is the specification of a level that reflects the trustworthiness of a requesting machine. Levels, evaluated in the context of a lattice, give an indication of the trust held towards machines. This is done before access is given for sensitive operations and other resources to subjects on whose behalf a machine is acting. Levels provide flexible access control and address the access control policy requirement for levels of trust directly.

Mandatory levels may be difficult to use for web services, as MAC is a policy that is associated with a high administrative burden. System administrators have to set policies by assigning security levels to all requestors and objects. In order to manage MAC for web services, several trusted administrators can be used for different tasks. Another solution is to automatically assign trust labels to requesting machines by a web services trust engine, so that the administration

burden is lessened. This research aims to address automated trust level assignments in order to provide for the making of autonomous access control decisions.

4.4.3 Role-based access control (RBAC)

RBAC (Ferrariolo & Kuhn 1992) can be considered an access control mechanism that has been designed to provide appropriate access control for commercial environments by reducing the complexity and cost of security administration. RBAC removes the requirement to repeatedly grant, deny and revoke permissions when a user moves to another position in a company. A fundamental difference between RBAC and DAC is that users cannot pass access permissions on to others at their discretion.

RBAC regulates the access of users to information on the basis of the activities users are to perform in the system. Roles are first identified for an organisation. A role can be defined as a set of actions and responsibilities associated with a particular job function such as “manager”. The roles that a user activates are typically not determined at the user's discretion, but rather by her or his assigned tasks, in compliance with the organisational protection guidelines or access control policies that are usually derived from laws, regulations or operating practices. Access permissions are assigned to roles rather than to individual users. In RBAC, relations are used to connect users (U), roles (R), and permissions (P) to one another (Coyne et al. 1996).

- The user-assignment (UA) relation, $UA \subseteq U \times R$, which is a many-to-many mapping from users to roles.
- The permission-assignment (PA) relation, $PA \subseteq P \times R$, which is a many-to-many mapping between permissions and roles.

When a user logs in to the system, she or he establishes a session. The concept of a session is a one-to-many mapping from users to roles. The user activates her or his assigned role(s) for the duration of a session. A session is thus an active subject. A user can exercise several roles at the same time or can be forced to assume only one role.

Several RBAC models have been proposed over the past few years. Coyne et al. (1996) specified four conceptual RBAC models, each model supporting different features that can be implemented. The basic model, $RBAC_0$ contains users, roles, permissions and sessions. $RBAC_1$ includes $RBAC_0$ with role *hierarchies* (Sandhu 1998). Hierarchies structure roles to reflect the lines of authority and responsibility in an organisation. $RBAC_2$ also includes $RBAC_0$, with *constraints* to restrict the assignment of users or permissions to roles, or the activation of roles in sessions. Constraints are used to specify application-dependent conditions and satisfy principles of least-privilege and separation of duties. Finally, $RBAC_3$ combines both $RBAC_1$ and $RBAC_2$, and provides *both* role hierarchies and constraints. Conceptually, roles are similar to security levels

(Eloff & Von Solms 1998), but are designed for commercial environments. Roles thus provide a good balance between DAC administration and MAC protection.

Mechanisms: RBAC is implemented by users-to-roles, and roles-to-permissions assignments.

Information: RBAC is defined with subjects, roles, and permissions that are set with regard to objects and actions.

Application to web services: As the RBAC model works well in dynamic environments (Joshi et al. 2001), it would be important to employ this mechanism in the web services access control policy. Roles have the ability to address cross-domain access control, since roles can be activated across independent domains by attributes found in credentials (Herzberg et al. 2000; Humenn & Kuo 2002). Such role activation can address requirements of autonomy, and loose coupling. Requirements of flexibility, efficient administration of subjects and attribute-based access control can be addressed similarly. For web services, a distinction must be made between a trust level and a role. A trust level identifies the trustworthiness of a machine, whereas a role identifies a type of subject. Important considerations involve the way in which role activation is done, and how roles are used in conjunction with requestor trust levels. A web services provider could maintain a subject and role database, but this will impede spontaneous SOAP exchanges. A more flexible solution is to activate roles through logical rules that evaluate the ability of subjects and the trust level of web services requestors.

4.4.4 Chinese Wall access control model

Over and above DAC, MAC and RBAC, there are a number of other access control models with specific mechanisms that can be used to provide better protection for web services. The specific application domain would determine choices. For instance, Brewer and Nash (1989) introduced the Chinese Wall access control model to specify confidentiality mechanisms for commercial environments in order to avoid conflicts of interest.

The Chinese Wall access control model recognises the importance of access history in protecting security. It can be seen as a special kind of dynamic separation of duty (Jajodia et al. 1997). In terms of the Chinese Wall access control model, objects are grouped into organisation datasets. Organisation datasets that are in competition are grouped together in conflict of interest classes. If a user accesses an object in an organisation dataset, she or he cannot be allowed to access any other object in an organisation dataset that appears to be in a conflict of interest class with it.

Mechanism: The Chinese Wall model is implemented by organising objects into datasets that are used in conjunction with conflict of interest classes (COIC) and a history of granted accesses. History information is kept in a two-dimensional matrix X, with a column for each object and a row

for each subject. An element $X_{s,o}$ is true if and only if subject s has previously accessed object o (Brewer & Nash 1989).

Information: Objects in datasets, subjects in conflict of interest classes, and a record of the history of accesses constitute information that is needed by the mechanism to operate.

Application to web services: Aspects of this model can be incorporated into the web services access control policy as required. If there are web services requestors who have a mutual problem of conflict of interest, datasets and conflict of interest classes can be created to ensure that they do not access information on a conflicting party. For example, eBooks is a web services provider that deals with many requestors who may be in conflict with one another, such as suppliers of books and other goods. If summarised retail information is made available to such requestors, it must not be to the detriment of other requestors who are in conflict with one another. These mechanisms can support flexible access control and conflict resolution policies.

4.4.5 Credential-based access control

Access control models discussed so far are designed to protect systems where identified subjects managed by an administrator are granted access to protected resources. When interactions occur between remotely located web services requestors and providers, authentication of subjects is no longer a viable option. Capability-based systems that do not rely on the identity of subjects but rather on capabilities have provided the foundation for credential-based access control. Recent developments (Bacon & Moody 2002; Biskup & Wortmann 2004) implement systems that make access control decisions about the properties of requestors found in credentials.

A credential is a digitally signed assertion by a credential issuer about the properties of one or more entities (Bina et al. 1994; Ching et al. 1997; Biskup & Wortmann 2004). A property might be an identity, or any non-identifying attribute such as a job title or a granted capability for a web services operation, expressed as an attribute. Credentials may also be anonymous of nature (Camenisch & Herreweghen 2002). Access control decisions are subsequently based on appropriate interpretations of attributes that are extracted from submitted credentials. Digitally signed credentials are verifiable and unforgeable.

Mechanism: A credential that contains one or more signed attributes, used in conjunction with access control rules that may be of the form (*attribute_list*, object, action).

Information: Credential-based access control requires signed attributes, objects and actions to operate.

Application to Web Service: For web services, digital credentials can be public key certificates (Housley et al. 1999), attribute certificates (ANSI 1999) (Johnston et al. 1998), and signed or unsigned assertions (Bonatti & Samarati 2002). Credentials are conveyed from the web services

requestor to the provider in an XML format in the SOAP header that accompanies the request. Web services requestors and providers should independently be able to issue and trust credentials. When credentials are presented with requests, web services providers should be able to decide whether access to web services operations should be granted or not. These decisions are based on the web services provider's interpretation of a requestor's attributes provided by XML credentials that accompany a SOAP request. Before web services can accept a credential, it is important to evaluate the trust that exists with the credential issuer, so as to ensure the validity of attributes contained by the credential. A consideration for web services is that requestors should be able to make requests without registering themselves with web services providers, if they present credentials that can be trusted.

Credentials can be used to implement a number of requirements from both the environmental and internal access control requirements. It provides for autonomy, loose coupling and attribute-based access control as web services requestors present their requests together with signed attributes of subjects in SOAP headers. Trust between web services providers and their requestors is incrementally established as credentials are presented.

4.5 WEB SERVICES ACCESS CONTROL SERVICE

Next, the discussion on access control requirements, mechanisms and information is summarised in order to define the web services access control service in greater detail. Figure 4.6 shows which environmental and internal access control requirements are addressed by current access control mechanisms as discussed. It also associates access control information with specific mechanisms. Requirements that are not addressed by current access control mechanisms can clearly be identified from this figure.

In order to allow the reader to interpret Figure 4.6 more easily, the related dimensions are summarised in a list in each case.

4.5.1 Dimensions of the web services access control service

4.5.1.1 Web services access control requirements

A Environmental access control requirements

- 1 Autonomy
- 2 Loosely coupled
- 3 Quality of service
- 4 Policy-based compatibility
- 5 Policy negotiation
- 6 Standards-based interaction

B Internal access control requirements

- 7 Flexibility
- 8 Efficient administration
- 9 Attribute-based access control
- 10 Trust levels
- 11 Exceptions
- 12 Conflict resolution

4.5.1.2 Access control mechanisms

- **ACL:** An access control list stores the subjects that have access rights to an object. Subjects can be organised into groups to make the assignment of access rights more manageable.
- **Lattice:** A security lattice (L, \leq) consists of a set of levels L and a partial ordering \leq so that a subject is associated with a maximum level and current level.
- **Role:** A role is a set of permissions associated with a particular job function.
- **COIC:** A conflict of interest class is a grouping of datasets from competing companies.
- **Credential:** A credential is a digitally signed assertion that contains attributes of an entity.

4.5.2.3 Access control information

- **Subject:** A subject is the entity that is granted access to an object.
- **Object:** An object is a resource that is the focus of protection.
- **Action:** An action is the type of access granted such as *read*, *write*, *execute*.
- **Level:** A level is a classification such as *secret* or *confidential*.
- **Role:** A role is a named organisational function such as *manager*.
- **History:** A history of accesses is a two-dimensional matrix with a column for each object and a row for each subject that shows if a subject has previously accessed an object.
- **Attributes:** An attribute is a name-value pair such as "age=30".

Each of these dimensions is shown as a side or plane of Figure 4.6, and is indicated by the legend in the figure. Three sides of Figure 4.6 are as follows:

- The green lines on the front side of the cube show the access control requirements (A + B).
- Access control mechanisms are displayed by the blue line on top of the diagram.
- The access control information used by a mechanism is shown on the side of the cube by means of the purple line.

A green block on the front plane shows that the specific access control requirement is related to the access control mechanism concerned. A blue block on the top plane of the diagram shows that the specific access control mechanism is related to the access control information displayed.

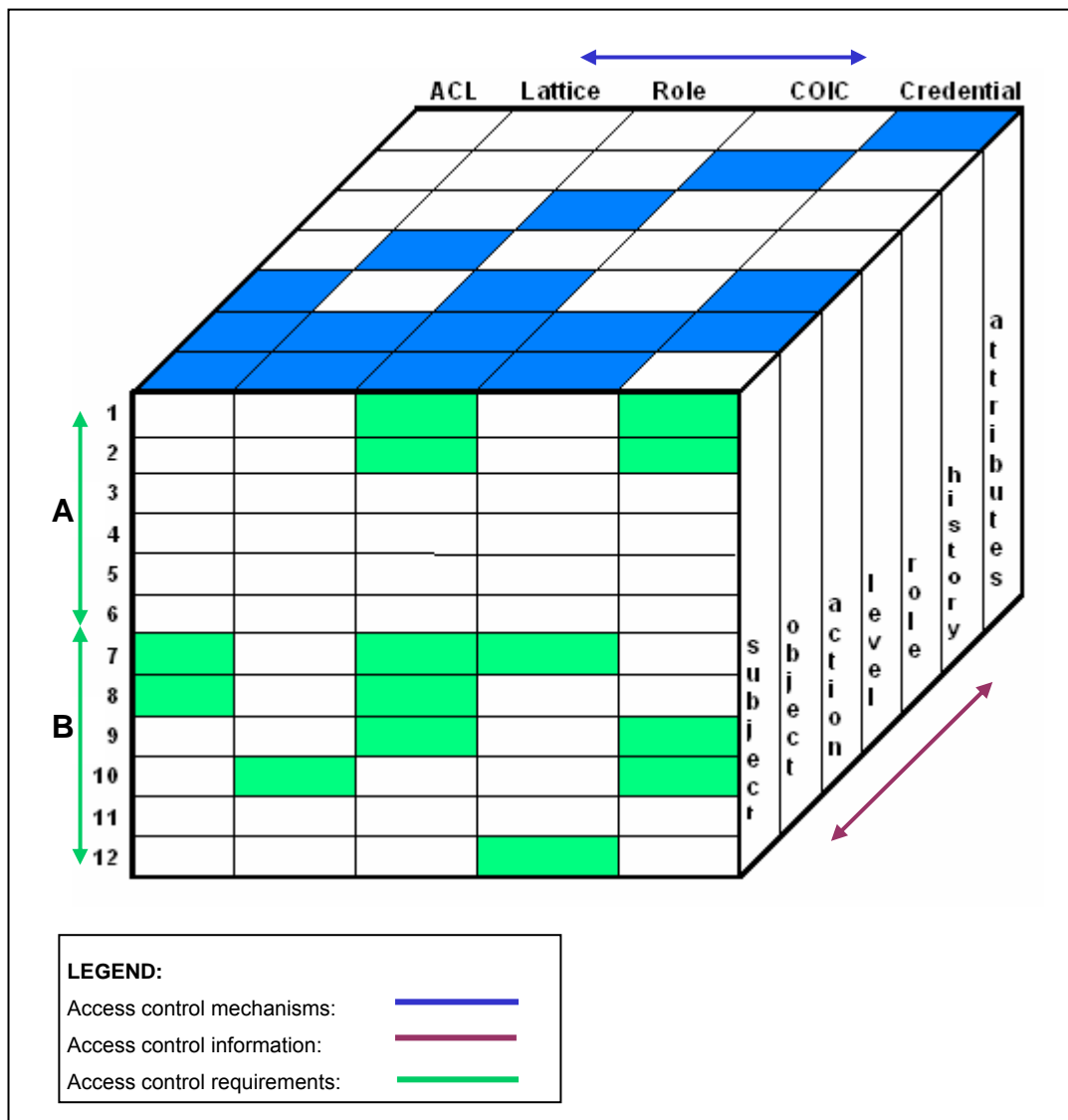


Figure 4.6: Web services access control service dimensions

To interpret Figure 4.6, consider each access control requirement in turn from top to bottom, to determine the extent to which current mechanisms are useful. As an example, the block marked with an **X** is now discussed. To interpret the meaning of this block, look at access control requirement 9 from group B, namely *attribute-based access control*. From the list of access control mechanisms, a mechanism that addresses this requirement is *credentials*, as was indicated earlier. The block has been marked in green to illustrate the relationship between the access control requirement and the access control mechanism. Furthermore, the three blue blocks of the credential access control mechanism (see the top right-hand side of the diagram) show that *credentials* is related to *objects*, *actions* and *attributes* (access control information). The requirement for attribute-based access control is also associated with *roles*, as is shown by the other green block for requirement 9. Its related access control information involves *subjects*, *objects*, *actions* and *roles*.

Each of the blocks marked in green similarly indicates that there are indeed relationships between the three aspects as discussed previously in this chapter.

4.5.2 Web services access control requirement analysis

Figure 4.6 is next analysed in order to determine access control concerns for the web services access control service. The environmental and internal access control requirements are therefore considered now.

4.5.2.1 Environmental access control requirements

Environmental access control requirements, indicated by section A in Figure 4.6, are addressed by mechanisms to a minimal extent only, as is revealed by the small number of green blocks marked for requirements 1 to 6. To address these requirements, new mechanisms are therefore needed by the web services access control service. Such mechanisms must be able to address requirements of autonomy, loosely coupled, quality of service, policy-based compatibility, policy negotiation and standards-based communication. A web services provider should be able to communicate access control information to web services requestors so that they would share a common understanding of access control requirements, required credentials and other information. Information should be exposed in a safe manner through a process of negotiation and agreement, so as not to compromise the resources of a web services provider. This fact provides the motivation for a new policy for the web services access control service, namely an *interface policy*, as is shown in Figure 4.7.

4.5.2.2 Internal access control requirements

Internal access control requirements 7 to 12, indicated by section B in Figure 4.6, are addressed by mechanisms found across a variety of access control models. Mechanisms to take note of are roles and credentials. To address internal access control requirements, new or adapted mechanisms used in conjunction with existing mechanisms are required to fully comply with all the requirements of the web services access control service. This motivates a seventh internal access control requirement for the web services access control service, namely that the access control service should be implemented by composite access control mechanisms. This can be achieved if the *access control policy* is of a virtual nature. Such a policy is not bound to any specific access control model or its associated mechanisms, and is machine-readable. The access control policy consists of declarative statements that are evaluated to determine access rights for subjects on objects, and is specified in a high-level policy language.

The focus of this research is next determined. Requirements 7, 8, 11, 12 have been greyed out in Figure 4.7. Although these requirements are important and should be included in a comprehensive web services access control service, they are not addressed in this research.

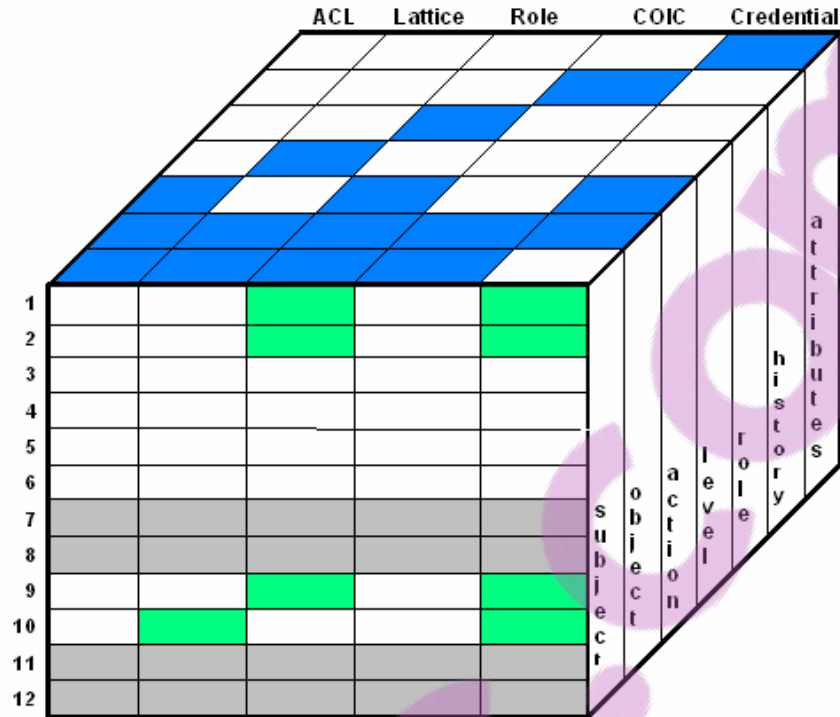


Figure 4.7: Web services access control service requirements not to be addressed

The current research thus focuses on the more specific web services access control requirements of autonomy, loose coupling, quality of service, policy-based compatibility, policy negotiation, standards-based interaction, attribute-based access control and trust levels.

Figure 4.8 summarises how the access control and interface policies have been derived from the environmental and internal access control requirements, after having been analysed by means of access control mechanisms.

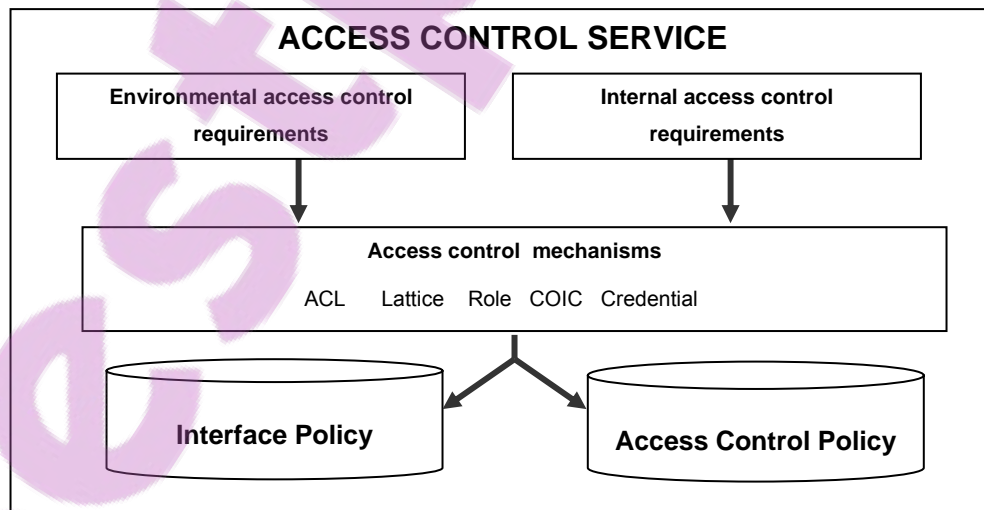


Figure 4.8 Access control service policies

4.6 CONCLUSION

This chapter highlighted the access control requirements to be addressed by the web services access control service. The chapter commenced by identifying the internal access control requirements of a web services provider. These requirements were added to the external access control requirements to give a total picture of twelve access control requirements.

The chapter proceeded with a background discussion on access control. Access control models, mechanisms and information were discussed with the aim of identifying possible models and mechanisms that could play a role in the protection of web service resources. The discussion was summarised in a figure (4.6) that clearly showed the discrepancy between current and desired access control mechanisms. From an analysis of this figure it was derived that access control for web services must be addressed at two policy levels: the *interface policy*, which is associated with the interface of the web service, and the *access control policy*, which is internal and kept private in order not to allow the web services to be compromised.

Before an access control service is designed, some outstanding issues are to be explored. As mechanism-independent, machine-readable policies are required, access control policy specification needs to be explored. The proliferation of policies demands an architecture in which proper decisions can be made. These issues will be discussed in Chapter 7 and Chapter 8 respectively.

Trust has been mentioned frequently in this chapter. As in the real world, different levels or degrees of trust are required to be assigned to requesting machines. Levels of trust can be established if web services are knowledgeable about the various properties that a machine possesses.

An investigation is conducted in Chapter 5 to determine requirements for web services trust.

5

Web Services Trust

Virtual applications, implemented with web services technology, have unique challenges that need to be overcome when access control is considered. The complexity of virtual applications, characterised by collaboration between unknown web services requestors and providers, is compounded by a lack of a central control authority. The lack of central control localises the responsibility for access control and other decisions at participating machines that support web services. This requires machines to be autonomous when making access control decisions. The inflexible nature of current web services access control solutions have led to new research, which highlights the management of trust relationships as a possible way to resolve these issues. The act of giving access to sensitive resources can in fact be considered as a refinement of a trust relationship. While trust is a well-established concept in information security, the current static nature of trust relationships predefined by an administrator is insufficient for the advancement of ad hoc collaboration over machine-to-machine interactions. In this regard, the concept of autonomous trust enables web services requestors and providers to reason about relevant information and evidence before making access control decisions.

The aim of this chapter is to investigate trust, so as to promote the inclusion of trust in the web services access control model. In order to achieve this, the current chapter commences with a background to the management of trust. A discussion on social forms of trust identifies a cognitive approach to trust, defined on the basis of information and reasoning, as a possible venue to pursue. Trust between web services is discussed in order to determine its alignment with human forms of trust. A conclusion rounds off the chapter.

5.1 THE MANAGEMENT OF TRUST

Trust between web services requestors and providers can currently be managed by different approaches such as trust management systems (Blaze et al. 1999), Liberty Alliance trust models (Boeyen et al. 2003), the WS-Trust specification (Della-Libera et al. 2003), trust negotiation systems (Winslett 2002), and computational trust as defined by the SECURE project (Bacon et al. 2004). These types of approaches are considered in the paragraphs that follow and then critically summarised. The characteristics of these approaches for machine-to-machine web services trust are listed next, and the discussion is concluded by reflecting on trust management as it would apply to web services providers.

5.1.1 Trust management systems

The term trust management has been commonly used since 1996 when it was first introduced by Blaze and others (1996; 1999a; 1999b). Trust management was then defined as “*a unified approach to specifying and interpreting security policies, credentials and relationships that allows direct authorisation of security-critical actions*”.

Trust management makes use of mechanisms such as identities, certificates, signatures and keys to establish trust relationships across domains. Even though the term *trust* is used in the title of these types of systems, it actually has a different meaning. Trust management does not refer to the problem of managing trust, but to the problem of managing access control performed with public keys (Grandison 2003).

A trust management system consists of three basic components (Blaze et al. 1996; 1999a):

- A language for expressing access control policies
- A language for specifying credentials
- A trust management engine that determines whether a request should be granted, given the local access control policies and set of credentials

When presented with a credential, the question that the trust management system attempts to answer is: “*Is a request R compliant with the local access control policies P given the set of credentials C?*” When trying to answer this question, a trust management system does not need to know the identities of parties accessing resources, as long as it knows that they are trusted to do so. This is achieved by the verification of credentials, which enables decentralised policy management through the delegation of authority (Yoa 2003). Resource owners delegate the rights of accessing its resource to parties by means of a digital credential. Those parties may in turn delegate this right to others, and the process may occur repeatedly. The last party may then present the set of credentials to the resource provider, where the trust engine will attempt to find

a chain of delegations from the set to make access control decisions. Grandison (2003) gives a very comprehensive overview of trust management systems. Some examples of trust management systems are SPKI (Ellison et al. 1999a, 1999b), SDSI (Rivest & Lampson 1996), KeyNote (Blaze et al. 1999), SD3 (Jim 2001) and Fidelis (Yoa 2003).

5.1.2 The Liberty Alliance trust model

The mission of the Liberty Alliance Project is to establish an open standard for federated network identity through open technical specifications. Liberty Alliance specifications (Liberty Alliance Project Specifications 2005) from the Liberty Alliance Project (Liberty Alliance Project 2005) define circles of trust, where organisations and parties in the inner circle are more trusted than those in outer circles. A circle of trust therefore recognises that different parties are trusted to different extents. Trust can be established between web services in accordance to a Liberty Alliance trust model (Boeyen et al. 2003). This can be achieved by using SAML assertions to move identities of users across domains.

The approach is defined over identities of web services requestors and providers. The social infrastructure that revolves around the issuing of certificates and the information recorded by certificate authorities make it possible to trust some keys and certificates more than others. Information to determine differences can be sourced by inspecting policies of certificate authorities. These policies specify how the identity of the party who was granted the certificate was validated, and the purposes for which a certificate can safely be used. The Global Grid Forum (2003) implements this approach, which is referred to as *qualified installation of keys*. A key is only installed if information ensures its trustworthiness.

Different forms of trust are enabled. For instance, *brokered* or *indirect* trust (Boeyen et al. 2003) involves two parties that have no pre-existing relationship or agreement, but that depend upon agreements they have with a third party. In the case of *community* trust (Boeyen et al. 2003) a party enters into a negotiation with an unknown party, but one that is part of a general class of businesses such as book stores. In this case, trust depends on the party who admitted the bookstore into the community.

5.1.3 WS-Trust

WS-Trust (Della-Libera et al. 2003) is part of the WS-security specification and was introduced in Chapter 2 on page 24. It aims to address trust between web services requestors and providers who may be unfamiliar with each other. The goal of WS-Trust is to enable web services requestors and providers to construct trusted SOAP message exchanges. A web services provider may require from an incoming message to prove a set of claims such as a name, key,

permission or capability. Such requirements are indicated in its policy as described by WS-Policy and WS-PolicyAttachment specifications. WS-Trust provides a simple request/response protocol for issuing, exchanging and validating security tokens so that it is possible to represent trust by brokered security tokens.

To establish trust by means of this approach, a web services provider's trust engine verifies that the claims in a security token are sufficient to comply with the policy, and that the message conforms to the policy. It verifies that signatures prove the attributes of the claimant, and finally that the issuers of tokens are trusted to issue the claims they have made.

5.1.4 Trust negotiation

Trust negotiation (Winslett 2002) (Bertino et al. 2004a) is an approach towards access control whereby access to resources is granted based on trust that is established between a web services requestor and provider. The term Automated Trust Negotiation (ATN) is also used to refer to this very recent development in distributed access control research. In trust negotiation, credentials that describe the ability of their owner are exchanged iteratively to build trust between negotiating web services requestors and providers. A move is made away from traditional access control rules defined by (object, subject, action) tuples towards the specification of access restrictions based on subject attributes.

The idea of using automated negotiation to establish trust is not new. A commonly known protocol, SSL, is an example of trust negotiation over the Internet. In SSL, a web services provider first discloses its credentials to a web services requestor in an attempt to establish trust. The web services requestor optionally submits its credentials to establish mutual trust with the web services provider. The established trust is limited in nature, as it is based only on the identity of web services requestors and providers, for the duration of a specific session.

In trust negotiation frameworks, a trust negotiation protocol defines the ordering of messages and the type of information that messages will contain. In the case of a trust negotiation strategy, however, the exact content of the messages is controlled, such as which credentials to disclose, when to disclose them, and when to terminate a negotiation. Examples are Trustbuilder (Child et al. 2002), (Jones et al. 2000), (Seamons et al. 2001) the Service Access and Information Release Framework (Bonatti & Samarati 2002), and X-Trust (Bertino et al. 2004).

5.1.5 Computational trust

The previously discussed approaches combine access control with trust, where trust is defined statically in an access control policy. For instance, the Automated Trust Negotiation approach

makes use of an explicit negotiation process, defined in a policy, to establish trust between web services requestors and providers. The SECURE project (SECURE 2003) (Bacon et al 2004) takes a different approach, and instead employs computational trust. A trust value is calculated, which can subsequently be used in an access control policy. Trust in a principal is computed by inspecting evidence relevant to the given context. Evidence consists of observations of previous interactions with a principal and recommendations from other principals.

For every decision that is made, the SECURE framework considers the trust it has in the requesting principal, and the risk of granting the request. Risk is seen as the combination of the costs and likelihoods of all the possible outcomes. Policies can be defined that differentiate between low-level trust and cost information on a per outcome basis, so that the level of uncertainty to be included can be controlled.

5.1.6 Summary of approaches used to management trust

Trust can be used in access control policies to grant access to web services requestors, based on a request and accompanying credentials contained in a message. The role of trust in access control decisions can be summarised at a high level as follows:

Table 5.1: High-level description of approaches to trust and access control

Trust Management	<i>I grant you access, because you have been delegated the right to access the resource by a public key that I trust, and you may have proven your ability by presenting verified and valid attributes.</i>
Liberty Alliance	<i>I grant you access, because I know who you are, I know the identity of the organisation from where your request originates, and the request is signed by a public key that I trust to a certain extent.</i>
WS-Trust	<i>I grant you access, because you have been delegated the right to access the resource by a public key that I trust, you may also have proven your ability by presenting verified and valid attributes, and it is in a format that I can understand.</i>
Trust negotiation	<i>I grant you further and advanced access, because my trust increase as you present more sensitive attributes with each consecutive interaction, signed by a public key that I trust.</i>
Computational trust	<i>I grant you access based on the extent to which I trust you, and I base my decision on the fact that your message may be signed by your public key, others trust you, and I continuously re-evaluate information and evidence in order to predict your future behaviour.</i>

The following characteristics can be listed:

- Required trust relationships are not easy to establish as they are negotiated, complex and time-consuming to implement, and manually configured by administrators.
- Trust is often based only on the verified identity of the other party.
- Trust negotiation assumes trust to be monotonic, that is, further disclosure of sensitive credentials does not negatively influence already formed trust.
- Trust is supported by a certification infrastructure that may not be widely accepted, since required certificate authorities can be either absent or inaccessible for some prospective web services requestors.
- Revocation of certificates and key distribution compound the problem of creating trust relationships.
- Trust very often fluctuates between no trust, complete trust or distrust, which is insufficient to use in an environment where different trust levels are required.
- For each interaction trust is formed by applications that check conditions in policies and verify certificates. Other than in the computational approach to trust, no history of previous interactions is maintained to allow better decisions in the future.

To address the limitations of creating trust through the verification and validation of public keys, Jøsang and Tran (2000) later extended the definition of trust management to read as follows: *“Trust management is the activity of collecting, codifying, analysing and presenting security relevant evidence with the purpose of making assessments and decisions regarding eCommerce transactions.”* By this definition, trust management and evolution move closer to the real world, as they are based on evidence that has been collected and evaluated.

Dealing with issues surrounding the encoding, analysis and presentation of evidence, Grandison (2003) next defined trust management as follows: *“The activity of collecting, encoding, analysing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships for Internet applications.”* Evidence could include credentials such as certificates for proof of identity or qualifications, risk assessments, usage experience or recommendations. A drawback of this work is that initial trust values are not computed, but are assigned by an administrator after information and evidence have been processed.

For web services trust, the following characteristics need to be addressed:

- Past experience and recommendations need to be evaluated for the purposes of trust computation.
- Uncertain and imprecise information needs to be included when trust is computed.

Next, trust management for web services is defined in more detail.

5.1.7 Trust management for web services

The trust between machines that support web services requestors and providers will provide a basis for all exchanges that take place between them. Web services requestors and providers that have forged strong relationships of trust with others over time have levels of goodwill towards each other that enable the sharing of more information, and the granting of further and advanced access. In contrast, ad hoc web services requestors may introduce themselves for the purpose of a once-off transaction. It would be unrealistic to expect from web services providers to treat well-known and unacquainted web services requestors in the same manner.

Because of the limited nature of trust, collaborating web services providers and requestors are faced with dilemmas such as the following:

- Does a web services provider deny access to all unacquainted web services requestors who are not trusted?
- What level of access does a web services provider give to web services requestors of whom nothing more than their identity is known?
- How does a web services provider initially distinguish between prospective web services requestors that may present a profitable opportunity and those that may become a liability?

Virtual applications supported by web services will only become a reality if autonomous machines maintain trust relationships with one another for the purpose of making decisions according to individual constraints and intentions. This will allow participants to feel in control, even though they may constitute only a small part of a virtual society. To address these questions, this research aims to extend the concept of trust management to address the static manner in which trust is assigned to others. The following **preliminary definition for trust management** is therefore proposed:

Definition (preliminary) – Trust management: *The automated collection, analysis and categorisation of information and evidence, with the purpose of determining a trust level that can be used for access control and other decisions.*

The preliminary definition will be extended to include not only the automated collection of information and evidence, but also to establish how information is analysed and categorised for the purposes of a trust calculation that will determine the level of trust for web services requestors.

The next paragraph provides a background to web services trust by discussing trust as it manifests between humans on the one hand, and between organisations on the other.

5.2 TRUST

To fully comprehend the process of trust formation and evolution, it is important to analyse trust models from other areas of research to understand underlying concepts. For this purpose, the next paragraph discusses social forms of trust for humans and organisations. The last paragraph then motivates the approach towards trust for web services, which includes elements of both human and organisational trust.

5.2.1 Trust for humans

Trust is an important aspect of human life and constitutes the basis for most decisions. It is mostly used unconsciously, with no assistance from third parties. To understand the fibre of trust between humans, the next sub-paragraphs describe trust, and properties and dimensions of trust. From this discussion, beliefs are identified as a basis for cognitive trust.

5.2.1.1 Trust

Trust is considered a relation consisting of three elements:

A trusts B about X.

Trust is therefore a directional relationship between A, called the *trustor*, and B, called the *trustee*, for a specific context X. The trustor, A, is a “thinking entity” in some form, whereas the trustee, B, can be anything from a person or physical entity, to abstract notions such as software or a cryptographic key (Jøsang 1996).

For instance, Figure 5.1 shows that Alice trusts Bob to drive the car. Another example of a trust relationship could be that Sue trusts the SSL protocol to encrypt a message.

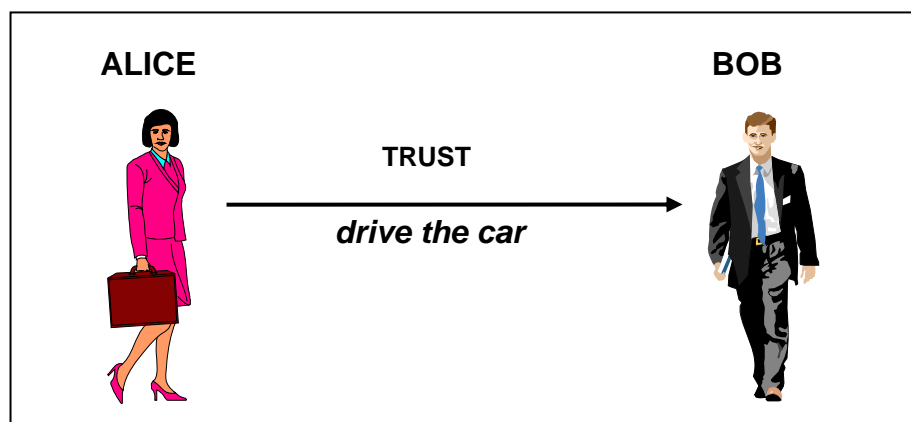


Figure 5.1: The trust relationship

5.2.1.2 Properties of trust

Several significant research contributions have been made in the field of trust, each revealing a very divergent point of view. According to Marsh (1994) the majority of work is from three different areas, namely sociology, psychology and philosophy. He identifies the following persons who have done significant work in this regard: Deutsch (1962), Luhmann (1979), Barber (1983) and Gambetta (1988). From these researchers' work, the following properties of trust are identified:

- Trust exists at both the individual and social level.
- Trust is a subjective notion that is not transitive.
- Trust allows the reduction of complexity when decisions are made.
- Trust is measurable and evolves over time.
- Trust is dependent on a specific situation where risk is accepted when interactions occur.

For this research, it would be important to determine whether similar trust properties are applicable to web services environments.

5.2.1.3 Dimensions of trust

Trust is seen to display both cognitive and emotional dimensions. The cognitive dimension focuses on the rational basis for trust, as it is a process of acquiring *information* and *reasoning*. In this process, the acceptance of information as truth leads to the formation of beliefs. Beliefs in others affect the cognitive willingness to depend on them. This dimension of trust is more prominent when participants do not know one another, or when participants are far removed from one another. A cognitive model of trust presented in Castelfranchi and Falcone (2003) shows beliefs as a basic component of a cognitive mental state of trust. Trust thus expresses beliefs in another party, where beliefs are based on the lack of contrary evidence (Gambetta, 1988). This statement makes clear the important role of information and evidence in trust formation and evolution.

5.2.1.4 Basis of trust

Cognitive trust is defined on the basis of beliefs. Beliefs as the basis for trust formation have been investigated in literature, and Chervany and McKnight (1996) conducted a very comprehensive study of trust over a wide range of disciplines in the social sciences. They identified six trust concepts, shown in Figure 5.2, which included categories of beliefs that are used in trust formation (Chervany & McKnight 1996). The interactions between these six constructs are described next.

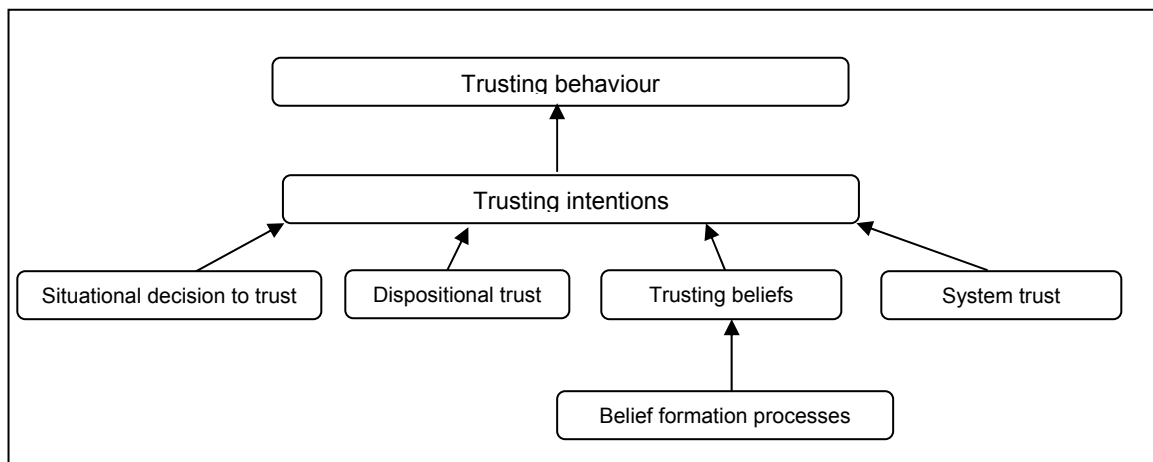


Figure 5.2: Trust formation

The aim of trust formation is for a trustor, the person who trusts, to exhibit trusting behaviour. *Trusting behaviour* is the extent to which the trustor depends on the other party and is manifested by her/his actions. It implies the acceptance of risks by the trustor. For a trustor to exhibit such behaviour, she/he needs to establish her/his trusting intention. *Trusting intention* is the extent to which the trustor is willing to depend on the other party. Trusting intentions are influenced by the situational decision to trust, dispositional trust, and beliefs such as trusting beliefs and system trust.

- *The situational decision to trust* is the trustor's willingness to trust in a given situation.
- *Dispositional trust* describes the general trusting attitude of the one who trusts, the so-called trustor. It is often referred to as "basic trust".
- *Trusting beliefs* indicate the extent to which a person believes that the other is trustworthy in the situation. Beliefs are formed via a *belief formation process*. A person is trusted because of trust in her/his benevolence, honesty, competence and predictability. Chervany & McKnight (1996) describe the following four categories of trust beliefs:
 - *Honesty* - the belief that agreements are made in good faith.
 - *Competence* – the belief that an entity has the ability to perform a task.
 - *Predictability* – the belief that the actions of an entity are consistent so that a forecast can be made about what such an entity will do in a given situation.
 - *Benevolence* - the belief that a party cares about the welfare of the other.
- *System trust* is based on the property of the system within which the trust relation exists and can be considered as a belief that is formed about the environment.

Humans use these trust concepts intuitively to trust, or distrust, as they interact with one another. Assumptions such as "it is very likely that Sue is trustworthy" are made fairly soon in the course of relationships. Although it is not possible for humans to determine Sue's trustworthiness exactly by stating that "the probability that Sue is trustworthy is .8", it may be quite possible for them to

estimate that “the probability that Sue is trustworthy is not .1”. Trust is approximated and clearly of a fuzzy nature.

Humans continually assess one another as they collect information through experiences, observations, and recommendations from others. Beliefs form over time and influence trust concepts. In turn, trust concepts influence one another. For instance, because of previous experiences, the generally trusting disposition of Jill has become very distrustful and has lowered her trust in the system, as well as in others.

The manner in which trust concepts influence one another is determined by human cognition. Axelrod (1972) introduced the concept of cognitive mapping to make explicit the way in which humans reason. In human cognition, causal inference has been shown to play an important role, and for that reason cognitive maps are defined on the basis of causal beliefs. Cognitive mapping can therefore be a useful tool to portray cognitive trust formation over trust concepts, defined on the basis of the beliefs of a human.

If these trust concepts can be formally specified, they can be used in trust computations by using fuzzy techniques. Previous research (Castelfranchi & Falcone 2002) have shown how humanistic forms of trust can intuitively be imitated by fuzzy techniques. For web services, it may be possible to relate these concepts to the environment of the web services requestor and provider, as well as to the behaviour of the web services requestors.

5.2.2 Trust for organisations

Considering the nature of web services, it is important to understand how trust is established between organisations. Market forces, social interactions, legal and assurance systems and insurance have an effect on trust relationships. A model of inter-organisational trust illustrates that trust is established in three stages (Ratnasingam 2001):

- Firstly, *competence trust* is established through the trust and security-based mechanisms that are embedded in e-commerce technologies to provide speed and real-time accurate information.
- Secondly, consistent positive behaviour from trading partners leads to credibility and reliability, which creates *predictability trust*.
- Lastly, *goodwill trust* focuses on organisational reputation and brand names, and is accomplished by enforcing best business practices.

5.2.3 Trust perspective taken by this research

This research maintains that trust is initially formed by “hard” mechanisms such as certificates and algorithms, but as time passes and positive experiences are recorded, “soft” mechanisms

such as human judgement create high levels of trust. The author of this thesis recognised layers of information that can be used for trust formation between web services requestors and providers. These layers and sources of information are depicted in Table 5.2. Initially, competence trust is formed on the basis of identity, implemented security mechanisms and best practice of partners as shown by information layers 1 and 2 at the bottom of the table. References and recommendations, shown in information layers 3 and 4, will further increase competence trust. Next, predictability trust is the result of established experience, as shown in information layer 5. Finally, after time, goodwill trust is formed, as is shown in information layer 6.

Table 5.2: Information layers and sources of trust formation

	Information Layer	Source of information
soft	6. Goodwill	Human judgement
	5. Experience	Volume of transactions, history, behaviour
	4. Recommendations	Situation-specific values
	3. References	Certificates, assurances, licences
	2. Technology	Security mechanisms, best practice
hard	1. Identity	Digital certificate, password, Kerberos ticket

The three stages of trust formation for organisations relate well to the categories of beliefs identified for humans by Chervany & McKnight (1996), as both recognise competence and reliability as components of trust. Information layers and related sources of information subsequently identified by this research are important for web services trust formation, as an investigation may be done to see if they can be gathered automatically. Information sources that can be gathered automatically can be bound to the trust concepts and to categories that have been identified in the previous paragraph, so that they can be used in a trust computation.

Next, formal definitions for beliefs and trust relationships for humans and organisations are defined.

5.2.4 Definitions for beliefs and trust relationships

Definitions for beliefs and trust relationships for humans and organisations are defined next (Abdul-Rahman 2004; Chervany & McKnight 1996; Castelfranchi & Falcone 2003; Grandison 2003).

Definition – Belief: *A belief is an entity's acceptance of something as truth.*

Definition – Trust Relationship: *A trust relationship exists if an entity holds beliefs over another entity that enables it to determine its trustworthiness. A trust relationship does not exist between strangers who have no knowledge of each other's existence.*

Trust properties, dimensions and concepts, and organisational trust have now been described. Trust for machines that support web services is next investigated to determine its synergy with human and organisational behaviour.

5.2.5 Trust for web services

The aim of web services technology is to establish inter-organisational infrastructures via machine-to-machine communication. In ensuing interactions, machines act on behalf of humans and organisations. The absence of human contact should cause machine-to-machine trust formation to be fundamentally different from its humanistic counterpart. Ideally, trust formation for machines should be able to borrow from elements of human trust, as the latter is so highly developed. To establish whether web services trust can indeed include human elements, the next sub-paragraphs highlight properties and dimensions of trust for web services, motivate an assessment approach as a basis for web services trust, describe the nature of web services trust, and finally, propose an autonomous approach towards web services trust.

5.2.5.1 Properties of trust

From the current body of research (Marsh 1994; Grandison 2003; Dimitrakos 2003), properties of trust similar to those mentioned for humans were identified for computing entities. Some of these properties can be summarised as follows:

- Trust is dependent on a specific context or situation where risk is accepted when interactions occur.
- Trust is a measurable belief that reflects its strength.
- Trust evolves over time through new experiences and observations.
- Trust is subjective.

Web services requestors and providers exhibit similar trust properties to those of humans and organisations. Unacquainted web services introduce themselves to one another, and form relationships with one another over time. Trust is dependent on the given situation where the perceived benefits gained are weighed against the cost and risks. Furthermore, trust is either bi-directional (between a web services provider and web services requestor) or uni-directional (where a web services provider may trust the web services requestor, but not vice versa).

The focus of this research is the trust that the web services provider holds towards its web services requestors, as it is used by the web services access control service in order to grant access to resources.

5.2.5.2 Dimensions of trust

Web services trust formation differs from its humanistic counterpart because of the absence of human contact. This excludes the emotional dimension of trust. Web services providers do have the ability to evolve trust over time through new experiences and observations. Experience and judgement can be used to form trust subjectively. For this reason, cognitive trust, defined on the basis of *information* and *reasoning*, can be investigated for the formation of machine-to-machine trust relationships. Cognitive trust is suited for web services environments, as it is used when participants do not know one another, or when participants are far removed from one another.

5.2.5.3 Basis of trust

It is not possible to directly portray the intuitive manner in which humans reason over beliefs. Machines that support web services providers can exhibit a form of intuitiveness if they can make inferences from incomplete information and can react in unfamiliar situations. This can be achieved if trust concepts can be identified, which can be meaningfully populated with information. For machines, intuitive reasoning over trust information is then replaced by a formal process of trust assessment and mathematical reasoning. Trust concepts are rather collections of entities originating from numeric values through a process of assessment, and are arranged together due to their similarity or related functionality. This enables machines to follow humanistic reasoning as trust concepts are defined to be fuzzy in nature. Machines can be programmed to include tolerance for imprecision, uncertainty and partial truth in its reasoning, similar to the way that humans intuitively do.

For a web services requestor it may be important to establish whether it trusts the intentions and competence of a web services provider, whereas a web services provider may need to establish whether the web services requestor is honest, reliable and creditworthy. Very often, web services may play the role of both requestor and provider. To accommodate these different viewpoints, the machine-to-machine trust formation process must cover a diverse range of trust concepts.

Relevant questions for trust between web services are:

- How is trust between web services represented within a particular context?
- Where does one find the information that is used to create trust?
- How does the performance of a web services requestor affect its perceived risk and resulting trust relationship?

Answers to these questions are the aim of the next chapter.

5.3 CONCLUSION

This chapter gives an overview of trust in order to provide a basis for important concepts that are used in the model presented in this thesis. The chapter commences by giving a critical overview of different approaches to the management of trust. The inclusion of information and evidence in trust computation, in order to counteract the limitations of creating trust through the verification and validation of public keys, is highlighted. A preliminary definition of trust management is given that identifies automation and trust assessment as important features.

Trust, as it is manifested between humans and organisations, is then discussed to determine if synergy between it and web services trust formation can be found. The properties, dimensions and basis of humanistic trust is discussed. Thereafter, organisational trust is discussed. The trust perspective taken by this research is subsequently identified. Finally, the properties, dimensions and basis of web services trust is described.

In order to provide the basis for web services trust, the next chapter gives an overview of characteristics of trust formation and evolution for web services entities. Characteristics are defined within a trust formation framework that is able to assign a level of trust to other parties. The framework defines an autonomous and phased approach to trust. The trust context, level, computational paradigm and manner in which trust is formed are also discussed.

6

Web Services Trust Formation Framework

This chapter presents a web services trust formation framework. The trust formation framework is characterised by a phased approach where trust is established over time. The context, level and computational paradigm of trust are defined, after which the main focus of the trust formation framework is described. The framework is aimed at a process of assessment defined on the basis of information. In order to establish trust concepts, an analysis is made of information sources found in the web services environment. The structure of trust concepts, defined by information sources, is made explicit by a taxonomy of these concepts. Formal definitions of trust management, trust assessment, trust relationships, trust types and trust concepts is given. Finally, the chapter is concluded.

6.1 TRUST FORMATION PHASES

Autonomous trust formation for web services requires a phased approach to allow trust to evolve (Dimitrakos 2003). The initial trust formation phase, or the way in which trust evolves from a state of ignorance to a state of either trusting or distrusting another, needs to be considered. Trust also evolves during the lifetime of the trust relationship. These aspects need to be considered in the light of web services environments. The development of trust for web services requires movement through the following phases, as shown in Figure 6.1:

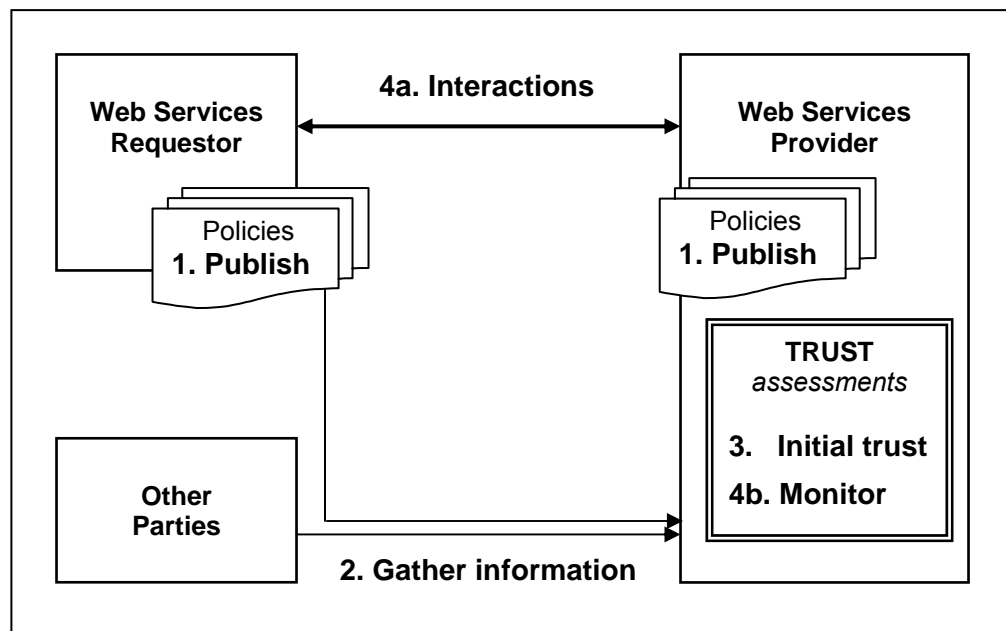


Figure 6.1: Phases of the formation of trust for web services

6.1.1 Publish trust information

Web services providers and requestors need to inform others how they will behave, and what they may expect from them, in order to establish a mutual trust relationship. Statements should be contained in machine-readable policies to enable others to determine how to comply with requirements, as shown by labels (1) in Figure 6.1. For interoperation, a shared understanding of information must exist between web services requestors and providers. Currently, no semantics exist for such trust formation.

6.1.2 Discover trust information

Providers of web services will trust web services requestors in which they have confidence. The evaluation of web services requestors should not just be based on their identity, but also on reputation, previous experiences and other evidence that proves competence. Information can be gathered by inspecting policies, or by evaluating recommendations and references from other parties, as shown by label (2) in Figure 6.1.

6.1.3 Trust formation

When a web services requestor or provider is selected, an initial level of trust needs to be formed, based on trust concepts that are established for the other party. As part of this process, the policies of both web services requestor and provider need to be inspected. The result of this

phase is a trust measure that indicates the basic level of initial trust that will be extended to the other party, shown by label (3) in Figure 6.1.

6.1.4 Trust evolution

Web services providers monitor all interactions – as shown by label (4a) in Figure 6.1– in order to evaluate the level of trust held towards others, shown by label (4b). If all transactions are processed smoothly, trust evolves so that future interactions may be granted access to more sensitive resources or more risky transactions, based on the new and higher level of trust.

The result of these phases is a level of trust that can be included in the access control policy.

In order to make trust formation and evolution possible, the context in which trust is formed is described in the next paragraph.

6.2 TRUST CONTEXT

In the real world, the context of each action plays an important role. For instance, a clerk may be trusted to hold the keys to the safe, but may not be trusted with confidential managerial information. This would also be true for web services requestors that access web services operations.

Enabling the trust mechanism of a web services provider to deal with several contexts would have a high cost in terms of complexity, as multiple trust values would have to be maintained within a variety of contexts. In contrast, trust in a single context is much simpler to maintain, as a single trust value is associated with each party that the web services provider interacts with. In the virtual application defined in Chapter 3, the eBooks web services provider interacts with all parties only in the context of the business functionality that it provides. To have a different trust context for each operation or grouping of operations is not a viable solution, as the information required to form a different trust value for each context is scarce (Sabater 2002).

What is rather required, is the ability to purposefully use each piece of information that is acquired. Different types of information may be used for different reasons. By analysing and categorising information, it can be made more useful for different situations. By creating meaningful trust concepts on the basis of which trust is defined, it would be possible to make decisions not only about the trust level, but also about trust levels of each trust concept, as may be required by different situations. For instance, if a transaction involving a large amount of money is performed, the creditworthiness of a web services requestor may be more important to consider than the level of goodwill held towards her/him.

To be able to use trust in a specific context, its level needs to be established. Considerations pertaining to a trust level are therefore highlighted next.

6.3 TRUST LEVEL

Trust has no natural physical measurements. Dasgupta (1988) parallels the value of trust with knowledge and information by stating that “*even though there are no obvious units in which trust can be measured, this does not matter, because in any given context you can measure its value, its worthiness. In this respect, trust is not dissimilar to commodities such as knowledge or information.*”

There are many different approaches to represent the level or measure of trust held towards another party. Some systems support arithmetic operations on recommendations and other evidence, so numeric quantification is more appropriate (Marsh 1994). It is also possible to use discrete verbal statements (Abdul-Rahman & Hailes 2000). However, there is still a problem relating to representation of ignorance, or the unknown, with respect to trust.

Gambetta (1988) was first to define trust in a way that it could be represented mathematically. The definition defined a subjective probability by which parties assess each other's trustworthiness. The definition made trust quantifiable in a range from 0 to 1, where 0 represents complete distrust and 1 complete trust. Marsh (1994) defined trust between -1 and 1 as either complete distrust or blind trust, with 0 as ignorance. Grandison (2003) represented the changing trust relationship between a trustor and trustee by a trust measure between 0 and 100, that reflects the degree of trust assigned to a relationship for a given context or action.

Poblano (Chen & Yeager 2003), a proposed trust model for the JXTA (Sun 2004) peer-to-peer platform, defines trust measures between -1 and 4, where -1 = *distrust*, 0 = *ignorance*, 1 = *minimal*, 2 = *average*, 3 = *good*, and 4 = *complete*. PGP (Zimmermann 2005) defines introducer trust levels as *don't know* = ignorant of public key's trustworthiness; *untrustworthy*, = public key not to be trusted to introduce others; *marginal* = public key can be trusted to introduce another key, but it is uncertain whether it is fully competent to do that; *full* = public key is always trusted to introduce another public key.

Trust Project (Golbeck 2003) is a social network that is built up from distributed data maintained on the semantic web. Within FOAF (Friend Of A Friend) files (The Foaf Project 2002), users include trust ratings for people they know using the FOAF Trust Module, a simple ontology for expressing trust ratings. The ontology has a vocabulary for rating people on a scale of 1 (low trust) to 10 (high trust) with the following descriptions: *distrustsAbsolutely*, *distrustsHighly*,

trustsModerately, distrustsSlightly, trustsNeutrally, trustsSlightly, trustsModerately, trustsHighly, and trustsAbsolutely.

For these examples it is clear that trust levels used by humans reflect their perceptiveness, as a wide-ranging set of levels can be used. For machines this is not possible, as their perceptiveness is restricted to information that is presented or sourced and processed by programs that are inadequate when compared to the human brain.

When a trust level is chosen, it should be useful for the purpose it will serve.

For this research, machine-formed trust levels need to reflect a measure of trust held towards web services requestors, in order to make better access control decisions. The access control policy of the web service access control service needs to refer to a few trust levels in order to grant access to sensitive resources. For this purpose, the discrete trust levels shown in Table 6.1 are chosen to indicate the trust that web services providers hold towards their requestors.

Table 6.1: Trust levels

HIGH
GOOD
MODERATE
LOW
IGNORANCE

In order to establish levels of trust, the next paragraph describes computational methods that can be used for this purpose.

6.4 TRUST COMPUTATION

For humans, trust values are not arrived at by using extensive computations. Humans are highly perceptive and combine trusting experiences and situations intuitively. Composite trust values are used over and over again in similar situations, which reconfirm or change their value. For machines this is not possible, and computational trust is the only viable course of action. Computational models for trust can broadly be classified into two categories: the cognitive approach and the mathematical approach (Esfandiari & Chandrasekharan 2001).

In the cognitive approach, trust consists of underlying beliefs, and trust is a function of the value of these beliefs. An example of the former kind of model is the one of Castelfranchi and Falcone (1998). The mathematical approach incorporates some aspects of game theory and the evolution

of cooperation models. An example of this kind of model is the one by Marsh (1994). Both approaches see trust as a variable that can be used as a threshold for action.

In this thesis, a cognitive approach towards trust computation is followed by a process of trust assessment of trust concepts.

There are a number of different ways in which trust can be computed (Boyd et al. 2005; Grandison 2003):

- *Simple summation of inputs*: The simplest way to compute a trust measure is to simply compute the sum and average of inputs for a category (McDonald & Pirzada 2004).
- *Bayesian systems*: Bayesian systems take binary ratings as input. Scores are computed by statistically updating beta probability functions (Jøsang & Ismail 2002).
- *Discrete trust models*: Discrete verbal statements are used, rather than continuous measures (Abdul-Rahman & Hailes 2000).
- *Belief models*: Belief theory is a framework related to probability theory but where the sum of all probabilities over all possible outcomes does not necessarily add up to 1. The remaining probability is interpreted as uncertainty (Jøsang 1999).
- *Fuzzy models*: Trust is represented by linguistic fuzzy concepts where membership functions describe to what degree a party is trustworthy or not (Manchala 1998).

For this research it is important to consider the fact that trust is a subjective and vague concept that is difficult to quantify. Since fuzzy logic allows reasoning with vague information and models the degree to which trust occurs, it may be better suited for use in this situation. Fuzzy cognitive maps (Kosko 1986) show potential for implementing a humanistic way of thinking about trust (Castelfranchi & Falcone 2002). It directly supports the approach to trust assessment that is adopted by this thesis. A fuzzy cognitive map can represent the trust assessment process of a machine in a symbolic manner, similar to the way in which humans cognitively manipulate belief and trust concepts. A web services provider can establish trust with others according to the structure of its fuzzy cognitive map, since this map enables it to make decisions based on observations and experiences. A fuzzy cognitive map (FCM) is illustrated in Figure 6.2.

Figure 6.2 consists of 4 nodes that represent concepts C1 to C4 that are related to one another. Nodes are assigned a value in the fuzzy interval range [0, 1]. Signed and weighted arcs represent the causal relationships that exist among concepts. The arcs of the graph represent the impact that one concept has on another and vary in the interval [-1, 0, +1]. The sign + or - indicates an increase or decrease in the effect that one concept has on another. Causal relationships are characterised by vagueness, as they represent the influence of one qualitative factor on another with linguistic variables. For example, the effect that C2 has on C1 is +0.85. This means that the designer of the map is *sure* that the effect of C2 on C1 is *very good*, or *very sure* that the effect of C2 on C1 is *good*.

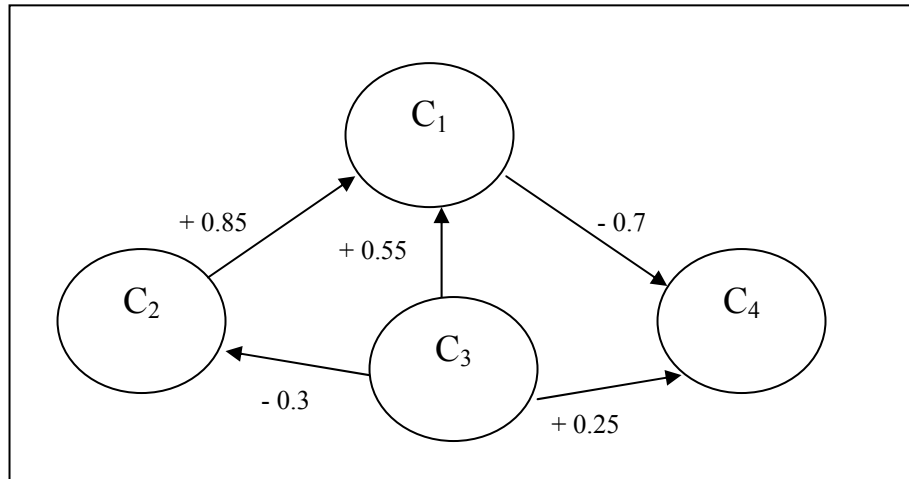


Figure 6.2: Fuzzy Cognitive Map (FCM)

To be able to populate the nodes of a fuzzy cognitive map with a value in the interval $[0, 1]$, information and evidence need to be aggregated. For this purpose, basic mathematical operations will be required to sum or average inputs. Finally, a discrete value such as “MODERATE” will be derived from a fuzzy cognitive map.

In order to understand the manner in which a fuzzy cognitive map forms trust, consideration is next given to the basis of trust for this research.

6.5 TRUST ASSESSMENT FOR TRUST CONCEPT FORMATION

The approach towards trust assessment defined in this research is characterised by *information* and *reasoning*. It requires that all aspects over which trust is formed and evolved, be identified. As mentioned earlier, trust is formed by the assessment of information, which leads to the formation of trust concepts. It is now important to consider what a trust concept consists of and what its source is.

In society, people form beliefs by collecting information through experiences, observations and recommendations. For distributed systems, trust should be based on information as far as possible (Jøsang 1996). For Internet applications, trust formation is defined as the act of collecting, codifying, analysing and presenting evidence that relates to competence, honesty, security or dependability, to be able to form trust relationships with others (Grandison 2003). Evidence may for instance consist of certificates for proof of identity, certificates describing competence, and risk assessments.

Similarly, trust concepts for web services requestors can be formed by sourcing, analysing and categorising the information available in the XML-based environment. The strength of the trust that is formed is determined by the quality of information that can be sourced. Decisions about who to trust is therefore based on the properties of a web services requestor and the security and trust requirements of a web services provider, defined in a policy.

Web services environments present unique opportunities for the assessment of information and for the consequent formation of trust concepts between machines. In order to enable integration between machines, a considerable amount of information is made available in different XML-based policies. Policies are accessible, as they are machine-readable and platform independent. The proliferation of web services specifications such as WSDL (Web Service Definition Language) (Christensen et al. 2000), UDDI (Universal Description, Discovery and Integration) (Bellwood et al. 2003), WS-Policy (Box et al. 2003), WS-Security (Atkinson et al. 2002), WS-Trust (Della-Libera et al. 2003), WS-SecureConversation (Anderson et al. 2005), WS-Addressing (Box 2004), BPEL4WS (Business Process Execution Language for Web Services) (Andrews et al. 2003) and WSLA (Web Service Level Agreements) (Dan et. Al. 2004) leads to an increase in the availability of XML-based information that covers different aspects of web services. Another useful source of XML-based information is SOAP events.

Automated machine-based assessment of XML-based information that leads to the formation of trust concepts is thus a reachable goal, as such information is readily available. Common sources of information that are used to form trust are environmental information, references, recommendations and experience. Each of these sources is defined in the context of web services in the paragraphs that follow.

6.5.1 Environmental information

To enable integration, both web services providers and requestors can describe what they offer to and demand from the other party. For instance, security mechanisms such as passwords and encryption algorithms are currently communicated in WS-Policy to others, through a set of security policy assertions defined within the WS-Security specification (Atkinson et al. 2002). To enable secure communication with another party, a web services provider needs to know whether requestors support WS-Security and which security tokens can be used. For instance, although a web services provider may support any combination of UsernameToken, Kerberos ticket, or certificate, it may prefer a certificate. A web services provider must also determine whether requestors require signed messages, and what token type must be used for the digital signatures. If encryption is required, the other party must know when to encrypt the messages, which algorithm to use, and how to exchange a shared key with the web services provider. These security requirements are addressed by the `<wsse:SecurityToken>`, `<wsse:Integrity>`,

and `<wsse:Confidentiality>` elements. This information allows web services providers to trust requestors, based on supported security mechanisms.

6.5.2 References

A primary concern for a web services provider is a lack of knowledge about a new web services requestor. A new web services requestor therefore first needs to prove its competence in a specific domain. If trusted authorities, through references, endorse a new web services requestor, it can be assigned a basic level of trust. References are statements in the form of certificates from independent third parties. The existence of references can be revealed to prospective web services requestors in WS-Policy documents. As a business publishes its own references, a web services requestor needs to confirm the validity of the information with the issuing party.

6.5.3 Recommendations

Since it is not possible for a web services provider to evaluate all aspects of a given situation when making a trust decision, such a provider can also rely on recommendations from others to form a trust relationship. A recommendation is an opinion obtained from another party pertaining to a specific situation or context, such as the delivery of goods or the quality of information provided. It is important to consider how much the third party may be trusted, and what trust may be extended to the web services requestor under consideration. A web services provider could publish in the WS-Policy document a list of partners from whom it would accept recommendations. Prospective web services requestors may use this list to get recommendations that will be trusted by the web services provider. A recommendation signed by the issuer is returned to the web services provider in a predefined format. As a recommendation is in a machine-readable format, with schema-defined context and values elements, the web services provider can understand it.

6.5.4 Experience

Trust is also created through the progressive gain of experience with others. Experience refers to the cumulative view of the result of interactions with a web services requestor in a context. All communication between machines that support web services requestors and providers occurs by means of SOAP requests and responses. By analysing SOAP messages, a basic profile of a web services requestor's reliability may be created.

It is important to note that reputation is not directly addressed by this research. The main difference between an autonomous trust system and a reputation system is that a reputation system produces a trust level as seen by a whole community, whereas autonomous trust reflects

the trust that the web services providers held towards another party (Boyd et al. 2005). Information sourced from reputation systems can be incorporated with other evidence that is collected by the web services provider's trust system.

It would be impossible to determine whether a web services requestor is truthful about its properties. However, information concerning dishonest behaviour can be recorded as a bad experience, and used in further trust formation.

Information for the purpose of trust assessments can thus be gathered automatically by processing environmental information, references from other parties and recommendations that may accompany requests, and also by recording experiences through inspecting SOAP messages. Mechanisms must exist at a web services provider and requestor to support the publication of policies, the interchange of references and recommendations, and the recording of experiences. Protocols ensure that messages are sent correctly, so that they are understood by communicating parties.

In order to use information for the purpose of trust formation, a basic taxonomy of trust concepts for trust formation is defined in the next paragraph.

6.6 TAXONOMY OF TRUST CONCEPTS

Motivated by the way in which humans trust, this research now proceeds to identify trust concepts for machine-based trust assessment. The structure of the required information is made explicit so that all web services providers and requestors will understand it. A taxonomy can assist the process of inter-operability and specifications for the trust formation process (Jasper & Uschold 1999). The taxonomy presented in Figure 6.3 defines relevant concepts for the formation of trust concepts.

Trust assessment, as described in this research, has the aim of evolving trust from a low level, where trust is brittle and fragile, to a high level, where trust is robust, and goodwill plays an important role.

This research now maintains that trust is formed by three high-level trust types:

- Firstly, before any web service interaction takes place, a web service assesses the properties of its *internal environment* to establish its confidence and its general disposition to others.
- Next, a web service assesses the properties of the *external environment* or institution within which the trust relation exists. This trust assessment increases the trust level to reflect moderate levels.

- Finally, a web service assesses the *other party's properties* over time to increase its trust level. The trust level evolves and can grow to a high level that reflects goodwill that may exist between web services.

Each of these three trust types is next described in more detail. Trust concepts are identified that contribute towards each trust type.

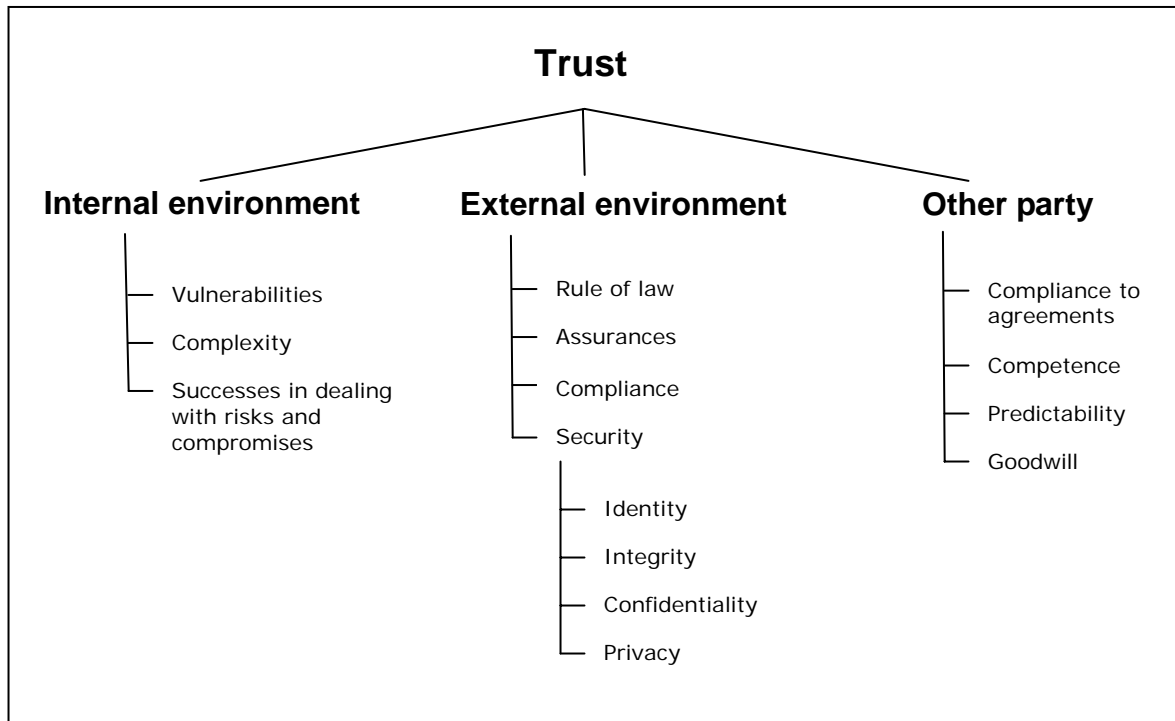


Figure 6.3: Taxonomy of trust types and trust concepts

Each of the trust concepts that may be used to form and evolve trust is described in more detail below.

6.6.1 Trust in the internal environment

This trust type is created from domain information that represents the expertise that exists within the environment of a web service provider. If a web services provider does not have the necessary expertise to conduct commerce over the Internet, the trust extended to its partners is affected. Risk analysis and confidence in own expertise may also play a role in trust formation, as is explained below:

- Vulnerabilities – risk analysis, as well as the management of risk, is well developed for computing environments. An organisation is able to perform risk assessments in order to determine its own vulnerabilities, perceived risks, costs and benefits. Armed with this

information, an organisation could promote the establishment of trust thresholds for risky transactions.

- Complexity – an application with high complexity is prone to pose more of a risk than one with a low complexity. This would influence the trust exhibited towards a web services requestor.
- Successes in dealing with risks and compromises – an organisation may feel that it has successfully dealt with risks and compromises and that it has the necessary expertise to adequately manage a risky endeavour with a web services requestor. This would influence the trust exhibited towards a web services requestor.

6.6.2 Trust in the external environment

This trust type is defined by the general practices that are supported by a web service. If well-accepted practices are in place, one partner will be more inclined to interact with another, as measures exist that provide safeguards. Risk is thus reduced in the face of misconduct. Rule of law, assurances, compliance and implemented security mechanisms may play a role in trust formation.

- Rule of law – parties who have a contract with one another have indirect trust in one another because the judicial system exists and will enforce the contract. Acts such as the US Computer Fraud and Abuse Act (1984), US Electronic Communications Privacy Act (1986) and the EU Data Protection Act (1994) provide further protection (Eloff & Granova 2003).
- Assurances – licenses, insurance policies and Service Level Agreements (SLA) provide additional safeguards to protect against risk.
- Compliance – standards revolve around specific measures in a number of different control areas of security. Compliance with regulations and standards such as Sarbanes-Oxley (2002) and ISO17799 (2005) can increase trust in another party. Security policies, procedures and standards, encapsulated by compliance to standards ensure smooth functioning of interactions.
- Security mechanisms – the extent to which web services offer security mechanisms and properties will affect a trust relationship. Security-based mechanisms such as authentication, integrity and confidentiality and privacy ensure timely, accurate and complete transmission and receipt of transactions. If a partner makes use of well-established security mechanisms such as digital signatures, well-known encryption routines, strict authorisation mechanisms and best business practices, a web service will be more inclined to process its requests.

6.6.3 Trust in the other party

This trust type is defined by the properties of a partner. As a web services provider interacts with a web services requestor, it gains information about the honesty, competence and predictability of the other. The establishment of these characteristics furthermore leads to a measure of benevolence held towards a web services requestor.

- Compliance to agreements is the belief that a web services requestor complies with pre-defined agreements. In order to establish honesty, recommendations from trusted parties may be taken into account. The main source of information will be the recording of experiences such as valid transaction requests that are sent.
- Competence is the belief that a web services requestor has the necessary skills to perform a task. Information that can assist a web services provider to gain confidence in a web services requestor includes certificates from third parties such as ISO 17799 certificates, licences, credit ratings, audit information and endorsements.
- Predictability is the belief that the actions of a web services requestor are consistent, so that a forecast can be made about how a web services requestor will behave in a given situation. This can be achieved by inspecting SOAP messages that are sent and received and by recording for instance the number of messages in error, the value of transactions, the number of transactions, and the validity of message details.
- Goodwill is the belief that a web services requestor cares about the welfare of the web services provider. It may be established over time as a web services provider realises the benefits gained from increased cooperation with the web services requestor. Setting of thresholds for the increase in honesty, competence and predictability can computationally determine goodwill. Such thresholds can activate different levels of goodwill.

It is easier to establish the competence and predictability than the honesty and goodwill of a web services requestor. These trust concepts are not equally important for all situations. For instance, when a payment is made, it may not be important to establish the goodwill of a web services requestor, but rather to determine her/his compliance to agreements and competence.

The taxonomy identifies trust as the top-level node, three trust types as intermediate-level nodes, and fifteen trust concepts as lower-level nodes. The structure of the taxonomy makes explicit the information about which trust is formed and about which it evolves. The manner in which information is assessed to populate trust concepts is a process that can be tailored to the specific requirements of a web services provider.

Next, formal definitions are given for web services trust.

6.7 DEFINITIONS: TRUST MANAGEMENT, TRUST ASSESSMENT, TRUST RELATIONSHIP, TRUST TYPE AND TRUST CONCEPT

Trust management, trust assessment, trust relationship, trust type and trust concept, are now formally defined for this research.

Definition – Trust management: *The automated assessment of information and evidence, relating to the properties of the internal environment, the properties of the external environment, and the properties of the other party, with the purpose of establishing trust concepts from which a trust level can be inferred for access control and other decisions.*

Definition – Trust assessment: *Trust assessment is an automated process that systematically gathers information and evidence related to trust concepts.*

A trust relationship only exists when a machine has populated trust concepts pertaining to a web services requestor. This implies that a trust relationship can only exist between machines if a machine has information about another party, however minimal it may be.

Definition – Trust relationship: *A trust relationship T is a tuple*

$$T = \langle X, Y, t, T_t \rangle$$

where T is the trust a web services provider X has in a web services requestor Y , for a given period t , defined over trust types T_t .

T is expressed as a trust level TL as follows:

$$T \subseteq \{ TL_i \} \text{ where } TL_i = \langle \text{ignorance, low, moderate, good, high} \rangle$$

Definition – Trust type: *Trust is formed by three trust types that are defined by the tuple*

$$T_t = \langle T_{t_{\text{internal env}}}, T_{t_{\text{external env}}}, T_{t_{\text{other}}} \rangle \text{ where}$$

$T_{t_{\text{internal env}}}$ *is the trust in the internal environment of web services requestor X ;*

$T_{t_{\text{external env}}}$ *is the trust in the external environment between web services provider X and web service requestor Y ;*

$T_{t_{\text{other}}}$ *is the trust in web services requestor Y .*

A trust concept is the manifestation of a trust assessment process that accepts information and evidence as truth by the machine that supports the web services provider.

Definition – Trust concept: *A trust concept is a fuzzified category of information or evidence that is populated with a value between 0 and 1. Trust concepts Tc_i relate to trust types Tt_i as follows:*

$$Tt_{\text{internal env}} = \langle tc_{\text{vulnerabilities}}, tc_{\text{complexity}}, tc_{\text{successes}} \rangle$$

$$Tt_{\text{external env}} = \langle tc_{\text{ruleoflaw}}, tc_{\text{assurances}}, tc_{\text{compliance}}, tc_{\text{security}} \rangle$$

tc_{security} *is formed from trust concepts related to security mechanisms as follows:*

$$tc_{\text{security}} = \langle tc_{\text{identity}}, tc_{\text{integrity}}, tc_{\text{confidentiality}}, tc_{\text{privacy}} \rangle$$

$$Tt_{\text{other}} = \langle tc_{\text{compl_agreements}}, tc_{\text{competence}}, tc_{\text{predictability}}, tc_{\text{goodwill}} \rangle$$

tc_{goodwill} *is formed from trust concepts as follows:*

$$tc_{\text{goodwill}} = \langle tc_{\text{compl_agreements}}, tc_{\text{competence}}, tc_{\text{predictability}} \rangle$$

6.8 CONCLUSION

This chapter focused on the formation of trust between web services that participate in virtual applications. The approach to trust formation is based on the assessment of information that is sourced and categorised. Trust evolves gradually and includes trust in the environment and the underlying control and support mechanisms. In addition, experiences with partners and recommendations from trusted referees influence a trust relationship. The approach defined here identified information sources, and trust concepts used in trust reasoning. The fact that trust is a subjective and vague concept that is difficult to quantify will be taken into account when choosing reasoning over it. Fuzzy cognitive maps are identified as a mechanism to allow reasoning over information. This enables the establishment of the level to which trust occurs.

The chapter identifies components of a trust formation framework that will be addressed by the model presented by this research. Definitions for trust management, trust assessment, trust relationship, trust level, a trust type and a trust concept is presented.

From this discussion it is clear that decisions about whom to trust and believe are based on the properties of a web services requestor and on the security and trust requirements of a web services provider that will be defined in a policy. The specification of an access control policy that includes trust statements is a requirement that was identified earlier in this thesis. The focus of the next chapter is on access control policy specification that addresses trust and other identified access control requirements.

7

Access Control Policy Specification

This chapter discusses access control policy specification for the web services access control service. In Chapter 4 the interface policy and access control policy were identified respectively. These policies need to be specified by policy specification languages, which need to be expressive and flexible in order to accommodate all identified access control requirements. The interface and access control policies are analysed to determine the purpose that they serve, and this leads to the identification of a set of policy characteristics that relate to a policy purpose.

The two main approaches towards policy specification languages are either logical or XML-based. When the logical approach is used, a modification of first-order logic (Halpern & Weismann 2003) or Datalog (Ceri et al. 1989) is made to meet access control constraints. Very specialised policy specification languages of an ad hoc nature are defined with the XML-based approach. Even though reasoning about credentials is often specified in logical languages (DeTreville 2002; Jim 2001; Hayton et al. 1998), XML-based languages are naturally suited for Internet-based environments, whereas SOAP messages and other relevant information are formatted in XML (Biskup & Wortmann 2004; Bertino et al. 2003). XML-based policy rules would be best for the coordination of cross-domain interactions. This chapter investigates both types of approaches to determine their suitability to the policies for the web services access control service.

The chapter commences with a discussion of the purpose that each policy serves, in order to determine whether either a logical or XML-based approach would be more suitable. Next, an example of a policy specification language for the purpose of policy publication is discussed. The chapter is concluded with a discussion on a policy specification language that can be used for access control reasoning.

7.1 PURPOSE OF WEB SERVICES ACCESS CONTROL SERVICE POLICIES

Web services access control service makes use of interface policy and access control policy specification languages where:

- The interface policy communicates requirements and capabilities of the web services provider to web services requestors, and vice versa. This is supported by access control policy publication.
- The access control policy describes the conditions under which an action is granted or denied. This is supported by access control policy reasoning.

The publication of a policy can be seen as its disclosure, but performed by means of web services specification languages, so that it can be understood by web services requestors. In order to protect sensitive information, this thesis refers to selective publication as a means to give requestors information they need at a point in time so that they can interact with the web services provider. Before these two types of language are discussed, a formal definition of a policy specification language is given to illustrate how it differs from a standard.

Definition - Policy specification language *A policy specification language is a language that describes in a complete, precise, verifiable manner the requirements, design, characteristics and behaviour of a policy.*

Definition - Standard *A standard is a policy specification language that has been approved by standards bodies such as OASIS (OASIS 2005) or W3C (W3C 2005).*

Policies for access control publication and access control reasoning are now discussed with reference to the type of policy specification language that would best support its purpose. Each discussion is completed by referring to examples of existing policy specification languages.

7.2 POLICY SPECIFICATION LANGUAGE FOR ACCESS CONTROL PUBLICATION

The web service access control service communicates access control, trust, and other requirements and capabilities through publication to specific web services requestors or to a public domain by means of the interface policy. The latter is basically a set of assertions about a web services provider, such as the type of encryption algorithm that is supported or the format of a credential that is required, so that web services requestors understand how to use web services operations.

The interface policy will typically specify SOAP message security requirements for authentication, confidentiality and integrity in the following way: *"The order operation requires either a X.509 v3 certificate or a username and password of the requestor. The message body must be encrypted with triple DES. The message body must then be signed using the RSA algorithm."*

For this research, access control requirements such as *"the order operation requires the subject's identifier on whose behalf the request is made"* need to be expressed. These requirements are domain specific and are used by the web services access control service in order to grant access to the requestor-order operation. In addition, assertions about the trust formation process must be made so that requestors can understand which types of information to provide. For instance, *"recommendations from trusted parties A, B or C can be provided in required format, found at URI www.eBooks.com/recommendation".*

The access control requirements identified in Chapter 2 demand an approach to publication to be described as follows:

- Policies should clearly and unambiguously state access control and trust requirements.
- Policies should be human-readable and machine-readable, to allow administrators to read and interpret policies, but also to limit human intervention where possible.
- Policies should be platform independent to allow machines to interoperate across boundaries.
- Policies should be defined by common terminology to ensure that both web services providers and requestors have the same understanding of a policy.
- Policies should be defined in a standards-based manner so that all parties can use and understand policies.

For these reasons, XML is next investigated as a means for policy publication.

7.2.1 XML

XML is a formal language that can be used to create a policy specification language to publish the interface policy to administrators and machines of web services requestors. When considering either the logical or XML approach, XML supports the mentioned considerations as follows:

- XML, a meta-language, is a W3C standard to define the storing, publishing and exchanging of information in a platform-independent manner.
- XML relies on Unicode (Unicode 2005), a standard for international language encoding so that information is not only exchanged between machines, but also across national and cultural boundaries.
- With XML, common standards for information exchange can be defined.

- The support XML enjoys from software vendors has rapidly made it the *de facto* format for data exchange across independent domains.

With XML, new languages are formed by the set of core concepts (Burman et al. 2000) as discussed below.

Elements

An XML element is a named construct that has a set of attributes and some so-called children. The children of an element can be other elements, literal text, comments or other types, and they are strictly ordered. Elements are written using brackets as follows:

```
<Policy>.....</Policy>
```

Attributes

Attributes are name-value pairs that are properties of an element that are not ordered. An element can have any number of attributes and is written as follows:

```
<Policy> PolicyName = "policy1"   URI = "www.abc.com" </Policy>
```

Comments

Comments are used as follows:

```
<!-- the following option is required -->
```

Literal text

Elements can contain character sequences that consist of Unicode characters.

```
<IssueDate>15 July 2005</IssueDate>
```

Document

An XML document consists of a strictly nested hierarchy of elements with a single root. The next document depicts a student credential.

```
<?xml version="1.0"?>
  <Student ID="3">
    <name>
      <surname>Smith</surname>
      <firstname>Sue</firstname>
    </name>
    <university>UP</university>
    <department>CS</department>
  </Student>
```

A policy specification language defined with XML makes the content, relationship and meaning of the interface policy clear. Web services providers and requestors that communicate with published interface policy documents need to agree on a set of element names, their valid content and the kinds of literal text that are permissible as attribute values and element content. An XML Schema ensures that elements and attributes of a policy document are defined as by the

document structure, and that the right type of data is used. The logical structure of subject attributes, credentials, references, recommendations and other information can unambiguously be defined.

To enable web services requestors to interpret and understand access control and trust requirements, XML schemas (defined by structures of XML elements and attributes and separate namespaces) need to be defined. The consequence for policy processors at both web services requestors is that they should be able to support the syntax of the interface policy directly by employing policy-assertion processing components.

Next, WS-Policy is discussed as an example of a policy specification language that is used to publish message security and other requirements and capabilities to web services requestors or providers.

7.2.1.1 WS-Policy

WS-Policy was developed by a group consisting of Microsoft (2004), IBM (2004), SAP (2004), BEA (2004), Verisign (2004) and Sonic Software (2004). It defines a basic set of constructs that can be used and extended by other specifications to describe a broad range of web services requirements, preferences and capabilities. WS-Policy refers to a set of three specifications:

- *WS-PolicyFramework*, referred to as WS-Policy (Bajaj et al. 2004a), which defines the overall model and syntax that can be extended by other specifications such as WS-Security;
- *WS-PolicyAssertions* (Bajaj et al. 2004b), which defines a basic set of assertions for policies, including whether an assertion is required or not; and
- *WS-PolicyAttachment* (Box et al. 2003b), which defines how to attach policy assertions to WSDL files.

The primary advantage of WS-Policy is that assertions are compact and easy to read (Anderson 2005). Assertions are domain specific, with each policy item or group of items having its own set of assertions. This requires separate XML schemas, defined by structures of XML elements and attributes and separate namespaces. Web services requestors comply with a policy by searching through and matching policy assertions. Because of its extensibility, WS-Policy has been identified as a possible mechanism to use in the management of autonomous computing systems (Ganek 2004).

Figure 7.1 is an example of a requirements policy for eBooks addressing message security. A policy expression is an XML serialisation consisting of a top-level container element `<wsp:Policy>`. The URI for the policy is "http://ebooks.com/policies#P1". This container encloses the policy assertions and identifies all namespaces being used. Assertions maintain

independence between web services providers and their requestors, as requirements for interaction are kept apart from the functionality of web services operations.

```

<wsp:Policy xml:base="http://ebooks.com/policies" wsp:Id="P1"
  xmlns:wsp="....."   xmlns:wsse="....." >

  <wsp:ExactlyOne>
    <wsp:All>
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:X509V3</wsse:TokenType>
      </wsse:SecurityToken>
    </wsp:All>
    <wsp:All>
      <wsse:SecurityToken>
        <wsse:TokenType>wsse:UsernameToken</wsse:TokenType>
      </wsse:SecurityToken>
    </wsp:All>
  </wsp:ExactlyOne>

  <wsse:Confidentiality wsp:Usage="wsp:Required">
    <wsse:Algorithm Type="wsse:AlgEncryption"
      URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
    <MessageParts>
      wsp:GetInfoSetForNode(wsp:GetBody(.))
    </MessageParts>
  </wsse:Confidentiality>

  <wsse:Integrity wsp:Usage="wsp:Required">
    <wsse:Algorithm Type="wsse:AlgCanonicalization"
      URI="http://www.w3.org/Signature/Drafts/xml-exc-c14n" />
    <wsse:Algorithm Type="wsse:AlgSignature"
      URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <wsse:SecurityToken>
      <wsse:TokenType>wsse:X509v3</wsse:TokenType>
    </wsse:SecurityToken>
    <MessageParts>
      Dialect="http://schemas.xmlsoap.org/2002/12/wsse#soap">
        S:Body
      </MessageParts>
    </wsse:Integrity>
  </wsp:Policy>

```

Figure 7.1: WS-Policy of eBooks for message security

Policy assertions are typed and can be either simple or complex. Simple assertions do not require special treatment when they are evaluated. An assertion that indicates that a specific field is to be presented in an interaction does so simply by its presence in the policy document. Complex assertions need to be compared with regard to policy operators. Policy operators enclose assertions, define priority using a preferences attribute, and specify the number of assertions from the list to be enforced. Options include `<wsp:All>`, `<wsp:ExactlyOne>` or `<wsp:OneOrMore>`. In the first part of the policy in Figure 7.1, enclosed by the `<wsp:ExactlyOne>` element, there are two alternatives for a security token from which one must be chosen. The security token can be requested through WS-Trust interactions and be transformed by a security token service to be in the published format. If the first alternative is selected, only the X509 token type is supported; otherwise, if the second alternative is selected, only the Username token type is supported. In the next part, it is stated that the body of a message must be encrypted with triple DES. Then the body must be signed using exclusive canonicalisation with the RSA algorithm, and an X509 security token must be included.

7.3 POLICY SPECIFICATION LANGUAGE FOR ACCESS CONTROL REASONING

The web services access control service reasons about rules stated in the access control policy. Reasoning requires logical processes to derive at decisions such as grant or deny. To ensure reliable and consistent decisions, the access control policy rules must not only be represented formally to ensure clarity, but formal reasoning ensures consistent decisions.

Access control policies that are described informally are often unclear and ambiguous. Consider the following statement from a high-level access control policy:

web services requestors may execute the search operation

This statement, in natural language, can be considered to be the policy that controls access to the search operation. From this statement it is not clear who is not granted access to the search operation, or whether any other operations other than *execute* are granted or forbidden. It therefore illustrates that the specification of an access control policy should be concise, exact and clear.

If the access control policy is more formally expressed in policy specification languages such as XrML (2001) or XACML, the interpretation of such syntax can still lead to ambiguities. This is because their semantics are described in English, and can be interpreted differently by policy administrators. As mentioned, a language such as XACML can be considered an attempt to create an access control policy specification language afresh, whereas logic-based languages are modifications of first-order logic.

By nature, access control decision making is a logical deduction process. Logical languages are attractive as access control policy languages, as the declarative nature of logic offers a good compromise between expressiveness and simplicity (Bonatti & Samarati 2003). Research on logics for access control shows its potential for ensuring access control specifications of high assurance for complex, distributed systems (Abadi 2003). What is required, is a logical policy specification language with clear syntax and semantics that can be used to represent the access control policy unambiguously, and that can reason about it. The logic should be expressive so that a variety of conditions can be captured, and it should manage queries efficiently.

Access control requirements for the access control policy demand an approach that includes the following:

- Policies should be defined independently from policies of other parties.
- Policies should be kept private so that they are not compromised.
- Policies should be expressive.
- Policies should be able to express both positive and negative access control rules.

- Policies should be able to reason conflicts that may arise from access control rules.
- Policies should be able to consistently make access control decisions about subject attributes and trust levels.
- Policies should be able to answer queries effectively.

These mentioned requirements may naturally be satisfied by a modification of first-order logic, as there have been numerous successful attempts to declaratively define access control policies with this approach. First-order logic is next investigated as a means for policy reasoning.

7.3.1 First order logic

Languages in first-order logic (FOL) are each a restriction of a many-sorted first-order logic with a different vocabulary that consists of a set of parameters such as quantifier symbols, predicate symbols, constant symbols, function symbols and others. First-order logic includes the following set of core concepts (Barwise & Etchemendy 2000):

Constant

A symbol whose referent has been fixed, e.g.

Mary, file.txt

Function

A rule for associating a member of one set with that of another, e.g.

operation(Search) = execute, member(Customer) = John

Predicate

A function from the subject of a statement to truth values, e.g.

grant(John, search), John(Customer), colour(Sky, Blue)

Variable

A symbol whose referent varies or is unknown, e.g.

x, y

Connective

A symbol that joins two or more propositions, e.g.

not (\neg), and (\wedge), or (\vee), implies (\Rightarrow), if and only if (\Leftrightarrow)

Quantifier

A symbol describing the number of objects from a domain that are asserted, e.g.

universal quantifier (\forall), existential quantifier (\exists).

Term

A constant symbol, a variable symbol, or an n-place function of n terms, e.g. x and $f(x_1, \dots, x_n)$ are terms, where each x_i is a term.

Atom

An atom is a simple proposition. If P and Q are atoms, then $\neg P$, $P \vee Q$, $P \wedge Q$, $P \Rightarrow Q$, $P \Leftrightarrow Q$ are atoms.

Sentence

A sentence is an atom, or, if P is a sentence and x is a variable, then $(\forall x)P$ and $(\exists x)P$ are sentences.

Well-formed formula

A well-formed formula (wff) is a sentence containing no free variables. All variables are bound by universal or existential quantifiers.

Access control policy languages are not directly stated in first-order logic, but rather in some variant of Datalog (Ceri et al. 1989). Datalog is a restricted form of logic programming with variables, predicates and constants, but without function symbols. Many benefits can be gained by using Datalog (Li & Mitchell 2003):

- The semantics of Datalog-based languages are declarative, unambiguous and widely-understood.
- Datalog has been extensively studied in both programming language and in the context of relational databases as a query language.
- The function-symbol-free property of Datalog ensures its tractability.
- Queries can be answered by efficient goal-directed evaluation procedures.

An access control policy defined as a Datalog program consists of a set of facts and rules. Facts are statements such as “Sue is a customer”. Rules are statements that allow the deduction of new facts from existing ones. An example of a rule is “If X is a customer, X may place an order”. Both facts and rules are represented as Horn clauses (Barwise & Etchemendy 2000) of the form

$$L_0 \leftarrow L_1, \dots, L_n,$$

where each L_i is a literal of the form $p_i(t_1, \dots, t_k)$ such that p_i is a predicate symbol and all t_j are terms. A term is either a constant or a variable. The left-hand side is called the head of the rule and the right-hand side its body. The body of a clause may be empty; in such cases, a fact is being stated. A literal, fact, rule or clause that does not contain any variables is called ground.

Policies can be defined in safe stratified Datalog (Damianou 2002), or Datalog with constraints (Li & Mitchell, 2003):

- *Safe stratified Datalog*: Stratified logic permits a constrained use of recursion and negation while disallowing those combinations that lead to undecidable programs. In a stratified program, clauses are ordered in such a manner that for any clause containing a negated literal in its body, there is a clause later in the program, which defines the negated literal.
- *Datalog with constraints*: The tractability of Datalog is a direct consequence of the absence of function symbols. This may be limiting when trust levels are required to be expressed. Datalog^C (Datalog extended with constraints) (Li & Mitchell, 2003) allows first-order formulas in one or more constraint domains, which may define trust hierarchies and time intervals used in the body of a rule. This allows more flexible access control reasoning about structures in a declarative language. Two simple examples of classes of constraint domains include
 - Equality constraint domains, represented by a set of constants and one predicate =;
 - Order constraint domains, represented by a linearly ordered structure and two predicates: = and <.

The next paragraphs investigate existing policy specification and enforcement languages and frameworks that can be used for logic-based access control specification and reasoning.

7.3.2 Examples of policy specification languages for access control reasoning

Languages that can be used for access control reasoning aim to support the expression and enforcement of access control policies. Examples include ASL (Jajodia et al. 1997), Keynote (Blaze et al. 1999a), SDSI (Lampson & Rivest 1996), SPKI (Ellison et al. 1999a, 1999b), Ponder (Damianou & Dulay 2001), Binder (DeTreville 2002), SD3 (Jim 2001), XrML (2001), Tower (Hitchens et al. 2001), the Security Policy Language (SPL) (Ribeiro et al. 2001), The Trust Policy Language (Herzberg et al. 2000), X-RBAC (Bhatti 2004) and XACML (Anderson et al. 2003) – to name but a few. Languages are able to express and enforce different access control policies within a single framework (Bertino et al. 1997). Many languages are designed for use in distributed systems where credentials are verified with cryptography. Not all of these languages have been designed or presented as logical systems, but have been influenced by logical work (Abadi 2003). Damianou (2002) provides a comprehensive survey on policy specification approaches. Access control policies can be expressed and processed in either logical languages or in XML-based languages. Each of these approaches will be discussed briefly next.

7.3.2.1 Logic-based languages

There have been many attempts to define policy specification and enforcement languages in a fragment of first-order logic such as Binder, SD3 and ASL. Policies are generally defined in Datalog. Although most of the efforts about access control policy specification focus on the use of formal logic, some approaches have been proposed for high-level logical languages such as SPL, Tower and Ponder, where access control is specified and enforced in a declarative object-oriented language. These languages are not formal and do not support automatic analysis or verification of security properties. There also exist custom languages such as SDSI/SPKI and Keynote that are influenced by logic. Next, a number of access control specifications are discussed to highlight important features of these languages. First, ASL and Ponder are described. Then, access control specification as used by frameworks and systems from the trust management community, such as SDSI/SPKI, Keynote, SD3 and the Service Access and Release framework are described to illustrate the specification of cross-domain access control integration.

a) ASL

The Authorisation Specification Language (ASL) (Jajodia et al 1997) is a logic-based policy specification language that is independent of any specific access control model. ASL is defined in Datalog, with stratified clause form logic. The next rule states that a subject may read file1 if the subject is in the Customer role. ASL is described in more detail later in this chapter.

```
cando(file1, s, +read) ← in(s, Customer)
```

b) PONDER

Ponder is a declarative, object-oriented language for specifying security and management policies. It allows policy types to be defined to which any policy element can be passed to create a specific instance. Ponder is considered a very complete language to express access control and other policies (Diaz et al 2002). However, Ponder is not a formal language and does not support automatic analysis or verification of security properties.

r denotes the target, file1, and s the user.

```
type    auth+ rule1 (subject <user> s, target <file1> r)
{
action read if belongs(s, r.Customer)
{
result = enable;
}
}
```

c) SPKI/SDSI

SPKI/SDSI is a credential based public key infrastructure that binds authorisations to public keys. (Ellison et al 1999a, 1999b), (Lampson and Rivest 1996). It supports a specification language that is used for distributed access control. SPKI/SDSI is a custom language, but can be viewed as a logical specification in a very rudimentary sense (Abadi 2003). SDSI (Simple Distributed Security Infrastructure) proposes the use of local names, and SPKI (Simple Public Key Infrastructure) deals with authorisation and delegation of authorisation. SPKI/SDSI does not require central control and allows principals to specify their own trust structures independent of each other.

SPKI/SDSI access control specifications support two kinds of credentials, namely *name certificates* to bind principals to names and *authorisation certificates* to bind authorisations to names. Besides name certificates and authorisation certificates, SPKI/SDSI also provides access control lists for specifying access control policies for a resource.

- *SPKI/SDSI name certificate* is used to bind principals to a name and is a document of the form <Keyholder,Name,Subject,Validity>.
- *SPKI/SDSI authorisation certificate* is used to bind an authorisation to a name and is a document of the form <Keyholder,Subject,Authorisation,Delegation,Validity> that is signed by the keyholder.
- *SPKI/SDSI access control list* for specifying access control policies for some interface. An ACL is a list of ACL entries which are documents of the form <Self,Subject,Authorisation,Delegation,Validity> and is not signed.

SPKI/SDSI specifications are very difficult to read and understand. Web service environments can employ SPKI/SDSI for integration, as no central certification authority is required. By using SPKI/SDSI, a Web service provider thus has the ability to create its own trust environment.

d) Keynote

KeyNote is a capability-based trust management system developed by Blaze et al. (1999a). It makes use of access control specifications that manage the delegation of authority for applications. The KeyNote engine is passed a list of credentials, policies, the public keys of the requester and an "Action Environment", which is a list of attribute-values that contains all the information relevant to the request. A credential is an authorisation assertion and describes the conditions under which one principal authorises actions requested by other principals. KeyNote policies delegate authority on behalf of the associated application to otherwise untrusted parties. To be able to integrate KeyNote into an application, it is important to identify attributes that can be used to gain access. The result of the KeyNote evaluation process is an application-defined string, where the simplest response could be "*authorised*".

An example of a KeyNote assertion is: (Blaze et al 1999b)

KeyNote-Version: 1

Authorizer: rsa-pkcs-hex:"1023abcd"

Licensees: dsa-hex "986512a1" || rsa-pkcs1-hex:"19abcd02"

Comment: Authorizer delegates read access to either of the Licensees

Conditions: (\$file == "etc/passwd" && \$access == "read") -> {return "ok2"}

Signature: rsa-md5-pkcs1-hex:"f00f5673"

Keynote does not establish trust between strangers, as clients are assumed to possess credentials that represent authorisation of specific actions with a known application server. It is thus not suited to web services environments where parties do not know each other.

e) SD3 (Secure Dynamically Distributed Datalog)

SD3 (Jim 2001a) illustrates how public keys are distributed under the control of a policy in a high-level language. It consists of a policy language, a local policy evaluator and a certificate retrieval system. SD3 predicates are prefixed with an issuer, which is a public key. This delegates the authority of predicate definition to the public key. An IP address can also be added to a predicate in order to refer to a remote policy. SD3 is a very general system and does not specify access control semantics. A SD3 policy can be implemented as follows (Jim 2001b):

```
AliceKey = Principal(DSA("P: 6e+1xVU3D5d..."));
BobKey = Principal(DSA("P: vO3gkdeTc0E..."));

PKD("alice",AliceKey):- ;
PKD("bob",BobKey) :- ;
PKD(user,key) :- AliceKey$PKD(user,key);
```

The policy file defines two public keys: *AliceKey* and *BobKey*. A policy *PKD* associates keys with user names. This is similar to a public key directory. *PKD* says that the key of the user *alice* is *AliceKey*, and the key of *bob* is *BobKey*. The last line says that *PKD(user,key)* holds if *AliceKey\$PKD(user,key)* holds. *AliceKey\$PKD* is the public key directory defined by Alice. This implies that whenever Alice's policy says that a user has a particular key, it will be true.

SD3 is modelled as a distributed query evaluation process, which assumes that all parties will provide credentials to support the query evaluation. For web services environments, such an assumption does not always hold.

f) Service Access and Information Release

Bonatti and Samarati introduced a uniform framework and model to regulate service access and information release over the Internet (Bonatti and Samarati 2002). The framework includes a portfolio and service protection language (PSPL), for expressing access control policies for services and release policies for client and server portfolios. PSPL has a well-defined

semantics and is monotonic. The language includes a policy-filtering mechanism to protect privacy during policy disclosure.

A portfolio contains both data declarations and credentials. Data declarations are not certified by a trusted third party. A client or server submits data declarations and credentials in order to obtain access to protected services. Credential- based access control is supported, even though the service provider has no previous knowledge of the service requester. PSPL is designed to enable the selective disclosure of credentials to disclose to gain access to sensitive resources. In PSPL, rules govern access to protected services. PSPL has rules governing services and rules governing portfolio data such as prerequisite rules, requisite rules and facet rules.

Servers specify service rules to regulate access to their services. As an example, a service rule dictates that the service *buy* can be granted to clients that submit a declaration stating their credit card number.

```
service_req(buy()) ← declaration(credit_card_number = X).
```

Both requestors and servers formulate release rules that dictate credentials and declaration release. As an example, a release rule dictates that the credit card information can be released, only in the context of buying requests and upon the reception of a non- disclosure agreement from the counterpart.

```
release_req(credit_card_info) ← declaration(no_disclosure="accept"),
                                current_service(buy()).
```

The approach defines a move away from traditional access control rules defined by (object, subject, action) tuples, to the specification of access restrictions. Disadvantages that exist are that requestors must understand the negotiation process, and must be possession of credentials as they are requested. The only capabilities for trust negotiation that are not represented in PSPL are support for declaring who should submit which credentials contained in a policy, support for transitive closure, and support for credential chain discovery (Child et al 2002). For web service environments, the framework is very useful, as the identity of users may not be known, but they may be in possession of required credentials and declarations.

g) SECURE

In the computational trust framework of SECURE, an explicit cost-benefit analysis is used to determine how much trust is required to offset risk. While a trust value is being calculated for a requestor, the access control manager reads the outcome costs for the action and checks any specified environmental constraints such as the time of day. A predicate call-out mechanism is used to include trust and cost/benefit computations within OASIS policy rules. There are three predicates used for this purpose. `Trust` retrieval uses the contexts associated with the action to

retrieve relevant trust values into rule parameters for a named principal. *Cost* retrieval retrieves the set of outcome costs into rule parameters for a named action and *Risk* thresholding fails if the trust is too low for the outcome's cost.

```
trust(principal, context; value?)
cost(action, outcome, cost?)
risk(value, trust, cost)
```

To determine whether a user is willing to store a given file on a particular server, the following Prolog policy is used. Three trust values (T - availability, confidentiality, integrity) and three cost value (Cost - unavailable, ignore_AC, too_strict) are determined. The request is granted if the risk is computed and trust found to be more than the cost.

```
privilegeRequest(fileAction_store,[ServerID,FileName]) ←
  trust(ServerID,availability,T_1), trust(ServerID,confidentiality,T_2),
  trust(ServerID,integrity,T_3),
  cost(FileName,unavailable,Cost_1), cost(FileName,ignore_AC,Cost_2),
  cost(FileName,too_strict,Cost_3),
  risk(unavailable,T_1,Cost_1), risk(ignoreAC,T_2,Cost_2),
  risk(tooStrict,T_3,Cost_3).
```

The specification of access control is very different to previously discussed languages, as complex policy decisions are made based on the context of trust and the cost associated with the request.

7.3.2.2 XML-based languages

Access control policies can also be expressed in XML. XML provides a uniform, platform-independent representation of organisational information. It can provide a mechanism for sharing and disseminating access control policies and information across heterogeneous applications. XML is thus very often seen as the natural choice for the basis of a common access control policy language, due to the ease with which its syntax and semantics can be extended to accommodate requirements of an organisation. It further enjoys widespread support from platform and tool vendors (Anderson et al. 2003). Recent proposals express access control policies as XML documents as exemplified by XACML, the Trust Policy Language, X-RBAC or X-TNL. Other recent advances can be found in the semantic web environment. Ontologies and markup are used to capture security information of web services. They are developed in OWL (Ontological Web Language) (Harmelen & McGuinness 2004), which implies that there is richer semantics for describing policies. For instance, KaoS includes an ontology for representing agents with rights and obligations and REI is used for describing policies in pervasive systems (Bradshaw et al. 2003).

a) XACML

XACML is a very sophisticated policy specification and enforcement language. XACML includes conditional authorisation policies, as well as policies with post-conditions to specify actions that must be executed prior to permitting an access. XACML has an extensible system of data types and functions to ensure interoperability. Included are combining algorithms, which define how to take results from multiple rules or policies and derive a single result. The *example policy* shown next is a very simple access control restriction that states that all requests have to be for *ServerABC*, for the *authenticate* operation.

```
<Policy PolicyId="ExamplePolicy"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
    rule-combining-algorithm:permit-overrides">
<!--Requests may only be made to ServerABC -->
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <ResourceMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">ServerABC
        </AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
      </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
<!--The action must be to authenticate -->
  <Rule RuleId="AuthenticateRule" Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            authenticate
          </AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="ServerAction"/>
        </ActionMatch>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

As illustrated here, policies in XACML can be complex and verbose, thus making it difficult to work directly with the language or policy files. It allows the specification of fine-grained access control rules. As policies from independent domains can be composed with each other, XACML can ideally be used when web services entities interoperate.

b) X-TNL

X-TNL (Bertino et al. 2003) is an XML-based language for trust negotiation that is used by the Trust-X (Bertino et al. 2004) framework. The language defines certificates and disclosure policies. Certificates are either signed credentials or unsigned declarations that carry information that is not sensitive. Credentials and declarations are grouped into datasets and X-Profiles that organise all the information of a particular subject. A novel aspect of Trust-X is the support for trust tickets that are issued upon the successful completion of a negotiation. The trust ticket is used by a requestor to speed up subsequent negotiations for the same resource.

Credentials define sets of properties that need to be evaluated during negotiation. They are defined by sets of templates called credential types, modeled by DTDs (Data Type Definitions), for specification of credentials with similar structure. An X-TNL credential is an instance of a credential type, as shown in the next example. Declarations are similarly structured.

```
<CS_Student credID="334B", SENS="NORMAL">
  <Issuer HREF="http://www.ABC.com" Title=CS_Student_Info/>
  <Name>
    <FName>Sue</FName>
    <LName>Smith</LName>
  </Name>
</CS_Student>
```

Disclosure policies state the conditions under which a resource can be released during negotiation. The next example shows a disclosure policy of a tertiary institution. It allows students of the CS department to park their cars without paying.

```
<policyspec>
  <properties>
    <certificate targetCertType = CS_Student>
    <certCond>
      //student_number[@code=CS.requestCode]
    </certCond>
    <certCond> /.../[position=student]</certCond>
    </certificate>
  </properties>
  <resource target="Park_Car"/>
  <type value="SERVICE"/>
</policyspec>
```

X-TNL is thus able to support different types of certificates in the form of credentials and declarations. X-TNL simplifies the specification of credentials by making use of credential templates. Uniqueness is ensured by the use of XML namespaces. X-TNL thus provides many features that can be used in support of web services access control.

7.3.2.3 High-level comparison of languages

Both logical languages and XML-based languages suffer from complexities that make them difficult to be used by administrators. The main difference between these approaches can be found in the manner in which policies are executed.

A logical program can be viewed as an executable formal specification of access control. In the environment there exists an efficient inference engine that provides a mechanism to automatically derive access control decisions at runtime. In addition, the correctness and consistency of an access control policy can be established. For the purposes of access control specification, formal validation and verification, logical languages are more suited. In complex web services environments, logical and formal specifications play an important role in detecting undesirable behaviour (Butler et al. 2002). Access control policy statements, translated into logic programs, can for instance be executed to analyse the effect of composing assertions and stating subjects attributes by means of existing access control rules. Formal specifications offer improvements in verification and validation, and can demonstrate the absence of undesirable behaviour by the access control policy of a web services provider.

Even though formal logic-based approaches are useful for analysing access control policies, they are relatively difficult to apply. Another disadvantage is that administrators view logical access control specifications as daunting, as they generally do not have a background in mathematical logic.

XML-based and high-level languages are generally easier to understand than formal logic-based approaches, but their correctness and consistency cannot be formally established. Predefined types and functions support XML-based policy specification languages and high-level programming languages. This requires an algebraic approach that uses policy specifications in executable programs for runtime checking. Although such approaches may be considered simpler to implement since XML with its related technologies and programming languages such as Java are well established, the chances of error due to faulty application code may increase. Next, ASL is discussed as an example of a policy specification language that is used to perform access control reasoning.

7.3.3 ASL

The Authorisation Specification Language (ASL) (Jajodia et al 1997) is a logic-based language that is independent of any specific access control model. It is supported by a framework that was comprehensively reported in (Jajodia et al 2001). The researcher finds ASL a comprehensive access control specification language because of a number of reasons:

- ASL is very flexible and expressive.
- ASL is independent of any access control model.
- ASL supports both closed policies, where all positive access control rules have to be specified, or open policies, where all negative access control rules have to be specified.
- ASL is defined in Datalog, with stratified clause form logic. It can thus express negative rules and still be decidable.

- ASL can be used to implement the access control policy requirements identified in Chapter 4, namely flexibility, efficient administration exceptions and conflict resolution.
- ASL can be extended to add new requirements of attribute-based access control and trust levels.
- ASL supports conflict resolution policies such as *denials take precedence* (where negative access control rules prevail over positive), *permissions-take-precedence* (where positive access control rules prevail over negative) and *nothing-takes-precedence* (where the conflict remains unsolved).
- ASL resolves inconsistencies among access control rules by using rules.

Disadvantages are that ASL can be complicated, because it consists of interdependent rules that must be fully understood by administrators. It only exists as a theoretical notation as there is no software support for it.

ASL defines an access control policy as a 4-tuple consisting of an object, user, role set and an action and is created from the following alphabet:

Constant Symbols: Every member of $O \cup T \cup U \cup G \cup R \cup A \cup SA \cup N$

O is a set of objects;

T is a set of object types;

U is a set of subjects;

G is a set of groups;

R is a set of roles;

A and SA is a set of unsigned and signed access control rules; and

N is a set of natural numbers.

Variable Symbols: Sets of variables $V_o, V_t, V_u, V_g, V_r, V_R, V_a, V_{sa}$ ranging over the sets

$O, T, U, G, R, 2^R, A, SA$, such that

$o \in O, t \in T, u \in U, g \in G, r \in R, a \in A, sa \in SA$ and $i, j \in N$.

Next, the predicates of ASL are described.

Table 7.1: ASL predicates

in (s1, s2).....	The direct membership relationship between subjects
dirin (s1, s2).....	The indirect membership relationship between subjects
typeof (o, t).....	The grouping relationship among objects
active (u, r).....	The role active for a subject
cando (o, s, sa).....	Access control rules explicitly inserted by the administrator
dercando (o, s, sa).....	Derived access control rules
do (o, s, sa).....	Conflict resolution policy and access control rules that subjects hold for objects
grant (o, s, r, sa).....	Accesses to be allowed or denied
done (o, s, r, a, t).....	Used to keep a history of rules executed by subjects on objects
error ().....	Used to enforce integrity

In Figure 7.2, an access control policy for eBooks is defined in ASL. A fragment of the access control policy is defined that governs the access of subjects to the Place_Order operation of eBooks.

<pre> dirin(s, customers) ← in(s, gold_customers) & in(gold_customers, customers) or in(s, standard_customers) & in(standard_customers, customers). cando(place_order, s, +exe) ← in(s, customers) or dirin(s, customers). </pre>
<pre> dercando(place_order, s, +exe) ← cando(place_order, s, +exe). </pre>
<pre> do(place_order, s, +exe) ← dercando(place_order, s, +exe). do(place_order, s, +exe) ← ¬dercando(place_order, s, -exe). </pre>
<pre> error() ← do(place_order, s, +exe) & active(s, fed_customers). </pre>

Figure 7.2: Access control policy for the place_order operation of eBooks

Four stages of access control are implemented consecutively. Firstly, propagation policies are defined so that the permission to execute *Place_Order* is propagated to subjects that belong to the *Customers* role hierarchy. Next, permissions are derived and a final permission is granted with the *do* predicate. Finally, integrity constraints are checked. The constraint specifies that a subject who is also active in the *Fed_Customers* role cannot place an order, as permission to do this is reserved only for subjects in the *Customers* role. Operations such as *Search*, *View_Order* and *Cancel_Order* would be treated in the same manner.

With the existing set of predicates, the access control policy cannot evaluate requests made on behalf of abilities, expressed as a set of attributes. For instance, for the *order* operation, the identity of the subject is presented to the web services provider, but this identity is not related to a local identity and is only used as a reference. A decision to grant access to the operation is rather made based on the trust level of the web services requestor, such as “good”. None of the current set of predicates can express access control rules that address trust levels. Extensions to implement these rules are described in Chapter 11.

7.4 CONCLUSION

This chapter identified two purposes for policy specification languages, namely access control publication and access control reasoning. The discussion highlights the fact that the interface policy should be specified in XML so that it can be both machine- and human readable across domains. WS-Policy is discussed as an example of an exiting policy specification language that can be used for the publication of access control requirements and capabilities.

For access control reasoning, a logical specification provides many benefits. This research identifies ASL as a comprehensive access control specification language that can be extended to allow autonomous access control decision-making. In practice though, logical specifications are not implemented, but access control specifications are rather materialised in XML-based languages. This research chooses to use logical specifications, as access control reasoning can be verified and easily understood.

8

Web Services Access Control Service Architecture

The web services access control service is added to the already existing operational web services architecture (Coyle 2002) as an integral part of the system design to ensure security of the whole system. As there is no central control when web services collaborate, the design of the web services access control service needs to include an autonomous approach to trust so as to provide a foundation for access control decisions. When considering the architecture of the web services access control service, it is therefore important to consider state-of-the-art access control and trust architectures in order to establish the best approach.

The main focus of this chapter is to identify components to be included in the web services access control service architecture. In order to identify relevant components, an investigation is conducted into access control standards and frameworks. Architectures for trust management, automated trust negotiation and computational trust are also described. The chapter is concluded with a list of those essential components and considerations for the web services access control service architecture that address access control requirements.

8.1 ARCHITECTURES FOR ACCESS CONTROL AND TRUST

Architecture describes high-level security designs in terms of the major components of a system and their interrelationships. To determine components of the web services access control service architecture and their interrelationships, a number of frameworks and architectures for access control and trust are investigated next.

8.1.1 Architectures for access control

Important approaches to consider as a basis for the web services access control service are the ISO 10181-3 Access Control Framework (ISO/IEC 1996) and the IETF (Internet Engineering Task

Force) policy architecture (Guerin et al. 2000) for policy-based access control. As the XACML architecture (Anderson 2003) is an important extension of the IETF policy architecture, its components are also considered. Finally, OASIS (Open Architecture for Secure Interworking) (Bacon & Moody 2002) is described and important features of access control architectures are summarised.

8.1.1.1 ISO 10181-3 Access Control Framework

The ISO 10181-3 Access Control Framework defines four components that participate in an access request: Initiators, Targets, Access Enforcement Functions and Access Decision Functions, as shown in Figure 8.1. ISO 10181-3 defines an architecture for access control that highlights interrelated components, but not an access control policy specification or an access control policy query language. An initiator requests access to a target resource. The initiator submits the request to a resource manager, which incorporates an Access Enforcement Function (AEF). The AEF is a specialised function that is part of the access path between an initiator and a target on each access and that enforces the decisions made by the Access Decision Function (ADF). The AEF submits the request, along with information about the initiator, to an ADF.

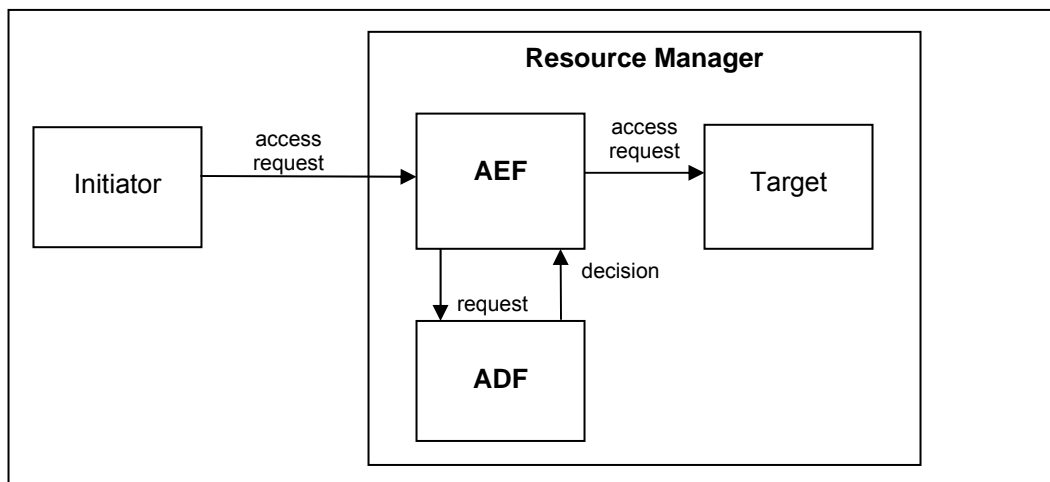


Figure 8.1: Components of the Access Control Framework

The ADF is also a specialised function. It makes access control decisions by applying access control policy rules to a requested action, as well as access control information, and the context in which the request is made. The ADF returns a decision to the AEF, and the AEF enforces the decision. The policy considers a variety of authorisation attributes when a decision is made. Attributes are classified as subject attributes, target attributes, request attributes and context attributes. For instance in eBooks, subject attributes may be an ID number or type of employee; target attributes may be the location of the file; request attributes may be the priority of the request, and context attributes may be the time that accesses are allowed. An important contribution of this architecture is that it makes apparent the distinction between decision-making and enforcement components.

8.1.1.2 IETF policy management architecture

RFC (Request For Comment) 2753 (Guerin et al. 2000) describes a framework for policy-based access control. It is a policy management architecture that is considered the best approach for policy management on the Internet. The main components of this architecture are Policy Enforcement Points (PEPs) and Policy Decision Points (PDPs), as shown in Figure 8.2. Also included is a policy management service that is used to specify, edit and administer access control and other policies, and a dedicated policy repository where policy information is stored and retrieved. The IETF does not define a specific language to express policies but rather a generic object-oriented information model for representing policy information following a rule-based approach (Damianou 2002). For example, generic objects such as system elements, logical and physical elements, systems, service and users are included by the model with their associated properties, operation and relationships.

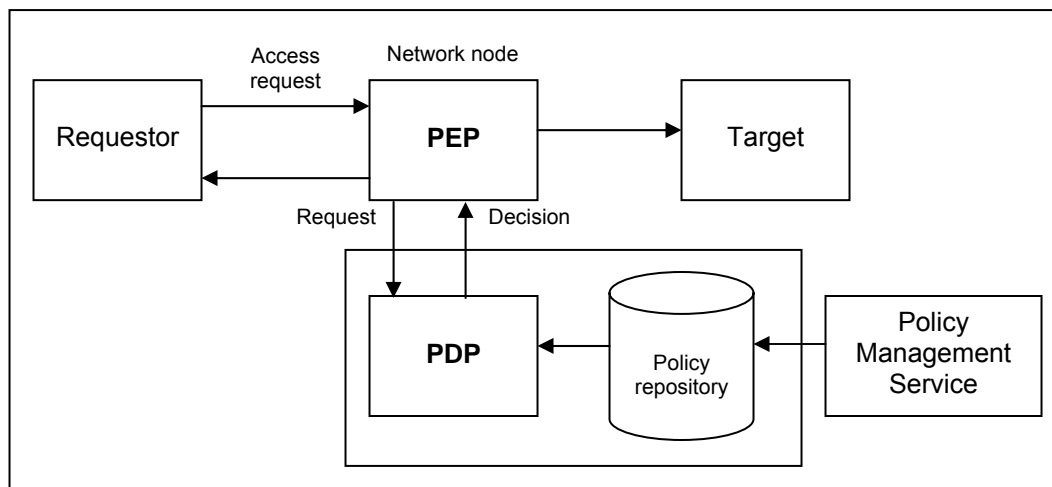


Figure 8.2: IETF policy management architecture

The PEP is a policy aware component that always runs at a network node. All interactions between components begin with the PEP. It interacts with requestors and enforces decisions it receives from the policy decision point. The PEP receives a message from a requestor that requires a policy decision. The PEP formulates a request for a policy decision, and sends it to the PDP. The PDP is a remote entity responsible for handling access requests and making decisions based on those requests. The policy language should ensure the unambiguous mapping of a request to an action. The PDP returns the decision and the PEP then enforces the policy decision by appropriately accepting or denying the request. The PDP may also return additional information to the PEP, which includes one or more policy elements. It should be possible for the PDP to ask the PEP to generate policy-related error messages as required.

The IETF policy architecture is extended by the work on XACML. A basic overview of the XACML architecture is shown in Figure 8.3.

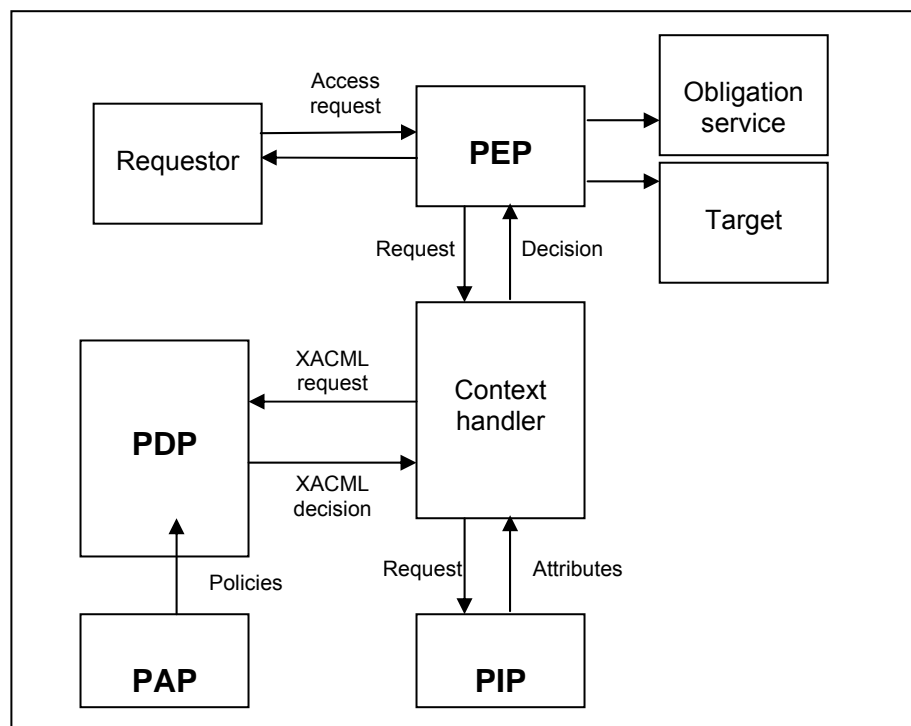


Figure 8.3: XACML architecture

An important difference with the IETF framework is the definition of XACML syntax to define not only access control policy rules, but also requests and responses between components. An XACML access control policy is also more than just rules of the form (subject, object action), as it includes rule-combining algorithms and obligations.

Additional components identified are the context handler that translates request attributes from their environment into XACML format, the Policy Information Point (PIP) from which additional attribute values can be requested for the subject or the resource, and the Policy Administration Point (PAP) that retrieves any number of policies, possibly from different domains, that are applicable to the access request. An access request is submitted by a requestor, and is intercepted by the PEP. The PEP sends the request to the context handler that translates the access request into an XACML request. The context handler may request from the PIP (Policy Information Point) attributes of the subject, resource, action and environment. The request is formatted and sent to the PDP. The PDP employs the PAP (Policy Administration Point) module to retrieve all applicable policies and evaluates the request against these policies. The PDP makes a decision and returns it to the context handler. The context handler converts the response to a format that can be understood by the PEP, with a set of obligations. Finally, the PEP fulfils the obligations and either grants or denies access.

XACML is a very sophisticated architecture that accommodates the distributed nature of subjects, objects, policies and other information in its design. As requests are sent from all kinds of environments, XACML has a “request translation” component to accommodate requests from

these often very different environments in order to enforce cross-domain decisions. A drawback is that access control rules from different domains must be defined in XACML syntax, and must refer to all subjects and objects in the same way.

8.1.1.3 OASIS

The Open Architecture For Secure Interworking (OASIS) from the University of Cambridge (Bacon & Moody 2002) is an example of an architecture that implements cross-domain role-based access control with certificates. Figure 8.4 shows the architecture of an OASIS service. The architecture consists of two main components: role activation and access control. Users are granted access to resources in other domains when they present valid role membership certificates (RMC). An RMC is created for users by a role activation component and is used by a remotely located access control component to give users access to resources.

In OASIS, targets are services that name their own roles. As roles are specific to a service, there is no need to administer roles globally. Roles are activated for the duration of a session, through credentials, as specified by a role activation policy. In addition, appointment certificates are created as long-lived credentials that can also activate roles.

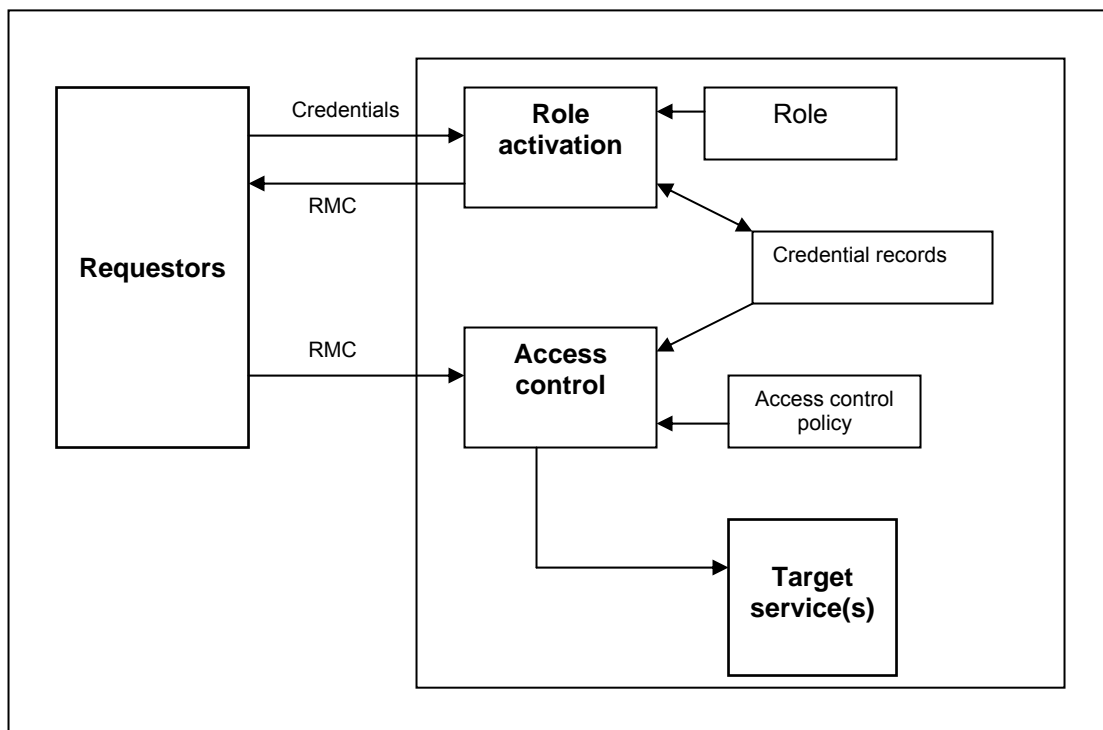


Figure 8.4: OASIS architecture

A user first activates a role by presenting its credentials to a target service. If all checks succeed, an encrypted RMC is issued and the service creates a credential record corresponding to it. This certificate represents the unforgeable, verifiable ability of the requestor. The requestor uses the

RMC with subsequent requests for services. A service validates the RMC, and checks any environmental constraints required by the access control policy. If successful, access is granted to the requestor.

Although OASIS is a cross-domain access control system, it illustrates how trust enables the movement of users across domains. In OASIS, access to resources is granted only if the content of a RMC can be trusted by verifying it cryptographically. OASIS also shows how abilities of users can be associated with roles to enable cross-domain access control. A drawback is that services are highly integrated to be able to understand the meaning of a role and to be able to create and validate a RMC.

8.1.1.4 Features of access control architectures

In summary, the following important features of access control architectures can be identified:

- The access control architecture is modular. Access control policy management, decision making and enforcement are separated.
- The policy enforcement point is highly specialised. It performs various functions that screens and protects the policy decision point from the environment.
- The architecture supports requests from heterogeneous environments. Requests can be translated into a format that can be understood by the policy decision point.
- The policy decision point considers a variety of attributes, and policies from different environments, when an access control decision is made.
- Support exists for a flexible access control policy specification and enforcement language.
- Support exists for standardised requests and responses between the policy enforcement and decision points.

8.1.2 Architectures for trust

Architectures that address trust in order to make better access control decisions are described next. An example in this regard is a trust management system, Fidelis (Yao 2003). This is followed by automated trust negotiation (ATN), an emerging approach that uses the properties of an entity to establish trust. As Trustbuilder (Trustbuilder 2001) is an important proposal in this field, its architecture is described next. In the third place, components that are used to create an autonomous approach to trust computation are illustrated by means of the SECURE project (SECURE 2003). These architectures represent the progression from binary trust that is created by the verification of public keys, to incremental trust that is created by the exchange of digital credentials, to autonomous trust that reflects the trust held towards another party by processing information, evidence and recording history. To conclude this section, the most important features of trust architectures are summarised.

8.1.2.1 Fidelis

In order for potentially distrusting principals to interoperate, the trust management system Fidelis (Yoa 2003) places strong emphasis on policy support supported by a policy language. In Fidelis a principal may be a person, an organisation, a computer process, or any other entity in some authority. The framework relies heavily on cryptographic mechanisms to identify principals, as public keys are used to identify principals.

Interoperation is realised by explicitly including trust as a feature of the architecture. In the case of Fidelis, trust is considered as a set of assertions that a principal holds with regard to another. A trusted piece of information is called a trust instance (Yao 2003). The trust instance quoted below states that organisation A (Org. A) believes that Sue is employed by them, and has an employee number 22888:

```
Is_employed("empid22888") : OrgA -> Sue
```

Trust conveyance allows principals to propagate trust statements to other principals. This allows principals to freely pass beliefs or assertions to others. A trust statement is represented as a public key credential, signed by the truster. It has a validity condition, which is defined by the truster to enable invalidation of outdated trust beliefs.

Credentials are trust instances that are fixed data structures, like membership cards. In the web services implementation of Fidelis, trust instances are represented in the Fidelis Interoperable Credential (FIC) format, which leverages the XML Signature standard to provide integrity guarantees. Policies, which are autonomously specified, administered and managed, interpret and provide the meaning for credentials. There are two kinds of policy in Fidelis: a trust policy that defines the relationships between trust statements and an action policy that relates an action to trust statements, to be used for access control.

Fidelis is implemented by a set of five communicating components (Yao 2003):

- The *conveyance* component allows trust instances to be exchanged between principals.
- The *trust inference* component encapsulates the evaluation of policies and answers queries made against these policies.
- The *credential management* component allows the management of trust instances, including collection, storage and retrieval in cases where this task cannot be performed by the principal.
- The *policy interrogation* component is designed to facilitate communication between strangers, so that unknown policies can be discovered through a query-based process. Nodes may either publish their ontology and policies or could support programming interfaces for interrogating and discovering their policies.
- The *trust agent* component is designed to automate communication between strangers. It provides an active interface on behalf of a principal. It automates the process of policy

interrogation and negotiation, and computes the disclosure set of credentials for requests.

Fidelis is an important step towards enabling potentially distrusting web services providers and requestors to interact securely. One of Fidelis's drawbacks is that it does not employ the concept of trust levels to allow a web services provider to treat trusted requestors differently according to policy rules.

8.1.2.2 Automated trust negotiation architecture

TrustBuilder represents one of the most significant proposals in the trust negotiation research area (Bertino et al. 2004a). It is a prototype system for negotiating trust between entities from different security domains, such as for military, business-to-business, and business-to-user interactions. Figure 8.5 shows the TrustBuilder architecture for trust negotiation. The architecture identifies the following components that are at its core, namely access control policies, negotiation manager and a strategy engine. Disclosure policies are access control policies that state the conditions under which a party can release a resource during a negotiation.

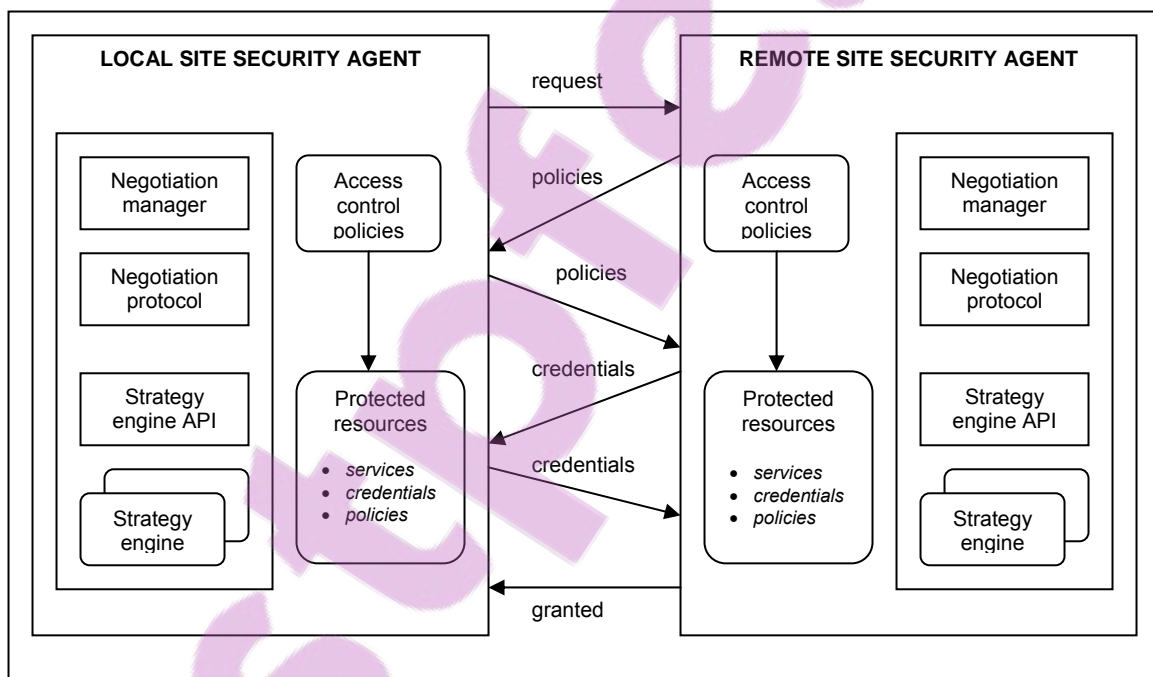


Figure 8.5: TrustBuilder architecture

The architecture supports a single protocol for establishing trust and is designed to support customised negotiation strategies. The goal of a trust negotiation strategy is to build trust through an exchange of digital credentials (Hess et al. 2002). The purpose of this exchange is to obtain access to protected resources. Policies govern the disclosure of both credentials and access control policies. Once the access control policy for a particular credential has been satisfied, a local negotiation strategy determines whether the credential is relevant to the current stage of the

negotiation. If so, it is disclosed. A credential or access control policy is disclosed if it has been sent to the other party in the negotiation, and a service is disclosed if the other party is given access to it.

8.1.2.3 Computational trust architecture

Architectures discussed so far accommodate trust with access control by focusing on a trust-management approach in which trust is determined either explicitly as in Fidelis, or incrementally as in Trustbuilder, for the purposes of a single interaction. In these approaches, evaluation of trust based on past experiences and other information is not considered. A different approach that can be followed is to include a trust component in the access control architecture so that the trustworthiness of requestors can be computed and be re-evaluated continuously. Instead of considering certificates as the basis of trust, certificates can rather be considered as one of many types of evidence to use when trust is computed.

An example of such an architecture can be found in the SECURE framework architecture (SECURE 2003), as shown in Figure 8.6.

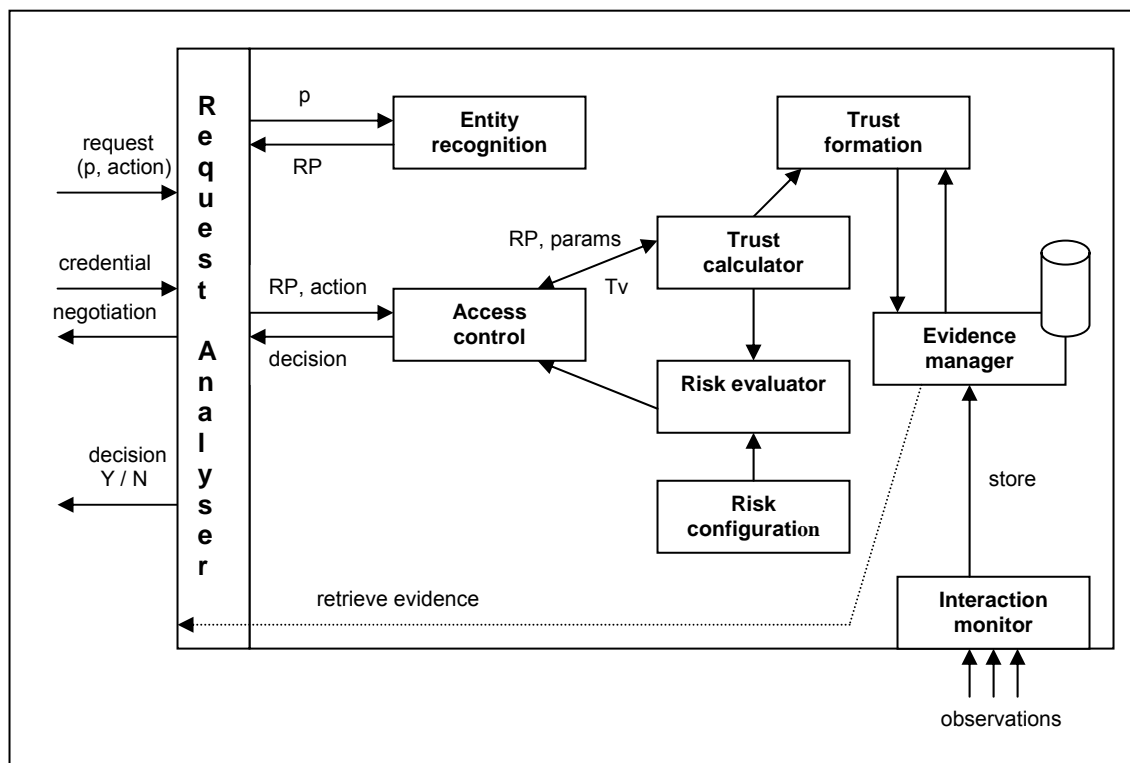


Figure 8.6: SECURE framework architecture

SECURE has the aim of implementing autonomous trust-based decision making for global computing. One focus of the project is Trust-Based Access Control (TBAC), which extends the work on OASIS role-based access control architecture. In the SECURE framework there are a number of components to consider, namely a Request Analyser and an Entity Recognition

component, with two main components that run in parallel: the Access Control Manager and the Evidence Manager (Cahill et al. 2004). The Access Control Manager has two sub-components, a Trust Calculator and a Risk Evaluator.

Firstly, all requests pass through the Request Analyser, which extracts from requests the specific action and determines its context. The Entity Recognition module next recognises *principals* (p). Entities are considered anonymous, as identity conveys little information about likely behaviour (Jensen & Seigneur 2005). The Entity Recognition module tries to recognise whether entities are trustworthy by using observable attributes such as multiple identities. The Access Control Manager grants or denies permissions for principals to execute *actions*, by enforcing requests against access control, privacy and evidence policies. For every decision, the Access Control Manager considers the trust it has in the requesting principal (p) and the risk of granting the request. A request by principal (p) may also include a list of *credentials*, which may include signed recommendations from other principals, and/or a list of referees whom the Trust Calculator may wish to contact for recommendations. The Access Control Manager looks up the relevant contexts for the requested action, and queries the trust calculator for a trust value (T_v) about p . This trust value is computed by examining *evidence* relevant to the current context. The Evidence Manager runs in the background and supervises evidence processing that is used to update trust and risk information. Evidence consists of *observations* of previous interactions with a principal, and *recommendations* from other principals.

8.1.2.4 Features of trust architectures

In summary, some important features that can be identified are the following:

- Trust is used in support of access control and other functions.
- Trust is formed by the evaluation of digital credentials.
- Trust formation is supported by policies that govern the disclosure of credentials.
- Where no central control authorities exist, trustworthiness of requestors is continuously evaluated by a trust component over experience and reputation.

8.2 WEB SERVICES ACCESS CONTROL SERVICE ARCHITECTURE

The architecture of the web services access control service incorporates several features from architectures that have been discussed here. In addition, environmental and internal access control requirements specifically need to be considered in the architecture design, which may necessitate adaptations from existing architectures. Requirements are now re-visited to determine their effect on components of the web services access control service. Environmental and internal access control requirements necessitate the different architectural components as discussed below:



Autonomy

The web services access control service architecture needs to accommodate autonomy by a policy-based approach that addresses autonomous access control decision-making. Policies address interactions between web services requestors and providers, the disclosure of information, as well as reasoning about access control rules.

Loosely coupled

The web services access control service needs to address loosely coupling by communicating information, by means of an interface and associated metadata, so that access control and trust information can be formatted by web services requestors, and be understood by the web services access control service.

Quality of service

The web services access control service architecture needs to address quality of service by the publication of information on its implementation of access control and other security services that protect messages.

Policy-based compatibility

The web services access control service architecture needs to accommodate policy-based compatibility by ensuring that there is a common understanding of access control requirements, credentials and other related information. This is realised by publishing XML schemas and ontologies that describe the structure of information such as credentials, subjects attributes, recommendations and references.

Policy negotiation

The web services access control service architecture needs to accommodate policy negotiation by a component that determines the compatibility of web services requestors and providers, and controls information and policy release for a given interaction.

Standards-based interaction

The web services access control service architecture needs to accommodate standards-based interactions by using standards-based message formats and policy specification languages that can be understood and processed by both web services requestors and providers.

Attribute-based access control

The web services access control service architecture needs to accommodate attribute-based access control by including a decision-making component that grant subjects access to resources based on attributes that are presented on their behalf by web services requestors. Attributes are presented to the decision-making component in a format that it understands.

Trust levels

The web services access control service architecture needs to support trust levels for web services requestor by a component that collects information and evidence, and continuously computes trust levels of web services requestors.

8.3 CONCLUSION

This chapter described a number of access control and trust architectures to establish a potential approach for the web services access control service architecture. Two important themes emerged from this discussion.

Firstly, it is apparent that policy-based management is becoming more important for environments such as web services. To address access control requirements, this research extends the policy-based approach by means of a policy that publishes access control and trust requirements and capabilities, to seamlessly integrate access control functionality between web services providers and requestors. The selective publication of the policy must be addressed so that the web services provider is not compromised.

Secondly, in trust architectures a move is made away from centralised trust formation – through the use of trusted parties – to environments where trust is managed in a decentralised, autonomous manner. In support of this trend, the web services access control service architecture employs a trust component that creates and manages a trust level for each requestor in order to make better access control decisions.

The current chapter concludes Part I, which constitutes the background discussion and identification of important concepts for this thesis. Next follows Part II, which commences with the model for the web services access control service.

PART II

9

The *WSACT* model an overview

Part I provides a background, critical overview and development of important concepts for this thesis. Firstly, access control requirements for web services environments are highlighted. A background on existing access control models and mechanisms is given, in order to identify the extent to which they can answer to the needs of the web services access control service. An analysis of access control requirements shows that the web services access control service is not satisfactorily addressed by current access control mechanisms. An important requirement to be addressed is the calculation of a trust level for web services entities. In this thesis, this requirement is discussed orthogonal to attribute-based access control. To provide a basis for the development of the model, a trust assessment framework is defined to allow a web services provider to calculate a trust level by reasoning over information and evidence. To complete the background discussion, policy specification languages for access control publication and reasoning, as well as architectural components required by the web services access control service are discussed.

Part II of this thesis sets out to develop an access control model to meet access control requirements by incorporating trust with access control – in one model. To introduce the proposed model, the present chapter gives an overview of the model titled **Web services Access Control incorporating Trust**, henceforth called the **WSACT** model.

9.1 WSACT - DESIGN MOTIVATION

In order to be able to answer the question “*Which request does web services provider P grant to subject S – on whose behalf web services requestor R is acting – if trust in R is low?*”, the WSACT model implements a number of components, namely:

- A component positioned at the perimeter of web services provider P, which manages all interactions related to trust information collection and access control with web services requestor R.
- A component internal to web services provider P, which makes access control decisions based on access control and trust policies.
- A component internal to web services provider P, which supports trust rules by calculating a trust level for web services requestor R over information stored in an information database.

A general overview of the main components of the model and their relation is depicted in Figure 9.1. The architecture consists of an authorisation interface, an authorisation manager and a trust manager. Together, these components constitute the access control service. The access control service should exhibit properties of an important access control concept - the reference monitor. To be considered a reference monitor, the access control service should be complete, isolated, and verifiable (Shirey 2000). Complete means that all requests are mediated by the access control service, isolated means that the access control service cannot be modified by other system entities, and verifiable means that it is small enough to be subjected to analysis and tests to ensure that it is correct.

In complex, distributed environments these properties cannot totally be addressed. For instance, the access control decision made by the access control service may not immediately be enforced, but may become part of an orchestrated access control decision made at a higher level of access control abstraction. The trusted implementation of the reference monitor as a security kernel is now replaced by the implementation of the access control service that is supported by encryption schemes and digital certificates. This ensures that components and communication between components are safeguarded. The access control service should thus carefully be secured so that it cannot be compromised in any way.

It is important to note that the model does not address the administration of policies. This would be an important feature of a comprehensive solution.

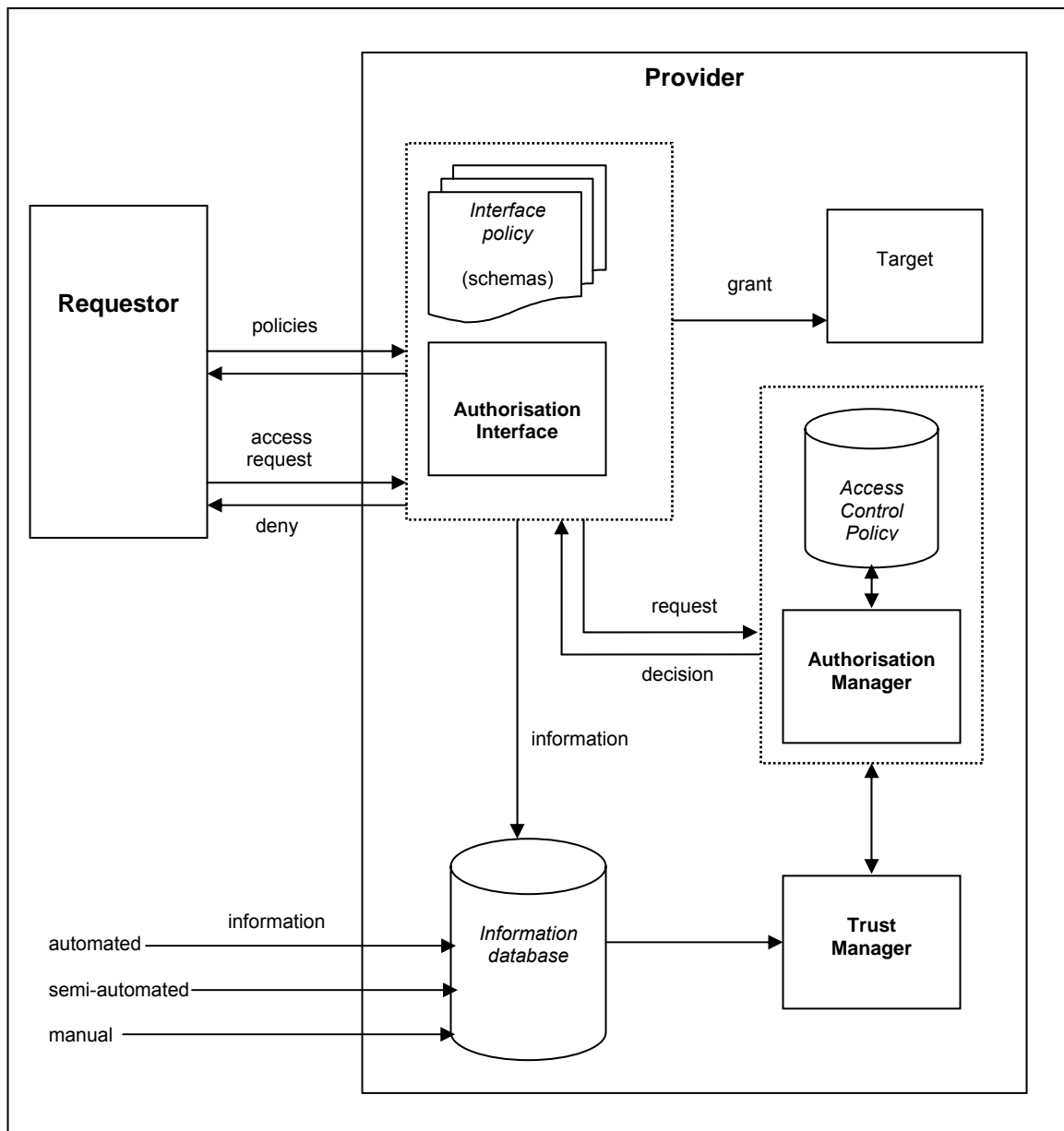


Figure 9.1: WSACT components

Requests sent to web services providers are formatted according to the functional description as defined by the interface document and the non-functional description found in the interface policy. The authorisation interface intercepts all requests. It inspects the header and body of the SOAP request and may optionally store information in the information database. It invokes the authorisation manager for an access control decision. Based on the latter's answer, requests are either granted or denied. In the background, the trust manager calculates a trust level for each web services requestor. Next, each component is described in more detail.

9.1.2 The authorisation interface

The authorisation interface is the first point of contact at the web services provider, as any requests sent to it are first intercepted here. It is an intermediary in the access path between a web services requestor and web services operation being requested. It is similar to the PEP as defined by the IETF policy-based access control framework. It thus enforces the access control decision. The authorisation interface is a highly specialised component that extends the functionality of a PEP by controlling not only access to web services operations, but also interactions related to the collection of trust information. The authorisation interface collects information and evidence that accompanies requests and stores it in the information database. It controls interactions between web services requestors and providers related to the exchange of policies.

It acts as a shield for the web services provider, as it attempts to ensure that only valid requests are passed to the authorisation manager. It formats access requests for the authorisation manager in syntax that the authorisation manager can understand. Additional credentials or declarations that are required when the decision is made are also formatted so that the authorisation manager can use them. The authorisation manager is contacted to determine whether the request may be granted. Based on the answer that is returned from the authorisation manager, the authorisation interface either passes the request to the web services operation to be executed or returns an error message to the web services requestor as described by SOAP faults in the interface document.

9.1.3 The authorisation manager

The authorisation manager constitutes the core component of the WSACT model, and is at the centre of decision making about access control policy. Because the authorisation manager is a very sensitive resource, it is made more secure by locating it away from the environment where SOAP requests over the Internet are received. It is similar to the PDP as defined by the IETF policy management architecture. The functions of policy decision making and policy enforcement are separated, as defined by the IETF framework for policy-based access control.

The authorisation manager bases all its decisions on access control and trust policies, as well as on requests that it receives. A decision of either grant or deny is returned to the authorisation interface. As required, the authorisation manager contacts the trust manager to determine the level of trust of the web services requestor.

The authorisation manager consists of two parts: an access control policy and an inference engine. Logical facts and rules define the access control policy. The access control policy extends role-based access control mechanisms to grant access to web services requestors

based on their level of trust. Subject attributes are further be used to grant access to web services operations.

9.1.4 The trust manager

The trust manager is a component not identified by the IETF framework for policy-based access control, and consists of two components: a fuzzification and a trust inference component. It bases its computation of a trust level on information contained in the information database.

Figure 9.1 indicates that the information database is populated from four different sources. Records are written by the authorisation interface, automatically by application entities, semi-automatically by a combination of application entities and human intervention, and manually by human intervention. The fuzzification component evaluates information found in the information database to produce fuzzified trust concepts in the range $[0, 1]$. These values are used to populate the nodes of a Fuzzy Cognitive Map, defined for each web services requestor. The trust inference component is dynamically invoked by the authorisation manager as it processes a request. It is passed the identification number of the web services requestor, and returns its trust level. Time-stamped information in the information database is removed when it has expired.

9.2 CONCLUSION

This chapter has presented a conceptual view of the WSACT model. It identifies interactions that occur between components of the model. The next three chapters are dedicated to the detailed specifications of the model.

Chapter 10 is dedicated to the first component of the web services access control service, namely the authorisation interface. The authorisation manager – the core component of the WSACT model – is subsequently described in Chapter 11. A description and discussion of the trust manager, which determines the trust levels of web services requestors, follows in Chapter 12.

Some theoretical aspects of the model are finally demonstrated in Chapter 13, and the development of a prototype based on the WSACT model is discussed.

10

WSACT

The Authorisation Interface

The development of the WSACT model commences by considering the first component, namely the authorisation interface. As mentioned before, the main functions of the authorisation interface are to manage interactions related to trust information, and to intercept all access requests and enforce access control decisions.

Initially, policies and information is sourced from web services requestors and other parties, where possible, in order to assess the environment in which the web service requestor and provider function. Subsequent interactions are continuously monitored for trust evolution by inspecting messages for new evidence.

For subsequent requests made to web services operations, information is extracted from messages for the purpose of formulating access requests. Access requests are presented to the authorisation manager, and resulting decisions are enforced.

This chapter considers the functionality of the authorisation interface. The first part of this discussion entails the role of the authorisation interface in collecting information for trust formation. The design of the interface policy is described with components that use and source information and evidence. Next, the authorisation interface is described in terms of the functionality that it provides for access control.

10.1 THE AUTHORISATION INTERFACE

Figure 10.1 depicts the authorisation interface and related data components in blue. The authorisation interface makes use of the interface policy to regulate interactions, updates the information database, and makes calls to the authorisation manager.

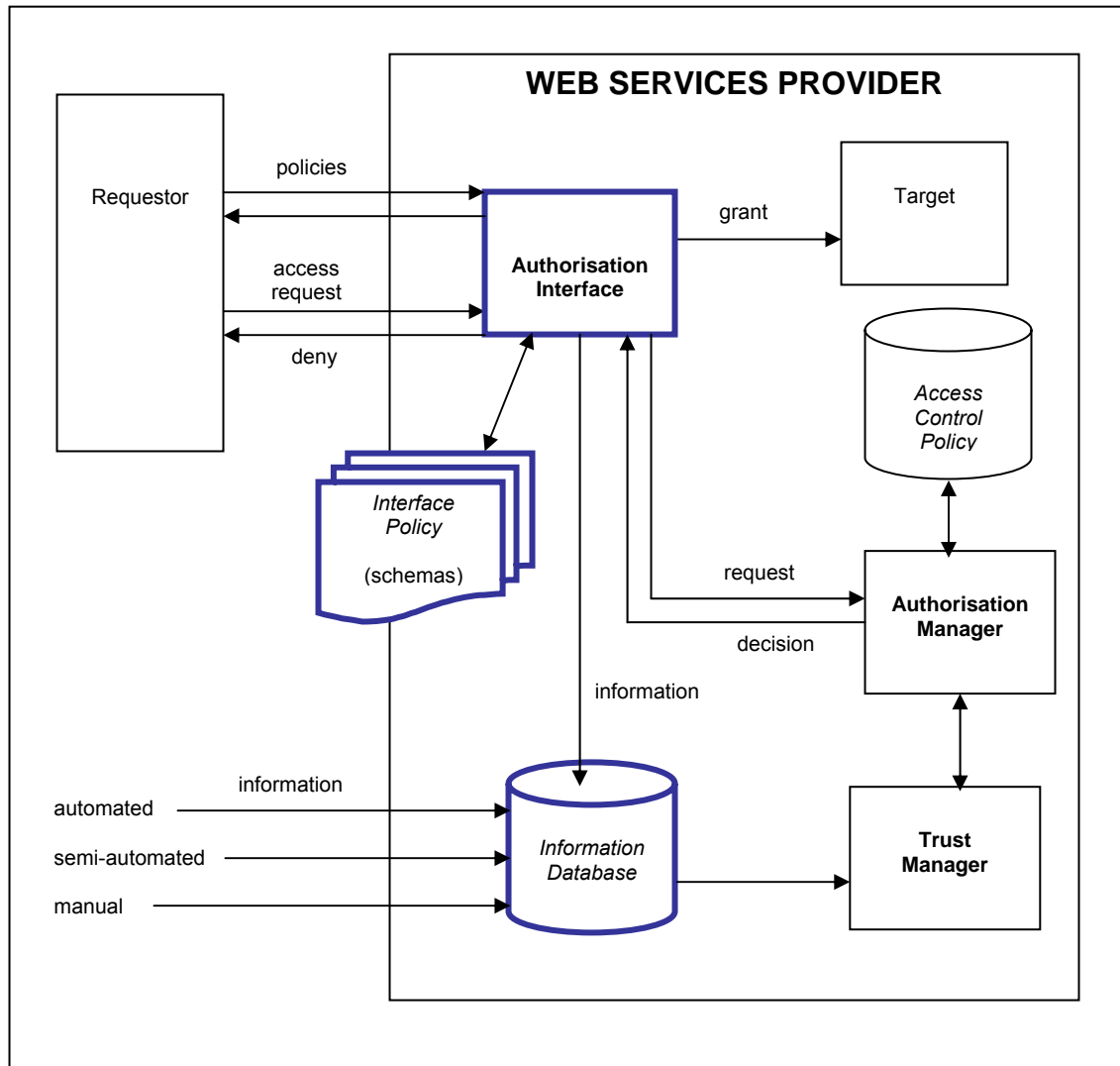


Figure 10.1: WSACT - the authorisation interface and related components

To create a complete and consistent definition of components of the WSACT model, a formal specification needs to be developed. This ensures the clarity and precision of the description of the system design. This is described in the next paragraph.

10.1.1 WSACT model specification in Z

A variety of formal specification languages can be used to define the WSACT model, such as the B notation (Abrial 1996), Z (Jacky 1997), VDM Specification Language (VDM-SL) (Jones 1990), and SDL (Specification and Description Language) (SDL Forum 2005). Each language is designed to serve a specific purpose. For instance, B is more focused on refinement to code than just formal specification. Z is found to be a popular specification language that is used in industry (Bowen & Hinchey 2005), as it can be used for mathematical modelling and unambiguous transmission of ideas between designers. The Z specification is well established, and has been adopted by ISO (International Standards Organisation) as an international standard for software specification (ISO 2002).

For this reason, all three components of the WSACT model are formally specified with the Z specification (Spivey 1992). Z is based on typed set theory and first-order predicate calculus (Jacky 1997). As the semantics of Z are mathematical and not computational, it is a powerful, flexible and extensible notation. A Z specification describes the state of the system, and operations that can change the state. System properties are modelled using set theoretic concepts and the necessary pre- and post-conditions for each system operation, or change of state, are stated explicitly. The specification describes what the system has to do, rather than how it is to be done. Because there is no built-in correspondence between the meaning of a Z specification and a computational model, the specification of a system in Z is a matter of convention.

The basic building block in any Z specification is the schema (Spivey 1992). Schemas provide structure for a specification and describe system operations and permissible system states. The schema calculus allows more complex components of a specification to be built from a set of previously defined and simpler schemas. Jacky (1997) and Spivey (1992) can be referred to for a comprehensive discussion of Z.

A convention that can be used for the specification of a system in Z is known as the Established Strategy. The Established Strategy (Barden et al. 1994) specifies a system as a state plus a collection of operations, each operation defined by a schema. It proposes that a Z specification be presented as follows:

1. Define all global constants and basic types, and give a natural language description of these.
2. Present the abstract state space, using the constants and basic types that have been defined.
3. Give an initial state of the system, and prove that such a state exists.

4. Introduce partial definitions of each of the system operations. Calculate the preconditions of the abstract operations on the state.
5. Draw up a table showing all the partial operations together with their inputs, outputs and pre-conditions for correct operation.
6. Define all schemas that present error conditions.
7. Use the Z schema calculus to make all partial operations complete.

For the purposes of the specification defined here, steps 1, 2 and 3 of the Established Strategy are followed. The design of the authorisation interface is now considered in Z.

10.1.2 Basic types

Following the first step of the Established Strategy, all global constants and types are defined. They are also described to clarify their use. Four categories of types are defined for initial trust formation, trust evolution, transaction trust and access control respectively.

10.1.2.1 Types for initial trust formation

The discussion that follows now refers to and focuses on the case study described in Chapter 3. It is assumed that both eLoans and eBooks expose information to each other in XML-based policies according to the structure of the interface policy. It is important to note that published interface policies do not provide a policy negotiation solution, but can merely be used to support such a process. A very naïve approach would be to expose all access control, trust and other requirements and capabilities to any potential web services requestor. Instead, negotiation based on trust levels of web services requestors is proposed. The publication of sensitive policies is controlled so that web services requestors are given access to sensitive policies according to the level to which they are trusted. Unknown web services requestors will initially not have access to interface descriptions of sensitive operations, or of sensitive access control requirements.

A first attempt is made by eLoans to incorporate eBooks into its application domain when it registers. Initial interactions that take place are of an introductory nature. In these interactions, functional and non-functional requirements and capability policy documents are exchanged and compared. eLoans may establish an initial level of trust in eBooks that enables it to commence with retail transactions for books. In the registration process, eLoans is identified by an identifier of some sort.

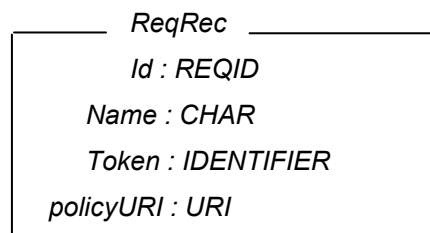
Web services requestors may be identified by username and passwords, tokens, or public keys. It is important that web services requestors are rather identified by public keys, as they can freely generate a public key pair that can be used to identify them. Keys cannot always be verified by

trusted third parties and web services providers may have to use their discretion when deciding to accept such public keys. If a public key of a web services requestor is accepted, the trust in it may initially be at a low level, but this may become higher as time passes and positive experiences are recorded. The use of a public key ensures that the identity and integrity of a web services requestor can be maintained. All requests and information are signed by the private key of the web services requestor. They would therefore be expected to ensure the safeguarding of their private keys. If a public key pair is used, the integrity and confidentiality of SOAP messages can be ensured by implementing a solution based on SSL.

The following basic types are defined after eLoans registers:

- *[REQID]* is a basic type for referring to a web services requestor.
- *IDENTIFIER :: = username, password | token | public key* is a basic type for authentication of web services requestors.
- *[URI]* is a basic type that is used to refer to the address of policies of web services requestors such as eLoans. This value is sent to eBooks as a parameter of the *registration* process.

The authorisation interface creates a record in the information database for each web services requestor as follows:



This information is used to form the initial trust level, as the identity of eLoans is known. The identity is not of high assurance, and the trust level calculated on the basis of this information reflects a trust level that is low. eLoans therefore needs to increase its trust with eBooks before it can be granted access to protected resources.

10.1.2.2 Types for trust evolution

In order to increase its trust with eBooks, eLoans needs to understand which information it should present to eBooks. The types defined by the interface policy are now considered. Types are domain dependent and the definition is only a possible structure of an interface policy.

[INTERFACEPOLICY] is a basic type that consists of a number of policy sections.

INTERFACEPOLICY == *seqPOLICYSECTION* where

POLICYSECTION ::= *ID* | *partners* | *recommendations* | *references*

The aim of each section of the interface policy is to state assertions on how trust can be formed.

[ASSERTION] is a basic type that is used to define each *POLICYSECTION*.

- *ID : ASSERTION*

ID identifies the policy uniquely. This enables the WSDL document to point to a specific policy. For example, P#123 identifies a policy of eBooks uniquely.

- *PARTNERSHEMA = = seqASSERTION*

Partner is the list of trusted partners that is published to inform web services requestors of their existence. The set of assertions {Partner#, CompanyName, CompanyURI, Signature} publishes the structure of required *PARTNER* information. It should be signed by eBooks to protect its integrity.

PARTNER = = *seqASSERTION*

Based on the given structure, a number of sets such as {P212, eCompanyABC, www.ABC.com, } are published by eBooks to give the identifier, name and location of their partners.

- *RECOMMENDATIONSCHEMA = = seqASSERTION*

A recommendation is made by a trusted partner and is signed. For a recommendation, additional types that may be defined are the context and value of the recommendation.

CONTEXT ::= *timely payment* | *correctly functioning application* | *efficiency* represent possible contexts in which recommendations can be made.

VALUE ::= *ignorant* | *low* | *moderate* | *good* | *high* are the set of possible value that can be used in a recommendation.

The set {RecommNo, IssuedBy, IssuedFor, Value, Context, CreationDate, ExpiryDate, Signature} publishes the structure of recommendations expected by eBooks.

RECOMMENDATION = = *seqASSERTION*

The set {Rec#1, eCompanyABC, eLoans, good, timely payment, 2005/07/15, 2005/12/31, VVXV45S88VSV} is a recommendation issued by Company ABC about eLoans which states that eLoans is good with timely payments. The date of creation, expiry and signature are added to the end of the set.

- *REFERENCESCHEMA = = seqASSERTION*

A reference is a statement that is made by a trusted party about a web services provider or requestor. eBooks publishes a list of references or sends them with messages to

eBooks so that eBooks can establish their authenticity in order to use them in trust formation. References can be categorised according to the following types:

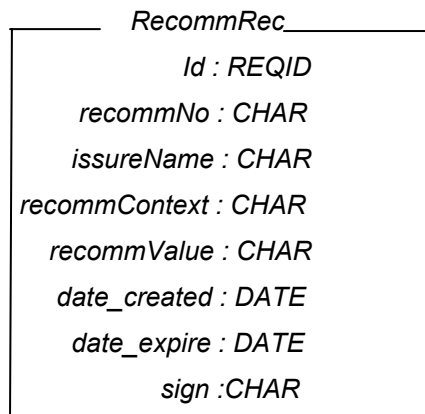
REFTYPE ::= audit information | credibility | insurance | best practise | endorsements

The set {RefNo, RefType, RefURI, Issuer, CreationDate, ExpiryDate, Signature} publishes the structure of how recommendations can be located by eBooks. References are signed by the issuing authority.

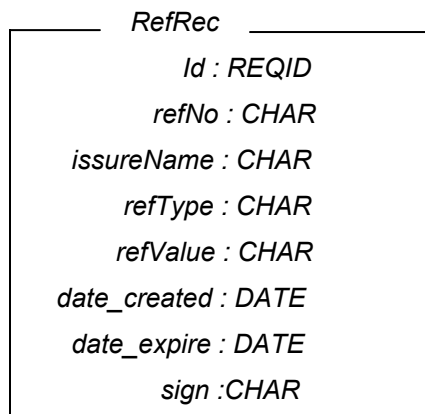
REFERENCE = = seqASSERTION

The set {Ref#5, endorsement, www.eBooks.com/reference#22, Authority CA, 2005/06/21, 2006/06/20} is a fifth type of reference, an endorsement, that is located at www.ABC.com/reference#22 and made by Authority CA, and it is valid between the dates specified.

The interface policy is accessed by eLoans in order to understand how to increase its level of trust. eLoans uses the list of published partners to source recommendations. For instance, if eLoans knows eCompanyABC, it can request recommendations from them to submit to eBooks. Recommendations can be sent with requests, to be added to the information database. A recommendation is stored in the information database by means of the following structure.



A reference is stored in the information database with the following structure.



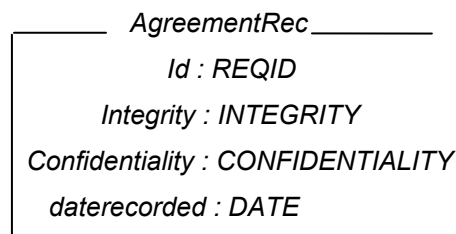
10.1.2.3 Types for trust in transaction security

The interface policy publishes requirements for transaction security such as integrity, confidentiality, privacy and service-level agreements. These requirements need to be adhered to by web services requestors to allow trust to be increased. Each message is inspected to determine its compliance to requirements.

For this discussion, integrity and confidentiality is considered, although many other requirements may be included as required. The following basic types are defined:

- *[REQID]* is defined as a basic type to refer to a web services requestor.
- *[INTEGRITY]* :: = *none* | *xmlsig#sha1* | *xmlsig#base64* | *xmlsig#hmac-sha1* | *xmlsig#rsa-sha1* are algorithms that may be used, but *xmlsig#sha1* may be preferred.
- *[CONFIDENTIALITY]* :: = *none* | *xmlenc#tripledes-cbc* | *xmlenc#aes128-cbc* | *xmlenc#rsa-1_5* are algorithms that may be used, but *xmlenc#aes128-cbc* may be preferred.

A record is created in the information database for each web services requestor, with the date on which it is recorded.



10.1.2.4 Types for access control

Types required for access control are now considered. Firstly, access control requirements are published in the *INTERFACEPOLICY* and are defined as follows:

SUBJCREDENTIALSSHEMA = *seqASSERTION*

The access control requirements are the subject credentials, defined as attributes, that are required in order to gain access to web services operations. The attribute set {ID, type} specifies that the identity of the subjects has been verified, and the type of subject. These attributes may be the access control requirements of the *order* operation.

SUBJCREDENTIALS == *seqASSERTION* is a type used to specify the attributes of the subject as they occur in the SOAP message.

The SOAP message is intercepted and its header and body inspected for information.

- The body contains a type, *OPERATIONNAME*, that is used to refer to the web services operation that is accessed. It is defined by the set $\{operationname, parameters\}$.
- The header is defined by the set $\{reqid, identifier, subjattributes, recommendation, reference\}$.

To determine if a request can be granted, the authorisation manager is invoked. It returns a result that is defined as follows:

ACCESSDECISION ::= grant | deny

If the request is denied, the authorisation interface returns a fault to the web services requestor as follows:

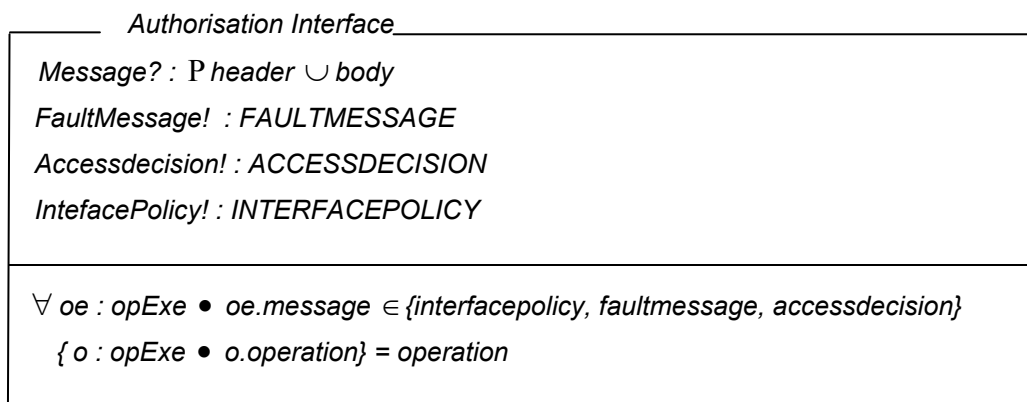
FAULTMESSAGE ::= requestdenied | invalidrequest | invalidparameter

A possible materialisation of the interface policy has been proposed by the researcher (Coetzee & Eloff 2005)

10.1.3 The abstract state space of the authorisation interface

State changes that need to be modelled include the interception of the SOAP message, the processing of its content, and return of a message to the web services requestors as required. The interface policy is a source of information required for these changes. The abstract state space of the authorisation interface is considered as a collection of data. The state of the system changes when operations are executed that transform data.

A set of $\{OPERATION\}$ can be used to refer to all operations. The result of these executions is the requested interface policy, an access decision, or a fault message.



10.1.4 Authorisation interface operations

The message is first evaluated by the *Evaluate Message* operation (shown in Figure 10.2) to determine its content. Next, three operations are performed. If the message is a request for an interface policy, the *Process Policy* operation determines the trust level of the web services requestor before exposing the relevant policy. If the header of the message contains trust information, the *Process Trust Info* is performed. Finally, the *Access AM* operation is invoked. All these operations are defined below.

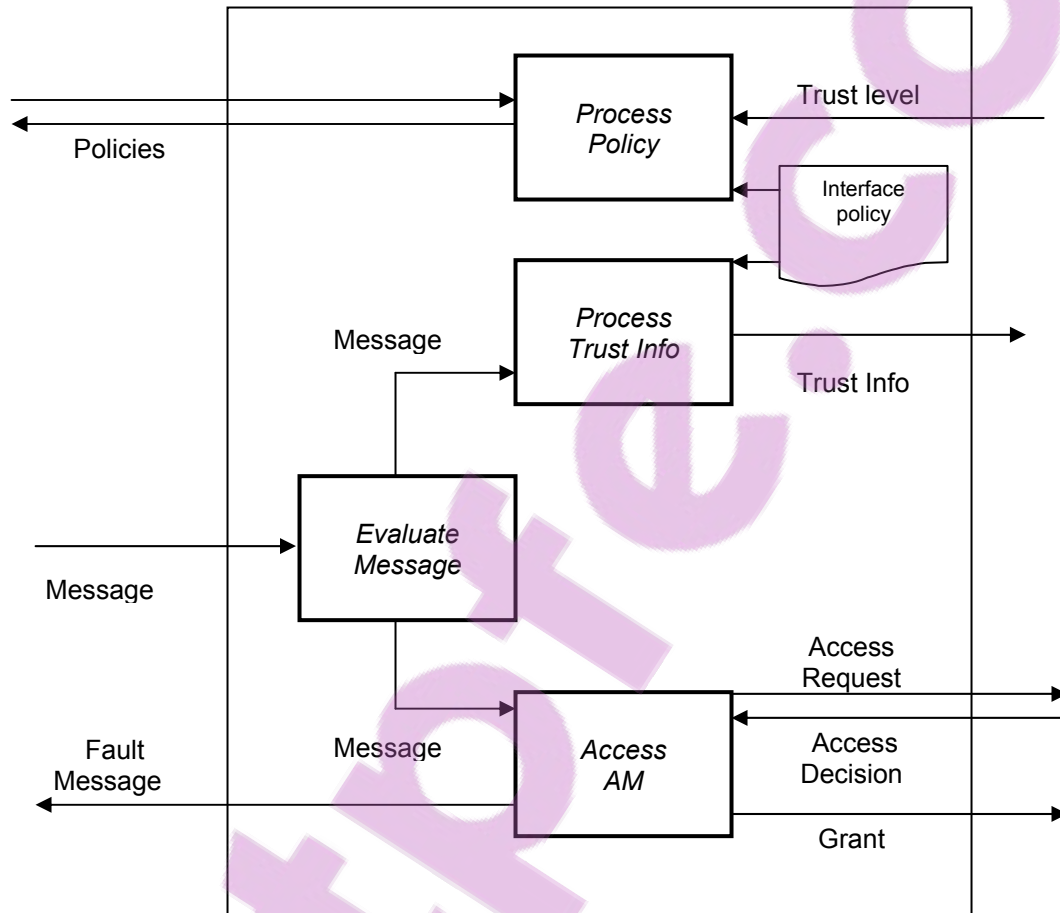


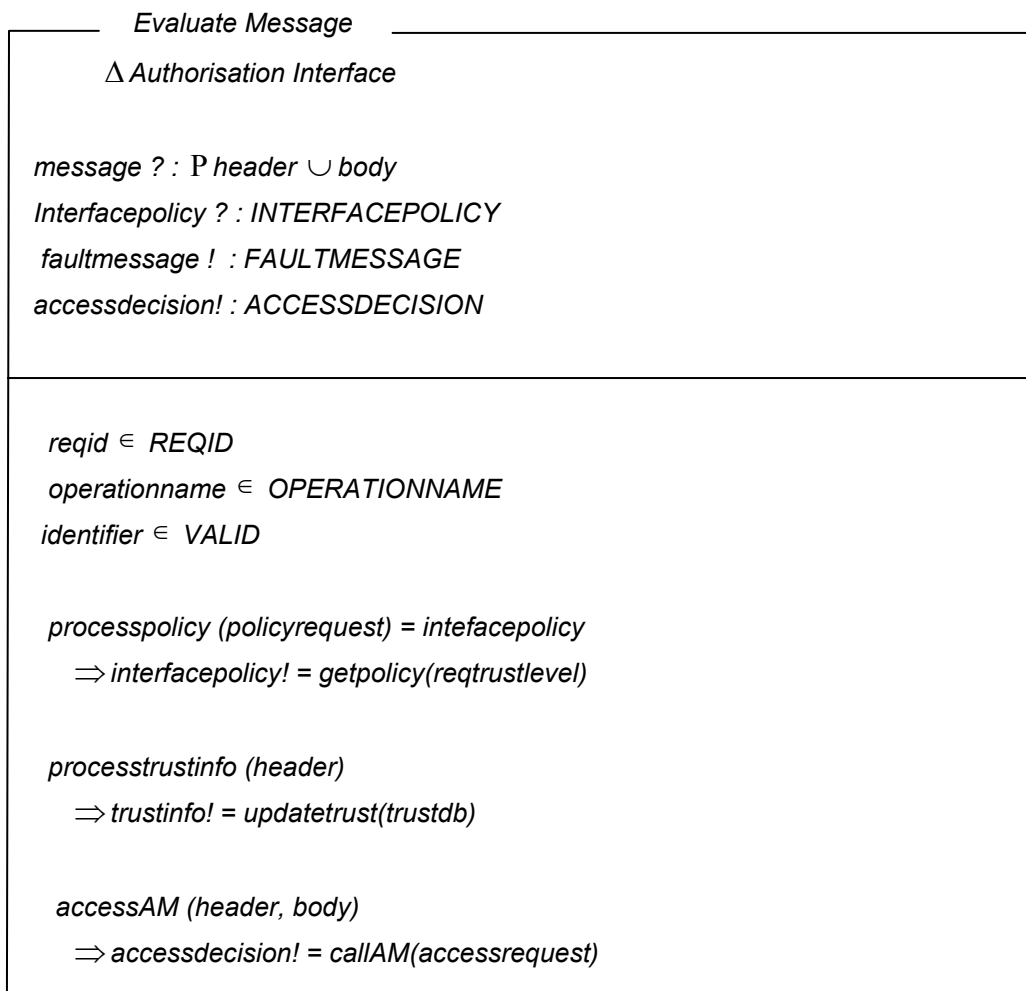
Figure 10.2: Authorisation interface operations

Functions such as validation of message fields and verification of public keys are performed, but are not discussed here.

10.1.4.1 Evaluate Message

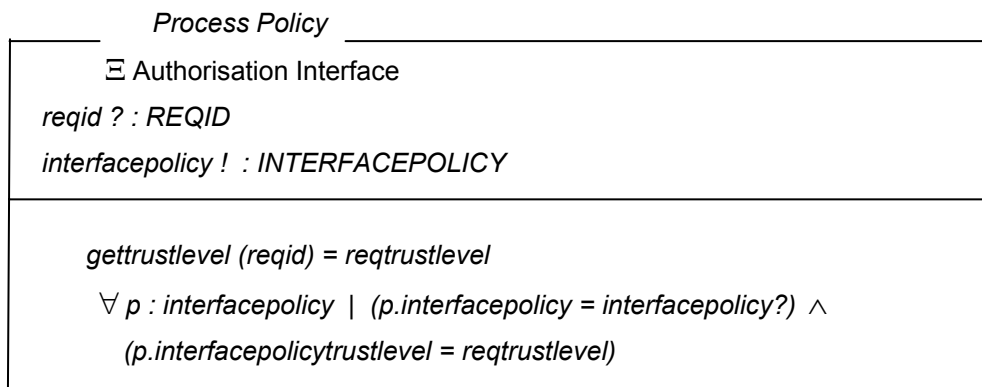
Evaluate Message intercepts the message and inspects both the header and body. A number of checks are made here. Firstly, the web services requestor is authenticated. The integrity of the message that is received is determined. The XML parser checks for well-formed and valid elements. Further checks are made to ensure that no malicious information is introduced, such as invalid parameters. Accompanying identifying keys are checked to ensure their authenticity.

If all checks are successful, *Process Policy* and *Process Trust Info* are invoked as required and the message is passed on. Lastly, *AccessAM* is invoked. If the access decision that is returned is deny, a fault message is returned to the web services requestor. Otherwise, the request is passed on to the web services operation.



10.1.4.2 Process policy

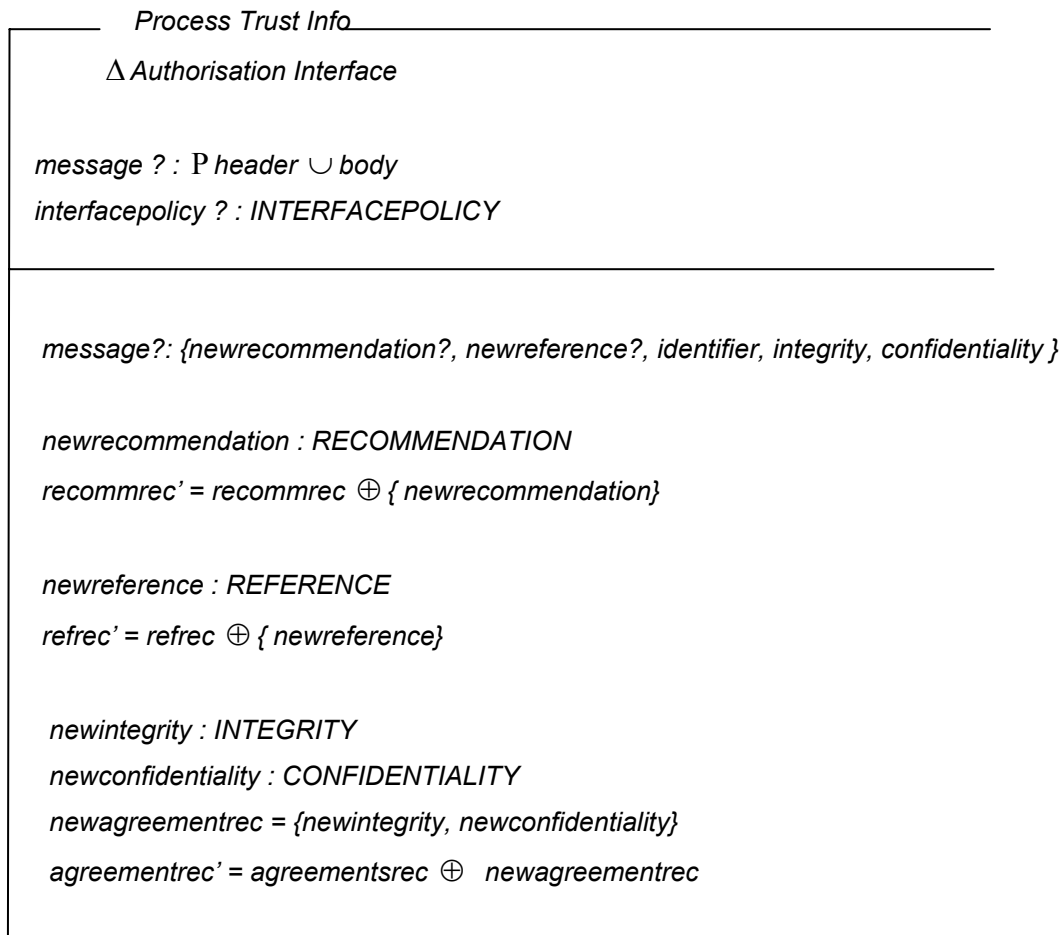
Process Policy controls the exchange of policies according to a protocol. Policies are selectively published to web services requestors according to their trust level. Publication of sensitive policies is private of nature. As trust levels of web services requestors increment, both functional and non-functional policies with the same trust level are exposed to grant web services requestors the ability to access operations of more sensitive nature. The URI of the policy or the policy is sent to the web services requestors as is required.



10.1.4.3 Process Trust Info

Process Trust Info receives the message and firstly checks the header for different types of information. A recommendation is inspected for validity, after which it is time stamped and saved in the information database. Reference information is inspected and used to access the reference from the specified URI. The reference is processed according to predefined rules, categorised, time stamped and stored in the information database. Any other information that is presented but that cannot be understood, is discarded.

The next phase of trust processing determines agreement with the requirements specified in the interface policy. The message is inspected to determine whether it complies with required security service requirements. In this process, the interface policy is consulted to determine requirements. If, for instance, specific elements of the messages must be encrypted with either 3DES or AES, but AES is preferred, and the message comes encrypted with 3DES, this information is recorded in the information database. Similar checks are performed for authentication, integrity and any other applicable services such as privacy, or other features such as SLAs (Service Level Agreements).

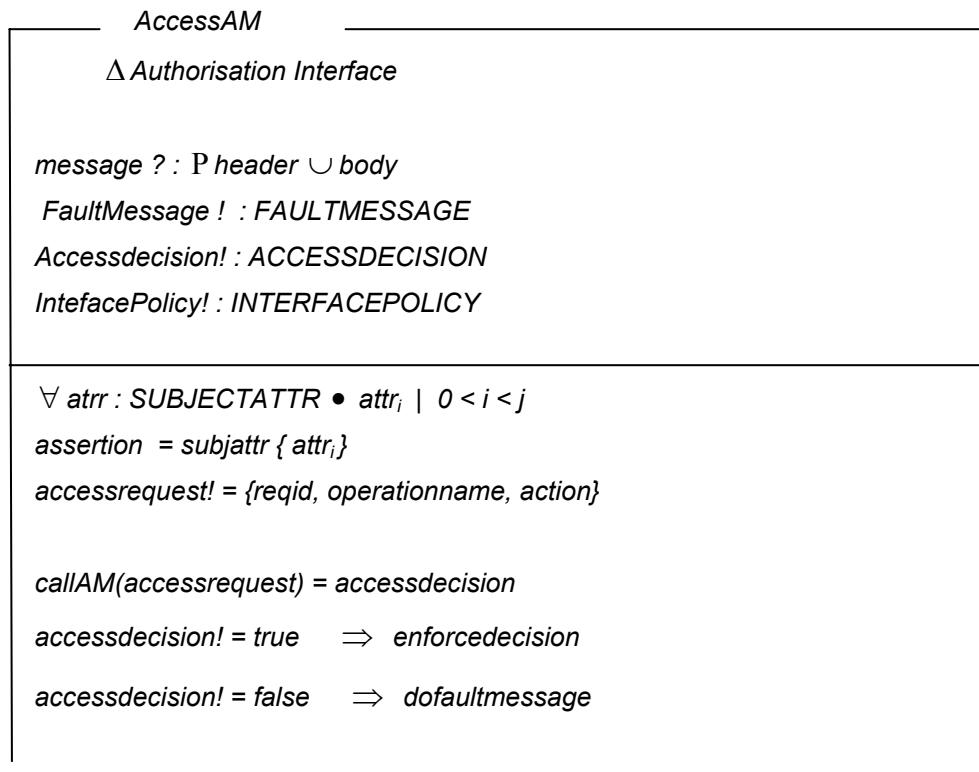


10.1.4.4 AccessAM

The final step is to invoke the authorisation manager with an access request. This is the task of the *AccessAM* operation. The SOAP message header and body are inspected to format an access request. The following elements from the message are used:

- The *reqid* on behalf of whom the request is made is found in the header.
- The *operationname* for which the request is made is found in the body.
- The *action* to be performed on the operation is determined. For most web services requests it would be execute.
- *SubjectAttributes* found in the header represent the subject making the request. Subject attributes are verified before they are passed to the authorisation manager.

An assertion asserts the subject attributes of the subject to the access control policy as required. An access request is passed to the authorisation manager. It returns a decision. If the request is denied, the *Access AM* operation returns a fault to the web services requestor, otherwise the web services operation is invoked.



10.2 CONCLUSION

The authorisation interface, a specialised access control component, addresses two important aspects: first of which are policy publication and trust information collection, the second are access control enforcement. This is an important difference from the approach followed by PEPs as defined in the IETF architecture.

The publication of information based on trust levels of web services requestors lessens the burden on trust negotiation over subject attributes. The approach defined here rather negotiates the publication of sensitive policies about the trust level of a web services requestor. Normally trust is created incrementally by the exchange and verification of sets of attributes defined in credentials. As trust increases, more and more sensitive resources are exposed. There is a high administrative burden to verify the credentials of each and every subject for each request that is made, and to create trust that exists for a single session between the subject and web service provider. Such an approach does not mirror the real world, where very often one is not only trusted as an individual, but also as a member of a community or organisation. The experience of past interactions also influences the subsequent decisions taken with regard to sensitive information.

The second important aspect addressed here is that of access control enforcement. An access request is formulated containing both the identity of the web services requestor and the subject attributes that represent the subject's ability. An access control decision must be made by the authorisation manager on the basis of these parameters. The obvious question is then how the trust level of a web services requestor and the attributes of a subject are used together when access control decisions are made. The next chapter focuses on this question.

11

WSACT The Authorisation Manager

The authorisation interface invokes the next component of the WSACT model, namely the authorisation manager. It is invoked by an access request that is formulated by the authorisation interface. Two main aspects are to be considered by the authorisation manager in order to process the request: subject attributes and web services requestor trust levels.

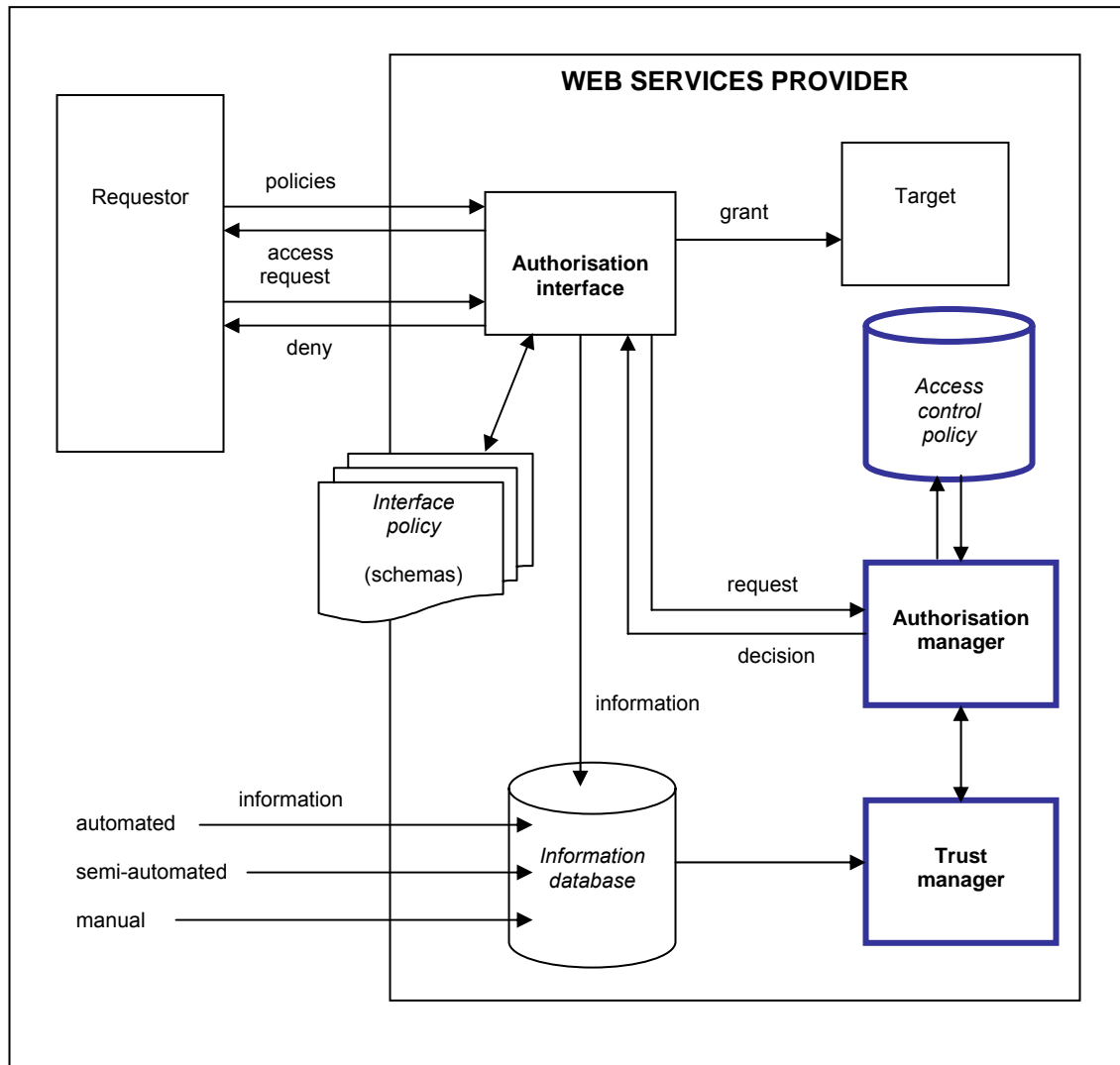
Access control models previously discussed do not address trust levels. In the authorisation manager, statements are to be included on web services requestor trust levels. This can allow a web services provider to grant advanced access to subjects of trusted web services requestors, rather than to subjects who make requests through web services requestors with whom a minimal level of trust has been established. Such flexibility gives a web services provider the ability to foster meaningful business relationships that portray humanistic forms of trust.

The access control language used to define the access control policy is based on Datalog (with constraints). It uniquely addresses attribute-based access control and trust levels in conjunction with each other. The access control language is small, and it has formal semantics for query evaluation and for access control reasoning.

The first section of this chapter identifies components that relate to the authorisation manager of the WSACT model. After that, the role of trust levels in access control decisions for the eBooks case study is described. This is followed by a discussion of important concepts of the authorisation manager of the WSACT model, during which rules and policies are defined by set, relations and predicates. The specification of the authorisation manager is finally given in Z, and the chapter is concluded.

11.1 THE AUTHORISATION MANAGER

Figure 11.1 depicts the authorisation manager and related components in blue. The authorisation manager reasons over rules defined in the access control policy. The trust manager is interrogated for the trust levels of web services requestors.



The access request is passed to the authorisation manager. It contains the name of the operation to be accessed, and the identity of the web services requestor making the request. Subject attributes used in the evaluation of the access request are added to the set of facts in the access control policy before an access decision of grant or deny is derived. Next, access control considerations for eBooks are identified before a formal definition of the access control policy is arrived at.

11.2 ACCESS CONTROL ADDRESSING ATTRIBUTES AND TRUST LEVELS

The authorisation manager answers the question “Which resources may a web services requestor *R*, acting on behalf of a subject *S*, access, if trust in web services requestor *R* is low/high?” The case study referred to earlier is revisited to add more detail to this question.

Figure 11.2 is an adapted version of Figure 4.2 that appeared in Chapter 4. The figure now includes eLoans as the web services requestor, and eBooks as the web services provider. The following considerations for access control for the web services provider, eBooks, were identified:

- A subject *S* delegates digital credentials, defined as sets of attributes, to eLoans. This could be in the form of an attribute certificate that is issued by the academic institution where the student is registered. The certificate may contain attributes such as student number, institution name, degree registered for, date of last registration. Cryptographically verified credentials such as these are trusted where the nature of this trust is binary – it either exists or not.
- A trust relationship exists from eBooks to eLoans. This relationship is manifested by a trust level that is calculated on the basis of information that has been sourced over a period of time.

Fundamental to access control for web services is attribute-based access control and delegation. This is highlighted by the first consideration mentioned above. The authorisation manager of the WSACT model addresses the trust relationship between eLoans and eBooks as a further consideration for autonomous access control decision making, even though the credentials of a subject are trusted. It may even be possible to not require subject attributes, but to base an access control decision only on the level of trust, or vice versa.

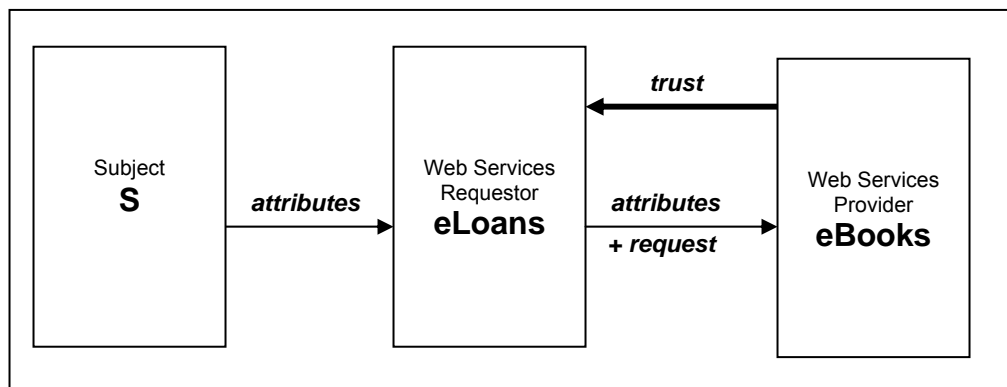


Figure 11.2: Access control entities for web services

Next, the role of trust levels are described as they relate to attributes, operations and web services requestors.

11.2.1 Web services operations, attributes and trust levels

Table 11.1 highlights the required trust level of the web services requestor, and the attributes the subject needs to possess to be granted access to an operation. It illustrates how trust levels and attributes are used together when access control decisions are made.

Table 11.1 Operations and required trust levels and subject attributes

Operation	WS requestor trust level	Subject attributes
Search	Low	Type of subject
Search-academic	Low	Student number, institution name, year of study and date of last registration
Order	Moderate	Identity, credit level
Make_payment	Moderate	Identity, position
List_specials	High	

The *search-academic* is an operation available to subjects who are students from recognised academic institutions. It illustrates that in some cases, the trust in the web services requestor is not as important as the attributes that the subject must possess in order to be granted access to the web services operation.

On the other hand, the *List_specials* operation is reserved for the subjects of web service requestors with a “high” trust level. Because a high trust level is present, subjects do not have to present credentials containing attributes to be granted access. *List_specials* is an operation that enables subjects to search for special offers on popular books that cannot be made available to the overall population of possible subjects, but that is reserved for the subjects of trusted partners. This means that a strong business relationship exists with such web services requestors, and it is fostered for future interactions. It reflects the true nature of business relationships where partners act according to their disposition towards each other. Disposition is reflected by a calculated trust level, where “high” trust equates to the existence of goodwill.

The *Order* operation illustrates that a “moderate” level of trust is required as the responsibility for the order lies with the web services requestor, who is liable for order payment. In addition, a signed assertion needs to be made on the identity of the subject, and its creditworthiness to be able to access this operation.

11.2.2 Web services requestors and trust levels

To illustrate the role of the trust levels of web services requestors in access control decisions, consider the following:

- eLoans is a trusted partner of eBooks. It has maintained a good relationship with eBooks over a number of years. eBooks has developed goodwill towards eLoans. The trust level of eLoans is “*high*”.
- eCompany is web services requestor who is not known to eBooks. eBooks only knows the identity of eCompany. The trust level of eCompany is “*low*”.

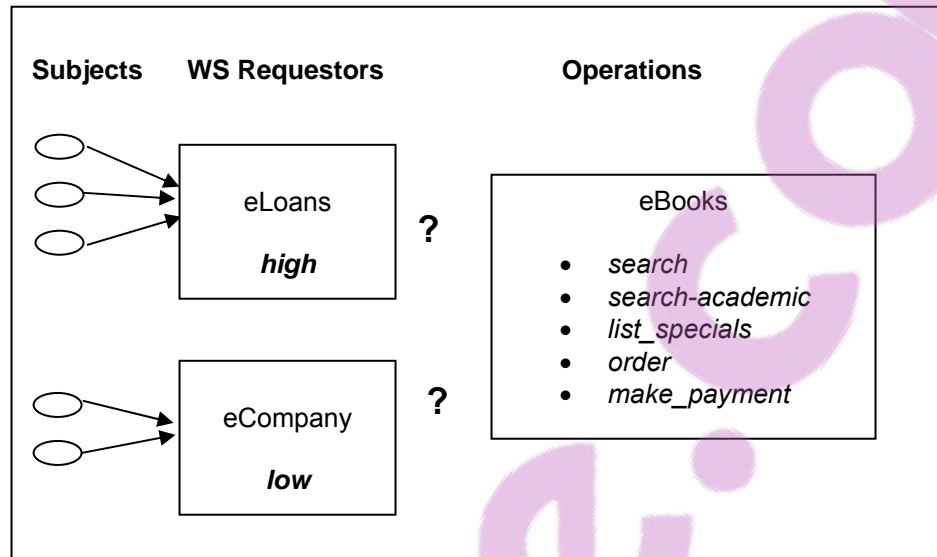


Figure 11.3: Trust levels of web services requestor

From the discussion in the previous two paragraphs, unique access control considerations evolve. In Chapter 4 attribute-based access control and trust levels were identified as internal access control requirements for web services providers. These two requirements are not independent of each other, but must be addressed in conjunction with each other, as each contributes to an advanced level of trust so that access can be granted to subjects. Figure 11.3 illustrates the problem of granting subjects from different web services requestors access to trusted web services operations. Subjects are represented by credentials, while web services requestors are represented by trust levels.

For instance, subjects request access to the *list-specials* operations. Even though subjects in the domain of eCompany may have valid credentials such as student numbers or credit ratings, they are not granted access as they are constrained by their relationship with eCompany, whose trust level is low. Subjects of eCompany are therefore limited to access only the *search*, or *search-academic* operation if valid credentials are presented. On the other hand, subjects originating from eLoans are granted access to the *list_specials* operations as the level of trust is “high”, similar to the trust level required by the *list_specials* operation.

Next, important concepts for the WSACT model are described.

11.3 WSACT AUTHORISATION MANAGER CONCEPTS

The authorisation manager of the WSACT model aims to provide a first step towards an access control model that takes into account the different trust relationships existing between web services requestors and providers that collaborate in a virtual community. The goal of the authorisation manager of the WSACT model is to express and enforce access control policies by including the calculated trust level of a web services requestor. At the same time, the authorisation manager is in line with current developments in attribute-based access control, as access is granted to subjects based not on their identity, but on their abilities that are expressed as sets of attributes. It is thus possible to specify that a subject be granted access to a particular web service operation from a certain web services requestor, but the same subject can be denied access if it makes the request from another web services requestor. Three concepts play an important role in this model: subject attributes, roles and trust levels. Each of these concepts is defined below, after which the rules and policies of the authorisation manager are described.

11.3.1 Subject attributes

Web services providers need to share their resources with many previously unknown entities. A subject possesses attributes that it can submit in order to obtain access to web services objects. The web services requestor is informed of the types of attributes that are needed by means of selective interface policy publication. Access control requirements are made available to web services requestors based on their trust level. Table 11.2 shows attribute requirements that are published per operation in the interface policy.

Table 11.2 Subject attributes per operation

Operation	Attributes required
Search	Assertion - Type of subjects such as employee, customer or student
Search-academic	Credential - Student number, institution name, year of study and date of last registration
Order	Assertion - Identity, credit level
Make-payment	Assertion - Identity, position of administrator making payment
List-specials	None

There are three possible interface policies that are defined according to trust levels. Initially, web services requestors who have a “low” trust level are exposed to only requirements for the search, and search-academic operations and operations that introduce minimal interdependency as described in paragraph 3.1.1.1 in Chapter 3. When a web services requestor reaches a “moderate” trust level, the existence of the order and make-payment operations and their

requirements will be exposed by selective publication of another interface policy. The existence of the list-specials operation is only made available to web service requestors with a “high” trust level. This operation does not require of subjects to be in possession of attributes.

Attributes are presented as assertions, or as credentials. In both cases, the content must be unforgeable and verifiable. This can be achieved if each credential is signed by the private key of the issuing authority, and if assertions made by web services requestors on the identity, name or age of the subject are signed by its private key. The web services provider needs the corresponding public key to ensure the validity of both credentials and assertions. Finally it can be said that the trust in a subject resides in the fact that its credential or assertion of the subject can be verified. The extent of this trust needs to be addressed by further trust assessment.

11.3.2 Roles

The authorisation manager of the WSACT model is role-based to make the assignment of permissions simpler and more scalable. Important RBAC (role-based access control) concepts are role hierarchies and role activation.

- *Role hierarchies* are an extension of RBAC, where roles inherit permission from subordinate roles. This allows the role hierarchy to reflect the hierarchy of the organisation roles. A partially ordered role-hierarchy RH exists, also written as \geq , where $x \geq y$ signifies that role x inherits the permissions assigned to role y . Inheritance along the role hierarchy is transitive, as multiple inheritance is allowed in partial orders.
- Role assignment is normally static as administrators assign a role to a user. For the WSACT model, dynamic *role activation* is important as the conditions under which each request is processed may be different.

There is furthermore a strong relationship between the concept of a trust level and a role. As already mentioned, a trust level is assigned to a web services requestor by the trust manager – based on evidence, information and past experience. Administrators of eBooks assign a trust level to each web services operation. Assigning trust levels to each and every web services operation and other resources will incur an administrative burden. Roles can simplify this assignment process if trust levels are assigned to roles, and roles are assigned to permissions.

The authorisation manager of the WSACT model thus support the assignment of trust levels to both roles and web services requestors. A next level of abstraction is thus introduced between subjects and operations. Administrators do not assign roles statically to web services requestors, but it is done dynamically, based on the evaluation of each request that is made. Role-activation rules specify the pre-requisite conditions and constraints that a web services requestor must

satisfy in order to enter a role. Roles are activated for the duration of the request that is processed.

A web services requestor that is assigned a trust level of “moderate” creates a session in which it activates a role that is at either a trust level of “low” or a trust level of “ignorance”. It thus dynamically activates roles in a role hierarchy that follows the structure of trust levels. Each session relates the web services requestor to many possible roles that are available according to the role/trust level hierarchy. It follows that access control - incorporating trust levels - enforces one-directional information flow in a lattice of trust levels as is required.

11.3.3 Trust levels

For access control to incorporate trust levels, it is important to accurately determine the trust level of web services requestors who make requests on behalf of subjects. The trust level of a web services requestor is calculated by the trust manager that continuously evaluates information and evidence. An important aspect of the authorisation manager of the WSACT model is that a trust level is retrieved by the authorisation manager as it is needed to make an access control decision.

Trust exists at different levels. Levels can be defined as the set $\{ignorance, low, moderate, good, high\}$, where $ignorance \subseteq low \subseteq moderate \subseteq good \subseteq high$. The set of all trust levels is denoted as TL. Trust levels are structured according to the following definitions:

Definition – Complete Partial Order: A complete partial order (TL, \subseteq) is a partially ordered set that contains a least element. This means that for trust levels there exists a least element, referred to as *ignorance*, which indicates that no information about a web services requestor exists.

Definition – Trust level lattice: There is a finite lattice of trust levels TL with a partially ordered dominance relation \geq and a least upper bound operator.

Using such a lattice of trust levels, it is possible to associate more than just one exact value with a web services requestor or role. These definitions allow the trust level of a web services requestor to be compared against the trust level of a role. The access control policy defines the access that is granted to subjects and web services requestors based on their trust level.

All concepts for the authorisation manager have now been defined. The rules and policies of the access control policy follow in the next section.

11.4 WSACT RULES AND POLICIES

The access control language of WSACT is based on ASL (Authorisation Specification Language). ASL, in turn, is based on Datalog, a rule-based declarative language that is widely understood. ASL is extended for WSACT with predicates to enable attribute-based access control and reasoning about trust levels. However, standard Datalog is not very expressive. For the WSACT model, it is important to evaluate trust levels by comparing them with one another. Datalog^C (Datalog extended with constraints) (Li & Mitchell 2003) allows first-order formulae in one or more constraint domains, which may be used to evaluate hierarchies in trust levels. The access control language is thus made more expressive as it allows reasoning over structures. For the WSACT access control policy language, the *constraint domain C* is a language of first-order formulae containing at least *true*, *false*, and the predicates =, >, <, ≤, ≥ between expressions that contain variables, and other constructs.

All sets and their relations used in the WSACT access control policy language are introduced in the paragraphs below; followed by the predicates used by the language.

11.4.1 Sets and relations

The following sets are used in the WSACT access control policy.

11.4.1.1 Subject attributes

SUBJATTR represents the set of subject attributes required to be granted access to resources. The WSACT access control policy maintains a list of attributes that are required to be granted access to an object, where an object is a web service operation, a collection of operations, a web service or a collection of web services that are exposed by the web services provider. *Subjattr* is an expression in the form *subjattr(attr₁, attr_j)*, where *subjattr* ∈ *SUBJATTR*, *subjattr* is the name of the assertion, and *attr₁, . . . attr_j* is the list of elements. For each 0 < i ≤ j, *attr_i=value*, where *value* is a variable.

An example of a set of subject attributes is credit card and an identification number. It is represented as *Payment_info(Credit_Card = "11003030302010", ID = "337552")*.

11.4.1.2 Web services requestors

REQUESTOR is the set of all requestors who require access to the system. A requestor is an expression of the form *requestor = value*, where *requestor* ∈ *REQUESTOR*, *requestor* is the identifier of a web services requestor, and *value* is a variable.

An example of a web services requestor may be requestor = "eLoans".

11.4.1.3 Web services objects

WSOBJECT is the set of all web services objects that require protection. The web services interface description contains information on the operations that can be performed so that the web services requestor is able to format a valid message. A web service object is an expression of the form $wsubject = value$, where $wsubject \in WSOBJECT$ and $value$ is a variable, and $wsubject$ is a web service operation, a collection of operations, a web service or a collection of web services that are exposed by the web services provider. Such relationships introduce a partial order \subseteq_{wsobj} on the set of web service objects, where operations are the minimal elements of the partial order.

An example of a web service object may be $wsubject = "Order"$.

11.4.1.4 Actions

SA is the set of all signed actions. A signed action is an expression of the form $sa = value$. Given a set of actions A , a set of values $sa \in SA$ is defined as $\{+a, -a \mid a \in A\}$. An example of a signed action would be $sa = "+EXE"$.

Remote users or applications would generally be allowed to execute a web services method, or access a server hosting a number of web services objects or an application.

11.4.1.5 Roles

ROLE is the set of all roles. A role is an expression of the form $role = value$, where $role \in ROLE$, and $value$ is a variable. An example of a role is $role = "visitor"$.

11.4.1.6 Trust levels associated with a role

TLROLE is the set of all trust levels associated with a role. The trust level is an expression of the form $tlrole = value$, where $tlrole \in TLROLE$, and $value$ is a variable.

An example of a trust level of a role is $tlrole = "low"$.

11.4.1.7 Trust levels associated with a web services requestor

TLREQ is the set of all trust levels assigned to a web services requestor. The trust level is an expression of the form $tlreq = value$, where $tlreq \in TLREQ$, and $value$ is a variable.

An example of a trust level of a web services requestor is $tlreq = \text{"low"}$.

The rules defined for the language have now been specified. It is assumed throughout that $subjattr \in \text{SUBJATTR}$, $requestor \in \text{REQUESTOR}$, $wsubject \in \text{WSOBJECT}$, $sa \in \text{SA}$, $role \in \text{ROLE}$, $tlrole \in \text{TLROLE}$, and $tlreq \in \text{TLREQ}$

11.4.2 Predicates

The predicates that define the access control policy are described in sections: access control predicates, service satisfaction predicate, role activation predicates, access derivation predicate and access request predicate.

11.4.2.1 Access control predicates

The scope of access control that is defined over the resources of the web service provider is defined in ASL syntax. For instance, an authorisation rule is a rule of the form:

$$\mathbf{cando}(wsubject, role, sa) \leftarrow L_1, \dots, L_n.$$

For each $0 < i \leq n$, L_i is either a **in**, **dirin**, or **typeof** literal. Authorisation rules are specified to grant or deny requestors access to web services objects. From these authorisations, further authorisations are derived through the application of other rules such as:

$$\mathbf{dercando}(wsobj, r, sa) \leftarrow L_1, \dots, L_n.$$

$$\mathbf{do}(wsobj, r, sa) \leftarrow L_1, \dots, L_n.$$

$$\mathbf{grant}(wsobj, s, r, sa) \leftarrow L_1, \dots, L_n.$$

$$\mathbf{done}(wsobj, s, r, sa, j) \leftarrow L_1, \dots, L_n.$$

11.4.2.2 Service satisfaction predicate

A new predicate *satisfied* is introduced to specify the attributes that are required to access objects defined by $wsubject$. The set of attributes is defined by $subjattr(attr_1, \dots, attr_j)$. A set of attributes need to be presented by a subject in order to be granted access to web services objects. A rule is specified as follows:

$$\mathbf{satisfied}(wsubject) \leftarrow \mathbf{subjattr}(attr_1, \dots, attr_j).$$

11.4.2.3 Role activation predicates

a) The trust level for each requestor is defined by the predicate:

$$\mathbf{reqTlevel}(requestor, tlrequestor).$$

The predicate is instantiated after interrogating the trust manager.

b) The trust level for each role is defined by the predicate:

roleTlevel(role, trole).

c) Role activation is defined by the following rule:

active(requestor, role) \leftarrow **reqTlevel**(requestor, treq),
roleTlevel(role, trole),
 ((trequestor > trole); (trequestor = trole)).

When the trust level of a web services requestor is the same or higher than the trust level of the role, the role is activated for the web services requestor.

It is important to ensure that a web services requestor activates only one role at a time, and that trust levels are carefully assigned to roles to avoid inconsistencies.

11.4.2.4 Access derivation predicate

A permission can be derived if a number of conditions are satisfied. Firstly, a rule, or a previously derived rule, should be present that allows the wsubject to be accessed by a role for a specified action. Next, the web services requestor must be active in the role. Finally, required subject attributes need to be present in the set of facts in the access control policy.

dercando(wsobj, requestor, sa) \leftarrow **cando**(wsobj, role, sa)
active(requestor, role),
satisfied(wsobj).

11.4.2.5 Access request predicate

Lastly, a query is presented to the authorisation manager. The access request is formulated by the authorisation interface, and never by users. The format of the access request is:

do(wsobj, requestor, sa) \leftarrow **dercando**(wsobj, requestor, sa).

The result of this query is always true or false. The answer is subsequently returned to the authorisation manager where it is enforced.

All sets, relations and predicates have been formally defined above. To illustrate the reasoning of the access control policy with regard to the eBooks case study, the authorisation manager is now formally described in Z.

11.5 AUTHORISATION MANAGER IN Z

Facts and rules that define the access control policy are now formally stated in Z. The specification is based on the specification of rule-based systems as provided by Jacky (1997). The contribution of this specification is twofold: consistency is maintained with all components of the WSACT model, and the reasoning of the access control policy is illustrated by means of the facts and rules of eBooks.

11.5.1 Facts

The state of the authorisation manager is the collection of facts that it takes to be true. Some facts are loaded into the system, and others are deduced. No distinction is made between these two categories in the authorisation manager state.

[*FACT*]

A basic type is used to declare facts. Particular facts are elements of this type.

A subset of all possible facts for eBooks is given for the purposes of this discussion. The facts are organised into access control rules, role trust levels, web services requestor trust levels and attributes required to be granted access to web services operations. Roles are the set {visitor, associate, client, partner, trusted_partner}, where “visitor” is the role with the least privilege. The facts are the permissions for the search, search_academic, place_order, make_payment and list_specials web services objects. Fact can also be deduced from role inheritance rules if so desired.

Access control rules:

```
search, visitor, +exe : FACT
search, associate, +exe : FACT
search, client, +exe : FACT
search, partner, +exe : FACT
search, trusted_partner, +exe : FACT
search_academic, visitor, +exe : FACT
search_academic, associate, +exe : FACT
search_academic, client, +exe : FACT
search_academic, partner, +exe : FACT
search_academic, trusted_partner, +exe : FACT
order, client, +exe : FACT
order, partner, +exe : FACT
order, trusted_partner, +exe : FACT
```

```

make_payment, client, +exe : FACT
make_payment, partner, +exe : FACT
make_payment, trusted_partner, +exe : FACT
list_specials, trusted_partner, +exe : FACT

```

Trust levels are defined as facts as follows: 0 = ignorance, 1 = low, 2 = moderate, 3 = good, 4 = high. The names of the roles reflect the increase in trust.

Role trust levels:

```

visitor, 0 : FACT
associate, 1 : FACT
client, 2 : FACT
partner, 3 : FACT
trusted_partner, 4 : FACT

```

The next facts state the trust levels associated with each of the requestors.

Requestor trust levels:

```

eLoans, 4 : FACT
eOther, 2 : FACT
eCompany, 2 : FACT
eOrgA, 3 : FACT

```

Next facts list which attributes are required by web services operations.

Attributes required:

```

search, student : FACT
search_academic, 920099777, UP, 2005/01/30 : FACT
order, 123445, good : FACT
make_payment, cc555678, admin : FACT
list_specials, none : FACT

```

No deduced facts have been included in these lists. They will be addressed when the rules of the authorisation manager are discussed.

11.5.2 The abstract state space of the authorisation manager

State changes that need to be modelled include the deduction of a conclusion relevant to a specific goal, and return of a result to the authorisation interface. The abstract state space of the authorisation manager is considered as a collection of data. The state of the system changes when operations are executed that transform data.

A set of $\{OPERATION\}$ can be used to refer to all operations of the authorisation manager. The result of these executions is an access decision.

<i>Authorisation Interface</i>
<i>Facts, Facts!</i> : <i>FACTS</i>
<i>Access request?</i> : <i>P FACT</i>
<i>Accessdecision!</i> : <i>P FACT</i>
$\forall oe : opExe \bullet oe.message \in \{accessdecision\}$
$\{ o : opExe \bullet o.operation \} = operation$

In the access control policy of the authorisation manager, rules associate a set of facts (called the *premise*) with a fact called the *conclusion*. If all premises are true, it can be concluded that the conclusion is true as well. In Z, the access control policy is represented as a global constant, which is a relation from sets of facts to facts.

As an example, the following scenario is described. eLoans makes a request to execute the order operation on behalf of a subject. The action is set to +exe. A conclusion needs to be inferred from the existing set of facts and rules defined in the access control policy shown next. In the access control policy:

- The first rule identifies that the client role must be used to access the order operation.
- The next rule activates the client role for eLoans as its trust level is 4, which is higher than 2, the trust level of the client role.
- The next rule grants eLoans access to the order operation, as valid subject attributes (id = "123445", credit level = "good") is present, and the correct role has been activated.
- The last rule evaluates to true as a derived access control rule is found for the access request that has been made.

access control policy : P *FACT* \leftrightarrow *FACT*

access control policy = {

order, client, +exe \mapsto *order, eLoans, +exe*

eLoans, 4 \wedge *client, 2* \wedge $4 > 2$ \vee $4 = 2$ \mapsto *eLoans, client*

order, id("123445"), credit_level("good") \wedge

eLoans, client \wedge

order, client, +exe \mapsto *order, eLoans, +exe*

order, eLoans, +exe \mapsto *order, eLoans, +exe*

}

The facts loaded in the access control policy are known to be true. *Infer* is a function from the initial set of facts to a larger set of facts. All of the initial facts also appear in the final set.

$infer = = (\lambda facts : P\ FACT \bullet facts \cup access\ control\ policy(|P\ facts|))$

Access control policy ($|P\ facts|$) is the set of all the conclusions of all the rules whose premises match some combination of the initial set of facts. The function *infer* is applied repeatedly until no more facts can be derived. This is the task of the function *entire*. The function determines the transitive closure of the child relation that is started with and is depicted by $^+$.

$entire = = infer^+$

Inconsistent facts need to be specified as well. For instance, positive and negative authorisations cannot both be derived for eLoans for the search operation.

inconsistent = = {

{search, eLoans, +exe},

{search, eLoans, -exe},

}

A set of consistent facts contains no more than one element from each set of mutually exclusive alternatives.

$consistent : P(FACT)$

$\forall facts : consistent; mutually_exclusive : inconsistent \bullet$
--

$\# (mutually_exclusive \cap facts) \leq 1$
--

The central activity of the authorisation manager is to consider data in order to deduce a conclusion that is relevant to a specific goal. This is modelled by the *Infer* operation.

<i>Infer</i>
$facts, facts' : P(FACT)$ $data, goals, accessdecision! : P(FACT)$
$(let\ all = complete(facts \cup data) \bullet$ $facts \subseteq facts' \subseteq all \wedge$ $accessdecision! = goals \cap all \subseteq facts')$

The access control policy is queried with the access request formulated by the authorisation manager. The operation is therefore goal-driven and it is implemented with backward chaining.

<i>Backward</i>
<i>Infer</i>
$access\ request? : P(FACT)$
$goals = queries?$

11.6 CONCLUSION

In this chapter the authorisation manager of the WSACT model was described. Both access control specification and reasoning have been addressed by the access control policy language that is defined in Datalog.

The access request presented to the authorisation manager is evaluated by the authorisation manager in a novel way. Both the set of attributes provided on behalf of the subject, and the trust level of the web services requestor are used to determine whether access can be granted. These two aspects complement each other in terms of the level of access that can be granted.

Role activation occurs based on the trust level of web services requestors. Roles are kept private so that the access control services of the web services provider are not compromised by subjects or web services requestors who try to activate known roles. Subjects and web services requestors use the published interface policy to determine which attributes to present, but this information does not give them access to the rationale behind the access control decision-making process.

An important element of the access control policy is the trust level of web services requestors. The trust level needs to be determined accurately, as it is used in many of the access control decisions. The trust level is dynamically retrieved from the trust manager for each access request where it is needed. The calculation of this trust level is discussed in the next chapter.

12

WSACT The Trust Manager

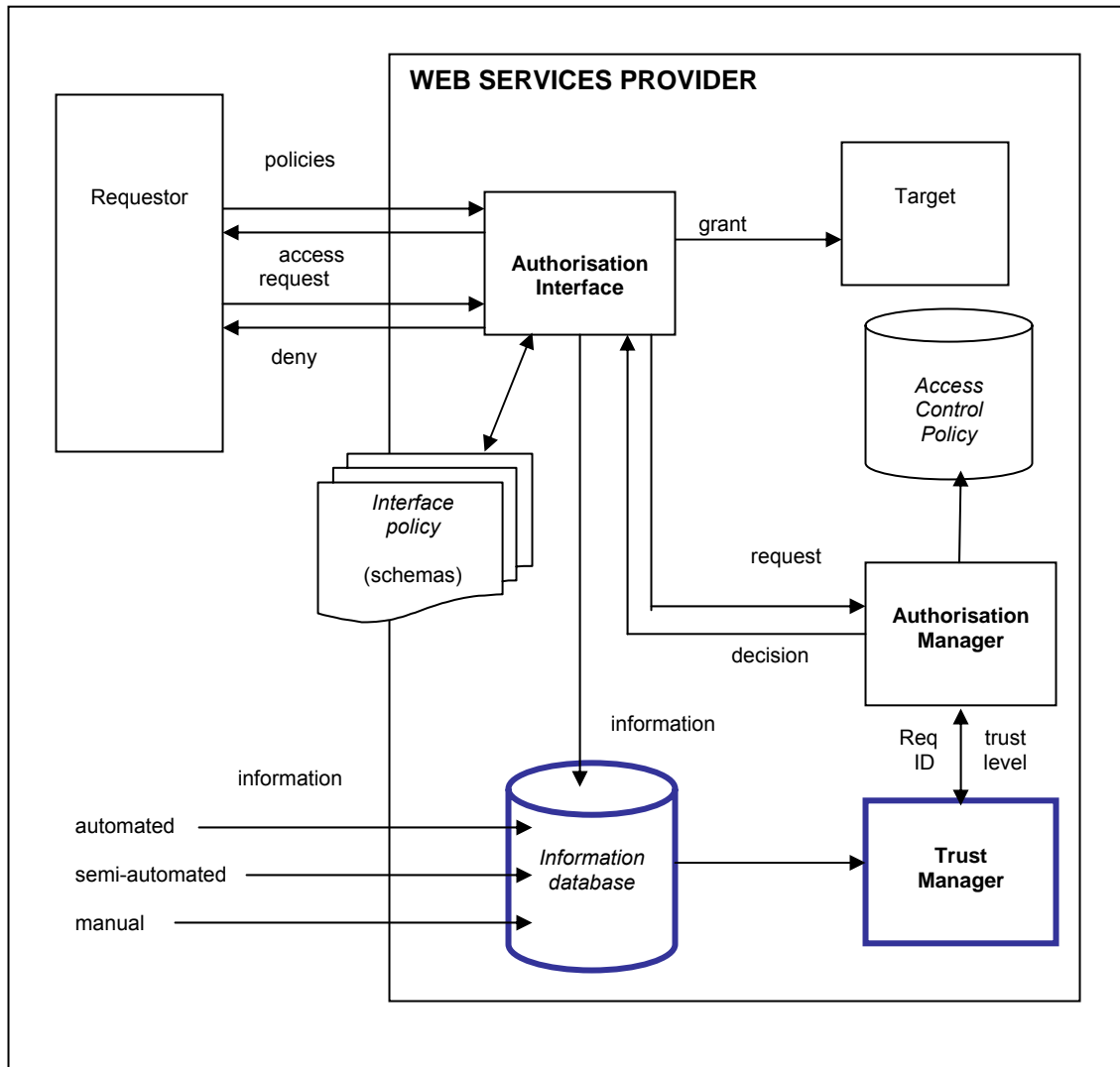
The authorisation manager invokes the trust manager, the third component of the WSACT model. The trust manager is invoked when access control rules defined in the access control policy are evaluated that needs the trust level of a web services requestor.

The motivation for the trust level is clear, as there are possibly millions of customers who become participants in environments provided by collaboration between web services requestors and providers. It is to be expected that the transactions that occur will vary in quality and value. In addition, the goods and services transacted are subject to different legal regulations, contractual agreements and security features. Web services entities need to be able to make distinctions between the behaviour, environment and characteristics of others. This research proposes that a trust level can be used to make these distinctions. A trust level gives a web services provider the ability to predict the behaviour of web services requestors. The proposed trust level is composed of a rich variety of structured information that is evaluated in an intuitive manner.

The first section of this chapter identifies external components that relate to the trust manager of the WSACT model. Internal components of the trust manager are made explicit next. This is followed by a discussion of concepts for the fuzzy cognitive map for web services trust, and a description of the structure of trust, trust types and trust concepts. Trust inference is addressed by a modified rule and by using the logistic threshold function, while the trust manager is described in Z, and operations are formally specified. Finally, the computation of trust levels is illustrated and ranges for trust levels are identified, before the chapter is rounded off with a conclusion.

12.1 THE TRUST MANAGER

Figure 12.1 depicts the trust manager and related data components in blue. The trust manager is interrogated by the authorisation manager in order to determine the trust level of a web services requestor. The trust manager makes use of the information database in its reasoning.



A request containing the identity of the web services requestor is sent to the trust manager. The latter inspects the information at its disposal and calculates the appropriate trust level. It replies to the request of the authorisation manager by indicating the trust level that is allocated to the web services requestor.

12.1.1 Trust manager components

The trust manager architecture consists of the components depicted in Figure 12.2 below.

- The main component of the trust manager is the *fuzzy cognitive map* that infers trust levels from trust concepts.
- The *trust database* stores trust concepts that are populated in two different ways:
 - *Automated*: The fuzzification process populates trust concepts with fuzzy values by interrogating and processing categorised trust information that is stored in the information database.
 - *Static*: Administrators use an interface to populate trust concepts with fuzzy values.

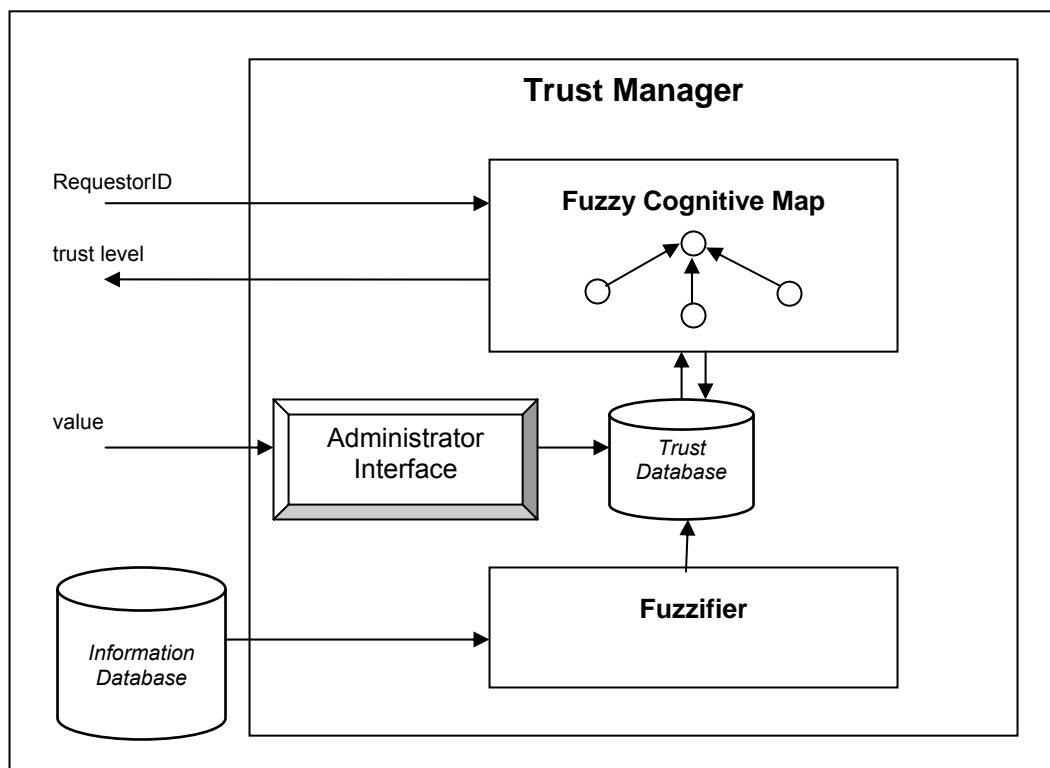


Figure 12.2: Trust manager architecture

The next paragraph gives a basic overview of the fuzzy cognitive map for web services trust. It also describes nodes and causal relationships, and the population of trust concepts, by either an administrator or by the automated fuzzification process.

12.2 FUZZY COGNITIVE MAP CONCEPTS FOR TRUST INFERENCE

An important facet of a fuzzy cognitive map is its structure. The fuzzy cognitive map for web services trust embeds the research on trust presented in this thesis. Nodes of the fuzzy cognitive map therefore represent trust, trust types and trust concepts.

Signed and weighted arcs represent the causal relationships that exist among trust, trust types and trust concepts, as shown in Figure 12.3. The arcs of the graph represent the impact that one trust type or concept has on another and vary in the interval $[-1, +1]$. Since humans are considered experts, they determine these causal influences (Kosko 1997). Each arc is drawn and weighted through intuition and may be modified through experimentation. Causal relationships are characterised by vagueness, as they represent the influence of one qualitative factor on another. Values are set by using linguistic variables such as “I believe that if trust in the internal environment increases, the increase in the trust level is *good*”. The linguistic variable is reflected by setting the arc weight to $+0.5$.

The above highlights an important feature of fuzzy cognitive maps. In fuzzy logic, linguistic labels are used to assign values, where each label may represent a range of values. Setting a label of HIGH may be interpreted to represent more than one numerical value. In fuzzy cognitive maps, however, crisp values are required. To accommodate this feature of fuzzy cognitive maps, the range $[-1, +1]$ may be subdivided into a number of linguistic labels such as VERY LOW, LOW, MODERATE, GOOD and HIGH, where each label is translated into a single numeric value.

The value of nodes varies in the interval $[0, +1]$ and reflects the degree to which the concept is active in the system at a particular time. Nodes are activated by either an automated process, which is the focus of this research, or else by an administrator, if automation is not possible. Again, crisp values are used to populate nodes to represent a linguistic label. A value of $+0.4$ for a trust concept such as implemented security mechanisms means that the trust manager is either *convinced* that the level to which security mechanisms are implemented is *moderate*, or else that it may be *possible* that the level to which security mechanisms are implemented is *good*. If a satisfactory threshold level is reached for a trust concept, its value is activated to 1, or otherwise to 0.

The fuzzy cognitive map allows the computation of trust, starting from trust concepts that are populated with values that represent knowledge that has been accumulated. Trust concepts represent a realistic view of the web services requestor and of the stability of the environment in which web services operations are to be used. Trust concepts embody the beliefs of the web services provider to provide a basis over which a trust level can be inferred. The level of trust is thus derived directly from the strength of trust concepts.

12.3 FUZZY COGNITIVE MAP FOR WSACT

The structure of the fuzzy cognitive map for web services trust is shown in Figure 12.3. It depicts the following:

- The trust level (C_1) in indigo, which is inferred from three nodes representing trust types.
- Three trust types in green:
 - Trust in the internal environment (C_2)
 - Trust in the external environment (C_3)
 - Trust in the other party (C_4)
- Trust concepts shown in green and aquamarine, represented by the remaining 15 nodes.

These trust concepts consist of

- two inferred trust concepts, shown in green; and
- thirteen trust concepts populated by fuzzy values, shown in aquamarine.

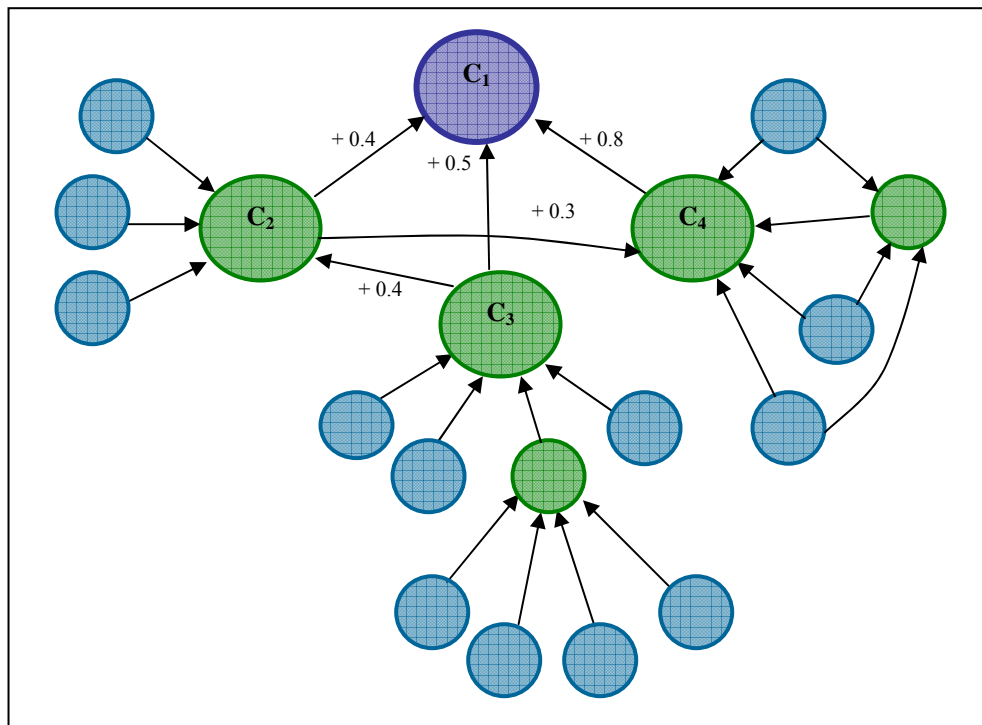


Figure 12.3: Fuzzy cognitive map for eLoans

The discussion dealing with the structure of the fuzzy cognitive map for web services trust is presented in a top-down mode. First, the inference of a trust level from the three trust types is discussed as it applies to eBooks and eLoans. A discussion of the population of related trust concepts of each of the three trust types follows subsequently.

12.4 TRUST

The fuzzy cognitive map models the interactions between the three types of trust that form the level of trust that eBooks holds towards eLoans. The nodes in Figure 12.3, representing trust and trust types, are as follows:

- C_1 - The trust level that eBooks infers for eLoans
- C_2 – The trust in the internal environment of eBooks
- C_3 - The trust in the external environment between eBooks and eLoans
- C_4 - The trust that eBooks has in eLoans

Causal relationships: The arcs of the graph represent the impact that one concept has on another and they vary in the interval $[-1, +1]$. For instance, a value $+0.8$ for the trust in eLoans (C_4) increases the trust (C_1) by 80%.

- $C_2 - C_1$: The trust in the internal environment of eBooks (C_2) has a moderate influence on trust as its weight is set to $+0.4$.
- $C_3 - C_1$: The trust in the external environment between eBooks and eLoans (C_3) has a moderate to strong influence on trust as its weight is set to $+0.5$.
- $C_4 - C_1$: The trust that eBooks has in eLoans (C_4) has a strong influence on trust, as the weight is set to $+0.8$.
- $C_2 - C_4$: The trust in the internal environment of eBooks (C_2) has a causal effect on the trust that eBooks has in eLoans (C_4) and its weight is set to $+0.3$. If eBooks is very sure about the risk of an endeavour and has expertise in dealing with that risk, eBooks may increase its belief in eLoans to reflect its self-confidence.
- $C_3 - C_2$: The trust in the external environment that exists between eBooks and eLoans (C_3) has a causal effect on the trust in the internal environment of eBooks (C_2) as its weight is set to $+0.4$. If eBooks is transacting with eLoans in a familiar environment, eBooks is bound to feel very sure, even though it may have vulnerabilities in its environment to consider.

Trust and the three trust types are defined as basic types as follows:

$[TRUST], [TRUST_INT_ENV], [TRUST_EXT_ENV], [TRUST_WS_REQ]$

The output variable, *trustlevel*, can be defined as a subset P of the set of integers Z . A definition for *trustlevel* is:

$| TRUSTLEVEL : P \ Z$

The output of the trust inference is treated as linguistic variables whose values are modelled as fuzzy sets. The generic symbol \mathcal{F} models a fuzzy set from a label to the real number interval $[0,1]$.

IGNORANCE : \mathcal{F} TRUSTLEVEL

LOW : \mathcal{F} TRUSTLEVEL

MODERATE : \mathcal{F} TRUSTLEVEL

GOOD : \mathcal{F} TRUSTLEVEL

HIGH : \mathcal{F} TRUSTLEVEL

The structure and population of nodes that represent the trust types of the fuzzy cognitive map are discussed next.

12.5 TRUST TYPES

Consideration is given to each of the three trust types. The focus of this research is to aim towards an automated process of trust assessment and inference. The trust in the other party (C_4) is most representative of this aim. Trust in the internal environment (C_2) and trust in the external environment (C_3) are partly populated by administrator intervention and their automation is the focus of future research. These two trust types form a basic level of trust based on risk evaluations, security mechanisms and guarantees, over which trust in eLoans can evolve. Their structure and the population of trust concepts over which they are inferred are described next.

12.5.1 Trust in the internal environment of eBooks

The focus of trust in the internal environment is to determine the self-confidence of eBooks. The trust in the internal environment of eBooks (C_2) is inferred from related trust concepts that have been populated by an administrator that determines the risk to which the web services entity is exposed, and the web service's ability to deal with risky interactions.

The concepts representing each node are as follows:

- C_5 - the vulnerabilities in the system of eBooks;
- C_6 - the complexity of the systems of eBooks; and
- C_7 - the successes of eBooks in dealing with risks and compromises.

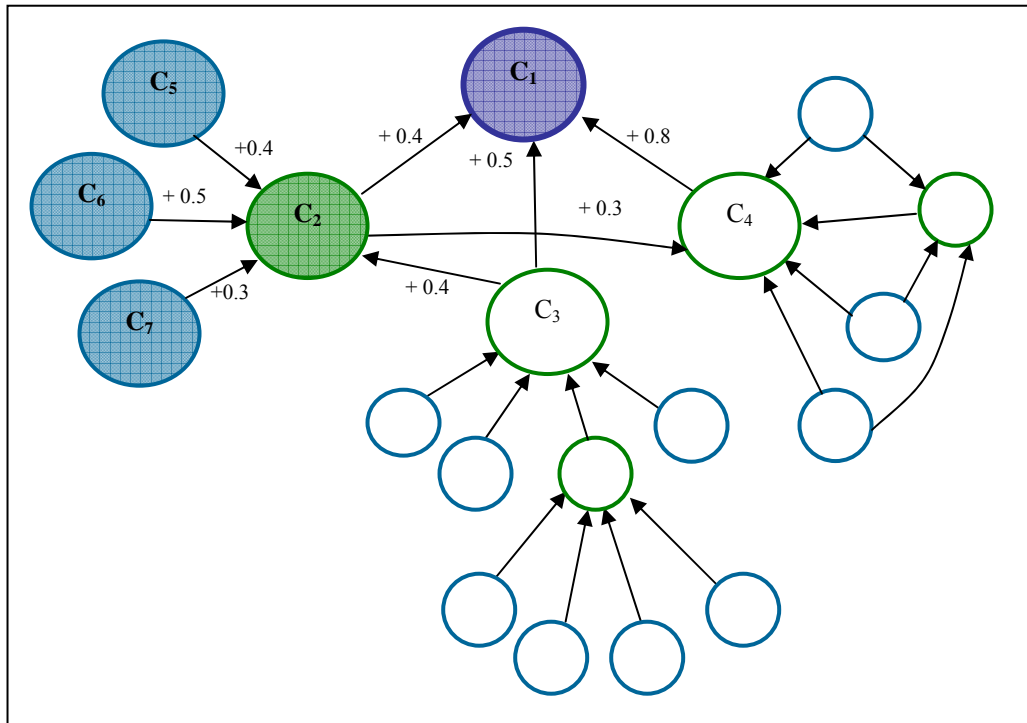


Figure 12.4: Trust in the internal environment of eBooks

The value of a trust concept is represented by a variable *fuzzyvalue*, defined as a subset of the set of integers. A definition for *fuzzyvalue* is:

| FUZZYVALUE : P Z

Each trust type is discussed next by describing its causal relationship and by investigating the population of its node. As this trust type is not populated in an automated manner, administrators inspect and analyse information in order to assign a value in the range [0,1]. Trust concepts C_5 to C_7 for eBooks are considered.

C_5 – Vulnerabilities

Causal relationship C_5 – C_2 : The level of vulnerabilities in the systems of eBooks has an effect on trust in the internal environment, as the weight is set to 0.4. This means that as the level of vulnerabilities decreases, trust in the internal environment increases by 40%.

Population of the trust concept C_5 : A fuzzy value in the range [0,1] is determined for this trust concept by evaluating the outcome of a vulnerability assessment. With vulnerability scanning, an automated scanning program or vulnerability scanner scans the network of computers hosting eBooks for a list of known weaknesses referred to as vulnerabilities (Venter 2003). A vulnerability scanner thus analyses the security state of the system on the basis of information collected at

intervals. After a scan is completed, the vulnerability scanner creates a report of the vulnerabilities found.

One or more administrators intuitively estimate a level for the occurrence of vulnerabilities for eBooks. The level is estimated after a thorough evaluation of the reports produced by the vulnerability scanner on the security state of the software and hardware hosting eBooks. A formal approach to vulnerability evaluation should be used, where the count of vulnerabilities, the vulnerability scanner used, and the different levels of risks posed by established vulnerabilities must be taken into account.

The result of the vulnerability evaluation is translated into linguistic terms LOW, MEDIUM, HIGH and VERY HIGH. The administrator of eBooks is presented with an interface that allows him/her to select one of these values. The interface stores a value in the field in the trust database representing this trust concept. Depending on the level of vulnerabilities, a threshold function activates the trust concept to either 1 or 0.

A record is created in the trust database with the following field:

<i>ReqTrustRec</i> <i>Id : REQID</i> <i>C₅Vulnerabilities : SINGLE</i>

C₆ – Successes in dealing with security risks and compromises

Causal relationship C₆ – C₂: The successes of eBooks in dealing with security risks and compromises have a positive effect on trust in the internal environment, as the weight is set to 0.5. This means that as more success is achieved in dealing with risks, trust in the internal environment increases by 50%.

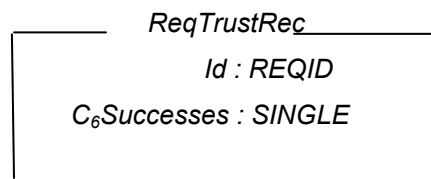
Population of the trust concept C₆: The purpose of this metric is to measure the number of vulnerabilities that have been identified and fixed, the time to resolution, and other information for measuring successes in progress when dealing with security risks and compromises. For instance, the average time to fix vulnerabilities and the total number of hours spent resolving vulnerabilities will play an important role in determining the metric. If the average time to fix vulnerabilities is three weeks, this may be deemed unsafe because exploits and worms are being developed within weeks of vulnerabilities being publicised.

Factors that can play a role in the determination of this metric are the number of users and devices compliant with each element of the security policy; the number of web services

requestors affected by service degradation or disruption or other compromises; data lost, modified or destroyed; decrease in network performance; increase in network utilisation; increases in waiting times during a network compromise, and time between compromise discovery and completion of system remediation. eBooks needs to chart the security team's performance to make sure the end result is risk reduction, especially to critical assets. Administrators should determine metrics to evaluate the successes and failures of different policies to improve security performance.

The result of determining successes is translated into linguistic terms LOW, MEDIUM, HIGH and VERY HIGH. The administrator of eBooks is presented with an interface that allows him/her to select one of these values. The interface stores a value in the field in the trust database that represents this trust concept. Depending on the level of successes, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional field:



C₇ – Complexity

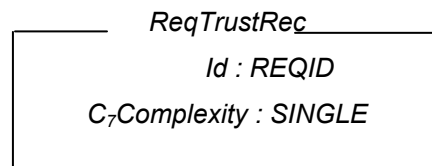
Causal relationship C₇– C₂: The level of complexity in the systems of eBooks has an effect on trust in the internal environment, as the weight is set to 0.3. This means that if the systems are not complex, trust in the internal environment increases by 30%.

Population of the trust concept C₇: Complexity determines the reliability, functionality and level of security vulnerabilities to be found in software systems. Both the inherent complexity of the web services operation and any additional complexity of the implementation play a role. The number of lines of code, and the number of functions or procedures implemented, are an important aspect of complexity because as they increase, complexity increases. A method that can be used to estimate complexity is the Function Point Analysis (FPA) (Albrecht 1979). FPA begins with the decomposition of the system of eBooks into its data and transactional functions. The data functions represent the functionality provided to eLoans by attending to their internal and external requirements in relation to the data, whereas the transactional functions describe the functionality provided to eLoans in relation to the processing of these data by the system. Before being expressed in points, the complexity of a function is characterised by the linguistic terms LOW, AVERAGE or HIGH, in accordance with its relative functional complexity. Upon completing a

point assessment of all functions, the application is then adjusted in accordance with the general characteristics of the system, which evaluates the general functionality of the application.

The result of FPA can be translated into linguistic terms LOW, MEDIUM, HIGH and VERY HIGH. The administrator of eBooks is presented with an interface that allows him/her to select one of these values. The interface stores a value in the field in the trust database that represents this trust concept. Depending on the level of complexity, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional field:



12.5.2 Trust in the external environment of eBooks

The focus of trust in the external environment is to determine the assurances that allow eBooks to collaborate with eLoans. The trust in the external environment that exists between eBooks and eLoans (C_3) is inferred from related trust concepts populated with aggregated information on the evaluation of assurances, contracts, implemented security mechanisms and best practice.

The concepts representing each node are as follows:

- C_8 – Rule of law
- C_9 – Assurances
- C_{10} – Compliance
- C_{11} – Implemented security mechanisms
 - C_{12} – Identity mechanisms
 - C_{13} – Integrity mechanisms
 - C_{14} – Confidentiality mechanisms
 - C_{15} – Privacy

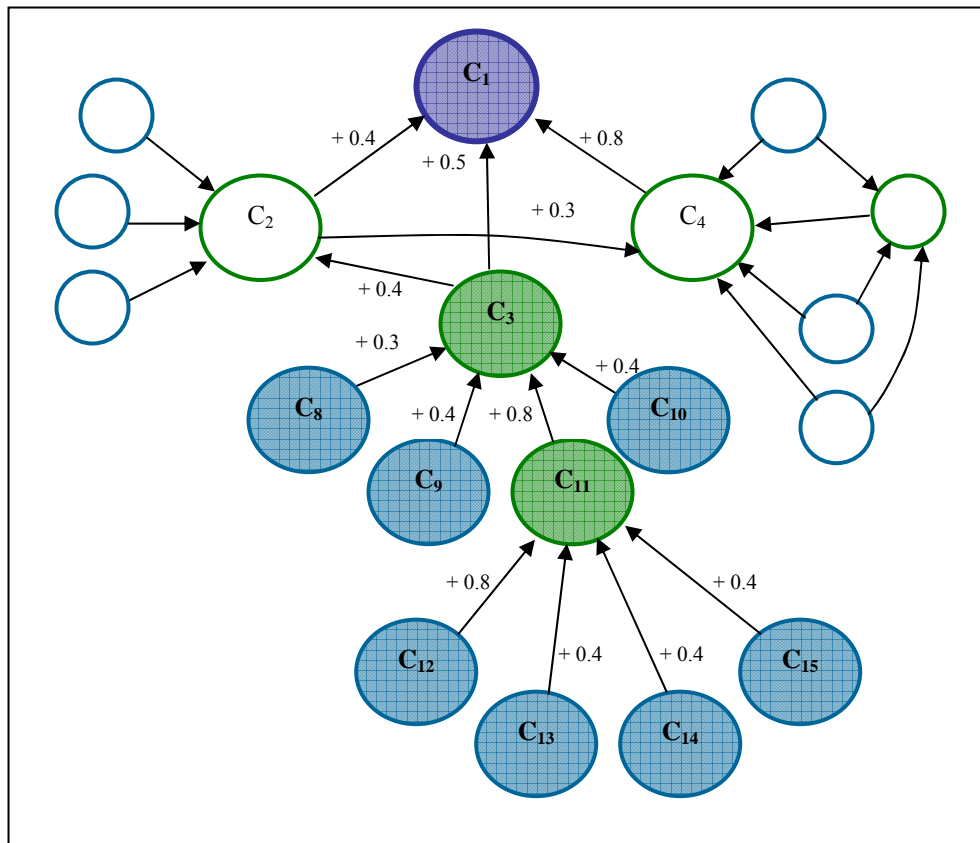


Figure 12.5: Trust in external environments between eBooks and eLoans

C₈ – Rule of law

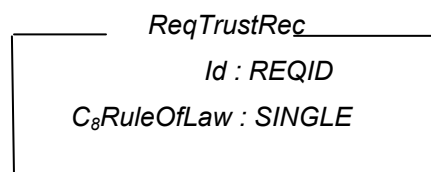
Causal relationship C₈ – C₃: The existence of rule of law has a positive effect on the willingness of eBooks to trust eLoans, as the weight is set to 0.3. This means that as there are more laws in the environment between eBooks and eLoans, trust in the external environment increases by 30%.

Population of the trust concept C₈: The rule of law can be defined as the institutional environment that establishes the basis for economic investment, production and exchange (Dedrick et al. 2005). It includes sound political institutions, an impartial court system, legal protection of property rights, enforceable contract laws, and citizens who have confidence in the legitimacy of these institutions and accept their authority in resolving disputes (Oxley & Yeung 2001). For web services, the rule of law plays an important role in reducing the risks of online transactions. When the rule of law is strong, web services requestors and providers know there is legal recourse in the face of online fraud, and there is effective punishment that offsets the need for reputation building.

To determine a metric for the rule of law that is applicable to a web services requestor such as eLoans is not an easy task. Administrators of eBooks should rather turn to formal studies that

have been conducted. Rule-of-Law is a survey-based assessment of the quality of the tradition of law enforcement and order in individual countries as reported in the study by La Porta and others (1997). It is an average of monthly indexes that range from 0 to 10, with a lower score for less tradition of law and order. Such an index can be used by administrators of eBooks, and can be transformed into a value between 0 and 1. The researcher foresees that such information will become more available in the future and that it will be exposed as a web services operation so that it can be retrieved dynamically. An administrator uses an interface to store a value in the field in the trust database that represents this trust concept. Depending on the level of rule of law, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional field:



C₉ – Assurances

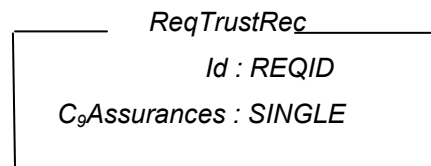
Causal relationship C₉ – C₃: The existence of assurances has a positive effect on the willingness of eBooks to trust eLoans, as the weight is set to 0.4. This means that as more assurances exist in the environment between eBooks and eLoans, trust in the external environment increases by 40%.

Population of the trust concept C₉: The existence of assurances such as licences, insurance policies and service level agreements (SLAs) provide protection against risk.

- Insurance and liability standards can be seen as tools for bolstering security. The lack of liability insurance may be seen as a significant barrier to organisational collaboration, as organisations that fail to adhere to agreed-upon standards are denied insurance (Crews 2005).
- Licence agreements are common provisions that can be seen as contracts upheld by the courts. Monetary losses are typically governed by such contractual agreements, while physical harm or property damage would be governed by more general liability law. Limitation-of-liability contracts are commonplace in allocating economic risk, as parties commonly give up certain rights to sue as a condition of receiving services in many contexts (Crews 2005).
- The establishment of machine-readable service level agreements ensures assurances with regard to the quality of the service to be provided, and business terms and conditions including pricing and penalties, to protect against risk.

The administrators of eBooks need to make a thorough investigation of their own position, as well as that of eLoans in terms of the existence of licences, insurance policies and service level agreements. The findings may lead to an intuitive decision on whether the existence of assurances is LOW, MEDIUM, HIGH or VERY HIGH. The administrator of eBooks is presented with an interface that allows him/her to select one of these values. The interface stores a value in the field in the trust database that represents this trust concept. Depending on the level of assurances, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional field:

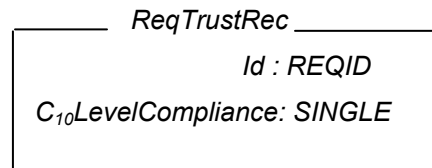


C₁₀ – Compliance

Causal relationship C₁₀ – C₃: The compliance to standards has a positive effect on the willingness of eBooks to trust eLoans, as the weight is set to 0.4. This means that higher compliance of eLoans with standards increases trust in the external environment by 40%.

Population of the trust concept C₁₀: To increase its trustworthiness, eLoans needs to implement controls or guidelines as contained in standard code of practice. Standards revolve around specific measures in a number of different control areas of security. Compliance with regulations and standards such as Sarbanes-Oxley (2002) and ISO17799 (2005) can be implemented. Regulatory compliance is demonstrated by mapping technical assessment results to specific regulations and standards to present key compliance metrics for different control areas. Assessment results are certified externally by an independent third party, who may be an individual or an organisation that has the approval of a national or an international body. These results can be conveyed to the administrator of eBooks. The results lead to an intuitive decision on whether the level of compliance is LOW, MEDIUM, HIGH or VERY HIGH. The administrator of eBooks is presented with an interface that allows him/her to select one of these values. The interface stores a value in the field in the trust database that represents this trust concept. Depending on the level of compliance, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional field:



C₁₁ – Implemented security mechanisms

Causal relationship C₁₁ – C₃: Implemented security mechanisms have a strong positive influence on the willingness of eBooks to trust eLoans, as the weight is set to 0.8. This means that as better security mechanisms are used to protect transactions and information moving over the network between eBooks and eLoans, trust in the external environment increases by 80%.

Population of the trust concept C₁₁: This concept is not populated by a fuzzy value, but is rather inferred from four trust concepts related to it, as shown in Figure 12.5.

It is important to note that all next trust concepts are automatically populated by values derived from an assessment process. Security mechanisms used by eLoans are the first trust concepts populated in this way. After eLoans registers with eBooks, the authorisation interface retrieves the interface policy of eLoans to inspect published information on security mechanisms that are supported by eLoans. Records are written to the information database, which specifies the types of mechanisms that are supported. The security mechanisms used in all SOAP interaction are *continuously* monitored and updated.

The creation of fuzzy values for trust concepts C₁₂ to C₁₅ is described next.

C₁₂ – Identity mechanisms

Causal relationship C₁₂ – C₁₁: The level to which identity mechanisms are implemented has a strong positive influence on the willingness of eBooks to trust eLoans, as the weight is set to 0.8. This means that as identity mechanisms of higher assurance are used, trust in the external environment increases by 80%.

Population of the trust concept C₁₂: eBooks supports any combination of username token, kerberos ticket or certificate as forms of identification, but prefers a certificate. The authorisation interface inspects the capabilities of eLoans and determines an initial level of compliance with its identity mechanism requirements. For instance:

- If eLoans identifies itself with a certificate, a record is written to the information database representing a HIGH level.

- If eLoans identifies itself with a kerberos ticket, a record is written to the information database representing a GOOD level.
- If eLoans identifies itself with a username token, a record is written to the information database representing a LOW level.

The fuzzification inspects records in the information database, recalculates a value for the trust concept by aggregating information, and stores a value representing any of the labels HIGH, GOOD or LOW. A value is stored in the field in the trust database to represent this trust concept. Each next interaction with eLoans is monitored to determine whether it complies with requirements. The value of the trust concept is adjusted accordingly. Depending on the type of identity mechanisms used, a threshold function activates the trust concept to either 1 or 0.

C₁₃ – Integrity mechanisms

Causal relationship C₁₃ – C₁₁: The level to which integrity mechanisms are implemented has a strong positive influence on the willingness of eBooks to trust eLoans, as the weight is set to 0.4. This means that as better integrity mechanisms are used, trust in the external environment increases by 40%.

Population of the trust concept C₁₃: eBooks may require that operations can be signed by either *xmldsig#sha1*, *xmldsig#base64*, *xmldsig#hmac-sha1* or *xmldsig#rsa-sha1*, but *xmldsig#sha1* may be preferred. eBooks inspects the capabilities of eLoans and determines an initial level of compliance with its integrity mechanism requirements. For instance:

- If eLoans uses *xmldsig#sha1*, a record is written to the information database representing a HIGH level.
- If eLoans uses *xmldsig#rsa-sha1*, a record is written to the information database representing a GOOD level.
- If eLoans uses *xmldsig#base64*, a record is written to the information database representing a LOW level.

The fuzzification process inspects records in the information database, recalculates a value for the trust concept by aggregating information, and stores a value representing any of the labels HIGH, GOOD or LOW. A value is stored in the field in the trust database to represent this trust concept. Each next interaction with eLoans is monitored to determine whether it complies with requirements. The value of the trust concept is adjusted accordingly. Depending on the type of integrity mechanisms used, a threshold function activates the trust concept to either 1 or 0.

C₁₄ – Confidentiality mechanisms

Causal relationship C₁₄ – C₁₁: The level to which confidentiality mechanisms are implemented has a positive influence on the willingness of eBooks to trust eLoans, as the weight is set to 0.4. This means that as better confidentiality mechanisms are used, trust in the external environment increases by 40%.

Population of the trust concept C₁₃: eBooks may require that operations are encrypted by either *xmlenc#tripleDES-cbc*, *xmlenc#aes128-cbc* or *xmlenc#rsa-1_5*, but *xmlenc#aes128-cbc* may be preferred. eBooks inspects the capabilities of eLoans and determines an initial level of compliance with its identity mechanism requirements. For instance:

- If eLoans uses *xmlenc#aes128-cbc*, a record is written to the information database representing a HIGH level.
- If eLoans uses *xmlenc#rsa-1_5* or *xmlenc#tripleDES-cbc*, a record is written to the information database representing a GOOD level.

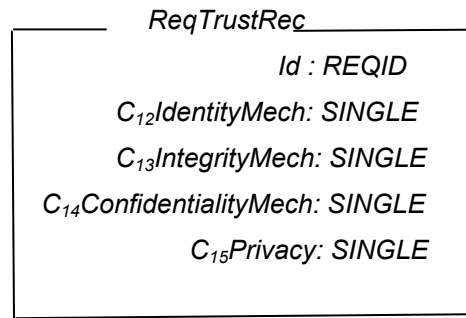
The fuzzification process inspects records in the information database, recalculates a value for the trust concept by aggregating information, and stores a value representing any of the labels HIGH, GOOD or LOW. A value is stored in the field in the trust database to represent this trust concept. Each next interaction with eLoans is monitored to determine whether it complies with requirements. The value of the trust concept is adjusted accordingly. Depending on the type of confidentiality mechanisms used, a threshold function activates the trust concept to either 1 or 0.

C₁₅ – Privacy

Causal relationship C₁₅ – C₁₁: The level to which privacy is implemented has a positive influence on the willingness of eBooks to trust eLoans, as the weight is set to 0.4. This means that if privacy is respected, trust in the external environment increases by 40%.

Population of the trust concept C₁₅: eLoans can publish an XML document in a policy, in for instance the Enterprise Privacy Authorisation Language (Ashley 2003), detailing how information is handled by its environment. An inspection of this policy by the authorisation interface results in records being stored in the information database. The fuzzification process analyses this information, and stores a value representing any of the labels LOW, MEDIUM, HIGH or VERY HIGH. The resultant value is to be stored in the trust database. Depending on the level of privacy mechanisms present, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following four additional fields:



12.5.3 Trust in eLoans

The focus of trust in eLoans is to determine its trustworthiness as an independent entity. As eBooks interacts with eLoans, it gains information about characteristics of eLoans that have been organised according to its compliance with agreements, competence and predictability. Over time, the establishment of these characteristics leads to a measure of goodwill. The trust in eLoans (C_4) is inferred from related trust concepts as depicted in Figure 12.6.

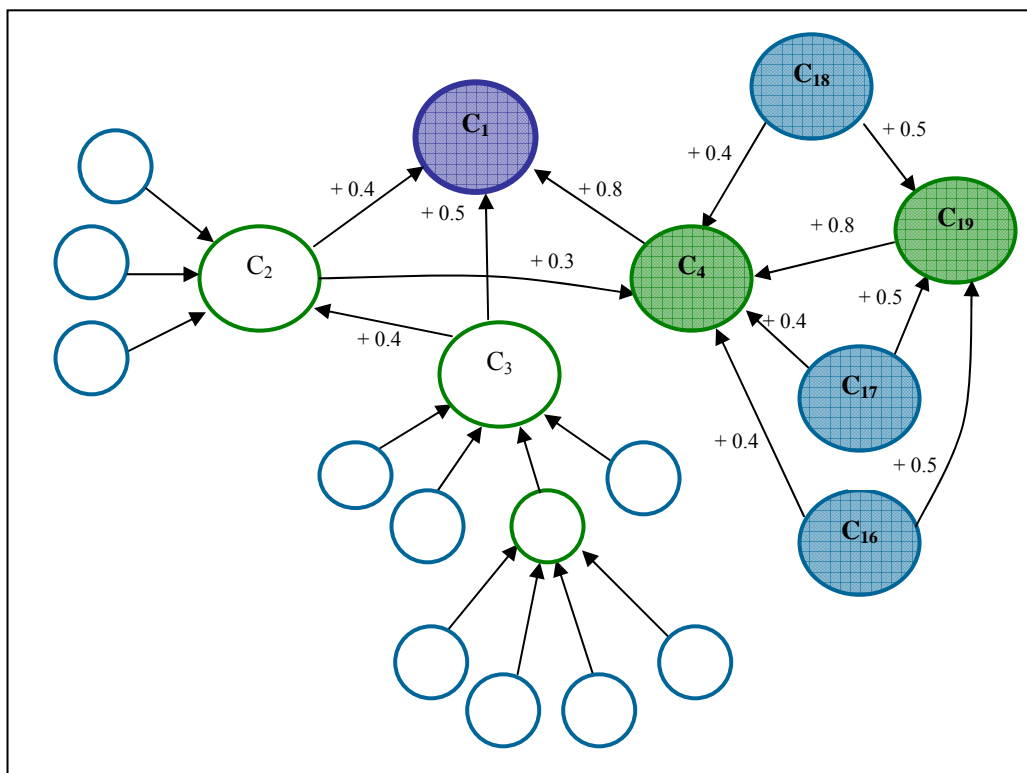


Figure 12.6: Trust in eLoans

The concepts representing each node are the following:

- C_{16} – Compliance of eLoans with agreements
- C_{17} – Competence of eLoans
- C_{18} – Predictability of eLoans
- C_{19} – Goodwill developed towards eLoans

C₁₆ – Compliance with agreements

Causal relationship C₁₆ – C₄: The compliance of eLoans with agreements has a positive effect on trust in it, as the weight is set to 0.5. This means that the more it complies with agreements, the more trustworthy it is and trust in eLoans increases by 50%.

Population of the trust concept C₁₆: Compliance with agreements is the belief that eLoans is honest in its interactions with eBooks. To ensure quality of service, a web services requestor and provider jointly define a machine-readable service level agreement (SLA) as part of a service contract that can be monitored by one or both parties. Such agreements are defined in XML-based policies with either WS-Policy or WSLA, and are monitored. The compliance of web services requestors with agreements can be determined by inspecting a large variety of parameters.

For this discussion, eBooks monitors the following two parameters:

- The SLA may state that no more than 10 transactions are allowed per minute. If this restriction is exceeded, a record is written to a database to indicate the level of transgression by using a factor between 1 and 10, where 10 represents the worst case.
- Security requirements may be stated in WS-policy documents. eBooks may for instance state that all communication for a specific web services operation must be encrypted with AES. If eLoans does not adhere to this requirement, it can be recorded to a database as an improper event.

Records are written to the information database according to a predefined set of rules. Records are aggregated to a value of between 0 and 1 to indicate the level of compliance and are saved by the fuzzification process in the trust database. For instance, if a value of 0.1 is derived, the level of compliance with agreements is LOW. A value of 0.8 means that the level of compliance with agreements is VERY GOOD. Depending on the level of compliance with agreements, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional fields:

<i>ReqTrustRec</i>
<i>Id : REQID</i>
<i>C₁₆CompIToAgreem: SINGLE</i>

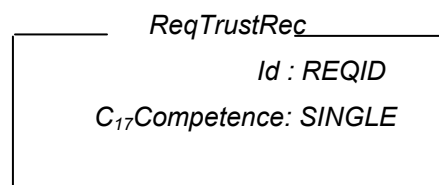
C₁₇ – Competence

Causal relationship C₁₇– C₄: The competence of eLoans has a positive effect on trust in it, as the weight is set to 0.5. This means that the more competent eLoans is, the more trustworthy it is and trust in it increases by 50%.

Population of the trust concept C₁₇: Competence is the belief that eLoans has the ability or necessary skills to perform a task. The competence of eLoans can be determined by evaluating recommendations and references that are submitted. Certificates state competence levels such as credit ratings, audit information, endorsements, ISO 9000 certification, privacy seals, or Better Business Bureau statements. Recommendations from other entities are also used to determine competence, but are only accepted from entities that are identified by a public key. The trust in the public key determines the weight assigned to a recommendation.

Recommendations, references and other statements are evaluated and written to the information database by the authorisation manager. The fuzzification process reads these records and derives a value of between 0 and 1 to populate node C₁₇. For instance, if a value of 0.1 is derived, the competence of eLoans is LOW. A value of 0.8 means that the level of competence is VERY HIGH. Depending on the level of competence, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional fields:



C₁₈ – Predictability

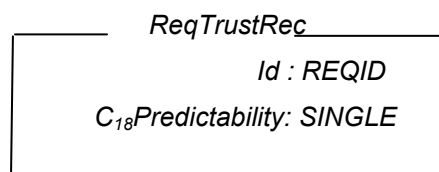
Causal relationship C₁₈ – C₄: The predictability of eLoans in respect of agreements has a positive influence on trust in it, as the weight is set to 0.5. This means that the more predictable eLoans is, the more trustworthy it is and trust in it increases by 50%.

Population of the trust concept C₁₈: Predictability is the belief that the actions of eLoans are consistent, so that a correct forecast can be made about how eLoans will behave in a given situation. This can be achieved by inspecting SOAP messages that are sent and received, and by recording for instance the number of messages, the number of messages in error, the value of transactions, date of transaction fulfilments, and the validity of message details. The focus of this

monitoring is to determine a score for each interaction from the time of its initiation until its fulfilment. If, for instance, an invalid credit card is submitted, the level of transgression is recorded by taking into account the value of the transaction.

Records are written to the information database for different types of interactions and events. The fuzzification process evaluates each interaction and creates a score of between 0 and 1 to populate node C_{18} . For instance, if a value between 0.5 and 0.65 is derived, the predictability of eLoans is GOOD. A value of 0.8 means that the level of predictability is HIGH. Depending on the level of predictability, a threshold function activates the trust concept to either 1 or 0.

A record is updated in the trust database with the following additional fields:



C₁₉ – Goodwill

Causal relationship C₁₉– C₄: The goodwill held towards eLoans has a strong positive influence on trust in it, as the weight is set to 0.8. This means that as goodwill increases, trust in eLoans increases by 80%.

Population of the trust concept C₁₉: Goodwill is the belief that eLoans cares about the welfare of eBooks. It is not established by an assessment of information, but is rather developed over time as eBooks realises the benefits gained from increased cooperation with eLoans. Thresholds are set for compliance with agreements, competence and predictability to computationally infer goodwill.

Causal relationships $C_{16} - C_{19}$, $C_{17} - C_{19}$, and $C_{18} - C_{19}$ reflect the fact that as eLoans complies with agreements, is found to be competent, and is regarded predictable, goodwill increases in each case by 0.5. This means that HIGH trust in eLoans can only be achieved if it consistently behaves well over a period.

12.6 TRUST INFERENCE IN WSACT

The earlier discussion illustrates the fact that the fuzzy cognitive map directly supports the approach towards trust assessment followed by this thesis, as knowledge – accumulated by components of the WSACT model – is exploited by its interactive structure. The structure and population of nodes of the fuzzy cognitive map have now been described. To determine how a web services provider can establish trust with others according to the structure of its fuzzy cognitive map, the inference of trust types and a trust level are considered next. Two important aspects need to be considered: the computation and the threshold function that transform values into the range [0, 1].

12.6.1 Computation

A fuzzy cognitive map consisting of n concepts is represented by a $1 \times n$ state vector A , which gathers the values of the n concepts; and by an $n \times n$ edge matrix E , with elements e_{ij} . The value of e_{ij} indicates how strongly concept C_i influences C_j . A_i represents the level of activation of a node. A discrete time simulation is performed by iteratively applying a summation and threshold process to state vector A . The activation level A_i for each concept C_i is calculated by rule 1 below (Kosko 1997):

$$A_i = f \left(\sum_{j=1}^n A_j e_{ij} \right) \quad (1)$$

A_i is the activation level of concept C_i at time $t+1$ and A_j is the activation of concept C_j at time t . f is a threshold function that transforms the summation into the interval [0,1].

Rule 1 focuses on the effect that interconnected concepts have on each other. To exploit knowledge and past experience that has been accumulated over time, it is now proposed that rule 1 is adapted, as suggested by Groumpos and Stylios (2004). The adaptation is shown in rule 2.

$$A_i = f \left(\sum_{j=1}^n A_j e_{ij} + \gamma A_i \right) \quad (2)$$

Rule 1 is adapted by including γA_i , where A_i is the activation of C_j at time t to represent the previous value of A_i . The coefficient γ represents the participation of the past experiences that are embodied by a trust concept in the calculation of the new value of a trust concept. Coefficient γ can take values in the interval $0 \leq \gamma \leq 1$, and can vary according to the desirable contribution of

the previous value in different circumstances, so the value of γ can change over time, so that $\gamma = \gamma(t)$. It has been proposed that during the training period of the fuzzy cognitive map, the value of γ should be near to 1, as a greater influence of past experiences ensures a smoother change in new values.

The current research proposes an adaptation to the manner in which rule 2 is used. In order to incorporate humanistic trust formation, γ is not set to 1 for all trust concepts, but is rather activated to 1 only for trust concepts that have been activated to 1 by the fuzzification process. The adaptation is included in rule 3.

$$A_i = f\left(\sum_{j=1}^n A_j e_{ij} + \gamma A_i\right) \quad \text{where } \gamma = 1 \text{ if } A_0 = 1 \quad (3)$$

The participation of past experience is thus dynamically imprinted into the state of the fuzzy cognitive map, to produce a trust level that more accurately reflects the influence of accumulated knowledge.

12.6.2 Threshold

Concept values are expressed on a normalised range denoting a *degree of activation* rather than an exact quantitative value. This is achieved by a threshold function that forces the concept value from unbounded values into a strict range. It is then possible to compare the state of nodes of the fuzzy cognitive map with one another. This mapping can also be seen as a variation of the fuzzification process in fuzzy logic. There are numerous threshold functions that can be used. For this research, the binary function, and logistic function (Kosko 1992; McNeill 2003) is considered. The purpose of the fuzzy cognitive map for web services trust will now be evaluated to determine the threshold function to be used.

12.6.2.1 Binary function

The simplest case is the binary function. This function is used to approximate concept-firing behaviour. The threshold takes on values 0 and 1 as follows:

$$f(x_i) = 0, \quad x_i \leq 0$$

$$f(x_i) = 1, \quad x_i > 0$$

Any positive nonzero concept value is forced to one, and any other concept value is forced to zero. The function is therefore used to activate the state of each trust concept, before the fuzzy cognitive map is run. Since the distinction is only between active and non-active states, this function is not suited to compute a trust level that needs to show a gradual increase in trust as experiences and evidence are accumulated.

12.6.2.2 Logistic function

The logistic function remains the most popular activation function (Kosko 1992). The fact that it is considered as the *maximum-entropy* function may account for its popularity. It creates a range of concept activation values between 0 and 1 and produces more variation in subsequent state vectors by means of the following calculation:

$$f(x_i) = \frac{1}{1 + e^{-cx_i}}$$

Here, c is a constant such that $c > 0$, where c determines the curve of the function. For this research, $c = 0.2$ is used, as it produces a wide range of trust levels, as illustrated by figure 12.7. For $c = 0.8$, $f(x_i)$ quickly approaches 1, and the range is limited for larger values of x . For $c = 0.1$, $f(x_i)$ does not approach 1, limiting the range of trust levels.

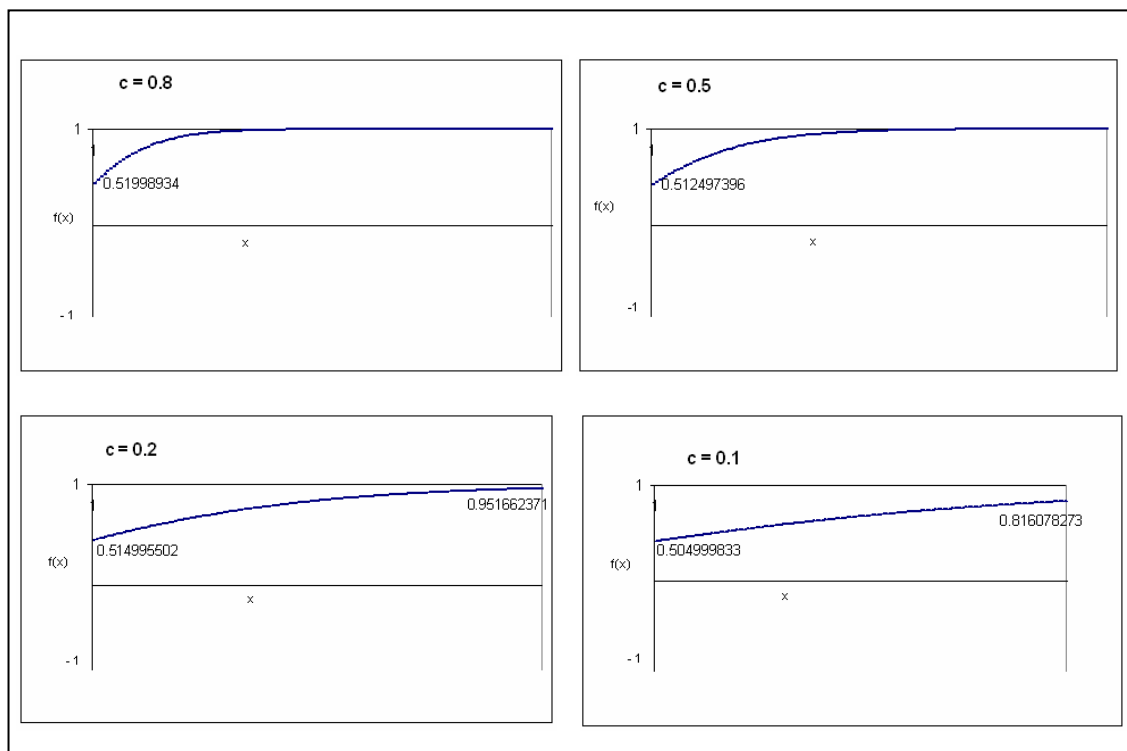


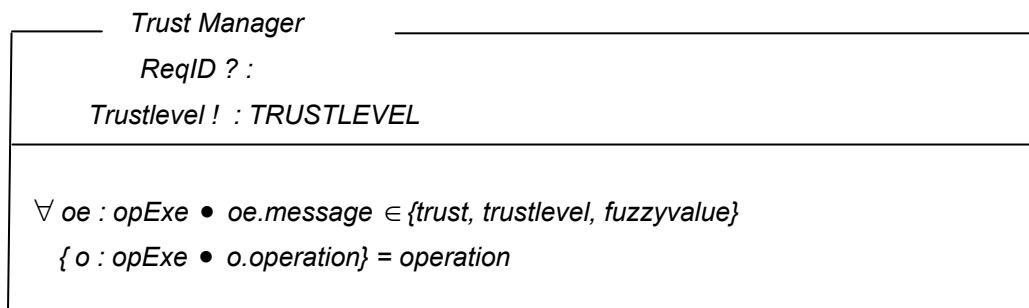
Figure 12.7: Activation by means of the logistic function for $c = 0.8, 0.5, 0.2$ and 0.1

Next, the formal specification of the trust manager is given in Z.

12.7 THE ABSTRACT STATE SPACE OF THE TRUST MANAGER

State changes that need to be modelled include the request to calculate the trust level for a web services requestor; the return of a trust level to the authorisation manager; trust computation by the fuzzy cognitive map; and the fuzzification of values for trust concepts. As before, the abstract state space of the trust manager is considered as a collection of data. The state of the system changes when operations are executed over data.

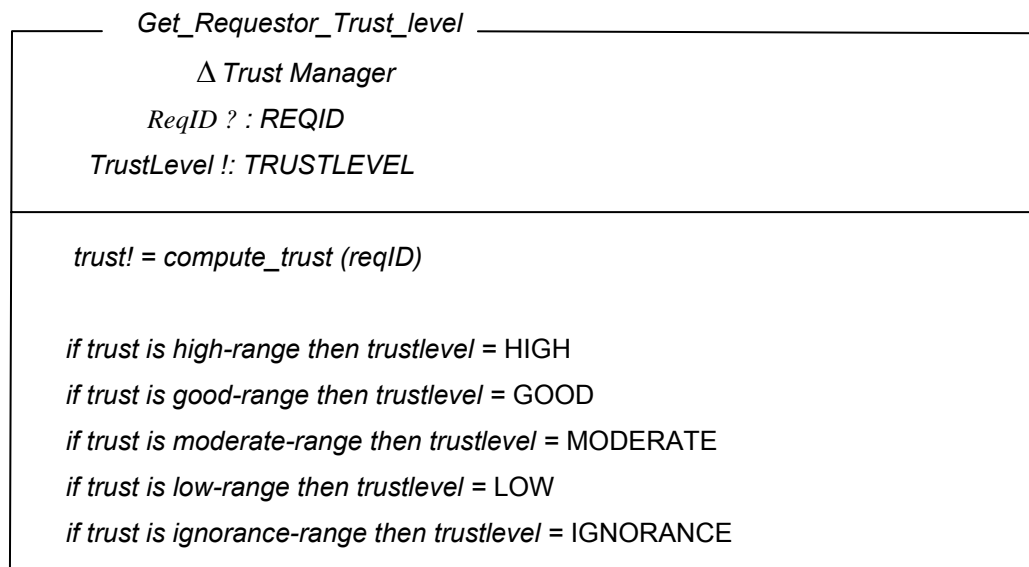
A set of $[OPERATION]$ can be used to refer to all operations of the trust manager. The result of these executions is the requested trust level.



The architecture defined in Figure 12.2 makes explicit all components that contain required operations of the trust manager. Operations needed are *Get_Requestor_Trust_Level*, *Compute_Trust*, and *Fuzzify_Trust_Concept*. These operations are defined in the paragraphs that follow:

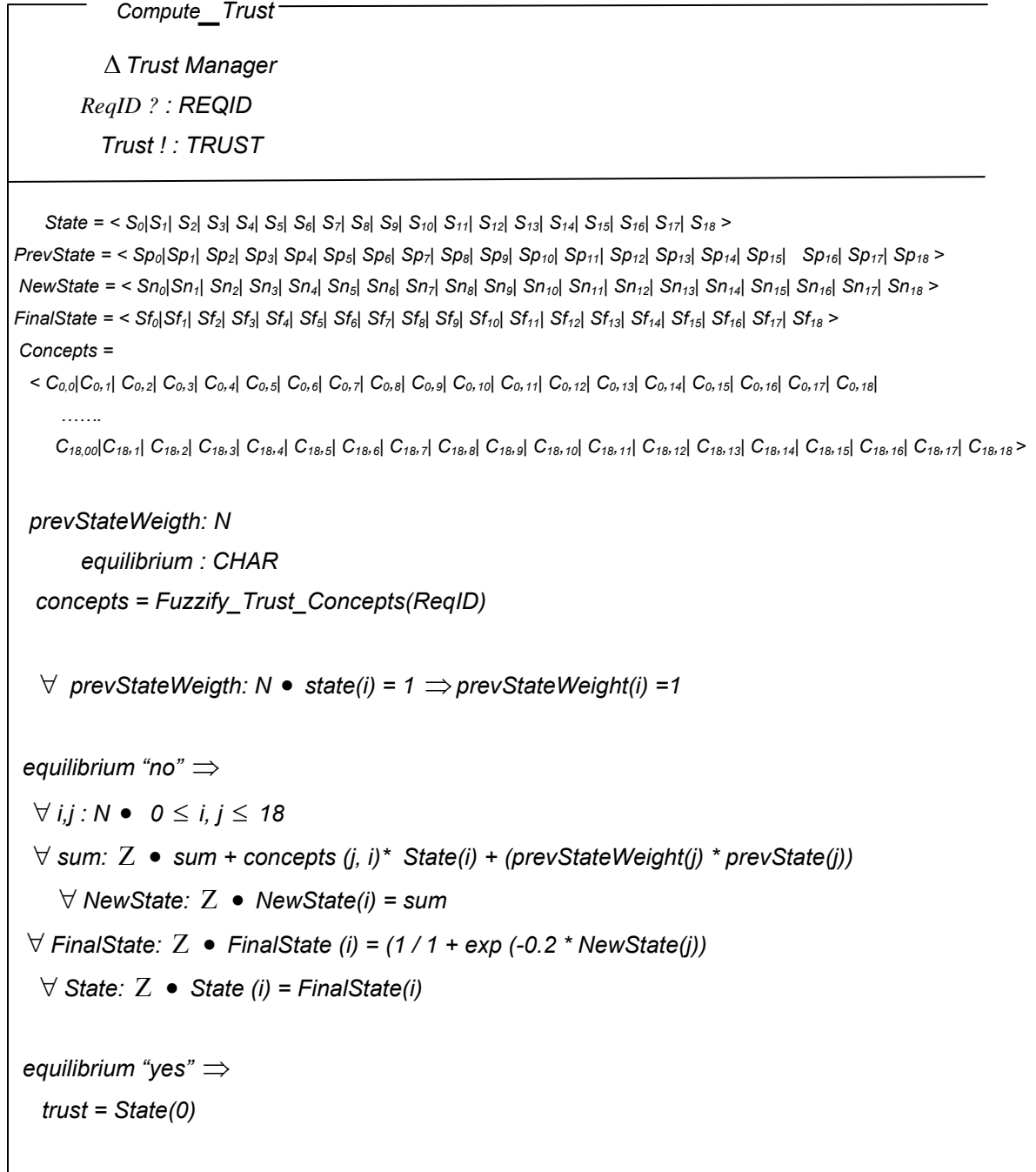
12.7.1 Get_Requestor_Trust_Level

The aim of this operation is to determine the trust level of a web services requestor in terms of linguistic variables.



12.7.2 Compute_Trust

The aim of this operation is to compute the trust level of a web services requestor by invoking the fuzzy cognitive map of a web services requestor. The operation is defined as follows:



12.7.3 Fuzzify_Trust_Concept

The aim of this operation is to get a fuzzy value in the range [0,1] for each of the trust concepts that are populated automatically. The specification is generalised for each trust concept to be fuzzified.

<p><i>Fuzzify_Trust_Concept</i></p> <hr/> <p>Δ <i>Trust Manager</i></p> <p><i>ReqID</i> ? : <i>REQID</i></p> <p><i>concept</i> ! : <i>Z</i></p> <hr/> <p><i>ReqRecs</i> \triangleright (<i>ReqID</i>)</p> <p>\forall <i>concept</i>: <i>Z</i> • <i>concepts</i>(<i>i</i>) = <i>fuzzify_info</i>(<i>ReqRecs</i>)</p>

The trust manager operations have now been specified in Z. Next follows a discussion of the computation of a trust level.

12.8 TRUST LEVEL COMPUTATION

The computation of a trust value that reflects the gradual increase in trust is illustrated in the paragraphs that follow. Trust concepts are populated consecutively from trust in the internal environment, trust in the external environment, and trust in the other party. The focus of this discussion is to determine ranges of trust values for the trust level.

Table 12.1 presents the consecutive increase in the trust value, as trust concepts are activated one after the other. Chapter 13 - the prototype - provides more detailed descriptions on changes in state vector A. In reality, any trust concept may be populated at any time. This aspect will be considered in the next paragraph.

Table 12.1: Increase in trust

Trust concept activated	Trust value
None	0,55
C ₅ - Level of vulnerabilities	0,58
C ₆ - Successes in dealing with risks and compromises	0,61
C ₇ - Complexity	0,64
C ₈ - Rule of law	0,67
C ₉ - Assurances	0,71
C ₁₀ - Compliance	0,75
C ₁₂ - Identity mechanisms	0,79
C ₁₃ - Integrity mechanisms	0,82
C ₁₄ - Confidentiality mechanisms	0,85
C ₁₅ - Privacy	0,88
C ₁₇ - Compliance with agreements	0,9
C ₁₆ - Competence	0,92
C ₁₈ - Predictability	0,94

An analysis of these results leads to the following intuitive conclusion about the range for each trust level.

IGNORANCE:	$trustlevel < 0.55$
LOW:	$0.55 \leq trustlevel \leq 0.64$
MODERATE:	$0.64 < trustlevel \leq 0.75$
GOOD:	$0.75 < trustlevel \leq 0.84$
HIGH:	$trustlevel \geq 0.85$

12.8.1 Ad hoc population of trust concepts

To determine how well the trust range applies when trust concepts are not populated consecutively, consider the case of web services requestor eCompany. eCompany presents credentials to increase its competence ($C_{17} = 1$). Other than that, no other information about eCompany is known. The trust in the internal environment of eBooks (C_5 and C_7) has been established. Trust (C_1) is computed to be 0.64, which equates to *low* trust, but borders on *moderate* trust. This seems reasonable, as a recommendation or reference may ensure that transactions of a generic nature may be conducted between eBooks and eCompany. Figure 12.8 shows state vector A as it reaches equilibrium.

State changes in trust, trust types and trust concepts

C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C0:	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
C1:	0.5	0.54	0.5	0.52	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.52
C2:	0.68	0.68	0.69	0.7	0.65	0.65	0.65	0.65	0.65	0.65	0.69	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.68
C3:	0.63	0.63	0.63	0.66	0.57	0.57	0.57	0.57	0.57	0.57	0.64	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.63
C4:	0.65	0.64	0.65	0.67	0.6	0.6	0.6	0.6	0.6	0.6	0.65	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.64
C5:	0.64	0.63	0.64	0.66	0.58	0.58	0.58	0.58	0.58	0.58	0.64	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.63
C6:	0.64	0.63	0.64	0.66	0.59	0.59	0.59	0.59	0.59	0.59	0.64	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.64
C7:	0.64	0.63	0.64	0.66	0.59	0.59	0.59	0.59	0.59	0.59	0.64	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.63
C8:	0.64	0.63	0.64	0.66	0.59	0.59	0.59	0.59	0.59	0.59	0.64	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.64

Figure 12.8: Activation of state vector A by C_5 , C_7 and C_{17}

If, on the other hand, implemented security mechanisms are activated to 1 ($C_{12} - C_{15}$), trust grows to 0.79, to reflect *good* trust. As the nature of such transactions dictates that a secure and safe environment is critical when transactions of a sensitive nature are conducted, this seems to be a fair conclusion. Figure 12.8 shows state vector A as it reaches equilibrium.

State changes in trust, trust types and trust concepts

C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C0:	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0
C1:	0.5	0.54	0.5	0.52	0.5	0.5	0.5	0.5	0.5	0.5	0.6	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.52
C2:	0.83	0.83	0.83	0.84	0.8	0.8	0.8	0.8	0.8	0.8	0.83	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.83
C3:	0.73	0.72	0.73	0.75	0.67	0.67	0.67	0.67	0.67	0.67	0.73	0.67	0.67	0.67	0.67	0.67	0.67	0.67	0.73
C4:	0.8	0.79	0.8	0.81	0.75	0.75	0.75	0.75	0.75	0.75	0.8	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.79
C5:	0.77	0.77	0.77	0.79	0.72	0.72	0.72	0.72	0.72	0.72	0.78	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.77
C6:	0.79	0.78	0.79	0.81	0.74	0.74	0.74	0.74	0.74	0.74	0.79	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.78
C7:	0.78	0.78	0.79	0.8	0.73	0.73	0.73	0.73	0.73	0.73	0.79	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.78
C8:	0.79	0.78	0.79	0.81	0.74	0.74	0.74	0.74	0.74	0.74	0.79	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.78

Figure 12.8: Activation of state vector A by C_5 , C_7 , C_{12} , C_{13} , C_{14} , C_{15} , C_{17}

These results show that reasoning over trust concepts to form a trust level can be implemented successfully with fuzzy cognitive maps. Administrators of a web service may need to perform

various experiments to determine the correct weights required for the fuzzy cognitive map of a web service.

12.9 CONCLUSION

This chapter dealt with the computation of trust for web services entities and commenced by discussing the components of the trust manager. The structure of the main component – the fuzzy cognitive map – was described in terms of the implementation of trust, trust types and trust concepts. All nodes were described, as well as causal weights.

The population of trust concepts for each of the three trust types was subsequently discussed. The aim was not to provide an exhaustive treatment of each trust concept, but rather an overview of how a trust concept can be populated. Some trust concepts were populated by means of administrator intervention and others by means of an automated process of trust assessment.

Trust inference was next addressed, and a modified rule was used to activate trust. The research in hand adapted this rule to implement the dynamic way in which trust is inferred. The threshold function that is used to determine a gradual increase in trust was described, after which the main components of the trust manager were also specified in Z. Ranges of trust levels were finally identified by experimentation.

13

WSACT Prototype Implementation

This chapter constitutes the conclusion of Part II of the thesis in hand through a demonstration of the deployment of the WSACT model in a prototype implementation. The prototype is a scaled-down version of the WSACT model. It consists of three components, following the design of the WSACT model. The implementation of the authorisation interface, authorisation manager, trust manager, and their integration in the web services environment make up the focus of this chapter. The chapter starts with a discussion of the aims of the prototype. Since the implementation tools of components differ, an overview of the implementation process is provided and the operation of the prototype is subsequently illustrated by means of the case study example. The three trust types are discussed as their related trust concepts are populated automatically by the trust assessment process and manually by administrator intervention.

13.1 THE AIM OF THE PROTOTYPE

The WSACT prototype aims to demonstrate the following features of the WSACT model:

- Attribute-based access control, used in conjunction with the trust level of a web services requestor.
- The automated assessment of trust information.
- The automated computation of a trust level for web services entities.

The focus of the prototype is on the calculation of the trust level of web services requestors and its use in the access control policy. In order to simplify implementation, the prototype was restricted to use only a representative sample of the information required for trust calculation.

13.2 IMPLEMENTATION OVERVIEW

The prototype was developed on the Microsoft ASP.NET (MS ASP.NET 2005) platform, as it provides built-in support for building and consuming standards-based web services. All code is

written in VB.NET (MS VB.NET 2005) and in Amzi! Prolog (AMZI 2005) as it can be used with the .NET platform. It is the view of the researcher that the WSACT model is indeed viable for implementation in the real world, as the nature of tool support ensured that the prototype was developed straightforwardly.

For the case study, the portals of eLoans and eCompany were both implemented in ASP.NET. The WSDL document of eBooks was used at both portals to create proxy classes in order to communicate with web services operations of eBooks. The proxy classes send SOAP request messages over HTTP to eBooks. By doing so, web services operations are invoked, as if they are local to the applications of eLoans and eCompany. Web services operations of eBooks are located behind the IIS (Internet Information Server 6.0) (MS IIS 2005) web server that is used by the prototype. The web server receives SOAP request messages as part of the HTTP POST request. It forwards these requests for processing to a web services request handler, which is part of the .NET framework. The request handler is responsible for parsing the SOAP request, and for invoking the web services operation. Each request is intercepted by the prototype implementation of the WSACT model before request processing is allowed. The implementation of each of the components of the WSACT model, executed before web services operations are invoked, is briefly described next.

13.2.1 Authorisation interface

The authorisation interface is the first component that intercepts and processes the SOAP request message. It is implemented as a class in VB.NET. Request interception is performed by creating SOAP extension classes that are placed between the request handler and the web services operation. The ASP.NET SOAP extension architecture provides an extension that can inspect or modify a message at specific stages in message processing on either the web services requestors or provider. The authorisation interface is thus implemented as a class that inherits from `System.Web.Services.Protocols.SoapExtension`, an abstract SOAP extension class.

The authorisation interface is inserted in the request-processing stream between the request handler and all web services operations for eBooks. This is done by adding entries to the appropriate `web.config` file as shown in Figure 13.1. An important benefit that is derived from this implementation is that the authorisation interface can be used in conjunction with the implementation of authentication, confidentiality and integrity mechanisms as specified by the WS-Security specification, as it makes use of the same architecture on the .NET platform. By setting a different priority on the implementation of each security mechanism, the order of events is controlled.

```
<webServices>
  <soapExtensionTypes>
    <add type="eBooks_WebService.AuthorizationInterface, eBooks_WebService"
        priority="1" group="0" />
  </soapExtensionTypes>
</webServices>
```

Figure 13.1: Web.config file of eBooks

The authorisation interface processes the SOAP request message before it is deserialised into types that can be understood by the ASP.NET environment. The header and body of the SOAP request message are inspected, so that information such as credentials, declarations and trust information can be extracted. Next, a declaration or credential is dynamically asserted to the access control policy of the authorisation manager, if present in the SOAP header. Finally, the authorisation manager is called with an access request. The result that is returned from the authorisation manager is either true or false. If the result is true, the request handler invokes the web services operation and creates the SOAP response. The web server takes the SOAP response and sends it back to the web services requestor as part of the HTTP response. If the result is false, a SOAP exception which indicates that the request has been denied permission is returned as a SOAP fault to the web services requestor as part of the HTTP response.

13.2.2 Authorisation manager

The access control policy of the authorisation manager is defined in Datalog. As Datalog is a subset of Prolog, Datalog statements can be parsed and executed by a Prolog interpreter. For the purpose of this prototype, the authorisation manager is defined in Prolog, a logical programming environment. A difference between Datalog and Prolog is that the order of statements in Prolog must carefully be considered.

The authorisation manager is developed in Amzi! Prolog (AMZI 2005). At the core of Amzi! Prolog is a runtime engine that can load and run compiled Prolog code. Compiled Prolog is stored in binary files that can be loaded by the Prolog engine. The access control policy is defined in a Prolog source file, which is compiled into a byte code file. All byte code files are linked to a file called AuthManager.xpl, which is called from the authorisation interface.

The interface from the authorisation interface to the authorisation manager is through the Logic Server of Amzi! Prolog. The Logic Server is the Prolog runtime engine, implemented as a Dynamic Link Library (DLL). It contains various function calls that enable the links between the authorisation interface and the authorisation manager. It provides the VB.NET authorisation interface with the ability to query access control policy rules of the authorisation manager, and retrieve an answer of either true or false as is required. Because the authorisation manager is

compiled, it executes extremely fast. This is because the declarative Prolog logic is compiled to run on a specialised Prolog virtual machine. Ideally, the authorisation manager can be used as an authorisation manager in a web environment, as multiple simultaneous authorisation managers can be executed in multiple threads.

13.2.3 Trust manager

The trust manager is implemented as a class in VB.NET. During the execution of rules of the access control policy, the authorisation manager needs to determine the trust level of a web services requestor. This is determined by calling a method of the trust manager class. A custom built-in Prolog predicate called *GetTrust* is defined for this purpose. The Logic Server has an interface for implementing custom built-in predicates called extended predicates, which can be implemented in languages that support function pointers. This makes it possible for Prolog to directly access methods of the trust manager class. The *GetTrust* extended predicate is added to the access control policy of the authorisation manager with the API function *AddPred*.

The trust manager is invoked by the first term of the *GetTrust* predicate, which contains the identity of the web services requestor. It computes the trust level for a web services requestor by consulting the fuzzy cognitive map that is implemented as a method of the trust manager. The fuzzy cognitive map is a simple structure that is mathematically represented with matrix vectors. The resulting vector of each fuzzy cognitive map cycle is computed from the multiplication of a vector with a matrix. This means that changes that occur in trust, trust types and trust concepts do not need the complete reconstruction of the structure, as they occur in the same matrix. As the computational cost of each cycle of the fuzzy cognitive map of a web services requestor is minimal, parallel processing can be implemented to increase the efficiency of the trust manager. When the cycles of the fuzzy cognitive map reach equilibrium, a trust level is determined. The trust level is unified with the *GetTrust* predicate so that it can be used in access control reasoning.

13.3 PROTOTYPE OPERATION

The prototype is web-based. A link to it can be found at http://csweb.rau.ac.za/staff/marijke/wsact/wsact_prototype.htm To make it user-friendly, links to the portals of eLoans and eCompany, the administrative trust interface of eBooks, and the initialisation of trust information are provided from this URL, shown in Figure 13.2.

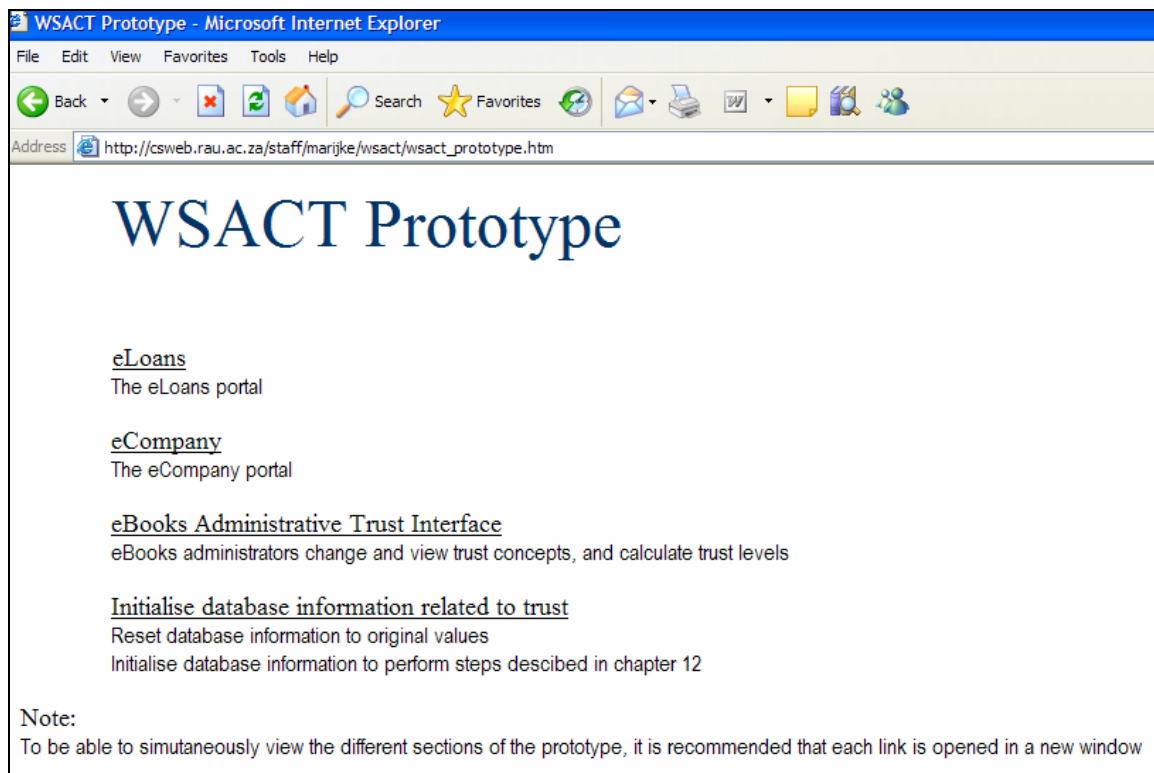


Figure 13.2: WSACT prototype

The case study is now revisited to illustrate the operation of the WSACT model. The discussion focuses on the two web services requestors – eLoans and eCompany – and on the web services provider eBooks. The discussion is focused on the automated assessment of information and the computation of a trust level, and the effect thereof on access control. The discussion is structured as follows:

- The trust manager – from where trust concepts and causal weights can be adjusted, and from which a trust level can be computed.
- Trust in the external environment between eBooks and eCompany – where the resulting trust level is computed to be at a *moderate* level – and its effect on access control.
- The trust in eLoans – where the trust level is computed to be *good* – and its effect on access control.
- Trust in the internal environment of eBooks – where the trust level is finally computed to be *high* as the result of manual administrator intervention.

13.3.1 The administrative trust interface of eBooks

The first component to be investigated is the trust manager. Administrators of eBooks have a web interface to view trust concepts and weights of causal relationships of each of the fuzzy cognitive maps that are stored in the trust database and to compute trust levels for web services requestors. The interface is accessed by selecting the option *eBooks Administrative Trust*

Interface from the first page of the prototype. The first page of the administrative interface is shown in Figure 13.3.

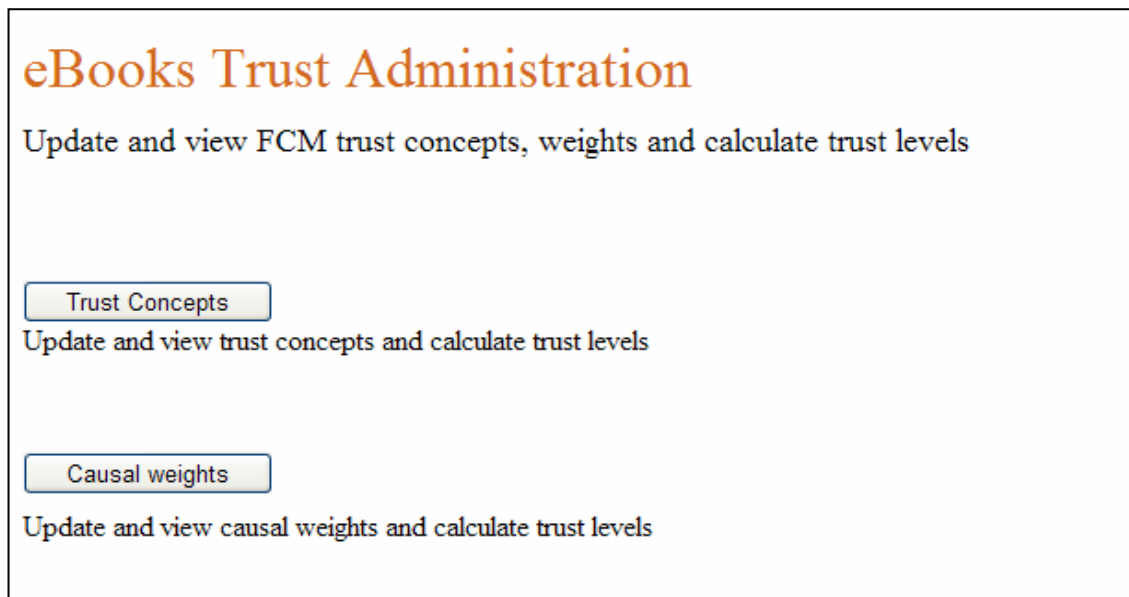


Figure 13.3: eBooks Administrative Trust Interface

As mentioned, some nodes of the fuzzy cognitive map are updated by an automated process of trust assessment and others manually by an administrator. To enable the simulation of different scenarios, administrators are given the ability to view and update all nodes and weights of the fuzzy cognitive map from this interface. In a real environment such abilities will be controlled with utmost care and they will only be available to persons of high responsibility. Figure 13.5 shows the page used to update nodes of the FCM representing trust concepts.

If no trust concepts have been set to 1, the computed trust level is 0.55, which represents *ignorance*, as defined on page 184 in Chapter 12. Next, trust relationships with eLoans and eCompany are considered.

13.3.2 Trust relationships of eBooks with eLoans and eCompany

For this discussion, both eLoans and eCompany have already registered with eBooks. Assume that there exists a person named Sue Smith, shown in Figure 13.4. She is both an employee of eCompany and a student registered with eInstitution, and the latter issued her with a digital credential as proof of registration. Because she is registered as a student, she has been granted a study loan by eLoans.

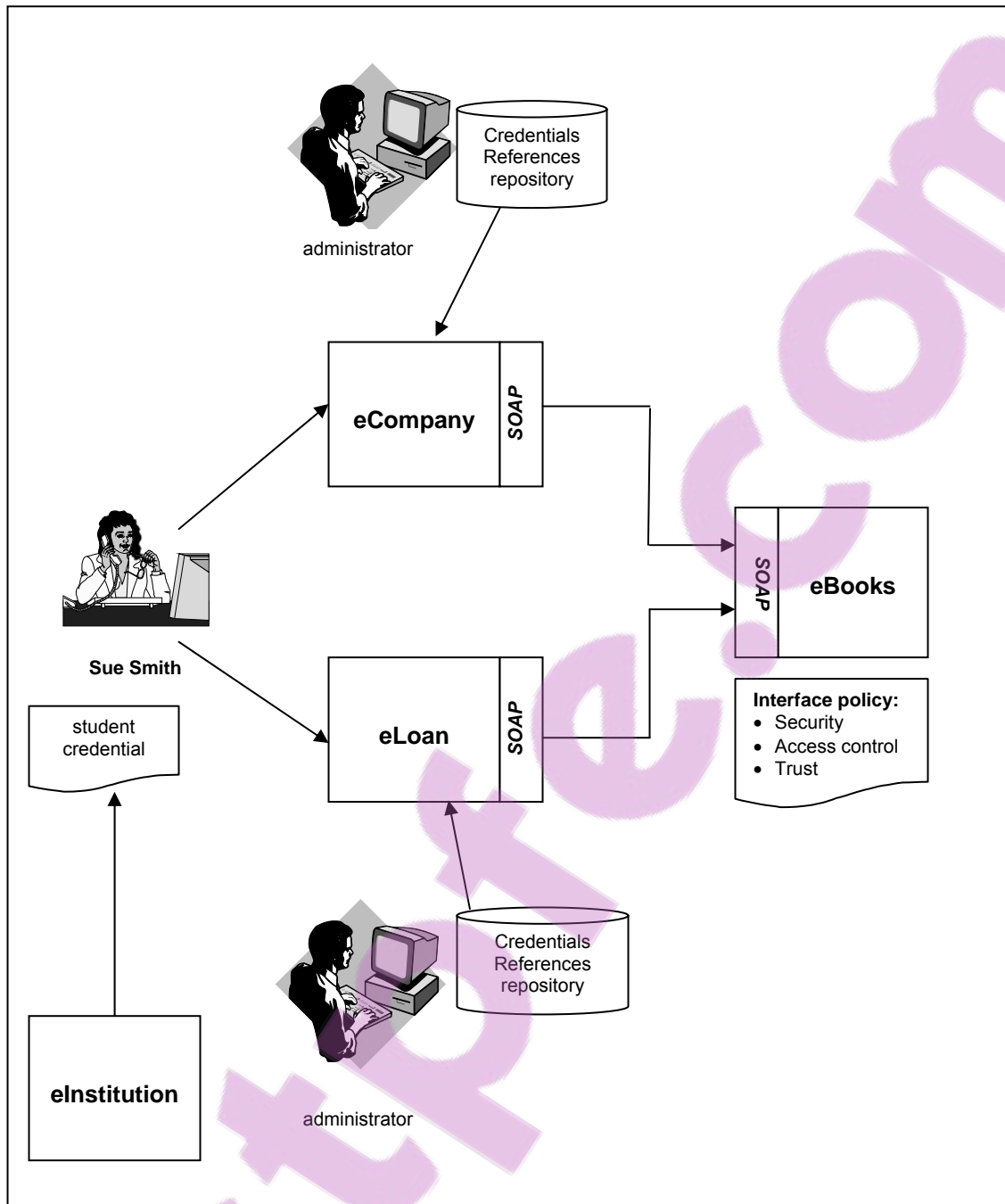


Figure 13.4: Sue Smith as participant in two virtual applications

eCompany and eLoans have in their possession credentials and references that they can use to increase their trust with eBooks. eBooks selectively reveals trust, access control and security requirements to others by its interface policies. This information allows an administrator shown in Figure 13.4 to submit credentials and references as is deemed necessary. Information is programmatically appended to SOAP headers, and the SOAP request message is protected by security mechanisms.

The discussion now sets out to illustrate the aims of the prototype that were defined at the start of the chapter. The trust relationship between web services requestors and providers, as well as the access granted to Sue Smith (based on the application from where she accesses eBooks), is a focus of this discussion. eCompany and eLoans are discussed consecutively, and the different features of the prototype are highlighted.

13.3.3 Trust in the external environment between eLoans and eCompany

This scenario highlights the second trust type, namely the trust in the external environment between the web services requestor and provider. It is illustrated by means of the trust relationship between eBooks and eCompany, and the consequent access allowed to Sue Smith, an employee of eCompany. The main focus is the trust in web services requestors when nodes $C_8 - C_{15}$, related to trust in the external environment C_2 , are populated by an automated process.

As mentioned earlier, eCompany registered with eBooks. During the registration process the node representing identity (C_{12}) is automatically set to 1. This occurs if the authentication mechanism used by eCompany fosters trust and complies with authentication requirements defined in the interface policy. To be able to view all trust concepts, the *Trust concepts* link can be selected from the *eBooks Administrative Trust Interface* page shown in Figure 13.2. Figure 13.5 shows trust concepts for both eCompany and eLoans. If C_{12} is 0, it must be manually set to 1 with the *Edit* button to perform the required calculation.

Trust concepts of the FCM Trust concepts can be reset from the "Initialise Database Information" page

	ReqId	C_5	C_6	C_7	C_8	C_9	C_10	C_12	C_13	C_14	C_15	C_16	C_17	C_18
<input type="button" value="Edit"/>	eCompany	0	0	0	0	0	0	1	0	0	0	0	0	0
<input type="button" value="Edit"/>	eLoans	0	0	0	0	0	0	0	0	0	0	0	0	0

Tip: Enter eLoans or eCompany

Figure 13.5: Activation of trust concept related to identity (C_{12})

Generally, trust is built over the cryptographically verified identity of the other party. If the node representing the identity (C_{12}) is set to 1, trust is computed to 0.58 as shown in Figure 13.6. The trust level is computed by selecting the *Calculate trust for:* link after entering "ecompany" in the text box. Figure 13.6 shows all state changes in trust, trust types and trust concepts where each column represents a trust concept from C_1 to C_{19} . Shaded columns represent trust and trust types that are inferred from trust concepts. Trust in the external environment (C_2) is 0.58, and trust in implemented security mechanisms (C_{11}) is 0.58.

State changes in trust, trust types and trust concepts

C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C0:	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C1:	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.54	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
C2:	0.59	0.59	0.6	0.61	0.55	0.55	0.55	0.55	0.55	0.55	0.6	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.59
C3:	0.58	0.57	0.58	0.6	0.52	0.52	0.52	0.52	0.52	0.52	0.58	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.57
C4:	0.58	0.57	0.58	0.6	0.53	0.53	0.53	0.53	0.53	0.53	0.58	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.57
C5:	0.58	0.57	0.58	0.6	0.53	0.53	0.53	0.53	0.53	0.53	0.58	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.57
C6:	0.58	0.57	0.58	0.6	0.53	0.53	0.53	0.53	0.53	0.53	0.58	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.57

Figure 13.6: Inference of trust level based on identity (C_{12})

It has been argued in this thesis that trust defined over the identity of another party does not reflect its real trustworthiness. It is also determined by many other factors such as the properties of the party and its environment. The trust level of eCompany (based on identity) is *low*, and it is not sufficient to allow it to transact business on behalf of its employees.

When eCompany first starts to interact with eLoans, it is not aware of the fact that it can provide an integrated and advanced service to its employees where the orders placed by employees can be managed. It knows that it can act as a gateway for its employees to operations of eBooks as described in paragraph 3.1.1 in Chapter 3. Employees are required to register themselves individually with eBooks, and are granted access to operations. For these transactions, employees are responsible for their own actions.

eCompany needs to increase its trust level to be able to be granted access to more sensitive operations, where it can act on behalf of its employees. It has registered with eBooks and has presented a digital certificate as identification. The node reflecting the identity mechanism thus reflects a high level (C_{12} is set to 1). After the company has registered, the rule of law of the country where eCompany is located is automatically retrieved by an application at eBooks, from a web services provider that offers such a public service. The rule of law is found to be at a high level and C_8 is set to 1.

As employees place orders by means of the eCompany portal application that is acting as web services requestor, eCompany is found to conform to security mechanisms for integrity (C_{13} is set to 1) and confidentiality (C_{14} is set to 1) for all interactions that required security mechanisms as defined in the interface policy. Interactions and transactions are monitored and records are written to the information database.

To simplify the operation of the prototype, it is assumed that the information database has been populated with records as if they were dynamically written during interactions. By selecting the

Load data for trust in external environment – 13.3.3 link on the *Initialise Database Information related to Trust* page, records are loaded into the information database. When selecting *Trust concepts* from the *eBooks Administrative Trust Interface* page, trust concepts can be viewed as in Figure 13.7. Trust concepts are set to 1 or 0 after records have been fuzzified by the trust manager.

Trust concepts of the FCM

Trust concepts can be reset from the "Initialise Database Information" page

	ReqId	C_5	C_6	C_7	C_8	C_9	C_10	C_12	C_13	C_14	C_15	C_16	C_17	C_18
<input type="button" value="Edit"/>	eCompany	0	0	0	1	0	0	1	1	1	0	0	0	0
<input type="button" value="Edit"/>	eLoans	0	0	0	1	0	0	1	1	1	0	0	0	0

Calculate trust for:

Tip: Enter eLoans or eCompany

Figure 13.7: Trust concepts set by fuzzifying information from database

The trust level of eCompany can be recomputed to include trust concepts C₈, C₁₂, C₁₃, and C₁₄ as shown in Figure 13.8.

State changes in trust, trust types and trust concepts

C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C0:	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0
C1:	0.5	0.5	0.51	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.58	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
C2:	0.73	0.72	0.72	0.74	0.69	0.69	0.69	0.69	0.69	0.69	0.73	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.73
C3:	0.66	0.65	0.65	0.68	0.6	0.6	0.6	0.6	0.6	0.6	0.66	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.65
C4:	0.69	0.68	0.68	0.71	0.63	0.63	0.63	0.63	0.63	0.63	0.69	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.68
C5:	0.67	0.66	0.67	0.69	0.62	0.62	0.62	0.62	0.62	0.62	0.68	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.67
C6:	0.68	0.67	0.67	0.7	0.62	0.62	0.62	0.62	0.62	0.62	0.68	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.67
C7:	0.67	0.67	0.67	0.7	0.62	0.62	0.62	0.62	0.62	0.62	0.68	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.67
C8:	0.67	0.67	0.67	0.7	0.62	0.62	0.62	0.62	0.62	0.62	0.68	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.67
C9:	0.67	0.67	0.67	0.7	0.62	0.62	0.62	0.62	0.62	0.62	0.68	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.67

Figure 13.8: eBooks' trust in external environment with eCompany

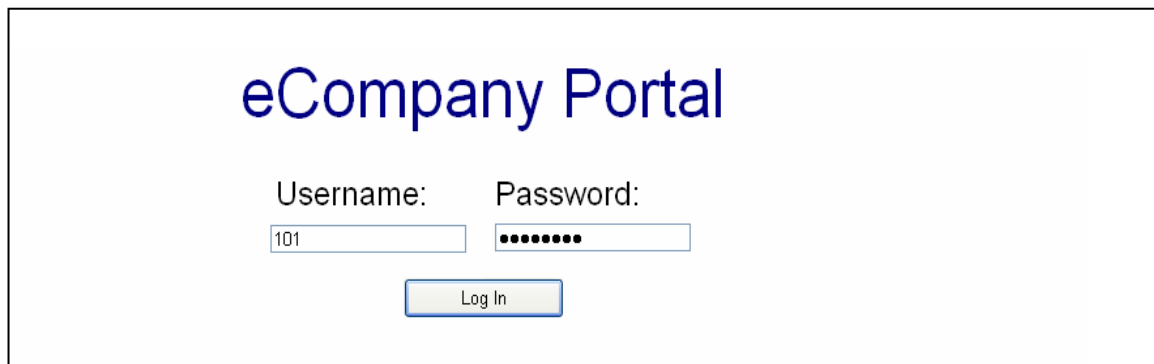
State vector A is computed until equilibrium is reached. The trust level is now 0.67, as shown in Figure 13.8. The trust level of eCompany has incremented from *low* to a *moderate* level, as defined by trust ranges in Chapter 12 on page 184. eCompany is now in a position to perform transactions that moderately trusted web services requestors are granted access to.

The operations that eCompany is granted access to, are addressed next. The WSDL document that is exposed to web services requestors with a *moderate* trust level, gives new information on web services operations that can be accessed at this level.

They are:

- The ability to not only perform a general search operation, but also to search for academic books if an employee possesses a student credential.
- The ability to place orders on behalf of employees.
- The ability to view orders placed on behalf of students.
- The ability to pay for orders on behalf of employees.

eCompany implements proxy classes to access all operations other than search-academic, as the latter is not deemed necessary for its environment. Sue Smith – employee number 101 – logs in to the eCompany portal with the password “password”, shown in Figure 13.9.



The screenshot shows a login form for the eCompany Portal. The title "eCompany Portal" is centered at the top in a large blue font. Below the title, there are two input fields. The first is labeled "Username:" and contains the text "101". The second is labeled "Password:" and contains ten black dots, indicating a masked password. Below these two fields is a rectangular button with the text "Log In" centered on it.

Figure 13.9: eCompany portal login

She is presented with an interface, shown in Figure 13.10, which enables her to purchase books at the expense of her employer. She might not know that she is buying books at eBooks, as the web interface she is working with is designed in line with the rest of the eCompany portal. She does not see the payment operation, as it is only made available to administrators of eCompany who have the privilege to make payments on behalf of eCompany.

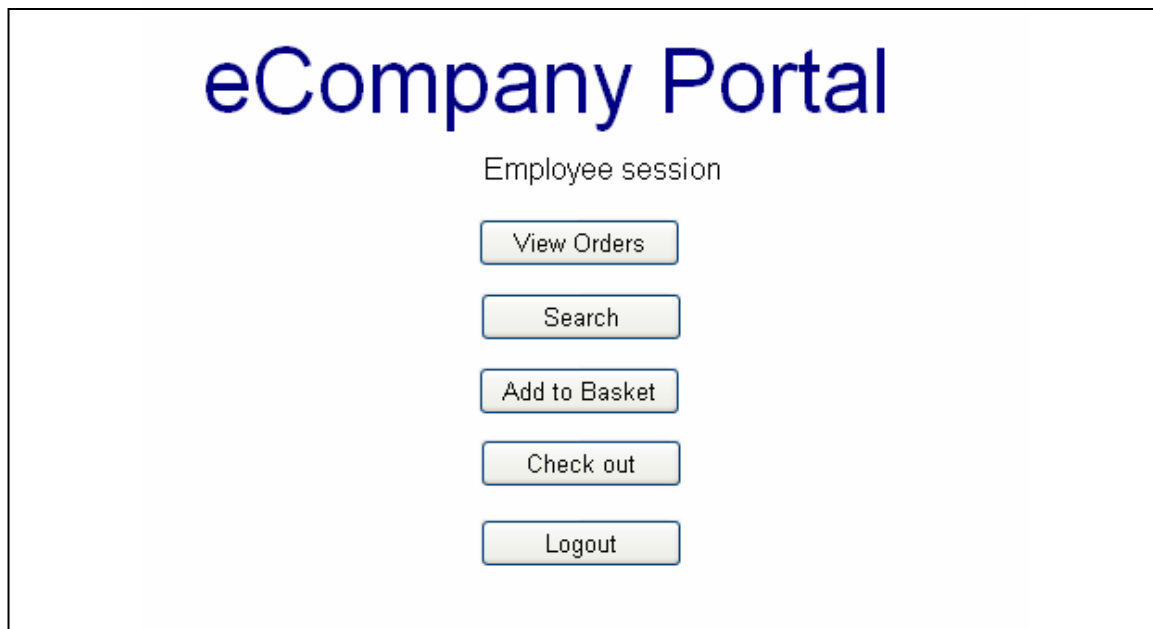


Figure 13.10: eCompany portal book order functions

The interface policy specifies that operations can be accessed if the identity of eCompany and the asserted ability of the subject (as defined by the web services requestor) are sent with the request in the header of the SOAP message. Sue selects the *search* operation, with “XML” as search parameter. The SOAP request message sent to eBooks is structured as shown in Figure 13.11. The SOAP header is named EBHeader, and contains the identity of the web services requestor (Appl_ID), and the type of subject making the request (Entity_Type).

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <EBHeader xmlns="http://tempuri.org/eBooks_WebService/_default">
      <Appl_ID>ecompany</Appl_ID>
      <Entity_Type>employee</Entity_Type>
    </EBHeader>
  </soap:Header>
  <soap:Body>
    <Search xmlns="http://csweb.rau.ac.za/staff/marijke/wsact/eBooks_WebService/..">
      <strSearch>XML</strSearch>
    </Search>
  </soap:Body>
</soap:Envelope>
```

Figure 13.11: SOAP message for search operation

The environment of eBooks receives the SOAP request message to invoke the search operation. Before the operation is invoked, the SOAP request message is intercepted by the authorisation interface as it arrives in the environment of eBooks. An access request is formulated for the authorisation manager by extracting the web services operation from the SOAP body and the web services requestor ID from the header. Figure 13.12 shows the access request.

```
do(search, ecompany, +exe).
```

Figure 13.12: Access request for search operation

The declaration made by eCompany on the identity of its employee is extracted from the SOAP header, and is added as an assertion to the access control policy of the authorisation manager. It is formatted as shown in Figure 13.13.

```
searchdecl(id('employee')).
```

Figure 13.13: Declaration

The access control policy that governs the granting of this request is shown in Figure 13.14. Only applicable rules that are used in reasoning about the request to *search* are shown.

```
cando(search, visitor, +exe).

reqtrustlevel(Requestor, X, Y) :-
    ask(reqtrustlevel, X, Requestor),
    trustlevelconvert(X, Y).

ask(Attr, TrustVal, _) :-
    !,
    TV = TrustVal.

ask(Attr, TrustVal, Gettrust) :-
    gettrust_value(Gettrust, TV),
    !,
    TV = TrustVal.

gettrust_value(P, TV) :-
    gettrust(P,TV),
    !.

trustlevelconvert(moderate, 2).
roletrustlevel(visitor, 1).
roletrustlevel(client, 2).

satisfied(search) :- searchdecl(id(Entity_Type)).

active(Requestor, Role):-
    roletrustlevel(Role, TlevelRole),
    reqtrustlevel(Requestor, TlevelReqs, TlevelReqi),
    ((TlevelReqi > TlevelRole);
    (TlevelReqi = TlevelRole)).

dercando(Object, Requestor, SignAction):-
    cando(Object, Role, SignAction),
    active(Requestor, Role),
    satisfied(Object).

do(Object, Requestor, SignAction):-
    dercando(Object, Requestor, SignAction).
```

Figure 13.14: A Prolog implementation of the access control policy for search operation

Request processing is performed as follows:

1. The **do** access request for the search operation, formulated by the authorisation interface, is evaluated to true, if a **dercando** predicate can be found for the search operation that evaluates to true.
2. The **dercando** rule specifies three conditions to be true, in order to evaluate to true. Conditions are italicised in the discussion that follows.
3. The **dercando** rule firstly requires that a **cando** rule exists for the search operation. The **cando** rule states that permission to execute the search operation is only granted to web services requestors who are active in the *visitor* role.
4. The **dercando** rule secondly requires that the web services requestor be active in the *visitor* role.
5. To activate the *visitor* for eCompany, the trust level of eCompany needs to be equal to or higher than the trust level of the *visitor* role. The trust level of the web services requestor is requested from the trust manager with the *GetTrust* predicate.
6. Trust levels of both the role and web services requestor are compared. eCompany is set in the *client* role, that is, in the role hierarchy above the *visitor* role.
7. The **dercando** rule thirdly requires that the declaration needed for the search operation be satisfied. In this case, it is the type of user which proves that he/she is indeed an employee of eCompany.
8. If all conditions evaluate to true, the authorisation manager returns *true* to the authorisation interface, else *false*.

During execution of step 4, the authorisation manager invokes the trust manager. A log file reveals that trust is computed for eCompany as shown in Figure 13.15.

```
Trust level of eCompany is moderate, as trust is = 0.67
```

Figure 13.15: Log file

Sue is therefore granted access to the search operation based on the combined influence of the following:

- The *moderate* trust level of eCompany that enabled the activation of a role with the required permission.
- Her asserted ability as employee.

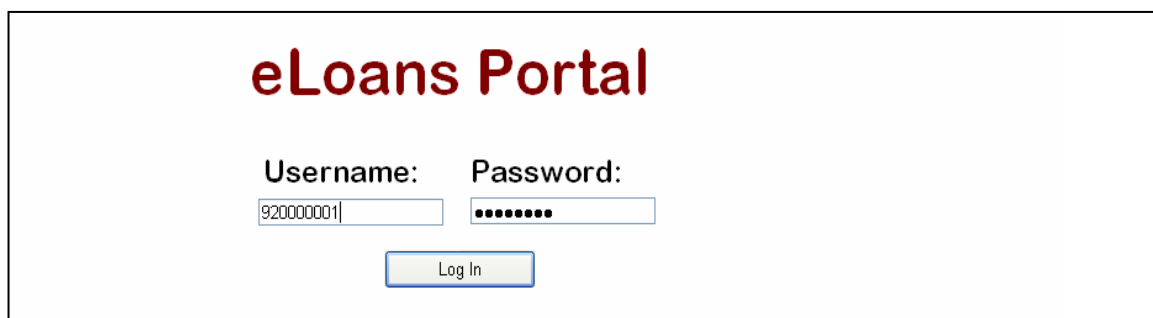
Next the operations of eLoans, a web services requestor whose trust level is *moderate*, are considered.

13.3.4 Trust in the other party – eLoans

The eCompany scenario has illustrated the combined effect of the abilities of the remote user and the trust in the web services requestor on access control decisions. The eLoans scenario has the aim of illustrating the following features of the WSACT model:

- Access control decisions based only on the ability of the remote user, where the ability is represented as attributes signed by a third party that are added to the SOAP request by eLoans.
- Access control decisions that are based only on the trust level of eLoans.
- The automated assessment of information and subsequent increase in the trust level to *good*, and then to *high*, so that operations reserved for highly trusted web services requestors may be accessed.

The WSDL interface document that is exposed to eLoans enables it to gain access to the same operations as eCompany. The same Sue Smith – registered as student 920000001 – logs in to the eLoans portal with password “*password*”, as shown in Figure 13.16.



The screenshot shows a login form for the eLoans Portal. The title "eLoans Portal" is centered at the top in a large, bold, red font. Below the title, there are two input fields. The first is labeled "Username:" and contains the text "920000001". The second is labeled "Password:" and contains a masked password represented by seven dots. Below these two fields is a button labeled "Log In".

Figure 13.16: eLoans login

She is similarly presented with an interface that enables her to purchase academic books at the expense of the loan that she has been assigned by eLoans. She might again not know that she is buying books at eBooks, as the web interface she is working with is designed in line with the rest of the eLoans portal. Her access to operations based on her abilities, as well as access based on the environment from where she makes requests is now considered.

13.3.4.1 Access based on a Sue’s abilities – search academic operation

Over and above the operations that have been implemented by eCompany, eLoans also implements the *search-academic* operation for its students. To be able to access this operation, the trust in eLoans is of no consequence, but Sue needs to present a digital credential from an academic institution to prove that she is indeed a registered student.

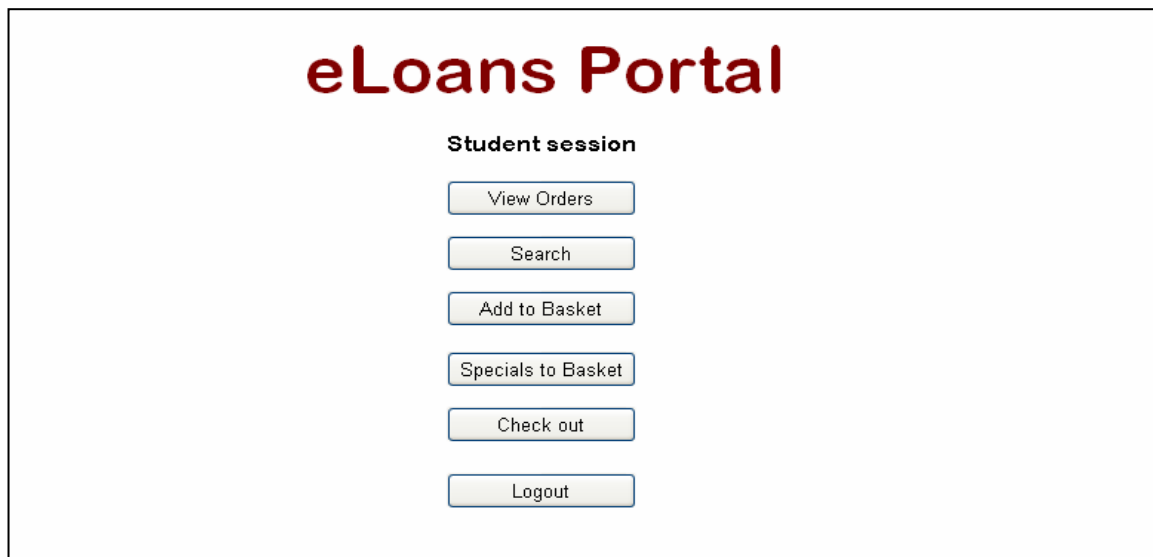


Figure 13.17: eLoans student search and order operations

Sue selects the *search* button in Figure 13.17, and is presented with the screen shown in Figure 13.18.

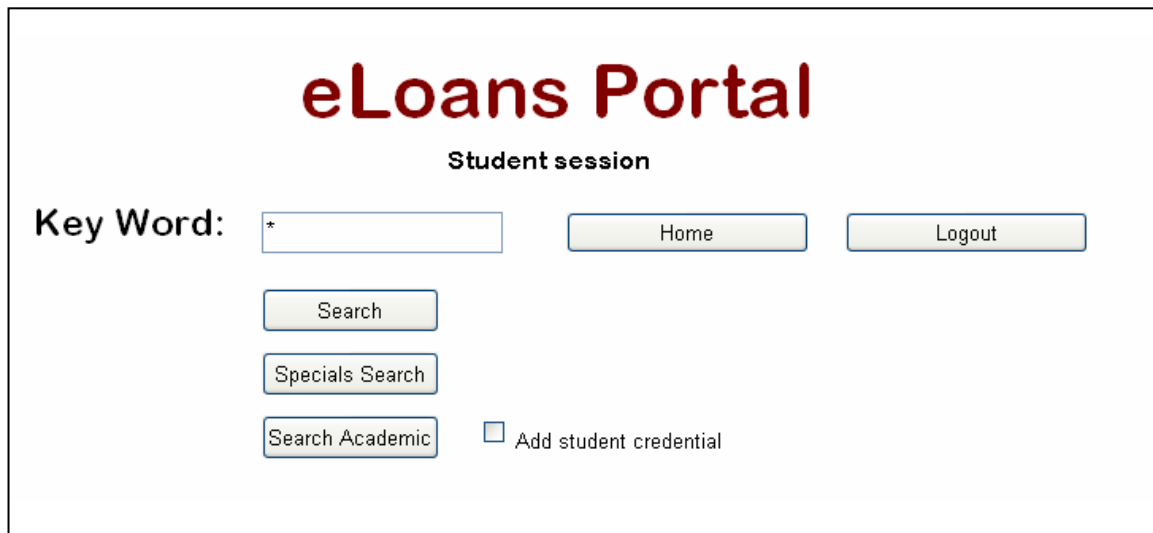


Figure 13.18: eLoans search operations

To be able to use the *search-academic* operation, Sue enters a search term and selects to add her pre-installed digital student credential, issued by eInstitution, to the SOAP request message. She is allowed to search academic books that are reserved only for students.

13.3.4.2 Access based on eLoans' trust level – specials search operation

The *specials search* operation is reserved by eBooks for highly trusted partners. This means that the trust level of web services requestors must be *high* to be able to access the operation. Figure 13.17 indicates the *specials search* operation on the web page of eLoans. In reality, eLoans

would not be aware of the existence of the operation until its trust level increases to *high*. At that point, it will be sent a new interface document that exposes operations reserved for trusted partners. If Sue clicks on the *specials search* button, she will not be granted access to the operation as the trust level of eLoans is not sufficient, and a SOAP exception will be thrown, as shown in Figure 13.19.

```

An exception has occurred at the service being processed
System.Web.Services.Protocols.SoapException: There was an exception running the extensions specified in the config file. --->
System.Web.Services.Protocols.SoapException: Access not authorised at eBooks_WebService.AuthorizationInterface.ProcessSOAPMessage(SoapMessage message) in C:\inetpub\wwwroot\BookService\AuthorizationInterface.vb line 108 at eBooks_WebService.AuthorizationInterface.ProcessMessage(SoapMessage message) in C:\inetpub\wwwroot\BookService\AuthorizationInterface.vb line 73 at System.Web.Services.Protocols.SoapMessage.RunExtensions(SoapExtension[] extensions) at System.Web.Services.Protocols.SoapServerProtocol.Init End of inner exception stack trace --- at System.Web.Services.Protocols.SoapServerProtocol.Initialize() at System.Web.Services.Protocols.ServerProtocolFactory.Create(Type type, HttpContext context, HttpRequest request, HttpResponse response, Boolean abortProcessing)

```

Figure 13.19: Error message

The next section assumes that the *specials search* operation is implemented by eLoans, but that it is not yet accessible by Sue because the trust level of eLoans is not sufficient. If a number of transactions are monitored to be successful, trust can automatically increase to a *high* level and the *specials search* operation becomes accessible.

Assume that the trust in the external environment with eLoans is similar to that of eCompany, as shown in Figure 13.20. By selecting the *Load data for trust in external environment – 13.3.3* link on the *Initialise Database Information related to Trust* page, records are loaded into the information database to set trust concepts shown in Figure 13.20.

Trust concepts of the FCM

Trust concepts can be reset from the "Initialise Database Information" page

	ReqId	C_5	C_6	C_7	C_8	C_9	C_10	C_12	C_13	C_14	C_15	C_16	C_17	C_18
<input type="button" value="Edit"/>	eCompany	0	0	0	1	0	0	1	1	1	0	0	0	0
<input type="button" value="Edit"/>	eLoans	0	0	0	1	0	0	1	1	1	0	0	0	0

Calculate trust for:

Tip: Enter eLoans or eCompany

Figure 13.20: Trust in the internal environment of eBooks and eLoans

State vector A is computed until equilibrium is reached. The trust level of eLoans is 0.67, which means that trust is at a *moderate* level. eLoans is now in a position to perform transactions that moderately trusted web services requestors are granted access to, as was indicated earlier.

Trust in eLoans needs to increment to *good* and finally to *high*, in order to be able to access the *specials search* operation. Trust concepts to be considered are *compliance to agreements* (C16), *competence* (C17) and *predictability* (C18).

To simplify the operation of the prototype, records have been created in the interface database for *compliance to agreements* and *competence* as if they have been written by the authorisation interface. Before the fuzzy cognitive map is run, the fuzzification process reads these records and sets trust concepts C16 and C17 to either 0 or 1, depending on whether the defined threshold for the trust concept is reached. Some records to compute the trust concept *predictability* are written during the interaction between eLoans and eBooks, to be used by the fuzzification process to illustrate automated trust assessment. Basic structures for the *REQUESTORS*, *COMPLIANCE*, *COMPETENCE* and *PREDICTABILITY* database tables, as well as their inter-relationships, are depicted in Figure 13.21. Each table is defined to contain the most basic information for the purposes of this prototype, and would be more complex in a real-world environment to comprehensively reflect many different types of information.

To repeatedly test the operation of this process, data needs to be initialised in the interface database. By selecting the *Load data for trust in other party – 13.3.4.2* link on the *Initialise Database Information related to Trust* page, records are loaded into the information database to set trust concepts. This loads data into the *COMPLIANCE*, *COMPETENCE* and *PREDICTABILITY* tables that are categorised according to a set of rules.

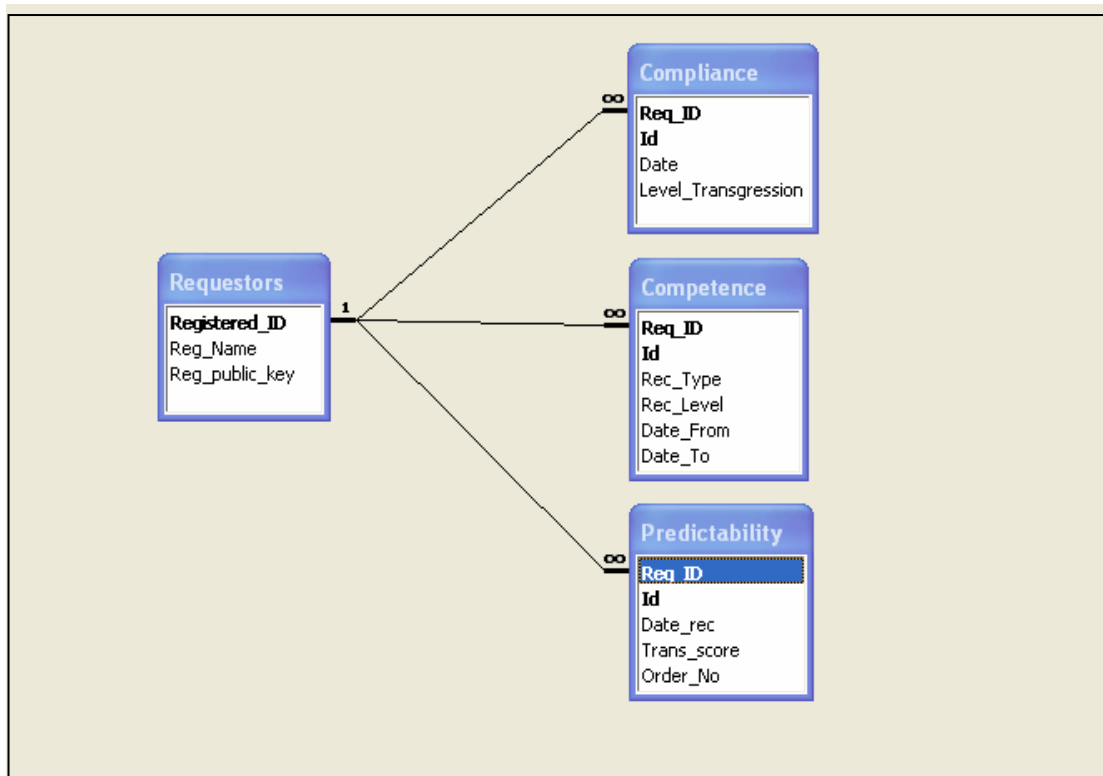


Figure 13.21: Tables from the interface database related to trust concepts

When the *COMPETENCE* and *COMPLIANCE* tables are populated with data, and related trust concepts C16 and C17 are set to 1, trust increases from 0,71 to 0,75. This increase is determined as follows:

The *COMPETENCE* table

In initial interactions, credentials such as recommendations, certificates of good conduct, Better Business Bureau statements, statements on financial status and other similar information are sent from eLoans to eBooks for analysis and records are added to the table. eBooks evaluates this information, and assigns a level to each statement according to a number of pre-defined rules. A threshold determines whether the information is sufficient to set the trust concept. In this prototype, a simple average is calculated if there are more than five types of records present that are valid. If the average level is 8 or more out of a total of 10, the trust concept is set to 1. Trust increases from 0,67 to 0,71.

The *COMPLIANCE* table

For each interaction between eLoans and eBooks, different types of factors are monitored. Examples are constraints specified in service level agreements, the implementation of security mechanisms, and information contained in SOAP messages such as invalid credit card numbers. Each interaction is monitored, and if a transgression occurs, the level of the transgression and the date are recorded. The average level of transgression is determined by taking the total number of interactions into account. If the level of transgression is 2 or less out of 10, the trust concept is set to 1. Trust increases from 0,71 to 0,75.

The *PREDICTABILITY* table

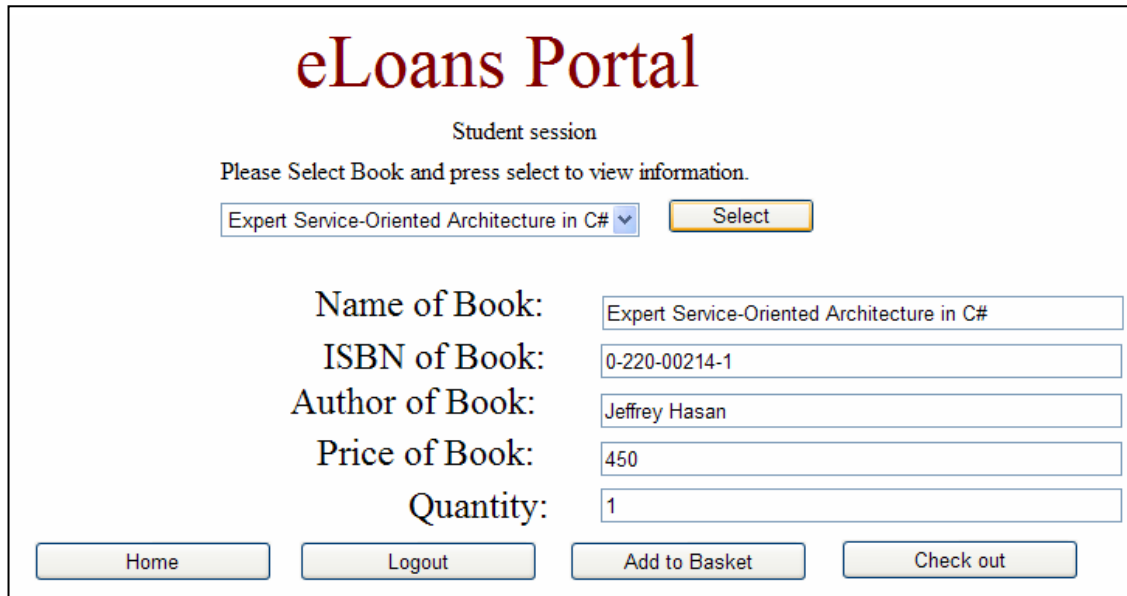
For the purposes of this prototype, the predictability of a web services requestor is determined by the payment of orders. For instance, if an order is paid for within 10 days or less, a score of 10 out of 10 is written to the database for that order. On the other hand, if the order payment is more than 90 days overdue, a score of 1 is written. Records are written as payment are made, and on a weekly basis for orders that are not paid. If eLoans has a transaction score of 8 or more out of 10, the predictability trust concept is set to 1.

Assume that records exist in the *PREDICTABILITY* table, but that the threshold is not met to set the trust concept to 1. Predictability is thus 0. The next section of the prototype illustrates the automated increment in trust as a number of orders are placed by Sue, and are immediately paid by the administrator of eLoans. Predictability is thus automatically set to 1.

Placement and payment of orders to set predictability

First, database tables are initialised in the interface database by selecting the *Load data for trust in other party* – 13.3.4.2 link on the *Initialise Database Information related to Trust* page. The trust level of eLoans is *good*. Sue is now ready to start placing orders.

Sue selects the *Add to Basket* option, shown in Figure 13.17. She is presented with the screen shown in Figure 13.22.



The screenshot shows the 'eLoans Portal' interface for a 'Student session'. The page prompts the user to 'Please Select Book and press select to view information.' A dropdown menu is set to 'Expert Service-Oriented Architecture in C#' with a 'Select' button next to it. Below this, there are five input fields for book details: 'Name of Book' (Expert Service-Oriented Architecture in C#), 'ISBN of Book' (0-220-00214-1), 'Author of Book' (Jeffrey Hasan), 'Price of Book' (450), and 'Quantity' (1). At the bottom, there are four buttons: 'Home', 'Logout', 'Add to Basket', and 'Check out'.

Figure 13.22: Adding items to the basket

Sue uses the drop-down list box to choose a book she is allowed to order. She selects the *Select* button to fill in other fields related to the book, and changes the quantity fields as is necessary. She adds the book to the basket by selecting *Add to Basket*. She may add a number of books to the basket in the same way. When she has ordered all books, she selects *Check Out*, shown in Figure 13.23, to finalise the order. Sue has placed an order for three books, and has the ability to delete an item if she so desires. By selecting *Order*, the order is finalised.

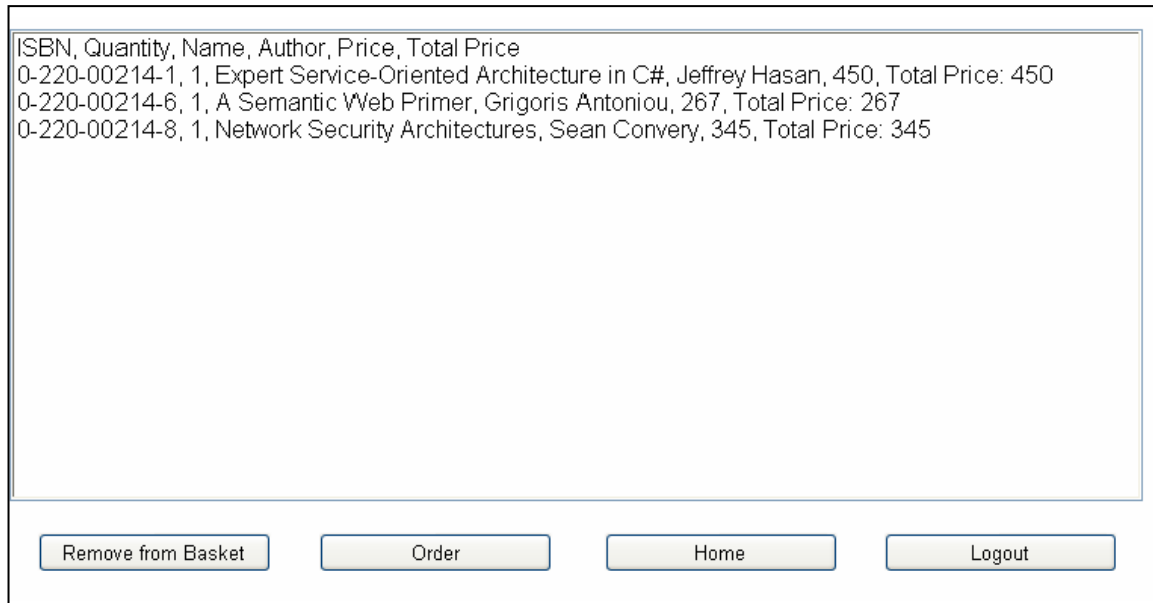


Figure 13.23: Finalise an order

The order is paid by the administrator of eBooks within 10 days, so that a transaction score of 10 can be written to the *PREDICTABILITY* table. In an operational environment, this fact will not be known to administrators of web services requestors so as to prevent them from manipulating trust levels. The administrator logs in at the login screen with username "root" and password "password". He/She is presented with the screen shown in Figure 13.24.



Figure 13.24: Administrator interface

From this screen the administrator views the order to verify who placed it. Figure 13.25 shows details concerning the last order to have been placed, order 79, by student 920000001 for eLoans. This information is directly retrieved from eBooks with the *view_order* operation.

eLoans Portal

Administrator session

Please Enter Application ID and/or Student ID to view corresponding orders.

Order Number:

Application ID:

Student ID:

Ono	Order_Date	ISBN	Qty	Name	Price	Author	Student_ID	Appl_ID
79	2006/01/27 12:00:00 AM	0-220- 00214-1	1	Expert Service-Oriented Architecture in C#	450	Jeffrey Hasan	920000001	eloans
79	2006/01/27 12:00:00 AM	0-220- 00214-6	1	A Semantic Web Primer	267	Grigoris Antoniou	920000001	eloans
79	2006/01/27 12:00:00 AM	0-220- 00214-8	1	Network Security Architectures	345	Sean Convery	920000001	eloans

Figure 13.25: Administrator – view order

After the order has been verified, it is paid. Figure 13.26 shows the payment screen where an order to be paid is selected. The *Continue* button is selected to proceed.

eLoans Portal

Administrator session

Please Select Order to be paid and press continue.

Order Number:

Figure 13.26: Administrator – select order to be paid

The administrator is presented with the screen in Figure 13.27 (which shows the total to be paid) and enters the credit card number. The *Process Payment* button is selected.

eLoans Portal

Administrator session

Please enter required information to make payment.

Order Number:

Amount to Pay:

Payment Method: ▼

Credit Card Number:

If paying with credit card

Figure 13.27: Administrator – enter credit card number

If the payment is processed successfully, the screen shown in Figure 13.28 is presented to the administrator.

Payment made successfully.

[Back to Admin homepage](#)

Figure 13.28: Administrator – successful payment

In the background, a record is written to the PREDICTABILITY table of the Interface database, as shown in Figure 13.29. A transaction score of 10 is recorded for order 79, as it was paid in less than 10 days.

Predictability				
Req_ID	Id	Date_rec	Trans_score	Order_No
eLoans	28	2006/01/27	10	79

Figure 13.29: Predictability table

Sue logs in again, and selects the *Specials Search* operation to see if the trust level of eLoans has increased to high, so that she can search for and order books at very special prices. If access is denied, the placing of orders and their payment are repeated until the trust concept for predictability is set. This can be monitored by viewing trust concepts from the *eBooks*

Administrative Trust Interface as shown in Figure 13.30. When this happens, the trust level is computed to be 0,79.

Trust concepts of the FCM Trust concepts can be reset from the "Initialise Database Information" page

	ReqId	C_5	C_6	C_7	C_8	C_9	C_10	C_12	C_13	C_14	C_15	C_16	C_17	C_18
<input type="button" value="Edit"/>	eCompany	0	0	0	1	0	0	1	1	1	0	0	0	0
<input type="button" value="Edit"/>	eLoans	0	0	0	1	0	0	1	1	1	0	1	1	1

Calculate trust for:
Tip: Enter eLoans or eCompany

Figure 13.30: Activation of predictability

The prototype now demonstrates an important aspect of the trust computation. At 0,79 the trust level is still *good*, but has not reached a *high* level. eLoans can behave as best it can, but its behaviour will not be able to increment the trust level to *high* because there are other factors that need to be taken into account. They are as follows:

- Trust concepts related to trust in the internal environment have not been set to 1. For instance, the self-confidence of eBooks is not sufficient to ensure that it trusts more easily, as trust concepts C6 (successes in dealing with risks) and C7 (complexity) have not been set to 1.
- The external environment does not support *high* trust, as trust concepts C9 (assurance) and C10 (compliance to standards) have not been set to 1.

Assume that eLoans determines that it has a low level of vulnerabilities. The trust concept C₅ is manually set by administrator intervention to 1, as shown in Figure 13.31. Also assume that eLoans informs eBooks of the fact that it has acquired insurance against loss. eBooks and eLoans agree to a service level agreement that govern behaviour in the face of misconduct. The trust concept C₉ is set by administrator intervention to 1, as shown in Figure 13.31. Trust finally computes to 0,85, which is at a *high* level.

Trust concepts of the FCM Trust concepts can be reset from the "Initialise Database Information" page

	ReqId	C_5	C_6	C_7	C_8	C_9	C_10	C_12	C_13	C_14	C_15	C_16	C_17	C_18
<input type="button" value="Edit"/>	eCompany	0	0	0	1	0	0	1	1	1	0	0	0	0
<input type="button" value="Edit"/>	eLoans	1	0	0	1	1	0	1	1	1	0	1	1	1

Calculate trust for:
Tip: Enter eLoans or eCompany

Figure 13.31: Activation of C₅ and C₉ by administrator

Sue tries the *Specials to Basket* option from the first screen. She is now successful as eLoans' trust level is *high* and access is allowed. The dropdown list box in Figure 13.32 shows only the few books that are available at special prices. Sue proceeds with the order by placing the books into her shopping basket. She may also add other books to the basket before finalising the order.

Figure 13.32: Order for special books

Sue is finally allowed to access this special option – not because of who she is, but because she is a member of a highly trusted web services requestor, with whom eBooks fosters stronger business relationships.

13.4 CONCLUSION

This chapter has demonstrated how access control incorporating trust, as described in the WSACT model, can be implemented in a prototype. The prototype has also illustrated how the required features for web services access control can be addressed. The incorporation of a trust level in the access control policy is presented as a viable solution to the problem of web services access control, where decisions of an autonomous nature need to be made, based on information and evidence.

The next chapter contains the researcher's final remarks about the WSACT model, the WSACT prototype implementation, and future work.

14

Conclusion

This thesis presents the WSACT model for web services access control. The model has evolved as the result of a scientific study into various facets of the problem area. In Chapter 1, the rationale behind the research is described, which leads to a number of research questions to be answered. A detailed study revealed the desired features of a web services access control service, stated as access control requirements.

In this chapter the researcher evaluates the extent to which the objectives of the research have been met by revisiting both the research questions and access control requirements. Finally, the chapter is concluded with the main contribution of the research, and suggestions on further research forthcoming from this work.

14.1 REVISITING THE PROBLEM STATEMENT

The main focus of this thesis is to specify an access control service for web services. The access control decision is given new focus by not asking “*Who may access this resource?*”, but rather “*Who is trusted to access this resource?*” In this work, it is identified that over and above attributes of users, the trust in a requesting machine plays an important role when access control decisions are made.

This thesis therefore endeavours to answer the following research questions:

What are the specific access control requirements of web services that need to be addressed?

In order to address this question, Chapter 2 gives a background to web services, with the aim of identifying requirements specific to the web services access control service. An evaluation of the web services environment highlights six environmental access control requirements namely autonomy, loosely coupled, quality of service, policy-based compatibility, policy negotiation and standards-based. These requirements are not met by current solutions.

Next, Chapter 3 describes a case study and concludes with a high-level access control policy for eBooks, a web services provider, to highlight questions faced by administrators of web services. These questions are revisited in Chapter 4, where the so-called internal access control requirements of web services are identified: flexibility, efficient administration, attribute-based access control, trust levels, exceptions, and conflict resolution. Of these, attribute-based access control and trust levels are identified as significant for this research. The proposition put forward in this research is that access control decisions made by the web services access control service are not based only on credentials presented by subjects, but also on the trust relationship with the requestor presenting the credentials. The extent to which a trust relationship plays a role in access control decisions is identified as an important access control requirement to be explored in the current research.

Altogether twelve access control requirements for web services are identified. For the purposes of this research, the list is reduced to include autonomy, loosely coupled, quality of service, policy-based compatibility, policy negotiation, standards-based access control, attribute-based access control, and trust levels.

How do current access control models meet the requirements of web services?

In order to address this question, Chapter 4 proceeds with a background discussion on access control. It describes access control models, mechanisms and information with the aim of identifying mechanisms that could be used to protect web service resources. The discussion is summarised to clearly show the discrepancy between current and desired access control mechanisms. From an analysis it is derived that access control for web services must be addressed at two policy levels: the interface policy and the access control policy.

How can access be granted to a diverse and ever-changing user population, as web services collaborate in virtual environments?

This question is answered by firstly identifying the lack of central control in web services environments in Chapter 5. The responsibility for access control and other decisions is thus shifted to participating machines, which need to be autonomous in nature. As the act of giving access to sensitive resources can be considered a refinement of a trust relationship, Chapters 5 and 6 take up the issue of trust to promote the adoption of an access control model that incorporates trust. The chapters thus addresses the requirement for trust levels, one of the access control requirements identified in Chapter 4. The discussion on trust culminates in a trust formation framework for web services in Chapter 6. From this framework it is clear that decisions about whom to trust are based on the properties of a web services requestor and on the security and trust requirements of a web services provider that is defined in a policy.

What policies are used by web services providers to communicate access control requirements and enforce access control decisions?

This question, which is dealt with in Chapter 4, identifies two policies required by the web services access control service. Chapter 7 in turn investigates policy specification languages to determine a suited policy specification language. It is identified that policy languages are required for two purposes: access control publication and access control reasoning. The interface policy is defined as a policy that needs to be published, for which the XML-based approach is found to be more suited. In Chapter 10, a conceptual design of the interface policy is presented in Z. The access control policy is used to reason about access control rules. For access control reasoning, ASL (Authorisation Specification Language) is identified as an access control specification language that can be extended with predicates to include reasoning about trust levels. Extensions are presented in Chapter 11.

How can trust be incorporated in an access control model for web services?

The WSACT model presented in Chapter 9 addresses the computation and use of a trust level throughout its design. Chapter 10 describes the publication of access control and trust requirements, and deals with the consequent categorisation and analysis of information to be stored in the information database. Chapter 11 describes extensions to ASL in order to incorporate roles, attributes and trust levels in access control reasoning. In access control reasoning, the trust level for a web services requestor is used to determine the access allowed. Chapter 12 describes the computation of the trust level by means of a fuzzy cognitive map that is populated by fuzzy trust concepts.

How can the model be deployed in the web services architecture?

This question is addressed in Chapter 8 and all the essential components and considerations for the web services access control service architecture are identified. Chapter 9 describes the main components of the web services access control service, namely the authorisation interface, the authorisation manager, and the trust manager. The prototype described in Chapter 13 identifies how each of these components is deployed in the Microsoft .NET web services architecture. The chapter also illustrates how the access control service can be integrated with current web services security specifications.

14.2 DOES THE MODEL MEET DESIRED ACCESS CONTROL FEATURES?

Chapters 2 and 4 identified eight access control requirements to be addressed. These requirements are discussed next in order to evaluate the extent to which they have been attained.

14.2.1 Environmental access control requirements

The focus of environmental access control requirements is to ensure that the access control service minimally impedes on the exchange of SOAP messages. The access control service can seamlessly be integrated with the web services architecture if environmental access control requirements are considered in its design.

Autonomy

The WSACT access control service is designed in such a way that access control decisions can be made autonomously by including incomplete, imprecise and fuzzy information in decision making. These decisions are implemented by means of the fuzzy cognitive map of the trust manager. The access control service keeps its access control policies private, and makes decisions independent from those of others.

Loosely coupled

The access control service is loosely coupled as dependencies between web services requestors and providers are limited to information that is expressed in the interface policy. Requirements are expressed for security mechanisms such as integrity and confidentiality, which are used in support of access control. Attribute requirements of subjects that are used to access web services objects are selectively published according to trust levels of web services requestors. Messages sent to and from the access control service are formatted according to standard web services specifications and published domain-dependent vocabularies. The WSACT model implements access control in an unobtrusive manner, as all requests are seamlessly intercepted.

Policy-based compatibility

Policy-based compatibility is maintained by ensuring that the publicised interface policy is described in a policy specification language defined with XML. This makes the content, relationship and meaning of the interface policy clear. The structure of information is made explicit by means of both standards-based and domain-dependent schemas.

Policy negotiation

The WSACT model does not address policy negotiation comprehensively. The model proposes that the authorisation interface should control the selective publication of both functional and non-

functional policies, according to the trust levels of web services requestors and the sensitivity of information contained in the policy.

Quality of service

The publication of access control and trust requirements by the WSACT model ensures that quality of service in support of information security is maintained. This allowed web services requestors who are aware of quality of service to select the best service.

Standards-based interaction

The WSACT access control service has been designed so that it can be used in conjunction with standards specification languages for message formats, and published access control and trust requirements. Specifications such as SOAP and SAML for message formats, WS-Security for security requirements, WS-Policy or WSPL for interface policy specification, and WS-MetaDataExchange for policy exchange can ensure that web services entities are able to communicate with one another.

14.2.2 Internal access control requirements

The internal access control requirements focused on ensuring that all resources exposed by web services operations could be accessed only by authorised parties. This was done by addressing the set of attributes provided on behalf of the subject and the trust level of the web services requestor as follows:

Trust levels

The authorisation manager of the WSACT model expresses and enforces access control policies by including the calculated trust level of a web services requestor. The access control policy interrogates the trust manager for the trust level of web services requestors when it is required. The trust level of the web services requestors grants partial access to the request for a web services object, by activation of a role. This requirement must be satisfied before the attributes of users are verified.

Attribute-based access control

After the activation of a role, the attributes of a user required to be granted access to a web services object are considered. The authorisation manager of the WSACT model grants access to subjects based not on their identity, but on their abilities that are expressed as sets of attributes. In some instances, only the attributes of a user were required. The approach does not support the iterative disclosure of credentials in a session, but rather requires that more sensitive attributes are presented as more sensitive operations are accessed. The trust in the user is supported by the trust in the web services requestor.

14.3 MAIN CONTRIBUTION

The main contribution of this thesis can be summarised as follows:

- The authorisation manager of the WSACT model is a first step towards an access control model that takes into account the different trust relationships that exist between web services requestors and providers.
- The publication of sensitive policies, based on the trust level of a web services requestor, lessens the burden on trust formation over the iterative disclosure of attributes of users. There is a high administrative burden to incrementally verify the credentials of each and every user for each request that is made. Such approaches do not mirror the real world, where one is very often not only trusted as an individual, but also trusted as a member of a community or organisation. By allowing web services entities to establish a level of trust between one another, users from each domain are allowed to make use of this trust to gain access to resources in another domain.
- The current research presents a first step towards an automated trust formation process for web services. The research also illustrates that much of the required information is available in machine-readable format, and demonstrates how it can be automatically gathered and assessed.
- The research makes an important contribution by composing a trust level from explicitly defined trust types. The structure of the fuzzy cognitive map defines how trust can be inferred from trust in the internal environment, trust in the external environment, and trust in the other party. An analysis of trust types leads to the rationale behind the inference of a trust level.

14.4 FUTURE RESEARCH

The proposed model achieved the set objectives to the extent described in the section above, but it suffers some limitations. These limitations provide opportunities to extend and support the work described in this thesis by a number of future research projects:

- As the interface policy becomes more sophisticated and dynamic, complex policy requirements and capabilities need to be processed by both web services requestors and providers. The implementation of sophisticated policy-processing points at each endpoint, to automatically negotiate about policies, would be an important element to complement the work presented in this thesis.
- The publication and processing of policy in this work is inflexible and requires human intervention. The definition of the interface policy by means of XML-related technologies such as XML Schema, RDF and RDF Schema, makes it possible that both humans and machines take advantage of the potential of the available information and greatly enhance automated policy integration.

- The question of distrust has not been addressed by this work. If a web services requestor transgresses past an unacceptable level, trust should not only decrease to low trust, but should change to become distrust. Distrust is considered the functional equivalent of trust, and can be used in the same manner to predict the behaviour of others.
- The thesis presented access control decision making at the level of a web services provider. In complex web services environments, the composition of web services requires that access control decisions of participants be combined at a higher level of access control abstraction. The work can be extended to include the composition of the decisions of web services providers at higher levels of compositions.
- In order to enable a fully autonomous access control service, an access control information translation service defined between web services requestors and providers would ensure that domain-specific access control information can be presented in a different domain, and that this can be understood and accepted.

Papers published in journals

Towards Web Services access control, Computers and Security, Vol. 23 no. 7, Elsevier publishers, UK, October 2004

An Access Control Framework for Web Services, Information Management and Computer Security, Emerald publishers, Vol 13, no 1, March 2005

Autonomous trust for Web Services, Internet Research, Vol 15, no 5, Emerald publishers, November 2005

Papers delivered at conferences

Access control for Web Services, ISSA 2003 (Information Security for South Africa), 9-11 July 2003, Sandton

Virtual enterprise access control requirements, SAICSIT 2003 (South African Institute of Computer Scientists and Information Technologists), IT research in developing countries, 17 – 19 September 2003, Indaba Hotel, Fourways.

Access control for networked Web Services, INC 2004 (The 4th International Network Conference), 6 – 9 July 2004, Plymouth, UK

A logic-based access control approach for Web Services, ISSA 2004 (Information Security for South Africa), 30 June – 2 July 2004, Gallagher Estate, Midrand

Access control for service-oriented computing, SATNAC 2004 (South African Telecommunications Networks and Applications Conference), 6-8 September 2004, Spier Wine Estate, Stellenbosch.

Metadata for trust in Service-oriented Architectures, ISSA 2005 (Information Security for South Africa), 29 June – 1 July 2005, Sandton, Johannesburg

Autonomous trust for Web Services, INC 2005 (The 5th International Network Conference), 5 – 7 July 2005, Samos, Greece

A Framework for Web Services Trust, SEC 2006, 21st IFIP International Information Security Conference "Security and Privacy in Dynamic Environments", 22 - 24 May 2006, Karlstad University, Karlstad, Sweden

Abadi M, Burrows M, Doorn L, Wobber E, (1999), Secure Network Objects. In J. Vitek and C. Jensen, editors, *Secure Internet Programming, Security Issues for Mobile and Distributed Objects*, pages 395–412. Springer

Abadi M. (2003) Logic in Access Control, in *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science*, June 2003, pp. 228-233.

Abdul-Rahman A. & Hailes S. (2000) Supporting Trust in Virtual Communities. In *Proceedings of the Hawaii International Conference on System Sciences*, Maui, Hawaii, 4-7 January 2000.

Abdul-Rahman A. (2004) A framework for decentralised trust reasoning, PHD thesis. Department of Computer Science, University of London.

Abiteboul S., Agrawal R., Klein J., Dayal U., Tsur S. & Weikum G. (2001) Are web services the next revolution in e-commerce? VLDB Conference, pp. 614-617, Rome, Italy.

Aguilar J. (2005) A Survey about Fuzzy Cognitive Maps Papers, (Invited Paper), *International Journal of Computational Cognition*, Vol. 3(2).

Albrecht A.J. (1979) Measuring Applications Development Productivity, *Proceedings of IBM Application. Dev. Joint Share/Guide Symposium*, Monterey, CA.

AMZI (2005) AMZI product page, <http://www.amzi.com/>, Accessed: 10 Sept 2005

Anderson A. (2004) An Introduction to the Web Services Policy Language (WSPL), In *5th IEEE International Workshop on Policies for Distributed Systems and Networks*. Yorktown Heights, New York, 7-9 June.

Anderson A. Anderson S, Adams C, Beznosov K, Brose G, Crocker S, et al. (2003) XACML 1.0 Specification, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, Accessed: 10 Feb 2004

Anderson S., Bohren J., Boubez T., Chanliou M., Della-Libera G., Dixon B. & Garg P. (2005) Web Services Secure Conversation Language (WS-SecureConversation), February, <ftp://www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf>, Accessed: 10 Oct 2005

Andrews T., Curbera F. et al. (2003) BPEL4WS, Business Process Execution Language for Web Services, Version 1.1, 31 March 2003, <http://xml.coverpages.org/WS-BPELv11-20030331.pdf>, Accessed: 10 Feb 2004

ANSI (American National Standards Institute) (1999) ANSI X9.45: Enhanced Management Controls Using Digital Signatures and Attribute Certificates, Washington, DC.

Apache Web Services (2005) ws.apache.org/ Accessed: 10 Oct 2005

Ashley P., Hada S., Karjoth G., Powers C. & Schunter M. (2003) *Enterprise Privacy Authorisation Language*, <http://www.zurich.ibm.com/security/enterprise-privacy/epal/specification/index.html> Accessed: 10 Nov 2005

Atkinson B. et al. (2002) Web Services Security (WS-Security), Version 1.0, 5 April 2002, <http://www.verisign.com/wss/wss.pdf> Accessed: 10 March 2003

Axelrod, R. (1972) *Framework for a General Theory of Cognition*. Berkeley: Institute of International Studies.

Bacon J. & Moody K. (2002) Towards open, secure, widely distributed services, *Communications of the ACM*, Vol. 45(6).

Bibliography

- Bacon J. & Moody K. (2002) Toward open, secure, widely distributed services. *Communications of the ACM*, 45(6) pp 59-64.
- Bacon J., Belokosztolszki A., Dimmock N., Eyers D., Moody K., Using Trust and Risk in Role-Based Access Control Policies, Proceedings of Symposium on Access Control Models and Technologies SACMAT04, (2004)
- Bajaj S., Box D., Chappell D., Curbera F., Daniels G. & Hallam-Baker P. (2004a) Web Services Policy Framework (WS-Policy), Sept 2004
<http://www.ibm.com/developerworks/library> Accessed: 8 Nov 2004
- Bajaj S., Box D., Chappell D., Curbera F., Daniels G. & Hallam-Baker P. (2004b) Web Services Policy Attachment (WS-PolicyAttachment), Sept 2004
<http://www.ibm.com/developerworks/library> Accessed: 21 Jan 2005
- Barber B. (1983) *Logic and Limits of Trust*. New Jersey: Rutgers University Press.
- Barker S. (2000) Security policy specification in logic. In *Proceedings of the International Conference on Artificial Intelligence*, Las Vegas, NV, pp. 143-148.
- Bartel M., Boyer J., Fox B., LaMacchia B. & Simon E (2002) XML Signatures,
<http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/> E Accessed: 10 Feb 2004
- Barwise J. & Etchemendy J. (2000) *Language, proof and logic*. Seven Bridges Press, USA.
- BEA (2004) Web Services Standards, <http://dev2dev.bea.com/webservices/standards.html>
 Accessed: 10 Feb 2005
- Bell D. & LaPadula L. (1973) Secure computer systems: Mathematical foundations, Tech report MTR-2547, vols I-III, MITRE Corporation, Bedford, MA.
- Bellwood T, Atkinson B, Cahuzac M, Cle´ment L, Colgrave J, Corda U, et al. (2003) *UDDI Version 3.0.1*, 14 Oct 2003, <http://uddi.org/pubs/uddi-v3.0.1-20031014.htm> Accessed: 10 Feb 2004
- Bertino E., Ferrari E. & Squicciarini A. (2003) X-TNL: An XML Based Language for Trust Negotiations, *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, IEEE CS Press, pp. 81-84.
- Bertino E., Ferrari E., Squicciarini A., (2004) Trust-X: A peer-to-peer framework for trust establishment, *IEEE Transactions on Knowledge and Data Engineering*, Vol 16, No 7
- Bertino E., Ferrari E., Squicciarini A., (2004a) Trust negotiation: Concepts, systems and languages, *Computing in science and engineering*, July/August 2004
- Bertino E., Jajodia S., Samarati P. & Subramanian V.S. (1997) A unified framework for enforcing multiple access control policies. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, Tucson, AZ.
- Bhatti R., Joshi J., Bertino E. & Ghafoor A. (2004) XML-Based Specification for web services Document Security, *IEEE Computer*, 37(4), April 2004, pp. 41-49.
- Biba K. (1975) Integrity considerations for secure computer systems, Tech. Rep. MTR-3153, MITRE Corporation, Bedford, MA.
- Bina E., Jones V., McCool R. & Winslett M. (1994) Secure Access to Data Over the Internet. In *Conference on Parallel and Distributed Information Systems*.

Biskup J. & Wortmann S. (2004) Towards a Credential-Based Implementation of Compound Access Control Policies, SACMAT 2004, 9th ACM Symposium on Access Control Models and Technologies, Yorktown Heights, New York, USA, June 2-4.

Blakley B., Byrne R., Huber R., Stokes E. & Rinkevich D. (2001) Access control model for LDAPv3, <http://www.ietf.org/proceedings/01aug/l-D/draft-ietf-ldapext-acl-model-08.txt>
Accessed: 13 March 2005

Blaze M., Feigenbaum J. & Lacy J. (1996) Decentralized trust management. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 164-173, IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.

Blaze M., Feigenbaum J., Ioannidis J. & Keromytis A. (1999a) The KeyNote Trust-management System, Version 2," IETF, RFC 2704, September 1999.

Blaze M., Feigenbaum J., Ioannidis J. & Keromytis A.D. (1999b) The role of trust management in distributed systems security. In *Proceedings of Fourth International Workshop on Mobile Object Systems: Secure Internet Mobile Computations*, no. 1603 in Lecture Notes in Computer Science, pp. 185-210, Springer-Verlag, July 1999.

Boeyen S., Ellison G., Karhuluoma N., Linn J., MacGregor W., Madsen P. & Sengodan S. (2003) Liberty Trust Models Guidelines, Version: 1.0, Editors: RSA Laboratories.

Bonatti P. & Samarati P. (2002), A unified framework for regulating access and information release on the Web, *Journal of Computer Security*, vol. 10(3), pp. 241-272.

Bonatti P. & Samarati P. (2003) Logics for Authorizations and Security, In *Logics for Emerging Applications of Databases*, Chomicki, Van der Meyden & Saake (eds), LNCS, Springer Verlag.

Booth D., Champion M., Ferris C., Haas H., McCabe F., Newcomer E. & Orchard D. (2004) WEB Services Architecture, W3C Working Group Note, 11 February 2004, <http://www.w3.org/TR/ws-arch/> Accessed: 25 Feb 2005

Box D, Curbera F, Hondo M, Kale C, Langworthy D, Nadalin A, et al. (2003) Web Services Policy Framework (WS-Policy), <http://www.ibm.com/developerworks/library/ws-policy/index.html> Accessed: 18 April 2004

Box D., Christensen E., Curbera F., Ferguson D., Frey J. & Hadley M. (2004) Web Services Addressing (WS-Addressing) W3C Member Submission 10 August 2004, <http://www.w3.org/Submission/ws-addressing/> Accessed: 11 Sept 2005

Box D., Ehnebuske D., Kakivaya G., Layman A., Mendelsohn N., Nielsen H.F, Thatte S. & Winer D. (2000) Simple Object Access Protocol (SOAP) 1.1, <http://www.w3.org/TR/SOAP/>
Accessed: 22 Jan 2003

Box D., Hondo M., Kaler C., Maruyama H., Nadalin A. & Nagaratnam N. (2003b) Web Services Policy Assertions Language (WS-PolicyAssertions)
<http://www.ibm.com/developerworks/library/ws-polas> Accessed: 9 Jan 2004

Boyd C., Ismail R. & Jøsang A. (2005) A survey of trust and reputation systems for online service provision, *Decision Support Systems* (in press).

Bradshaw J.M., Jeffers R., Montanari R., Suri N., Tonti G. & Uszok A. (2003) Semantic web languages for policy representation and reasoning: A comparison of KAoS, Rei and Ponder. In *2nd International Semantic Web Conference*, Sanibel Island, Florida, USA, Oct. 2003.

Bibliography

Bray T., Paoli J., Sperberg-McQueen C.M. & Maler E. (2000) Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October.

Brewer D.F.C. & Nash M.J. (1989) The Chinese Wall security policy," in Proc. IEEE. Symposium on Security and Privacy, pp. 206-214.

Burman I., Navarro A. & White C. (2000) *Mastering XML*, Sybex, USA.

Butler R.W., Chandramouli R. & Kuhn, D.R. (2002) Cost effective use of formal methods in verification and validation, Foundations 02 Workshop on Verification & Validation, Columbia, MD, Oct 2002.

Cahill V., Jensen C.D., Chen Y., Gray E. & Seigneur J. (2004) SECURE Framework Architecture (Beta), Technical report, Computer Science Department, The University of Dublin, Trinity College <https://www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-07.pdf> Accessed: 10 Feb 2005

Camenisch J. & Herreweghen E. (2002) Design and Implementation of the *Idemix* Anonymous Credential System. In *ACM Conference on Computer and Communication Security*, Washington D.C.

Casati F. & Shan M.C. (2001) Models and languages for describing and discovering e-services. SIGMOD Conference, p. 626, Santa Barbara, Calif., USA.

Castano S., Fungini M., Martella G. & Samarati P. (1995) *Database security*, Addison-Wesley.

Castelfranchi C. & Falcone R. (1998) Principles of trust for MAS: Cognitive anatomy, social importance and quantification. In *Proceedings of the International Conference on Multi-Agent Systems*, Paris, France, pp. 72-79.

Castelfranchi C, Falcone R., Pezzulo G. (2002) A Fuzzy Approach to a Belief-Based Trust Computation., in *Trust, reputation and security theory and practice*, Bologna, Italy, July, Lecture notes in Computer Science, Vol 2631

Castelfranchi C. & Falcone R. (2003) Social Trust: A cognitive approach. In *Trust and Deception in Virtual Societies* by Castelfranchi C. & Yao-Hua Tan (eds). Kluwer Academic Publishers, pp 55-90.

Ceri S., Gottlob G. & Tanca L. (1989) What You Always Wanted to Know About Datalog (And Never Dared to Ask), *IEEE Transactions on Knowledge and Data Engineering*, pp. 146-166.

Chen R, Yeager W 2003 Poblano: A distributed trust model for peer-to-peer networks. Technical report, Sun Microsystems. <http://www.jxta.org/docs/trust.pdf> Accessed: 23 Oct 2005

Chervany N.L. & McKnight D.H. (1996) The meanings of trust. Technical Report 94-04, Carlson School of Management, University of Minnesota.

Child E., Jacobson J., Mills H., Seamons K., Smith B., Winslett M., Yu T. & Yu L. (2002) Requirements for Policy Languages for Trust Negotiation. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, Monterey, California.

Ching N., Jones V., Slepchin I. & Winslett M. (1997) Using Digital Credentials on the World-Wide Web. *Journal of Computer Security*, pp. 255–267.

Christensen E., Curbera F., Meredith G. & Weerawarana S. (2000) Web Services Description Language (WSDL) 1.1, <http://www.w3.org/TR/wSDL> Accessed: 10 March 2003

Bibliography

- Coetzee M & Eloff J. H. P. (2004) Towards Web Services access control, Computers and Security, Vol. 23 no. 7, Elsevier publishers, UK, October 2004
- Coetzee M & Eloff J. H. P. (2005) Autonomous trust for Web Services, Internet Research, Vol 15, no 5, Emerald publishers
- Coetzee M & Eloff J. H. P. (2005B) An Access Control Framework for Web Services, Information Management and Computer Security, Emerald publishers, Vol 13, no 1, March 2005
- Colan M. (2004) Service-Oriented Architecture expands the vision of web services, <http://www-106.ibm.com/developerworks/webservices/library/ws-soaintro2/>
Accessed: 10 Oct 2004
- Coyle F.P. (2002) *XML, Web services and the data revolution*, Addison-Wesley.
- Coyne E.J., Feinstein H.L., Sandhu R. & Youman C.E. (1996) *Role-Based Access Control Models*. IEEE Computer, vol. 29(2), pp. 38-47
- Crews C.W. Jr (2005) Cybersecurity Finger-pointing Regulation vs. Software Liability, Information Security and Insurance, www.cei.org
- Damiani E., De Capitani Di Vimercati S. & Samarati P. (2002) Towards Securing XML Web Services, *XML'02*, November 22, Washington DC, USA.
- Damiani E., De Capitani Di Vimercati S., Paraboschi S. & Samarati P. (2001) Fine-grained access control for SOAP e-services, Proceedings of the 10th International World Wide Web Conference (WWW10) , HongKong, May 1-5.
- Damianou N. & Dulay N. (2001) The Ponder Policy Specification Language. *Policy 2001: Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, Springer-Verlag.
- Damianou N. (2002) A policy framework for management of distributed systems. PhD Thesis, University of London, London, UK.
- Damianou N. (2002) A policy framework for management of distributed systems. PhD Thesis, University of London, London, UK.
- Dan A., Davis D., Kearney R., King R., Keller A., Kuebler D., Ludwig H., Polan, M. Spreitzer, and Youssef A., (2004) Web Services on demand: WSLA-driven Automated M. Management, IBM Systems Journal, Special Issue on Utility Computing, Volume 43, Number 1, pages 136-158, IBM Corporation, March
- Dasgupta P. (1988) Trust as a commodity. In *Trust*, Diego Gambetta (ed.). Basil Blackwell.
- De Capitani Di Vimercati S. & Samarati P. (2000) Access Control: Policies, Models, and Mechanisms. In *Foundations of Security Analysis and Design* (Tutorial Lectures). R. Focardi and R. Gorrierieds, Springer-Verlag, pp. 137-196, September 2000.
- De Silva C. W., (2000) *Intelligent Machines – Myths and Realities*. CRC Press, New York, USA.
- Dedrick J., Kraemer K.L. & Shih C. (2005) Rule of law and the international diffusion of e-commerce, *Communications of the ACM*, Volume 48(11), November 2005, pp 57-62.
- Della-Libera G. et al. (2003) Web Services Trust Language (WS-Trust), <http://www.ibm.com/developerworks/library/ws-trust/index.html> Accessed: 10 July 2004

Denning D.E. (1976) A lattice model of secure information flow, *Communications of the ACM*, vol. 19(5).

DeTreville J. (2002) Binder, a logic-based security language. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pp. 105-113.

Deutsch M. (1962) Cooperation and Trust: Some theoretical notes. In *Nebraska Symposium on Motivation*, M.R. Jones (ed.) Nebraska University Press.

Diaz G., Duflos S., Gav V. & Horlait E. (2002) A comparative study of policy specification languages for secure distributed applications, *DSOM 2002, IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Montreal, Canada.

Dillaway B., Epstein J., Ford W., Fox B., Hallam-Baker P., LaMacchia B. & Lapp J. (2001) XML Key Management Specification (XKMS), <http://www.w3.org/TR/xkms/>
Accessed: 7 June 2004

Dimitrakos T. (2003) A Service-Oriented Trust Management Framework. In *Trust, Reputation, and Security: Theories and Practice*, Vol. 2631, pp. 53-72. Editors: Rino Falcone, Suzanne Barber, Larry Korba and Munindar Singh, *Lecture Notes in Computer Science*, Springer-Verlag.

E. Bertino, E. Ferrari, and A. Squicciarini, (2003) X-TNL: An XML Based Language for Trust Negotiations, *Proc. 4th IEEE Int'l Workshop on Policies for Distributed Systems and Networks*, IEEE CS Press, pp. 81–84.

Ellison C. M, Frantz B, Lampson B, Rivest R. L, Thomas B. M, Ylonen T, (1999a), Simple public key certificate. <http://world.std.com/~cme/html/spki.html>.

Ellison C. M, Frantz B, Lampson B, Rivest R. L, Thomas B. M, Ylonen T, (1999b), SPKI certificate theory. Internet RFC 2693

Eloff J.H.P. & Granova A. (2003) Computer Crime Case Analysis, *Computer Fraud and Security*, Oct. 2003, Elsevier Advanced Technology.

Eloff J.H.P. & Smith E. (2000) Cognitive fuzzy modeling for enhanced risk assessment in a health care institution, *IEEE Intelligent systems and their applications*, Vol. 14(2), pp 2-8.

Eloff J.H.P. & Von Solms S.H. (1998) Information Security, Rand Afrikaans University

Esfandiari B. & Chandrasekharan S. (2001) On how agents make friends: Mechanisms for Trust Acquisition. In *Proceedings of the 4th workshop on Deception, Fraud and Trust in Agent Societies*, Montreal, Canada, pp. 27-34.

Ferraiolo D. & Kuhn R. (1992) Role-based access control, *15th National Computer Security Conference*, <http://csrc.nist.gov/rbac/> Accessed: 12 July 2005

Fielding R., Gettys J., Mogul J. & Frystyk H. (1997) Hypertext Transfer protocol - HTTP/1.1, Network Writing Group, Request for Comments, no. 2068.

Fontana J. (2003) IBM, Microsoft publish Web Services identity spec, *Network World Fusion*, 8 July, <http://www.nwfusion.com/news/2003/0708ibmmsspec.html> Accessed: 5 Aug 2004

Frier A., Karlton P. & Kocher P. (1996) The SSL 3.0 Protocol. Netscape Communications.

Gambetta D. (1988) Can we trust Trust?, Chapter 13, pp. 213-237. Basil Blackwell. Reprinted in electronic edition from Department of Sociology, University of Oxford.

Bibliography

Ganek A (2004) Keynote address, Vice President, Autonomic Computing, IBM Software Group, *IEEE 5th International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)* June 7-9, 2004, IBM Thomas J Watson Research Center, Yorktown Heights, New York.

Global Grid Forum (2003), <http://www.gridforum.org/> Accessed: 4 March 2004

Golbeck J. (2003) Trust and Reputation in Web-Based Social Networks, <http://trust.mindswap.org/trustOnt.shtml> Accessed: 8 Oct 2005

Gollmann D. (1999) *Computer Security*, John Wiley and Sons, England.

Gong L. (1989) A Secure Identity-Based Capability System. In *IEEE Symposium on Security and Privacy*, Oakland, CA.

Gottschalk K., Graham S., Kreger H. & Snell J. (2002) Introduction to web services architecture, *IBM Systems Journal*, Volume 41(2).

Grandison T.W.A. (2003) Trust Management for Internet Applications, PhD Thesis, Imperial College of Science, Technology and Medicine, University of London, Department of Computing.

Groumpos P.P. & Stylios C.D. (2004) Modeling complex systems using Fuzzy Cognitive Maps, *IEEE Transactions on systems, man and cybernetics – Part A: Systems and Humans*, Vol. 34(1).

Guerin R, Pendarakis D., Yavatkar R. (2000), RFC 2753 – a framework for policy-based admission control, www.faqs.org/rfcs/rfc2753.html Accessed: 6 June 2005

Hallam-Baker P., Hodges J., Maler E., McLaren C. & Irving R. (2003) SAML 1.0 Specification, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security Accessed: 29 April 2004

Halpern J.Y. & Weismann V. (2003) Using first-order logic to reason about policies, *16th IEEE computer security foundations workshop*, p. 187.

Harmelen F. & McGuinness D. (2004) OWL Web Ontology Language, W3C Recommendation 10 February 2004 <http://www.w3.org/TR/2004/REC-owl-features-20040210/> Accessed: 13 Aug 2005

Hayton, R.J., Bacon, J.M. & Moody, K. (1998) Access Control in an Open Distributed Environmen, In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, USA, pp. 3-14, May 1998.

Herzberg, A., Mass, Y., Michaeli, J., Naor, D. & Ravid, Y. (2000) Access Control Meets Public Key Infrastructure, or: Assigning Roles to Strangers. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California, USA, 14-17 May 2000.

Hess A, Jacobson J., Mills H., Wamsley R., Seamons K.E. & Smith B. (2002) Advanced Client/Server Authentication in TLS, *Proceedings: Network and Distributed System Security Symposium*, San Diego, California, 6-8 February 2002.

Hitchens, M. & Varadharajan, V. (2001) Tower: A Language for Role Based Access Control. In *Proceedings of the Policy Workshop*, HP Labs, Bristol, UK, Springer-Verlag, 29-31 January 2001.

Housley R., Ford W., Polk W. Solo D. (1999), Internet X.509 public key infrastructure certificate and CRL profile, RFC 2459, Internet Engineering Task Force, <http://www.ietf.org/rfc/rfc2459.txt>

Humenn P. & Kuo C.J. (2002) Dynamically Authorized Role-Based Access Control for Secure Distributed Computation, ACM Workshop on XML Security, Nov. 22, 2002, Fairfax VA, USA.

IBM (2004) SOA and Web services
<http://www-136.ibm.com/developerworks/webservices>, Accessed: 15 Nov 2004

IBM UDDI Business Registry (2004) <https://uddi.ibm.com/ubr/registry.html> Accessed: 6 June 2005

IBM Web Services (2005) www-106.ibm.com/developerworks/webservices/ Accessed: 16 Oct 2005

Imamura T., Dillaway B. & Simon E. (2002) XML Encryption, <http://www.w3.org/TR/xmlenc-core/> Accessed: 6 July 2005

ISO (1996) ISO 10181-3 Access Control Framework.

ISO 17799 (2005) Information Technology – Security Techniques – Code of practice for Information Security Management, <http://17799.standardsdirect.org/iso17799.htm>

ISO 7498-2. (1989) Information Processing Systems — Open System Interconnection — Basic Reference Model – Part 2: Security Architecture.

Jajodia S., Samarati P. & Subramanian V.S. (1997) A logical language for expressing authorisations. In *Proc. Of the 1997 IEEE Symposium on Security and Privacy*, Oakland, CA.

Jajodia S., Samarati P., Sapino M. & Subramanian V.S. (2001) Flexible Support for Multiple Access Control Policies, *ACM Transactions on Database Systems*, 26(2), June 2001, pp. 214-260.

Jasper R. & Uschold M. (1999) A Framework for Understanding and Classifying Ontology Applications. In *Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, Stockholm, Sweden, August 1999.

Jensen C.D. & Seigneur J. (2005) The role of identity in pervasive computational trust: Privacy, Security and Trust within the Context of Pervasive Computing, The Kluwer International Series in Engineering and Computer Science, Vol. 780.

Jim T (2001b), SD3, <http://www.research.att.com/~trevor/sd3.html> Accessed: 21 April 2005

Jim, T. (2001), SD3: a trust management system with certified evaluation. in IEEE Symposium on Security and Privacy. Oakland, California, USA: IEEE Computer Society. <http://www.research.att.com/~trevor/papers/JimOakland2001.pdf> Accessed: 21 April 2005

Jones V., Seamons W. & Winsborough K. (2000) Automated trust negotiation, Technical Report. TR-2000-05, Department of Computer Science, North Carolina State University.

Jøsang A. (1996) The right type of trust for distributed systems. In *New Security Paradigms Workshop*.

Jøsang A. (1999) Trust-based decision making for electronic transactions. In *Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99)*, L. Yngström and T. Svensson (eds) Stockholm University, Sweden.

Jøsang A. & Tran N. (2000) Trust Management for E-Commerce, *Virtual Banking 2000*, security.dstc.edu.au/papers/virtbank2k.pdf Accessed: 24 April 2005

Jøsang A. & Ismail R. (2002) The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia, June 2002

Bibliography

- Joshi J.B.D, Aref W.G., Ghafoor A. & Spafford E.H. (2001) Security models for web-based applications, *Communications of the ACM*, Vol. 44(2) p 38.
- Kaye D. (2003) *Loosely Coupled: The Missing Pieces of Web Services*, Rds Associates Inc.
- Kosko B. (1986) Fuzzy Cognitive Maps, *International Journal of Man-Machine Studies*, Vol. 24, pp. 5-75.
- Kosko B. (1997) *Fuzzy Engineering*, Prentice Hall, Upper Saddle River, New Jersey.
- Kosko B. (1992) *Neural networks and fuzzy systems*, Prentice-Hall, USA.
- Kudo M., Jajodia S., & Subrahmanian V.S. (2001) Provisional authorization, e-commerce security and privacy, Anup Ghosh (ed.) Kluwer Academic Publishers, Boston, pp 133-159.
- La Porta R., Lopez-de-Silanes F., Shleifer A. & Vishny R.W. (1997) Legal determinants of external finance, *Journal of Finance*, Vol. 52(3), pp 1131-1150.
- Lampson B. & Rivest R.L. (1996) SDSI - a simple distributed security infrastructure <http://theory.lcs.mit.edu/~cis/sdsi.html>.
- Lampson B. (1971) Protection, in *Proceedings of the 5th annual Princeton Conference on Information Science and Systems*, Princeton University, pp 437-443.
- Li N. & Mitchell J.C. (2003) Datalog with constraints: A foundation for trust-management languages. In *Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL 2003)*, vol. 2562 of *Lecture Notes in Computer Science*, pp. 58-73, Springer-Verlag.
- Liberty Alliance Project (2005), <http://www.projectliberty.org/> Accessed: 21 Sept 2005
- Liberty Alliance Project Specifications (2005) <http://www.projectliberty.org/resources/specifications.php> Accessed: 21 Sept 2005
- Luhmann N. (1979) *Trust and Power*. Wiley.
- Lyu M.R. & Rajaraman C. (1992) Some coupling measures for C++ programs. *Proceedings of the eighth international conference on Technology of Object Oriented Languages and Systems*, pp. 225-234. Santa Barbara, California, US.
- Manchala D.W. (1998) Trust Metrics, Models and Protocols for Electronic Commerce Transactions. In *Proceedings of the 18th International Conference on Distributed Computing Systems*.
- Marsh S. (1994) Formalising Trust as a Computational Concept, PhD Thesis, University of Stirling, UK.
- McDonald C. & Pirzada A.A. (2004) Establishing Trust In Pure Ad hoc Networks. In *27th Australasian Computer Science Conference*, The University of Otago, Dunedin, New Zealand.
- McNeill F.M. (2003) A Rich Collection of Squashing Functions, Technical Report, Fuzzy Systems Engineering, <http://www.fuzzysys.com/squash2.pdf> Accessed: 8 Jan 2005
- Microsoft (2004) Metadata Specifications Index Page <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsmetaspecindex.asp> Accessed: 21 June 2005
- Microsoft UDDI Business Registry Node (2004) <http://uddi.microsoft.com/> Accessed: 21 Sept 2005

Bibliography

- Microsoft Web Services (2005) msdn.microsoft.com/webservices/ Accessed: 21 Sept 2005
- Microsoft White Paper (2001) Global XML Web Services Architecture, http://www.getdotnet.com/team/XMLwebservices/gxa_overview.aspx Accessed: 21 Sept 2005
- MS ASP.NET (2005) ASP.NET resources, <http://msdn.microsoft.com/asp.net> Accessed: 21 Sept 2005
- MS IIS (2005) Internet Information Server product page, <http://www.microsoft.com/WindowsServer2003/iis/default.msp> Accessed: 21 Sept 2005
- MS VB.NET (2005) Visual Basic resource, <http://msdn.microsoft.com/vbasic/default.aspx> Accessed: 21 Sept 2005
- Newcomer E. (2002) *Understanding Web Services*, Addison-Wesley, USA.
- OASIS (2005) Organisation for the Advancement of Structured Information Standards, www.oasis-open.org/home/index.php Accessed: 21 Sept 2005
- Orchard D. (2004) Achieving Loose Coupling, <http://dev2dev.bea.com/pub/a/2004/02/orchard.html> Accessed: 8 April 2004
- Oxley J.E. & Yeung B. (2001) E-commerce readiness: Institutional environment and international competitiveness, *Journal of International Business Studies*, Vol. 32(4), pp 705-723.
- Parisi-Presicce F., Sandu R. & Zhang X. (2004) A Runtime-enforced Policy Approach to Control the Execution of Mobile Codes, George Mason University, Technical Report, 2004.
- Ratnasingam P.P. (2001) Interorganizational trust in Business to Business e-commerce, PhD thesis, Erasmus University, Rotterdam.
- Ribeiro C., Zuquete, A. & Ferreira. P. (2001) SPL: An access control language for security policies with complex constraints. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'01)*, San Diego, California, February 2001.
- Rivest R. & Lampson B. (1996) SDSI - A Simple Distributed Security Infrastructure, October 1996.
- Sabater J. (2002) Trust and reputation for agent societies, PHD thesis, University of Barcelona, Spain.
- Sandu R. (1996) Access control: The neglected frontier, *Proceedings of the 1st Australian conference on Information Security and Privacy*, Wollongong, Australia, June 23-36.
- Sandhu, R.S. (1998) *Role Activation Hierarchies*. In *Proceedings of the Third ACM/NIST Role-Based Access Control Workshop*, Fairfax, Virginia, USA, ACM Press, 22-23 October 1998.
- SAP (2004) Web Services Policy, <http://ifr.sap.com/ws-policy/index.html> Accessed: 21 Sept 2005
- SAP UDDI Business Registry (2004) <http://uddi.sap.com/> Accessed: 21 Sept 2005
- Sarbanes-Oxley (2002) Sarbanes-Oxley Act of 2002, <http://news.findlaw.com/hdocs/docs/gwbush/sarbanesoxley/072302.pdf> Accessed: 21 Sept 2005

Bibliography

Schmidt T., Wippel G., Glanzer K. & Fürst K. (2005) Security System for Distributed Business Applications, *International Journal of Web Services Research*, 2(1), pp 77-88, Jan-March 2005.

Seamons E., Winslett M. & Yu T. (2001) Interoperable Strategies in Automated Trust Negotiation. In *ACM Conference on Computer and Communications Security (CCS)*. Philadelphia, Pennsylvania, USA: ACM, <http://isrl.cs.byu.edu/pubs/ccs2001.pdf>

SECURE (2003) The secure project, <http://secure.dsg.cs.tcd.ie> Accessed: 21 Sept 2005

SECURE (2003), The SECURE project at Cambridge, www.cl.cam.ac.uk/research/SRG/opera/projects/secure Accessed: 21 Sept 2005

Shirey R. (2000), RFC 2828 - Internet Security Glossary, www.faqs.org/rfcs/rfc2753.html Accessed: 21 Sept 2005

Shohoud Y. (2002) Real world XML Web Services for VB and .NET developers, Addison-Wesley.

Sonic Software (2004)
http://www.sonicsoftware.com/solutions/learning_center/standards_watch/index.ssp
Accessed: 21 Sept 2005

Spivey J.M. (1992) The Z Notation: a reference manual,
<http://spivey.oriel.ox.ac.uk/mike/zrm/index.html> Accessed: 18 June 2005

Sun (2004) www.sun.org/software/jxta Accessed: 21 Sept 2005

The Apache Software Foundation (2005) Apache HTTP server,
<http://httpd.apache.org/docs-2.1/howto/auth.html> Accessed: 21 Sept 2005

The Foaf Project (2002) <http://foaf-project.org/> Accessed: 17 Oct 2004

The Internet Society (2004) WebDAV Access control protocol,
<http://webdav.org/specs/rfc3744.html> Accessed: 21 Sept 2005

Trustbuilder (2001) TrustBuilder: Access Control and Authentication for Open Computing Systems, <http://dais.cs.uiuc.edu/trustbuilder/> Accessed: 21 Sept 2005

Unicode (2005), Unicode home page, www.unicode.org Accessed: 21 Sept 2005

Venter H.S. (2003) A Model For Vulnerability Forecasting, Rand Afrikaans University, Johannesburg, PhD thesis.

Verisign (2004) **VeriSign: Enabling Trusted Web Services**
<http://www.verisign.com/spotlight/02/0219/index.html> Accessed: 21 Sept 2005

W3C (2005) World Wide Web Consortium, <http://www.w3.org> Accessed: 21 Sept 2005

Winslett M. (2002) An Introduction to Trust Negotiation. Nixon & Terzis (eds), In *Proceedings of the First International Conference, iTrust* Heraklion, Crete, Greece, May 28-30, Springer.

XML Schema (2004) Part 0: Primer Second Edition, W3C Recommendation 28 October 2004
<http://www.w3.org/TR/xmlschema-0/> Accessed: 3 Jan 2005

XrML (eXtensible Rights Markup Language) (2001) version 2.0. At <http://www.xrml.org/>. Accessed: 21 Sept 2005

Bibliography

231

Yao, W.T, (2003), Trust Management for widely distributed systems, PHD dissertation, Jesus College, University of Cambridge

Zimmermann P.R. (1995) *The Official PGP User's Guide*. Cambridge, MA, USA: MIT Press.