A Security Gateway for Web Service Protocols

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Dong Huang

aus Shanghai, China

Tag der mündlichen Prüfung:	25. Juni 2007
Erster Gutachter:	Prof. Dr. Jacques Calmet
Zweiter Gutachter:	Prof. Dr. Hartmut Schmeck

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Dissertation ist bisher keiner anderen Fakultät vorgelegt worden.

Ich erkläre, dass ich bisher kein Promotionsverfahren erfolglos beendet habe und dass keine Aberkennung eines bereits erworbenen Doktorgrades vorliegt.

Acknowledgements

Firstly, I would like to thank my supervisors, Prof. Dr. Jacques Calmet and Dr. Jorge Cuellar for their constant dedication, guidance, advice and inspiration. This thesis would not have been possible without their constructive critisms and motivation. I am also very grateful to Prof. Dr. Hartmut Schmeck for his advice, encouragement and guidance.

Many thanks to both Dr. Cuellar and Prof. Calmet for arranging for financial support during my PhD studies, I would like to express my sincere gratitude to their kindness and the opportunity that they gave me to make the research at Siemens AG, Corporate Technology. Special thanks go to Dr. Joerg Abendroth who has contributed useful comments and advice to the work presented here.

Many thanks to my friends and colleagues in the University of Karlsruhe (TH), CT IC3 of Siemens and Corporate Quality Consulting GmbH. They include: Yi Yang, Shane Bracher, Mike Elschner, Holger Haas, Dr. Monika Maidl and Helga Scherer.

Other friends who deserve a mention include: Jia Ma, Qi Zhu, Cheng Wang, Minhua Chen and Ming Yang. Thanks for all the encouragements communicated via snail mails, emails, and phone calls. Special thanks to Dr. Xiujuan Yu for her timly interventions and words of encouragements I was able to go through the most difficult periods. To all other friends whose names I have forgotten to mention, thank you.

Lastly, I owe a great debt to my family for supporting my decision to study in the Germany, their sacrifices and their endless support over the years, especially my mother Mei Huang.

The work presented in this thesis was partly funded by Siemens AG, Corporate Technology in Munich.

Zusammenfassung

Die Verbreitung von Web Services und service orientierten Architekturen führen zu einem dramatischen Wandel im Aufbau von internen Systemen und deren Kommunikation mit externen Systemen.

Um die Kosten von Software zu senken und gleichzeitig ihre Leistungsfähigkeit zu steigern, setzen immer mehr Firmen und Organisationen Webservices ein. Eines der größten Probleme, welches die Industrie von dem Einsatz dieser Standardarchitektur abhält, ist die Sicherheit.

Als Antwort auf diese Sicherheitsbedenken erarbeiten IBM und Microsoft einen Plan für sichere Webservices und einen Zeitplan für die Entwicklung von Entwurfsmustern für sichere Webservices, die einen Schutz der Nachrichtenübertragung im Webserviceumfeld gestatten.

Standardisierungsorganisationen wie OASIS und W3C haben SAML und XACML vorgeschlagen um ein XML-Schema für Zugriffs- und Berechtigungsstrukturen sowie ein XML basierendes System für den Austausch von Nutzerrechten und weiteren Attributen als Standard zu verabschieden.

Trotz des großen Aufwands, der getätigt wurde, kann im Moment noch nicht von ausreichendem Sicherheitniveau gesprochen werden. Die Herausforderungen zwischen Geschäftstätigkeit und IT, sich regelmäßig verändernden Organisationsstrukturen und flexiblen Geschäftsprozessen/Geschäftsmodellen anzupassen, stellen zusätzliche Anforderungen an ein skalierbares, flexibles und wartbares Sicherheitssystem. Zwei Fallbeispiele zur Begründung unserer Forschungsarbeit:

- Im Kapitel 404 des Sarbanes Oxley Act (SOX) ist beschrieben das alle Firmen mit mehr als 75 Millionen US$ Marktkapitalisierung die Wirksamkeit ihres internen Kontroll- und Regelsystems nachweisen müssen. Trotz der Aufregung zur Sicherstellung der Einhaltung des SOX ist eine Einführung eines angemessenen Zugriffskontrollsystems vorteilhaft. Die Rollen basierte Zugriffskontrolle (RBAC) ist eine solche Möglichkeit. In großen Organisationen stellt sich jedoch den Sicherheitsbeauftragten als Hauptfrage wie kann er die sich ständigen Änderungen in gesetzeskonformer Weise zeitgerecht in das System übernehmen.
- Adaptive Geschäftsprozesse entstehen aus der Tatsache heraus, dass Geschäftsanwendungen von monolithischen Einzelapplikationen zu service orientierten Architekturen migrieren. Flexible Anpassungen wegen

sich ändernder Geschäfts-Anforderungen entstehen durch Auf- und Abbau von Aufgaben (Abläufen) virtueller Organisationen. Um die sichere Ausführung von Geschäftsprozessen zu gewährleisten, ist ein verteiltes semantisches Gerüst / Rahmenwerk Voraussetzung. Die Nutzung desselben semantischen Gerüstes erlaubt es verschiedenen Systeme zu kommunizieren. Speziell für die Sicherheitsanforderungen der (Web-) Prozesse, ist es notwendig Einschränkungen und Möglichkeiten in einer gemeinsamen, verständlichen Art und Wiese zu definieren.

Diese Forschung sieht die Lieferung innovativer Sicherheitslösungen für die virtuelle, service-orientierte und entstehende Unternehmen voraus, welche eine sichere Ausführung von Geschäftsprozessen und eine vertrauensvolle Zusammenarbeit zwischen Organisationen ermöglicht. Weiterhin hat ein effizientes Authorisierungs-Management von diesen Lösungen folgende Voraussetzungen: (1) Abbildung von Geschäftsabläufen in eine Sicherheits-Policie und (2) ermöglichen der Zusammenarbeit von unabhängig verwaltenden Einheiten/Organisationen.

Diese Arbeit beschreibt Gracia, ein Sicherheits-Gateway welches entwickelt wurde den Authorisierungsanforderungen von service-orientierten Architekturen in Unternehmen zu entsprechen. Ein Sicherheits-Gateway ist ein Ausdruck der sich auf ein einheitliches Sicherheitskonzept (Gerüst) bezieht , die Integration von sicherheitsrelevantem Wissen und die Auswertung und Durchsetzung von (policy compliances“). Auf dem Konzept einer virtuellen Wissens Community basierend (VKC) und einer generischen Abstraktion für Agenten basiertes Wissensaustausch (AOA), stellt Gracia ein generisches Platform um Wissen auszutauschen zwischen unterschiedlichen Einheiten oder Unternehmen bereit. Gracia enthält eine eigene Sprache für die Spezifikation von Sicherheitsanforderungen, die Gracia Policy Language (GPL), welche nach der Syntax und Semantik der Web Ontology Language (OWL) und der Semantic Web Rule Language (SWRL) und der Inferenzmodell für die Auswertung von Policies abgeleitet ist/wird. Mit dieser Sprache und dem dazugehörigen Interferenzmodell ist Gracia in der Lage Policies zu modellieren, welche sowohl Geschäftsregeln als auch regulatorische Einschränkungen (Compliance) erlauben.

Die zweite Hälfte dieser Arbeit beschreibt das Design und die Implementierung des Gracia Gerüstes für die standard webservice Plattform. Diese Arbeit verfolgt 2 Ziele, erstens die praktische Demonstration/Machbarkeit von

Gracia und zweitens zu erforschen, ob ein Policy basierter Sicherheitsgateway für den unternehmensweiten Einsatz möglich ist.

Eine wichtige Voraussetzung in solch einem System ist die Wissensintegration, die den Anteilseignern der Geschäftsprozesse erlaubt ihr Wissen zu teilen und eine sicherheitsorientierte Herangehensweise an die Prozessebene erlaubt. Diese Voraussetzung, ein Wissens-Integrations-Gerüst für Gracia zu erstellen, wird aufbauend auf das VKC Gerüst von IAKS an der Universität von Karlsruhe entwickelt.

Abstract

The advent of Web Services and service-oriented architectures is fundamentally changing the way we build our internal systems and how internal and external systems interact with each other. To reduce the costs of software systems while at the same time increasing the capabilities of the systems, more and more companies and organisations are adopting their IT systems to Web Service technologies. One of the most important problems, which prevents the industry from producing and implementing a standards-based architecture, is Security.

In response to security concerns, IBM and Microsoft have collaborated on this proposed Web Services security plan and roadmap for developing a set of Web Service Security specifications that address the problem of how to provide protection for messages exchanged in a Web service environment. Standard organisations like OASIS and W3C have also proposed SAML[1] and XACML[2] to provide an XML schema for representing authorisation and entitlement policies and an XML-based framework for communicating user authentication, entitlement-, and attribute information.

However, despite the fact that a lot of effort has been put into solving security concerns, it is still not able to be said that the web service technology is secure enough. Huge gaps between business and IT, frequently changing organisation structures and flexible business process/model give us new challenges for building up a system with a scalable, flexible and easily manageable security framework. This research is mainly motivated by the following two facts in the real world.

- Under Section 404 of Sarbanes-Oxley (SOX), public companies that have a market capitalisation of more than $75 million, must attest to the effectiveness of internal controls and audit processes. One of the few aspects of SOX compliance that makes sense is the adoption of a proper access control method. Role-based access control (RBAC) is one of such methods. In a large organization, the key question for the Security Manager is how to keep the consistence of RBAC policy with respect to regulatory compliance and reflect the changes in the real world to RBAC policy in time.

- Adaptive Business Process is emerging from the fact that business applications are moving from standalone systems to service oriented architectures, flexibly adapting to changing business needs and serving them optimally through building

[1] http://www.oasis-open.org/committees/security/

[2] http://www.oasis-open.org/committees/xacml/

and dissolving task-driven virtual organisations. To enable secure execution of business processes, we need firstly a shared semantic framework. Utilising the same semantic framework enables various systems to understand each other. Particularly for the process-level security requirements, the first step is to specify the constraints and capabilities in a commonly understandable way and with the same vocabulary.

This research envisions the delivery of innovative security solutions for the virtual, service-oriented-, and evolving enterprise, enabling secure execution of business processes and trusted collaboration between organizations. Furthermore, effective management of authorisation in these solutions requires: (1) the mapping of business level policies into security policies; (2) enabling collaboration between independently administered domains/organizations.

This thesis describes **Gracia** - security **G**ateway for **R**isk **A**nd **C**hange **I**nformation m**A**nagement, a security gateway designed to address the authorisation needs for service-oriented enterprise computing. A security gateway is a term coined to refer to a united policy management framework for the specification of security policies, the integration of security-related knowledge, and the evaluation and enforcement of policy compliances. Based on the concept of the Virtual Knowledge Community (VKC) and a generic abstraction for agent-based knowledge exchange (AOA), Gracia provides a generic platform for sharing and exchanging knowledge among different domains and organisations. At the heart of the Gracia framework is a language for the specification of security policies, the Gracia Policy Language (GPL), which shares the syntax and semantics with Web Ontology Language (OWL)[3] and Semantic Web Rule Language (SWRL)[4], and the inference model for evaluating policies expressed in GPL. With the policy language and its inference model, Gracia is able to model policies, which can express both business rules and constraints from the regulatory compliance.

The second half of the thesis describes the design and implementation of the Gracia framework for the standard web service platform. The goal of this work is twofold: Firstly, to demonstrate the practical feasibility of Gracia, and secondly, to investigate the use of a policy-driven security gateway for enterprise system. An important requirement in such systems is knowledge integration that allows stakeholders of business processes to share their knowledge and enable trust management during the knowledge sharing process. Addressing this requirement, a knowledge integration frame-

[3] http://www.w3c.org/TR/owl-features

[4] http://www.w3.org/Submission/SWRL/

work for Gracia is developed on top of the VKC framework[5] from IAKS at the University of Karlsruhe. To enforce the Gracia Policy in the real world web service systems, we have also exploited the possibility to convert Gracia Policy into standardised Web Service security policies such as XACML and WS-Policy[6]. Although full mapping between these languages is still infeasible, our approach proves the potential usage of semantic policy language (GPL) in web services security scenarios.

[5] http://avalon.ira.uka.de/iaks-calmet/papers/SOIC2006.pdf
[6] http://www.ibm.com/developerworks/library/ws-polfram

Contents

Chapter 1

Introduction

1.1 Introduction

During the last decade, the IT industry has been increasingly faced with the task of integrating networked information sources and applications across heterogeneous hardware and software platforms. Initially, the main driving factor for this trend was the demand for the integration of different state-of-the-art systems with incompatible legacy IT systems within the intranets of large organisations, such as systems for ordering, parts tracking and billing. Another reason was the advent of the Internet in the commercial arena during the mid-1990's, which enabled new forms of information access and exchange, both for businesses and consumers.

The advent of Web Services and service-oriented architectures(SOA) has fundamentally changed the way we build our internal systems and how internal and external systems interact with each other. Web services are software components that can be accessed via the Internet using popular web mechanisms and protocols such as the hypertext transfer protocol (HTTP). Public interfaces of Web services are defined and described using extensible markup language (XML) based definitions. The emerging Web service technology has been receiving an enormous amount of attention and dissemination both in the academic and the industrial world since the publication of the first Web service specifications in 2000. The primary reasons for the immense interest are: (1) the standardisation of the Web service technology is driven by leading industrial companies and organisations. Competing companies work together in order to ensure the interoperability of Web service standards. The specifications for Web services are free of royalty. Software vendors and users need not pay any license fees for their applications based on Web service specifications. (2) The Web service technology is

fully based on XML and Internet technologies such as HTTP, TCP/IP. Therefore, Web services are independent of hardware, programming, software and operating platforms. This provides developers and software vendors greater flexibility when developing either Web service infrastructures or Web service applications.

In order to reduce the costs of software systems whilst at the same time increasing their capabilities, more and more companies and organisations are adapting their IT systems to Web Service technologies. One of the major problems that prevents the industry from producing and implementing a standards-based architecture is Security. In response to security concerns, IBM and Microsoft have collaborated on this proposed Web services security plan and roadmap for developing a set of Web Service Security specifications that address the problem of how to provide protection for messages exchanged in a Web service environment. Standard organisations like OASIS and W3C have also proposed SAML [88] and XACML [84] to provide an XML schema for representing authorisation and entitlement policies and an XML-based framework for communicating user authentication, entitlement and attribute information.

However, despite the fact that a lot of effort has been put into solving security concerns, it is still cannot be assumed that the web service technology is adequately secure. Huge gaps between business and IT, frequently changing organisation structure and flexible business processes/models provide new challenges for building up a system with a scalable, flexible and easily manageable security framework. From the following three real world scenarios, we introduce the motivation of this research.

1.2 Research Motivation

1.2.1 Regulatory Compliance

Understanding the issues surrounding regulatory compliance can be a difficult and frustrating endeavor. Most developers or system administrators do not have a legal background and regulators generally don't have a background in IT. The result is a failure of communication: The language and requirements described in legislation are not easy to pin to explicit software requirements. The following table provides a brief description of each act and where it is applicable:

Sarbanes-Oxley

In the wake of corporate financial scandals such as the Enron disaster of 2001, U.S. Congress passed the Sarbanes-Oxley act (commonly abbreviated as SOX). SOX was

Act	Target
Sarbanes-Oxley	Privacy and integrity of financial data in publicly traded corporations
HIPAA	Confidentiality, integrity, and availability of health care information.
BASEL II	Confidentiality and integrity of personal financial information stored by financial institutions. Availability of financial systems. Integrity of financial information as it is transmitted. Authentication and integrity of financial transactions.

Table 1.1: Regulatory Requirements

signed into law in 2002. Its purpose is to give investors more confidence in the financial reporting process of publicly-traded companies by putting controls in place to ensure the confidentiality and integrity of financial data. The act applies to companies that are publicly traded in the United States, but has far-reaching international applicability due to the fact that many large foreign companies are also traded on the U.S. stock exchange. The key part of the SOX act for developers is Section 404 entitled ”Management assessment of internal controls”. This section requires management to take responsibility for the integrity of financial information by evaluating IT systems and processes and producing evidence that the company has done a reasonable job at keeping sensitive information safe. Whilst SOX doesn't address IT directly, the implications for IT are huge, given that most financial data flows through computerised information systems and Programmer's code.

SOX has been a major driver for IT security. Section 302 of the Sarbanes-Oxley act requires that the CEO and CFO issue periodic statements certifying that adequate controls are in place for the control of financial information within their organisation. Unawareness of a vulnerable system is no longer a defense because top executives now have to attest to the implementation of proper controls. To be SOX compliant, companies must have regular external audits that assess the controls that are in place to ensure that data is accurate, unaltered, and offers a true representation of the company's financial position. SOX has promoted significant investment into IT and IT security. For most companies, most of this information-flow takes place through IT systems. This

means that IT needs to provide assurance that the data:

- Cannot be altered by unauthorised individuals.
- Cannot be viewed by unauthorised individuals.
- Is available when needed by authorised individuals.

Compliance with the SOX Act requires that companies implement practiceable access controls on data and service and ensure that system vulnerabilities are patched, disallowing unauthorised modification or leakage. One framework that is commonly used to help IT comply with the needs of SOX is COBIT (Control Objectives for Information and Related Technologies)[1], an open standard published by the IT Governance Institute and the Information Systems Audit and Control Association[2].

HIPAA–Health Insurance Portability and Accountability Act

The Health Insurance Portability and Accountability Act (HIPAA)[3] was passed in 1996 by the U.S. Congress. It established federal regulations that force doctors, hospitals, and other health care providers to meet baseline standards when handling electronic protected health information (ePHI), such as medical records and medical patient accounts.

Before the enactment of HIPAA, personal information that accumulated in various private databases was thought to be the property of the organisation that owned the database. The major underlying concept of HIPAA is the notion that the owners of databases are not necessarily the owners of the data contained therein - they are only intermediaries. This is a fundamental paradigm shift, because HIPAA compliant organisations must ensure that record owners are guaranteed:

- Access to their own records and the right to request the correction of errors.
- Prior knowledge pertaining to how their information will be used.
- Explicit consent from the involved individuals before ePHI can be used for marketing.
- The right to request that any form of communication between organisations and individuals is kept private.

[1] http://www.isaca.org/cobit
[2] http://www.isaca.org
[3] http://www.cms.hhs.gov/hipaa

- The right to file formal privacy-related complaints to the Department of Health and Human Services (HHS) Office for Civil Rights.

There are several provisions of HIPAA, divided into two title sections. Title I deals with the portability of health insurance coverage for employees and families when changing jobs, and was also created to ensure that employees with pre-existing medical conditions cannot be unfairly denied medical coverage. Title II contains the "Administrative Simplification" provisions and has three subcategories: privacy provisions, HIPAA Electronic Data Interchange (HIPAA/EDI), and Security Provisions. The security provisions are the primary concern for developers due to the fact that they contain specific technical compliance objectives. The security provisions obligate companies to:

- Ensure the confidentiality, integrity, and availability of all ePHI that the health care entity creates, receives, transmits, or maintains.
- Prevent disclosure of any ePHI information that is not permitted or required.
- Ensure that system information is available for auditing trails.
- Ensure that authentication is in place so that specific workers or entities are who they say they are.

BASEL II

BASEL II[4] is officially known as the International Convergence of Capital Measurement and Capital Standards. It is a framework established by the Basel committee, a consortium of Central Governing Banks from several countries. The purpose of BASEL II is to revise the existing international standards used to measure the viability of a bank's capital. The previous BASEL accord is considered to be out of date because there are several realities of modern banking that are not adequately reflected in the regulations. For example, the original BASEL accord does not take into account arbitrage across markets.

The majority of BASEL II is worded for banking professionals. Given that BASEL II is an international standard, it is written in such a way that it can be applied to a variety of banking systems worldwide. This fact is the reason why the requirements are quite vague from an IT perspective, at least in terms of actionable compliance targets. However, there is a substantial collection of information available that describes risks and mitigation steps for electronic banking and financial services. The following process steps will help ensure BASEL II compliance:

[4] http://en.wikipedia.org/wiki/Basel$_{I}I$

- Prevent improper disclosure of information.
- Prevent unauthorised transactions from being entered into the computer system.
- Prevent unauthorised changes to software during routine development and maintenance that allow fraudulent transactions to be generated, leave certain kinds of operations unchecked, or disable logging with the goal of bypassing auditing so that actions may proceed unnoticed.
- Prevent the interception and modification of transactions as they are transmitted through communication systems such as telephone, data, and satellite networks.
- Prevent interruption of service due to hardware or software failure.

The following strategies can help ensure that the above process steps are satisfied.

- Change Management: It is important that developers be held responsible for changes that are made to software systems. Modifications to bank software that allow fraudulent transactions to be executed have not been perpetrated by viruses from external sources, but rather by disgruntled developers or employees. Accountability measures need to be put into place to prevent people from inserting arbitrary changes to critical bank software systems. Code reviews should be conducted frequently by teams of developers and testers to ensure changes are justified. Source control systems should be put into place to prevent unauthorised changes, ideally by enforcing the use of access control lists, or at the very least, by flagging recent changes for review so that a clear auditing trail can be constructed in the event of suspected wrong-doing.
- Authentication: It is necessary to ensure that the transactions that take place within banks are executed solely through legitimate agents. Authentication is the primary means of ensuring that agents are who they claim to be. When setting up authentication systems, the utilisation of strong passwords is a must. ”Guest” accounts and anonymous access should be disabled. Passwords should be stored using a sufficiently strong form of encryption to protect the credentials from attackers who might gain access to them. Technologies such as Secure Socket Layer (SSL) and digital certificates should be employed. The enforcement of password policies that re-set credentials on a semi-regular basis is also recommended.

Policy-based Access Control

From a fundamental information security and controls perspective, it is clear that Web service security is crucial to regulatory compliance. The requirements for regulatory compliance apply to any system that processes or maintains data. Given that most data is stored, accessed, and maintained in electronic format which is often accessed through Web service components, there is a significant correlation between this information and Web Services.

As with most information security initiatives, the requirements for regulatory compliance[5] are policy-driven in areas such as:

- User authentication
- Password management
- Access controls
- Input validation
- Exception handling
- Secure data storage and transmission
- Logging
- Monitoring and alerting
- System hardening
- Change management
- Application development
- Periodic security assessments and audits

If security policies designed to maintain regulatory compliance are not put into place and enforced with adequate business processes and technical controls, Web services can easily expose systems to danger. Regarding all of the attention that regulatory compliance has been receiving, one of the few aspects that makes sense is the adoption of a proper access control method. In a large organisation, the key question posed to the security manager is how to maintain the consistency of access control policy with respect to regulatory compliance and reflect the changes in the real world to access control policy in time. The other requirements also include:

[5] http://www.spidynamics.com/spilabs/education/articles/sarbanes-oxley.html

- enabling user single sign-on (SSO) to the applications and content they are authorised to see
- easily extending access control to include Web services applications via standards such as Security Assertation Markup Language (SAML) [88]
- centrally managing access to applications and information via policy, providing a single point of policy enforcement and audit of access for all users
- applying rules, in accordance to corporate policy, for the augmentation of internal controls for regulatory compliance, including time-specific restrictions or access control based on the location of the originator.

1.2.2 Security in Adaptive Business Processes

Business processes integrate systems, partners, and people to achieve key strategic and operations objectives. Adaptive business processes can be defined, refined, and optimised to respond to changing business environments, government regulations and competitive pressures. Through the evolution of mainframes, Management Information Systems (MIS), packaged applications, J2EE-based application platforms, business process management systems (BPMS), Service Oriented Architectures (SOA) are the current focus.

Business process management systems have traditionally not been designed for dynamic environments requiring adaptive response. Currently, the need for adaptive business process is being driven by the demands of e-commerce in both B2B and B2C. Initial B2B automation activities were centered around Electronic Data Interchange (EDI) initiatives. More recent work in the B2B space has focused on the development and deployment of ebXML (electronic business XML). With both EDI and ebXML the collaborating business partners predefine the terms of their electronic interaction. In comparison, views toward virtual organisations require flexible, on-the-fly alignment of business partners; in other words, adaptive capabilities.

The currently available business process management systems span a wide range of capability. This is not surprising considering that businesses in any sector can benefit from business process management. For example, the insurance industry has benefited greatly from document management systems, which reduce physical paperwork, increase the availability of documents and control the flow of information during processing. Collaborative and production-oriented business processes are distinguished by measures of structure and centricity. Collaborative business processes are

information-centric. Typically, human interpretation of information drives the business process in a loosely structured manner. In comparison, production business processes are process-driven due to their highly repetitive nature. To achieve the efficiency required of production business process, the processes are highly structured. The requirements of adaptive business process fall between these two broad categories [20].

Adaptive business process needs to react to changing environmental conditions. Currently, businesses change their processes through two primary mechanisms: Business Process Reengineering (BPR) and Continuous Process Improvement (CPI). BPR is the periodic analysis and subsequent redesign of the intra- and inter-business processes used by an organisation. BPR is used to overhaul processes in order to create operational efficiencies that improve quality and save time and cost. Conversely, CPI focusses on continuous improvement through the application of small and orderly changes. Business processes are continuously examined in order to find ways to increase quality and reduce waste. Adaptive business processes respond to changing conditions through adaptive change. Adaptive change is not constrained by measures of frequency or impact [20].

Current business processes initiatives have embraced the Web service model. Given the current state of technology, Web service-based business processes are typically deployed behind corporate firewalls and are used for intra-organisational workflow. The reason for this is that Web service specifications are weak in regards of security issues and transaction management. Inevitably, as standards evolve to address these deficiencies, business processes will shift from the domain of intranets to that of the Internet. This transition will be accompanied by a new set of problems.

- Security: To enable the secure execution of business processes, we first need a shared semantic framework. Utilising the same semantic framework enables various systems to understand each other. Particularly for the process-level security requirements, the first step is to specify the constraints and capabilities in a commonly understandable way and with the same vocabulary.
- Transaction Management: When an intranet-based business process system executes, it does so with a collection of services that are owned and managed by the same organisation. In this environment, service interruptions are infrequent and typically scheduled due to consolidated system management. In contrast, Internet-based business processes must be designed for resilient operation as service partners periodically become unavailable due to decentralised system management and the lack of network service guarantees. The evolution from intra- to internet-based business processes will increase the design and run-time

complexity, since the coordination mechanism must become more faults-tolerant.

1.2.3 Service Platform for Mobile Network

In 3G, the convergence of telecom and datacom environments will encourage service providers and operators to change the way in which services are created. Today, service creation focusses on network technology, but in 3G applications content will drive service development.

IP Multimedia Subsystem (IMS)[6] is a set of specifications that describes the Next Generation Networking (NGN)[7] architecture for implementing IP based telephony and multimedia services. IMS defines a complete architecture and framework that enables the convergence of voice, video, data and mobile network technology over an IP-based infrastructure. It fills the gap between the two most successful communication paradigms, cellular and Internet technology. The vision for IMS is to provide cellular access to all the services that the Internet provides.

IMS architecture supports a wide range of services that are based on SIP[8] protocols. An IMS architecture delivers multimedia services that can be accessed by a user from various devices via an IP network or traditional telephony system. The underlying network architecture can be divided into three layers (Device Layer, Transport Layer, and Control Layer) plus the service layer. Service providers are eager to allow their customers to be able to develop and implement services that alleviate the existing service resources described above. However, many enterprise application developers may have an IT background but are not familiar with complex telephone protocols (e.g. SIP, ISDN etc.); and they need a simple API for services creation and development. The solution is Parlay X SOA (Service-Oriented Architecture)(Figure 1.1)[9], which has been defined by Parlay Group in 2003 in order to provide a set of simple-to-use, high-level, telecom-related Web services. The idea behind Parlay X is to provide Web services in a context that is already widely adopted and understood by a large number of developers and programmers and to do so in an environment where there are a variety of development tools available. With the Parlay X SOA Web services interfaces, the application developers can access and influence the existing IMS services more easily through Web services. The Parlay X SOA Web services are connected to the telecommunication network via either the **Open Services Access - Gateway** (OSA-GW) or directly through

[6] http://www.3gpp.org/specs/numbering.htm

[7] http://www.itu.int/ITU-T/ngn/fgngn/index.html

[8] http://www.sipknowledge.com/SIP$_RFC.htm$

[9] http://www-128.ibm.com/developerworks/library/ws-soa-ipmultisub1

data service components over IP Protocols.

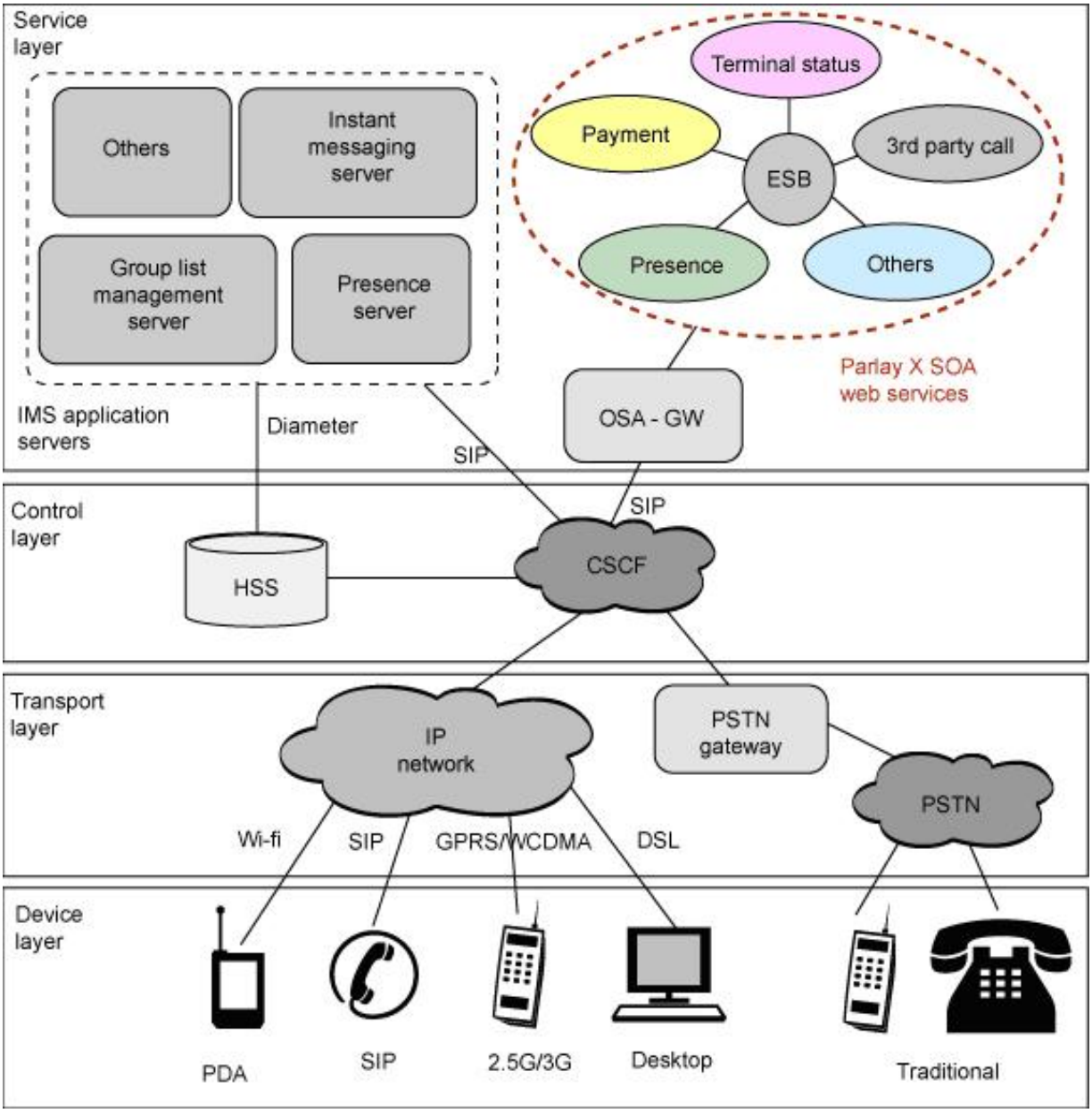

Figure 1.1: A high-level view of the Parlay X SOA web services within the IMS architecture

SPICE[10] (Service Platform for Innovative Communication Environment) addresses the remaining unsolved problem of designing, developing and executing efficient and innovative mobile Service creation/execution platforms for networks beyond 3G. This project

[10]http://www.ist-spice.org

will research, prototype and evaluate an extendable overlay architecture and framework to support easy and quick service creation, test and deployment of intelligent mobile communication and information services. Building on significant advances in IT technologies, the SPICE platform will support multiple heterogeneous execution platforms allowing for new, innovative services to be spread across different operator domains and over different countries, realising a variety of business models. For users, operators and service providers, the SPICE project will turn today's confusing heterogeneity into a rich and easily manageable service environment by exploiting the diversity of device capabilities and fostering service adoption. The SPICE approach will broaden business opportunities in the communications and associated business sectors. Service continuity from fixed to mobile access and seamless roaming of services across operators and networks is far from being a reality. In this context, the main challenges in SPICE are:

- To provide end-users with a means of communication and tailored applications anywhere, anytime and on any device;
- To supply service providers and non-professional users with service enablers that facilitate and accelerate application development.
- To allow operators to take up the role of Service Provider
- To build a user-transparent infrastructure that hides the complexity of services and applications crossing over different access domains, offers a diversity of services and copes with the various access network technologies.

The SPICE Project is structured into eight technical work-packages (WP). WP6 (Service Access Control and Trust Management) focusses on all aspects related to controlling access to the service platform for users and third party service providers. This includes providing a security framework to support user and service authentication, authorisation, non-repudiation in single- and multi-domain-environments and also methods for management and enforcement of service-level agreements. The SPICE platform is required to provide interfaces and a service that will take in two access policies for each of the combined services and outputs another access policy that will be used for the combined service.

1.3 Security Challenges

Traditional approaches to security - specifically network firewalls - are insufficient at dealing with the emergence of new threats. In order to attain a better understanding of these threats and the solutions proposed in later chapters, this section provides some background information about the field of IT security. A detailed definition of computer security is provided by Bruce Schneier in the foreword to Ross Anderson's book on Security Engineering [4]. The definition concentrates on the characteristics of an attacker, the entity whose actions lead to a dangerous situation:

Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary trying to ensure that things fail in the worst possible way at the worst possible time . . . again and again.

Fundamental to computer security are the concepts of confidentiality, integrity and availability (CIA), which refer to keeping data secret, ensuring data remains intact and ensuring systems are responsive. The four implementation domains representing the foundation of security implementations [58] are: *Authentication*, *Authorization*, *Privacy* and *Audit*.

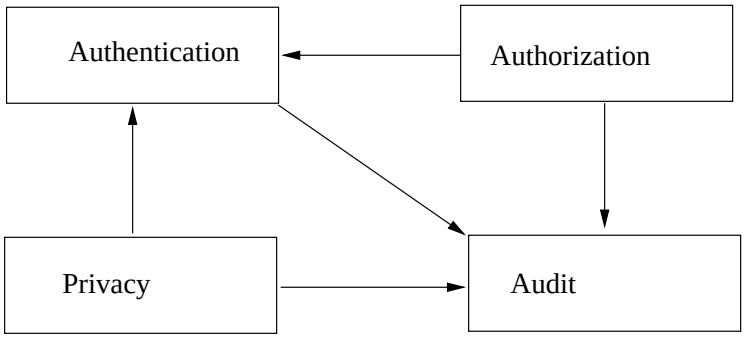

Figure 1.2: Dependencies between Security Domains

These security implementation domains are inter-dependent and Figure 1.2 illustrates the dependencies between them. For example, it is not possible to implement authorisation without authentication. Likewise, both authorisation and authentication rely on auditing to ensure that security violations are captured for analysis, etc. These domains are described in this section. In particular this thesis focusses on the security of service-oriented systems and therefore provides a brief introduction to SOA security. This section then finishes with a brief note about the following security aspects: *Internal*, *External* and *Legal Security*.

1.3.1 Authentication, Authorisation, Privacy and Audit

Authentication is concerned with the identification of the communication or collaboration participants. Authentication can be explicit or implicit. In the former case a party has to explicitly authenticate all incoming requests, whilst in the latter it ”trusts” the invoking parties to perform authentication. Such notion of trust can be used to describe trust zones. There are several authentication approaches employed in today's systems[11]:

- End-user Web applications usually use web-based authentication mechanisms ranging from simple basic authentication requiring input of user credential in the HTTP header to sophisticated mutual certificates authentication schemas, with the majority of authentications based on either corporate directory services or specialised applications databases.
- Rich client applications usually implement programmatic authentication, requiring the consumer to explicitly supply credentials, which are then validated using either local operating system security, corporate directory service, or an underlying database.
- Mainframe applications usually use Resource Access Control Facility (RACF)[12] based authentication mechanisms.

Authorisation domain ensures that, once the identity of the requester has been established, it is possible to limit their access to operations, data, or other resources. Authorisation assumes authentication - it is necessary to know who the requester is in order to decide what they are allowed to do. There are two primary approaches to implementing authorisation:

- Authorisation of Individual Parties: Every party (user) can be assigned an explicit set of access rights to functions. This authorisation approach was widely used for standalone, rich-client applications with a limited amount of users. However, it doesn't scale well for Web-based applications with a significant user population.
- Authorisation through Roles: Various roles can be created for each application and access rights can be assigned to these roles instead of individual parties. When each party is authenticated, the credentials applied to the party are mapped to the party's role (usually through groups in the directory service), which then

[11] http://orchestrationpatterns.com/serviceSecurity
[12] http://www.ibm.com/eserver/zseries/racf

determines whether or not the party is authorised to access a particular function. This authorisation approach scales well. Role-based authorisation provides the foundation for the security implementations of many component-based systems.

There are several distinctly different concerns associated with the privacy domain:

- Identification of the end user(s) and/or service(s) that created the message/document: This is usually achieved by signing the message/document, potentially using multiple signatures. This implementation relies on a number of common standards for digital signatures and is used in B2B commerce, government commerce, and even more widely in corporate email communications.

- Ensuring the message's privacy, either by sending it through a secure channel or by encrypting or otherwise securing the content: Two examples of such implementation can be seen in the sending of plain text messages over a secure transport such as HTTPS or TLS or encrypting and sending a message over an insecure channel.

Auditing should provide the necessary capabilities for storing all of the service's messages. Whilst auditing alone does not necessarily suffice for the purposes of non-repudiation, auditing of the message exchange (proof of content) along with authentication (proof of origin) generally provides the required level of proof.

1.3.2 SOA Security

Organisations have existing security infrastructures in place. These infrastructures protect resources on diverse platforms and provide implementations for all or some of the security domains described above. This SOA model breaks both application and application-oriented security boundaries. The services abstraction layer brings together multiple disparate applications into one cohesive service environment. This shift requires a new security approach, above and beyond that of the existing application security implementations.

Application-Level Security

Exact definitions of application-level security vulnerabilities vary. SPI Dynamics report [30] defines application-level security vulnerabilities as those weaknesses created when the constituent components of an application are combined together.

The report by Gartner [91], ”Application-Level Firewalls Required”, describes how current security measures are targeted at the network-level. They operate by blocking network packets destined for vulnerable services. Gartner predicts that, in the future, systems will be attacked primarily at the application-level; i.e. rather than attack the system hosting the service (where the system is protected by a network-firewall and security is at it's strongest), hackers will instead attack the service itself.

1.3.3 Security Aspects

Strong security mechanisms are required for two reasons: on the one hand money and digital signatures are an attractive aim for attacks and on the other hand it is important to create trust in the system. These challenges were tackled by a three-level security categories to increase the reliability of a system [21].

- External security: This level deals with the protection of a system against outside attacks. The availability of the system, authorisation mechanisms and encrypted communication are used at this level.
- Internal security: Once the external security barrier has been broken, the internal security mechanisms allow identification and elimination of intruders by monitoring the system. At this level so-called malicious players which come from inside the system can also be detected.
- Legal security: E-business includes transactions in order to achieve valid contract conclusions. A legal instance must be provided, whose actions are similar to those of a notary and which on the one hand assures the service consumer that the service provider is really authorised to provide the service and on the other hand supports service level negotiation. This instance might differ between countries, states or even companies, depending on local law or policies, but it always requires security in order to protect the service provider, service consumer and mediator.

1.4 Requirements of the Security Gateway

This section identifies the research issues and the requirements of a security gateway that enables policy enforcement and specification for Web services in a service-oriented environment. The security gateway should also define access privileges for service

consumers in terms of authorisation to use or access the shared resources. Furthermore, the management of the knowledge of services consumers and providers is important to ensure that appropriate actions are executed in response to events triggered by changes in context and occurrence of security violations, e.g., malicious attempts to access the shared resources.

We will now discuss the requirements of the security gateway for Web services protocol in terms of policy specification to enable the establishment of knowledge sharing, access control and trust in knowledge from members of the services environment.

1.4.1 Policy Specification

Policies have been used in many application areas for managing distributed systems. This is because a policy-based approach is flexible, scalable and permits adaptation to changes in security requirements and context by dynamically loading and removing policies from the system without interrupting its function. It is therefore important to identify the required policies for Web services protocols and provide a consistent way of expressing them. The following is a list of policies that are required for our security gateway:

- Authorisation policies regulate the behaviour of service partners by controlling access to the services.
- Obligation policies manage and adapt to context changes. When an event occurs, obligation policies trigger the execution of actions for applying these changes.
- Policies to express security or other requirements such as cardinality, separation-of-duty constraints and other law constraints.

1.4.2 Trust and Knowledge Management

It is also important to discuss trust between service knowledge sources in SOA. It should be noted that knowledge-sharing in SOA cannot rely upon the availability of a fixed network infrastructure, but rather participants can only rely on their own knowledge and information held by other peers in the service transaction processes. Consequently, they need to establish some form of trust-relationship between each other in order to ascertain the trustworthiness of the information relayed in the services life cycle.

1.5 Thesis Contribution

This thesis describes a semantic policy-based security gateway to support the secure establishment and management of a Web services environment. There are several innovations that constitute the contribution of this thesis. The first one is the novel use of policies to define the security and regulatory constraints for the establishment of a policy framework. The second innovation is the use of evidence-based trust brokering protocols to assist the knowledge sharing in SOA by relying on the security information available in Web services as well as information provided by other agents in the so-called Virtual Knowledge Community, which will be described in the following chapters. The last innovation is the design of the security gateway in order to implement our ideas and also provide a prototype for our work.

We propose to use a *Gracia Policy* to define the regulatory requirements, the characteristics that Web services must exhibit in order to be eligible to interact with business processes as well as the policies governing their behaviour for security purposes. This approach is well suited to the message level as it requires relatively little consideration of the process level and avoids the need for negotiation. Furthermore, it allows additional information to be conveyed as part of the *Gracia Knowledge Base* (e.g., Airport Security Regulation), provided that the knowledge-owner is trusted. It also allows for the establishment of trust-relationships between the knowledge sources. We use ontology to classify the knowledge concepts and instances. For the moment, we represent the ontology about the security constraints and domain knowledge in a description logic language called OWL-DL [79] and the rules in Semantic Web Rule Language (SWRL) [46].

An evidence-based trust brokering protocol was designed in order to assist with trust management in the *Gracia Knowledge Base*. Trust assertions can be calculated with evidence in order to attest to the reputation of the knowledge source which they have previously verified or they know about.

As a proof-of-concept, this thesis will document a prototype implementation of the security gateway for web service protocols, which covers the establishment and management of knowledge communities as well as the enforcement of access control policies. The implementation also includes a Gracia Policy Editor to support the creation of Gracia Policy and the Knowledge Management Tool for managing the Gracia Knowledge Base based on Virtual Knowledge Community (VKC) [21].

1.6 Dissertation Outline

Chapter 3 introduces the related work on Web services security and knowledge management in SOA. Chapter 3 begins with a discussion of the Web services protocols and the different architectures used to apply them. Related business processes specification language, agent-based knowledge management and business rules will also be discussed.

Chapter 4 defines the concept of security gateway and presents a real-world scenario including regulatory requirements and policy specification. An overview of the author's security gateway-**Gracia** (*security **G**ateway for **R**isk **A**nd **C**hange **I**nformation m**A**nagement*) will be provided. The reader will be introduced to the Gracia policy specification with reference to the required policies with formalisation of security constraints and access control information. Chapter 4 will then conclude with a description of the Gracia Knowledge Base and the evidence-based Trust Brokering Protocols.

Chapter 5 introduces different approaches to the specification of security policy based on the logic and semantic web technology. Additionally, the Gracia Policy language and the representation of security constraints in a formal and semantic way will be discussed.

Chapter 6 expands on the evidence-based trust brokering protocols and provides details about the protocols based on the VKC. It also explains the hybrid approach to the Gracia Knowledge Base.

Chapter 7 includes a description of the architecture and evaluation of the framework by presenting the implementation of the knowledge management and the access control model. Two supporting tools will also be introduced, namely the Gracia Policy Management Tool and the Gracia Knowledge Management Tool. Additionally, this chapter presents the evaluation of the security gateway in terms of its efficiency.

Chapter 8 provides a conclusion and suggests directions for future work.

Chapter 2

Background

2.1 XML Web Services and Security

According to the Web service specifications available at W3C[1], the definition of a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface defined by machine-processible format (specifically WSDL) [24]. Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages [40], typically conveyed using HTTP with an XML serialisation in conjunction with other web-related standards. Considering Web services itself as a layer, it could be placed between the application and transport layer in terms of the Internet Model.

2.1.1 Web Services Protocols

SOAP (Simple Object Access Protocol) [40], WSDL (Web Services Description Language) [24], and UDDI [13] are the standards for Web services. SOAP has been accepted and is being standardised by the W3C. WSDL has been submitted to the W3C for standardisation as the standard language for the description of Web services. UDDI is the standard directory service for the Web service.

SOAP

SOAP is a lightweight protocol initially proposed by Microsoft, IBM, and other organisations. The SOAP protocol supports XML document exchange and provides a convention for Remote Procedure Call (RPC) using XML messages. SOAP specifies a wire

[1] http://www.w3.org/2002/ws/

protocol for facilitating highly distributed applications. SOAP is similar to DCOM(Distributed Component Object Model) and CORBA[2] in that it provides an RPC mechanism for invoking methods remotely. SOAP differs in that it is a protocol based on open XML standards and XML document exchange rather than being an object model relying on proprietry binary formats. Both DCOM and CORBA use binary formats for their payload. The SOAP gateway performs a similar function to DCOM and CORBA stubs - translating messages between the SOAP protocol and the language of choice. As a result, SOAP offers vendor, platform, and language independence. With SOAP, developers can easily bridge applications written with COM, CORBA, or $EnterpriseJavaBeans^{TM}$ (EJB)[3]. In regards to specification, SOAP is composed of three parts:

- A framework describing how SOAP messages should be constructed
- A set of encoding rules for exchanging data types
- A convention for representing remote procedure calls

SOAP was never intended to provide a complete distributed object architecture. SOAP does not mandate a specific object model, therefore the specification does not address such issues as distributed garbage collection, message batching, objects-by-reference, object activation, or type safety. Although it is possible to handle request and response messages, the SOAP specification does not directly handle asynchronous communication. It is, however, possible to implement asynchronous communication using the SOAP protocol; for example, Microsoft's BizTalk messaging profile uses SOAP as its wire format and implements asynchronous message exchange.

SOAP is clearly becoming one of the de facto standards for Web services. Because of its platform and language independence and ease of integration, many companies are embracing SOAP as a backbone for their Web services strategy. Since the initial release of the SOAP 1.1 specification in May 2000, more than 30 SOAP toolkits and development environments have been developed to support a wide variety of development platforms (J2EE, .NET, CORBA) and languages (Java, C-sharp, C++, Visual Basic, Python). Additionally, the W3C has formed a working group to draft the next generation of the SOAP protocol.

WSDL

WSDL is a language designed for describing the capabilities of Web services. Proposed by IBM and Microsoft, WSDL combines the best of IBM's NASSL (Network Accessible

[2]http://www.omg.org/docs/formal/04-03-12.pdf

[3]http://java.sun.com/products/ejb/index.html

Services Language) and Microsoft SOAP Contract Language. WSDL is based on XML and is a key part of the UDDI initiative. The WSDL document specification helps improve interoperability between applications, regardless of the protocol or the encoding scheme. The WSDL 1.1 specification defines WSDL as ”an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages.” Essentially, a WSDL document describes how to invoke a service and provides information about the data being exchanged, the sequence of messages for an operation, protocol bindings and the location of the service. A WSDL document defines services as a collection of endpoints, but separates the abstract definition from the concrete implementation. Messages and port types provide abstract definitions for the data being exchanged and the operations being performed by a service. A binding is provided to map to a concrete set of ports, usually consisting of a URL location and a SOAP binding.

UDDI

Ariba, IBM, and Microsoft developed the first version of UDDI, the Universal Description, Discovery and Integration specification. As the name suggests, UDDI allows a business to describe the services it offers and to discover and interact with other services on the Web.

UDDI is also a cross-industry open specification that is built on top of existing standards like TCP/IP, XML, HTTP, DNS, and SOAP. At the heart of UDDI is the UDDI Business Registry, an implementation of the UDDI specification. With the registry, a business can easily publish the services it offers and learn what services other businesses are offering. The registry is created as a group of multiple operator sites. Although each operator site is managed separately, information contained within each registry is synchronised across all nodes. Business entities describe information about a business, including their name, description, services offered, and contact information. Business services provide more details about each service being offered. Each service can have multiple binding templates, each describing a technical entry point for a service; for example, mailto, http, ftp, fax, and phone. Finally, *tModels* describe the particular specifications or standards a service uses. With this information, a business can locate other services that are compatible with its own system.

2.1.2 Web Service Security Standards

Security is important for Web services due to the fact that more and more comprehensive services are being made available via the Internet and Web services are becoming

increasingly interconnected ”ad hoc” across organisations over the Internet. This environment is more hostile than the environments covered by more traditional middleware such as CORBA. In fact, the lack of security features is currently considered the most significant inhibitor for the large-scale adoption of Web services.

Apart from the typical security challenges of large-scale distributed applications across heterogeneous systems and administrational domains, Web services have to deal with the fact that XML documents contain links to other documents. It is currently unclear how these links can be secured. At the moment there is no complete security model for Web services. Some of the more promising emerging specifications are described in the following paragraphs.

The W3C consortium standardised several specifications related to cryptography: The XML Digital Signature (XML-DSIG)[4] provides integrity, signature assurance and non-repudiation for whole or partial documents. The XML Encryption (XML-ENC)[5] standard encrypts and decrypts digital content for whole or partial documents, and the XML Key Management Specification (XKMS)[6] provides a method for accessing public key infrastructure services for distributing, registering, and validating public keys.

The Web Services Security specification (WS-Security) [65] provides a set of mechanisms to help developers of Web Services secure SOAP message exchanges. Specifically, WS-Security describes enhancements to the existing SOAP messaging to provide quality of protection through the application of message integrity, message confidentiality, and single message authentication to SOAP messages. These basic mechanisms can be combined in various ways to accommodate the creation of a wide range of security models using a variety of cryptographic technologies.

WS-Security also provides a general-purpose mechanism for associating security tokens with messages. However, no specific type of security token is required by WS-Security. It is designed to be extensible (e.g. support multiple security token formats) to accommodate a variety of authentication and authorisation mechanisms. For example, a requestor might provide proof of identity and a signed claim that they have a particular business certification. A Web service, receiving such a message could then determine what kind of trust they place in the claim. Additionally, WS-Security describes how to encode binary security tokens and attach them to SOAP messages. Specifically, the WS-Security profile specifications describe how to encode Username Tokens, X.509 Tokens, SAML Tokens , REL Tokens and Kerberos Tokens as well as how to include

[4] http://www.w3.org/TR/xmldsig-core/
[5] http://www.w3.org/TR/xmlenc-core/
[6] http://www.w3.org/TR/xkms2/

opaque encrypted keys as a sample of different binary token types. With WS-Security, the domain of these mechanisms can be extended by carrying authentication information in Web services requests. WS-Security also includes extensibility mechanisms that can be used to further describe the credentials that are included with a message. WS-Security is a building block that can be used in conjunction with other Web service protocols to address a wide variety of application security requirements.

Message integrity is provided by leveraging XML Signature and security tokens to ensure that messages have originated from the appropriate sender and were not modified in transit. Similarly, message confidentiality leverages XML Encryption and security tokens to keep portions of a SOAP message confidential.

Security Assertion Markup Language (SAML) [88] is an extension of WS-Security from OASIS. SAML specifies the language used to exchange identity, attribute and authorisation information between parties involved in web service communication in an interoperable way. SOAP is used as the SAML request/response protocol transport mechanism. SAML requests and responses reside within the SOAP body. SAML could help in achieving Single-sign-on (SSO). SSO is a mechanism where the authentication context of a consumer can be maintained across multiple services.

In addition to WS-Security, web service security specifications also include : WS-Policy [6], which defines the rules for service interaction, and WS-Trust [97], which defines trust models for secure exchanges.

Web Service Policy (WS-Policy)

The WS-Policy and WS-Policy Attachment specifications offer mechanisms for a) representing the capabilities and requirements of Web services as Policies, b) determining the compatibility of policies, c) naming and referencing policies, and d) associating policies with Web service metadata constructs such as service, endpoint and operation.

A policy expression is the XML representation and interoperable form of a Web Services Policy. A policy expression consists of a Policy wrapper element and a variety of child and descendent elements. Child and descendent elements from the policy language are *Policy*, *All*, *ExactlyOne* and *PolicyReference*. Other child elements of Policy, All and ExactlyOne are *policy assertions*. (The Policy element plays two roles: wrapper element and operator.) Policy assertions can contain a nested policy expression. Policy assertions can also be marked optional to represent behaviors that may be engaged (capabilities) for an interaction. The optional marker is the *wsp:Optional* attribute which is placed on a policy assertion element.

Web Services Policy language defines two forms of policy expressions: *compact* and *normal form*. The compact form is less verbose than the normal form. The compact form is useful for authoring policy expressions. The normal form is an intuitive representation of the policy data model[7].

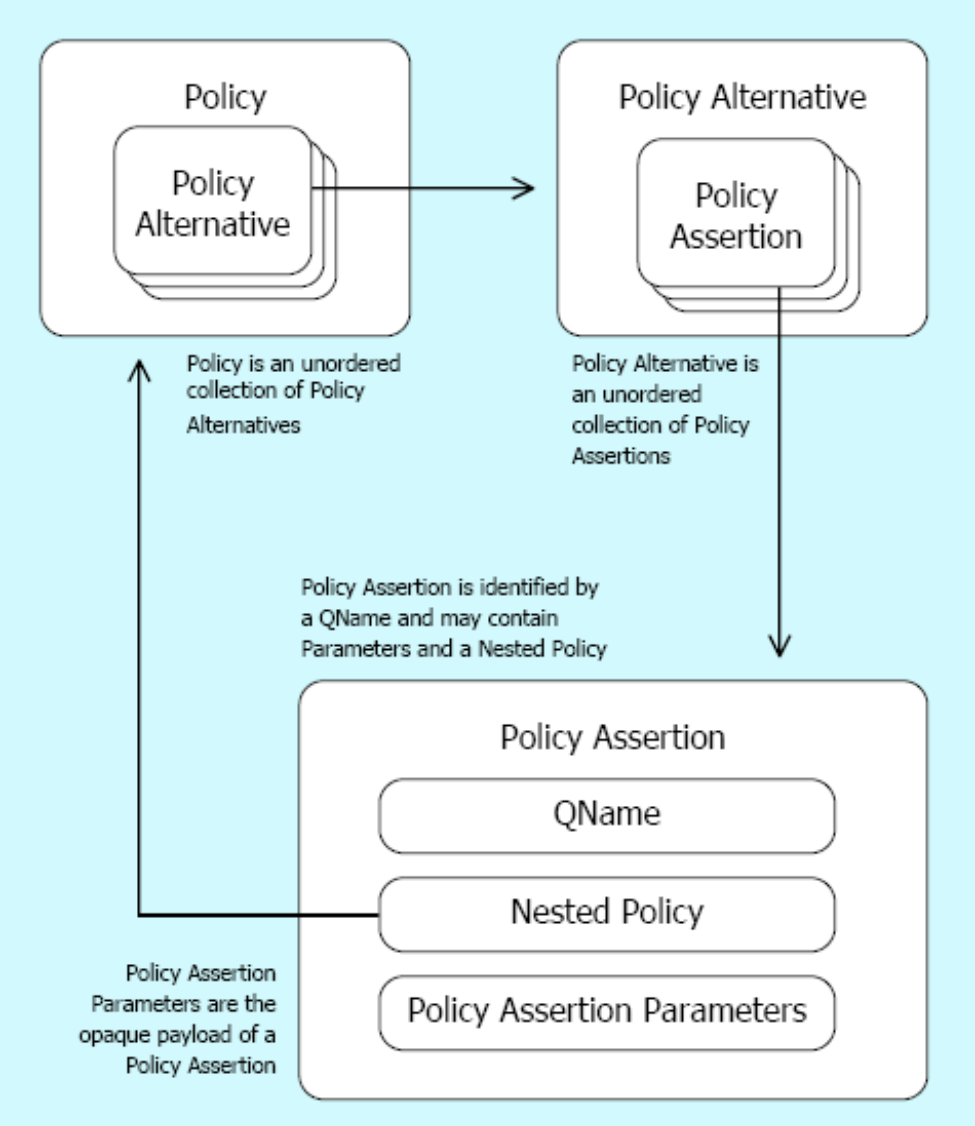

Figure 2.1: WS-Policy Data Model

The following list shows a service provider policy and a requester policy, where these policies have been normalised and take the following form:

[7] http://2006.xmlconference.org/proceedings/122/presentation.pdf

Listing 2.1: Service Provider Policy

```
<wsp:Policy wsu:Id="ProviderPolicy">
   <wsp:ExactlyOne>
        <wsp:All>
        <nsSecurityAssertion level="high"/>
        <nsReliableMessagingAssertion/>
        </wsp:All>
        <wsp:All>
        <nsSecurityAssertion level="medium"/>
        <nsTransactionAssertion/>
         </wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Listing 2.2: Service Requester Policy

```
<wsp:Policy wsu:Id="Requester_Policy">
   <wsp:ExactlyOne>
        <wsp:All>
        <nsSecurityAssertion/>
        <nsReliableMessagingAssertion timeout="100"/>
        <nsReliableMessagingAssertion retries="3"/>
        </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

The Extensible Access Control Markup Language (XACML) specification

XACML [84] is an OASIS standard that describes both a policy language and an access control decision request/response language. The policy language is used to describe general access control requirements and has standard extension points for defining new functions, data types, combining logic, etc. The response always includes an answer as to whether the request should be allowed using one of four values: *Permit*, *Deny*, *Indeterminate* or *Not Applicable*.

In a typical situation, the user wishes to carry out an action in relation to a particular resource. The user typically submits a request to the entity that protects the resource (like a filesystem or a web server), otherwise known as a Policy Enforcement Point

(PEP). The PEP then forms a request based on the requester's attributes, the resource in question, the pending action and other information pertaining to the request. The PEP then sends the request to a Policy Decision Point (PDP), which examines the request in relation to applicable policies and supplies an answer as to whether or not access should be granted. The answer is returned to the PEP, which then allows or denies access to the requester. Note that the PEP and PDP may both be contained within a single application or distributed across several servers. In addition to providing request/response and policy languages, XACML also carries out the task of finding a policy that applies to a given request and comparing the request to the policy in order to supply a yes/no answer.

At the root of all XACML policies is a *Policy* or a *PolicySet*. A PolicySet is a container that can hold other Policies or PolicySets, as well as references to policies found in remote locations. A Policy represents a single access control policy, expressed through a set of Rules. Each XACML policy document contains exactly one Policy or PolicySet root XML tag. Because a Policy or PolicySet may contain multiple policies or Rules, some of which may evaluate to different access control decisions, XACML needs some way of coordinating the decisions each one makes. This is done through a collection of Combining Algorithms. Each algorithm represents a different way of combining multiple decisions into a single decision. There are Policy Combining Algorithms (used by PolicySet) and Rule Combining Algorithms (used by Policy).

A *Target* is basically a set of simplified conditions for the Subject, Resource and Action that must be met for a PolicySet, Policy or Rule to apply to a given request. A policy can have any number of *Rules* which contain the core logic of an XACML policy. At the heart of most Rules is the *Condition*, which is a boolean function. If the Condition evaluates to true, the Rule's Effect (a value of Permit or Deny that is associated with successful evaluation of the Rule) will be returned. Evaluation of a Condition can also result in an error (Indeterminate) or the decision that the Condition doesn't apply to the request (NotApplicable). A Condition can be quite complex, built from an arbitrary nesting of non-boolean functions and attributes.

Attributes are named values of known types that may include an issuer identifier or an issue date and time. Specifically, attributes are characteristics of the Subject, Resource, Action, or Environment in which the access request is made. A Policy resolves attribute values from a request or from some other source through two mechanisms: the *AttributeDesignator* and the *AttributeSelector*. An AttributeDesignator lets the policy specify an attribute with a given name and type, and optionally an issuer as well, and then the PDP will look for that value in the request, or elsewhere if no matching values can be found in

the request. There are four kinds of designators, one for each of the types of attributes in a request: Subject, Resource, Action, and Environment. Because Subject attributes can be broken into different categories, SubjectAttributeDesignators can also specify a category to look in. AttributeSelectors allow a policy to look for attribute values through an XPath query.

Listing 2.3: XACML policy example

```
<Policy PolicyId="SamplePolicy"
        RuleCombiningAlgId="#rule-combining-algorithm:permit-overrides">
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <ResourceMatch MatchId="#function:string-equal">
        <AttributeValue
           DataType="#string">PrinterService</AttributeValue>
        <ResourceAttributeDesignator
        DataType="#string"
        AttributeId="#:resource:resource-id"/>
      </ResourceMatch>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>

  <Rule RuleId="LoginRule"  Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <ActionMatch
        MatchId="#function:string-equal">
```

```
          <AttributeValue
          DataType="#string">login</AttributeValue>
          <ActionAttributeDesignator
          DataType="#string"
          AttributeId="ServiceAction" />
        </ActionMatch>
      </Actions>
    </Target>
    <Condition FunctionId="#function:and">
      <Apply FunctionId="#function:time-greater-than-or-equal"
        <Apply FunctionId="#function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="#time"
               AttributeId="#environment:current-time" />
        </Apply>
        <AttributeValue DataType="#time">10:00:00</AttributeValue>
      </Apply>
      <Apply FunctionId="#function:time-less-than-or-equal"
        <Apply FunctionId="#function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="#time"
              AttributeId="#environment:current-time" />
        </Apply>
        <AttributeValue DataType="#time">18:00:00</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="FinalRule"  Effect="Deny" />
</Policy>
```

A simple example Policy is listed in List 2.3. Its Target says that the Policy only applies to requests for the service called ”PrinterService”. The Policy has a Rule with a Target that requires an action of ”login” and a Condition that applies only if the Subject is trying to log in between 10am and 6pm.

2.1.3 Architectural Approaches

There are several broad architectural options for implementing security in the context of web services [89]: *XML Gateway*, *Interceptor*, and *custom code*.

XML Gateway

XML Gateway (sometimes called XML firewall or XML proxy) is a software package or hardware appliance that filters XML traffic and blocks unauthorised traffic before it can reach a protected service (see Figure 2.2) [74].

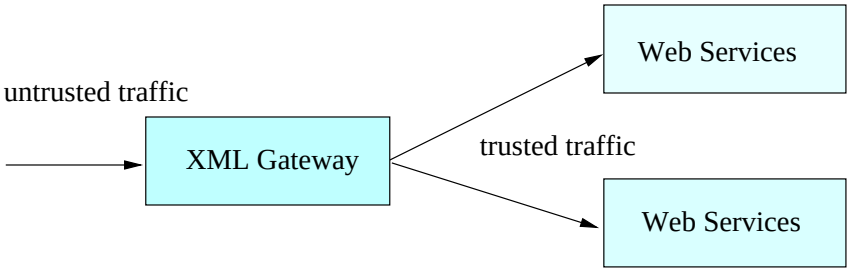

Figure 2.2: XML Gateway

An XML Gateway enforces access control rules by processing security tokens contained within incoming messages, and by ensuring that the XML format and content are appropriate for the target. It may use the SAML to establish the authentication status of an end user or to request attribute information, which is used to make an access-control decision. XML Gateways strive to re-use the existing security infrastructure, including the preconfigured users, groups, and roles. To do this they typically contain security adapters to existing security technologies such as LDAP directories, traditional firewalls, and PKI. Otherwise, the overhead of re-keying rules and user profiles into an XML Gateway would be prohibitive. XML Gateways often provide additional functionality such as data transformation, content-based routing, load balancing and service traffic monitoring, etc.

The drawback of this architecture is that it leaves the service endpoint addresses unprotected. If the incoming traffic bypasses the gateway and reaches the service endpoint, it also bypasses the security provisions implemented by the gateway. Consequently, using this architecture usually requires additional security provisions, for example restricting the physical nodes from which the service endpoint can receive service traffic, mutual authentication between the gateway and the service endpoint, etc [74].

Interceptor

The Interceptor represents an alternative architecture. Lightweight Interceptors co-located with service endpoints are used. They employ platform-specific hooks such as ISAPI Filters, JAX-RPC handlers, MQ exits, etc. to intercept service traffic. When a service request arrives at the service endpoint it is first processed by the Interceptor, which evaluates the security rules before passing the request to the service (see Figure

2.3) [74].

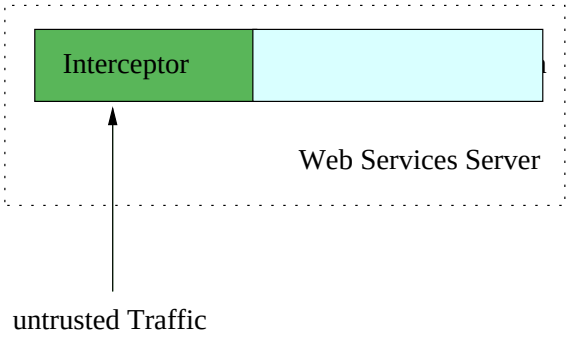

Figure 2.3: Interceptor

Unlike the Gateway (which requires the traffic to flow through an infrastructure component) the Interceptor enforces security at the service endpoint itself. Policy-aware environments such as WSE [45] supply an interesting variation on the Interceptor idea. In these environments the interceptors are built into the environment itself and can be configured using either WS-Policy files or custom configurations.

Although this architecture solves the ”last mile” problem it requires security configurations for every Interceptor (i.e., service endpoint). This requirement can cause significant management overhead when the security policy changes. A modified Interceptor architecture can alleviate this problem by combining an interceptor with a centralised security service (see Figure 2.4) [74].

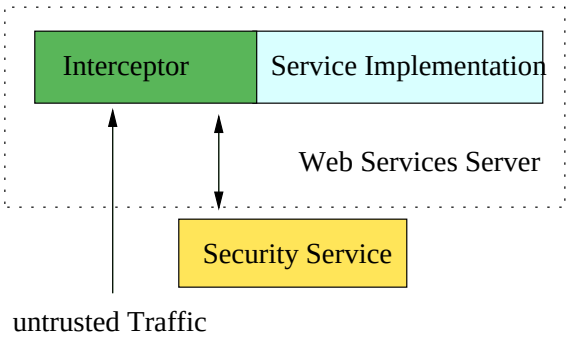

Figure 2.4: Interceptor with centralised Security Service

In this configuration the Interceptor delegates security processing to a specialised security service, which performs the actual processing of security rules. This centralises security processing similarl to the XML Gateway scenario, allowing the movement of the computationally intensive functionality such as cryptography to dedicated hosts.

This minimises the impact on invocation performance and provides a central point of management and reporting for service security processing.

Custom Code

The third option entails writing security code into the service itself. For example, the .NET Common Language Runtime (CLR) includes implementations of XML Signature, XML Encryption, and WS-Security standards. Similarly a set of libraries providing security functionality is available on the J2EE platform. These could be adapted to implement security. The arguments for and against custom-coding revolve around the typical buy versus build decision. While the Gateway and Interceptor models provide centralised management, auditing, and the ability to make global changes for multiple services, this functionality must be custom-built for home-grown platform-specific solutions.

2.2 Corporate Knowledge Management

An enterprise can be considered in a distributed computational paradigm. Knowledge Management is generally considered to have begun in the 1950's when Alfred Sloan divisionalised General Motors. With knowledge management, the unmeasurable must be measured.

In practice, knowledge management often encompasses identifying and mapping intellectual assets within an organisation, generating new knowledge for competitive advantage within an organisation, making vast amounts of corporate information accessible, sharing of best practices, and technology that enables all of the above - including groupware and intranets.

The concept of corporate knowledge has two different aspects. The first relates to business and refers to the amount and quality of know-how and information that is available within a company. The second aspect relates to the framework of information technology and is tightly linked to the diverse variety of paradigms that are used to represent knowledge. With the common use of e-business nowadays these two aspects are becoming somewhat interrelated. From the IT point of view one can view an enterprise as a distributed computational paradigm. Multiagent systems have been invented to tackle such problems within artificial intelligence but are nowadays also seen as a management methodology.

Corporate knowledge generally covers several domains such as document/knowledge management, discussion forums, skill management and knowledge engineering. The

main paradigms used to represent knowledge are logic, frames or semantic nets. Knowledge Representation (KR) has long been considered one of the principal elements of Artificial Intelligence and a critical part of all problem-solving [87]. One of the most important developments in the application of KR has been the development of so-called frame-based KR languages or systems (proposed by Minsky in 1981 [81]). Frame-based systems are knowledge representation systems that use frames as their primary means of representing domain knowledge. A frame is a structure for representing a concept or situation. Attached to a frame are several different types of information, for instance, definitional and descriptive information and how to use the frame. While frame-based KR languages vary from each other to some degree, they share some common characteristics: (1) frames are organised into hierarchies; (2) frames are composed of slots (attributes) for which fillers (scalar values, references to other frames or procedures) have to be specified or computed; and (3) properties (fillers, restriction on fillers, etc.) are inherited from superframes to subframes in the hierarchy according to the inheritance strategy. It is important to note that, besides the inheritance relation, frames do not provide the explicit concept of relations between frames but they use the concept of slot for representing these relations.

Based on the original proposal, several knowledge representation systems have been built and the theory of frames has evolved. Important descendants of frame-based representation formalisms are description logics that capture the declarative part of frames using logic-based semantics. Moreover, the object-oriented paradigm has adopted the organisational principles introduced by frame-based systems.

An important branch in AI that is concerned with KR mechanisms deals with the so-called *ontologies* [38]. An ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts. It is used to reason about the objects within the domain. Ontologies are typically used to describe domain vocabularies and are actively used in artificial intelligence, the semantic web, software engineering and information architecture as a form of knowledge representation about the world or a part of it. The typical elements described by an ontology are:

- Individuals: the basic or ”ground level” objects
- Classes: sets, collections, or types of objects
- Attributes: properties, features, characteristics, or parameters that objects can have and share
- Relations: ways that objects can be related to one another

Attributes are typically used to represent primitive values whereas relations are used to relate non-primitive classes. Some common relationships between concepts are the classification (instance-of, member-of), aggregation (part-of), generalisation (is-a, subclass-of, a.k.a. subsumption), and partitioning (group, context) of the concepts.

2.2.1 Agent-Oriented Abstraction

The agent-oriented abstraction paradigm is introduced in [22]. It is a high-level abstraction for agent modelling and covers the concepts of agents, annotated knowledge, utility functions and society of agents. Indeed, AOA relies on Weber's classical theory in Sociology [113]. The model extends the abstraction capabilities of the existing Agent-oriented Programming paradigm. Abstraction can be viewed as a methodology to express a system in terms of one selected theoretical framework. The word system covers software or hardware components of a computer system.

AOA associates the usual features to agents (ability to perceive, reason, act, communicate) and it is compliant with a societal approach of agents. AOA assumes that agents are entities consisting of both a knowledge component and a decision mechanism system. The knowledge component covers any piece of information available in an enterprise from the technology required to design and produce goods to management decision policy through human relations and internal or external communication. It is partitioned into four components, also called annotations: *ontology*, *communication*, *cognition* and *security*.

- Ontology. This annotation covers ontologies and knowledgebases. It is divided into classes corresponding with structured, unstructured, complete or incomplete knowledge/ontology. Specialisation of such classes is obtained via a query, information retrieval or data mining system. Additionally, languages such as Knowledge Query and Manipulation Language (KQML) [33] or Knowledge Interchange Format (KIF) [37] may provide a specialisation of these classes.

- Communication. Communication is divided into classes that describe the main facets of the communication mechanisms between agents and the external world. Coordination is achieved through subclasses for cooperation, planning (both distributed and centralised), competition or negotiation. The specialisation of these classes is performed when implementing protocols, for instance interaction protocols or task distribution protocols. These protocols can be synchronous or asynchronous. The number of possible protocols is enormous. It is even possible to

design a communication protocol by means of an agent system connecting the communicative entities within an agent system. Message types are also among the specialisations found in communication. A specialisation for negotiation can be based upon the market mechanism, a contract net approach or a blackboard system.

- Cognition. There is obviously a cognitive component among the possible annotations of knowledge. Three main classes exist: semantics, pragmatics and models. Semantics implies a formalised representation of the cognition found in an agent. This translates into specialisations that could be logic-based or a well defined view maintenance mechanism. The class "'pragmatics"' relates to common-sense reasoning and reasoning that is not based on a sound theoretical model. The class "'models"' covers any cognitive meaning that although well captured in models is by no way theoretically certain. This leads to specialisation in terms of speech act for instance where although the separation into locution, illocution and per locution is well established, there is no proof it is a theoretically sound simulation of human communication. It is at best a meaningful approximation. This annotation covers the implementation features appearing under the following headlines: descriptive, prescriptive, reactive, pro-active, personal, conventional, objective or subjective, speaker's versus hearer's versus society's perspective. The BDI model (Belief, Desire and Intention) is set within this annotation.

- Security. This annotation could also be labelled integrity. It recognises that any system exists in a real world where laws and rules govern the actions of agents. An important class is obviously security. Whether for mobile or static agents, security issues are becoming overwhelmingly important. The specialisation of this class is achieved, for instance, through the use of classical encryption methods or firewall technologies. A second class concerns laws and regulations. Whether within a country or a company, any representative agent is required to obey laws and regulations. Specialisations relate to the existing regulations, directives and laws.

2.3 Business Rules

According to the Business Rules Group[8], "a business rule is a statement that defines and constrains some aspect of a business. It is intended to assert business

[8] http://www.businessrulesgroup.org/

structure or to control or influence the behavior of the business. The business rules that concern the project are atomic, that is, they cannot be broken down further". Modelling business rules as separate entities offers great flexibility. Particularly in the e-commerce domain, this can be a valuable advantage considering that the business analyst, who ideally authors the business rules, does not need to have programming knowledge to change the rules. Typically, changing the business rules happens more often than changing the large e-commerce applications. Moreover, extracting the business rules from the business logic leads to a better decoupling of the system, which, as a consequence, increases maintainability. One of the most important facts about business rules is that they are declarative statements, i.e. they specify what has to be done as opposed to how it has to be done.

Three basic types of business rules have been identified: integrity rules (also called "integrity constraints"), derivation rules (also called "deduction rules" or "Horn clauses"), and reaction rules (also called "stimulus response rules", "action rules" or "event-condition-action (ECA) rules"). A fourth type, deontic assignments, has only been marginally discussed (in a proposal for considering "authorisations" as business rules)[9].

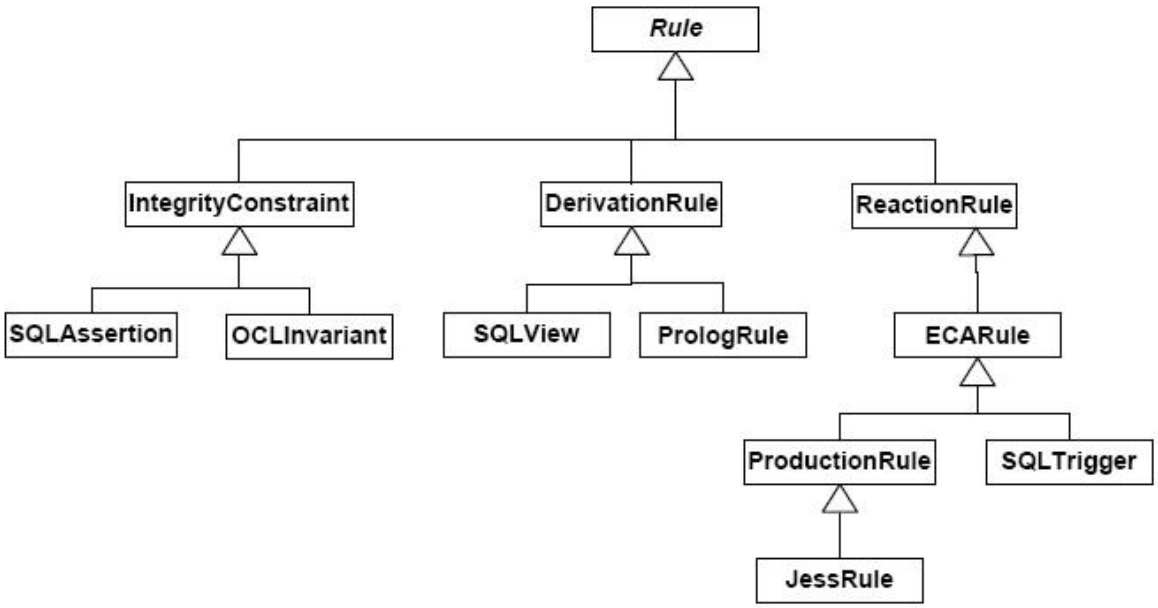

Figure 2.5: The Rule Type Hierarchy

[9]http://oxygen.informatik.tu-cottbus.de/rewerse-i1/GRML.pdf

- An integrity rule is an assertion that must be satisfied in all evolving states and state transition histories of an enterprise viewed as a discrete dynamic system. There are state constraints and process (or behavior) constraints. State constraints must hold at any point in time.
- A derivation rule is a statement of knowledge that is derived from other knowledge by an inference or a mathematical calculation. Derivation rules allow capturing terminological and heuristic domain knowledge about concepts whose extension does not have to be stored explicitly because it can be derived from existing or other derived information on demand.
- Reaction rules are concerned with the invocation of actions in response to events. They state the conditions under which actions must be taken; this includes triggering event conditions, pre-conditions, and post-conditions (effects). They define the behavior of a system (or agent) in response to perceived environment events and to communication events (created by communication acts of other systems/agents). In general, reaction rules consist of an event condition, a state condition (or precondition), an action term, and a state effect (or post-condition) formula. Thus, they can also be called Event-Condition-Action-Effect (ECAE) rules, subsuming Event-Condition-Action (ECA) and Condition-Action (or production) rules as special cases.
- Deontic assignments of powers, rights and duties to (types of) internal agents define the deontic structure of an organisation, guiding and constraining the actions of internal agents.
- In some situations business rules can be described very simply with a single sentence. In other situations this isn't the case. Table 2.1 presents one way of fully documenting the rule "*Tenured professors may administer student grades*".

2.3.1 Business Rule Engine

Rule engines implement rules evaluation, which focusses on managing a collection of facts (or knowledgebase) and evaluating rule sets comprised of one of several predicates. A rule engine manages a large numbers of facts and efficiently evaluates predicates that work on these facts.

A typical rule engine consists of several components:

- A rule base contains all the rules for execution.

Name	Tenured professors may administer student grades
Identifier	BR0212
Description	Only tenured professors are granted the ability to initially input, modify, and delete the grades students receive in seminars that they and only they instruct. They may do so only during the period in which a seminar is active.
Source	University Policies and Procedures Doc ID: U1701 Publication date: August 14, 2000
Related Rules	BR12 Qualifying For Tenure BR65 Active Period for Seminars BR200 Modifying Final Student Grades

Table 2.1: Business Rule Example

- The working memory holds the data with which the rule engine operates.
- The Pattern Matcher decides which rules from the rule base apply, given the content of the working memory.
- An inference engine works in discrete cycles and is used to find out which rules should be activated in the current cycle (including the activated ones from previous cycles) by using the pattern matcher.

All activated rules from the conflict set are ordered to form the agenda (the list from which the right hand side will be executed). The process of ordering the agenda is called the conflict resolution. To complete a cycle, the first rule on the agenda is fired and the entire process is repeated. Typical rule engines either implement a forward or backward-chaining strategy. Backward chaining works backwards from a conclusion to be proven to determine if there are facts in the working memory to prove the truth of the conclusion. Forward chaining takes all the facts stored in the working memory and attempts to apply as many rules as possible. The inference engine works from the initial content of the workspace towards the final conclusion.

The Java Rule Engine API specified by the Java Specification Request (JSR)

94 is a Java community effort defining an API for rule engines in the Java world [56]. The JSR 94 specification standardises a set of fundamental rule engine operations. These operations include parsing rule sets, adding objects to the inference process, firing rules and getting objects from the engine as a result. The main goal of the JSR 94 specification is to integrate different rule engines with a simple adapter into client applications without settling on one rule engine vendor. The specification provides two main parts, a rule administration API and a rule runtime API. The administration interface is primarily used for loading and managing rule sets (a collection of rules for solving a specific task). The runtime API is for executing specific rule sets by using a rule session, which represents the connection between the client and a specific rule engine. The main critique of rule engine vendors is the lack of a standard rule representation format.

In the area of business rule engines, a number of different tools are available. Drools[10] is one of the most popular open source Java business rule engines. The Drools rule engine uses an enhanced version of the Rete algorithm [35] called the Rete-OO algorithm. Drools is a purely forward chaining system. Jess[11] (Java Expert System Shell) is a robust and mature expert system shell for developing rule-based applications in Java. Jess has many unique features including backward chaining and working memory queries and can directly manipulate and reason about Java objects. Both rule engines do not allow a distributed rule execution and can only be used as a library that is linked to the application.

2.3.2 Semantic Web Rule Language

The Rule Markup Initiative aims to provide a standard rule language and to provide an interoperability platform integrating various business rule languages, inference systems and knowledge representation paradigms. It has gained increasing momentum within the standard, academic, and industrial communities.

RuleML [96] is an XML-based Rule Markup Language that has been proposed by the Rule Markup Initiative as a canonical language for publishing and sharing rules on the Web. RuleML is implemented with XML Schema, XSL Transformations (XSLT) and reasoning engines. The RuleML Initiative collaborates with several standards bodies including W3C, OMG and OASIS. Because it is XML-based, RuleML inherits some of its benefits directly from XML, including platform inde-

[10] http://www.drools.org
[11] http://www.jessrules.com/jess/index.shtml

pendence and interoperability. RuleML is also extensible, a prime example being its combination with OWL to form the Semantic Web Rule Language (SWRL) [46] and its Object-Oriented extension called OO RuleML [16].

RuleML is also translatable into and from other Semantic Web standards via XSLT. Various tools are also available, including OO jDREW [7], Mandarax[12] , and NxBRE4[13].

RuleML covers the entire rule spectrum, from derivation rules and transformation rules to reaction rules and integrity checking. RuleML can thus specify queries and inferences in Web ontologies, mappings between Web ontologies, and dynamic Web behaviors of workflows, services, and agents. The rule language is based on XML and is low-level. As an example, consider the following rule:

"The discount for a customer buying a product is 5.0 percent if the customer is premium and the product is regular"

This rule is expressed in RuleML as follows:

Listing 2.4: RuleML Example

```
<imp>                                                    1
  <_head>
    <atom>                                               3
      <_opr><rel>discount</rel></_opr>
      <tup><var>customer</var>                           5
        <var>product</var>
        <ind>5.0percent</ind></tup>                      7
    </atom>
  </_head>                                               9
  <_body>
    <and>                                                11
      <atom>
        <_opr><rel>premium</rel></_opr>                  13
        <tup><var>customer</var></tup>
     </atom>                                             15
     <atom>
       <_opr><rel>regular</rel></_opr>                   17
```

[12]http://mandarax.sourceforge.net
[13]http://www.agilepartner.net/oss/nxbre

```
      <tup><var>product</var></tup>
    </atom>
   </and>
  </_body>
</imp>
```

where the tags <_head> and <_body> represent the action and the condition of the rule respectively. Because RuleML is XML-based, it is hard for domain experts to adopt. This appears to have been the critical factor determining the success of this language, as recognised by [44]. In this thesis, the TRANSLATOR [44] initiative will be presented which is aimed at raising the level of RuleML abstraction. TRANSLATOR is an open source tool with the ability to automatically translate natural language-like sentences written in Attempto Controlled English [36] into RuleML rules. Likewise, we can also imagine adding a layer on top of our high-level rules to allow for their expression in a natural language-like format.

An SWRL rule formally expresses an if-then rule using the OWL syntax. A rule expresses the implication relationship between the classes of body (the ”if” clause) and head (the ”then” clause), both of which consist of kinds of atoms. More complete syntactical information can be found in the SWRL specification. For the sake of readability, a rule is often written in the following form:

$antecedent \Rightarrow consequent,$

where both *antecedent* and *consequent* are conjunctions of the atoms that are written as $a_1 \wedge ... \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., ?x).A sample rule is defined as follows.

$hasParent(?personA, ?personB) \wedge hasBrother(?personB, ?personC) \Rightarrow hasUncle(?personA, ?personC).$

Using SWRL syntax, the sample rule could be written as follows.

Listing 2.5: SWRL Example

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom
              swrlx:property="hasParent">
     <ruleml:var>personA</ruleml:var>
     <ruleml:var>personB</ruleml:var>
   </swrlx:individualPropertyAtom>
```

```
  <swrlx:individualPropertyAtom
             swrlx:property="hasBrother">
    <ruleml:var>personB</ruleml:var>
    <ruleml:var>personC</ruleml:var>
  </swrlx:individualPropertyAtom>
</ruleml:_body>
<ruleml:_head>
  <swrlx:individualPropertyAtom
             swrlx:property="hasUncle">
    <ruleml:var>personA</ruleml:var>
    <ruleml:var>personC</ruleml:var>
  </swrlx:individualPropertyAtom>
</ruleml:_head>
  </ruleml:imp>
```

Chapter 3

Design Issues and Overview

3.1 Introduction

In recent years, a significant number of companies have implemented e-business solutions. This is due to the fact that an investment in e-business technologies promises a competitive advantage through the integration of processes and lower transaction costs. Companies are enclosing traditional computing tasks, such as database access or commercial transaction systems, in wrappers as software services to connect them to the Internet. New tasks, such as computerised auctions and e-marketplaces, have also been introduced as business services. The concept of Web Services is thought to be the next generation of e-business architectures for the web. Such a Service-Oriented Architecture (SOA) describes principles for the creation of dynamic, loosely-coupled systems based on services without a specific implementation.

Both Web services and Semantic Web services consider services as fundamental and central entities. Service descriptions play a central role in publishing services for use, categorising them and finding the right service to satisfy a need. In [108], three different aspects are considered in regards to service descriptions: (1) *functional*, (2) *behavioral* and (3) *non-functional*.

The functional description contains the formal specification of what exactly the service can do. The behavioral description contains the formal specification in regards to how the functionality of the service can be achieved. The non-functional description captures constraints over the previous two. The most important issues surrounding the non-functional descriptions are those of security, authentication and privacy. These issues relate to the exchange of information necessary for the

service to be consummated. Legal and contractual issues between the provider and requester of the service also play an importan role.

A policy consists of information that can be used to modify the behavior of a system [72]. Policies describe the capabilities, requirements and constraints of a service. Non-functional service descriptions are often used to formalise or specify assertions in policies.

Two problems arise from the application of web services in e-business:

1. *Change and Risk Management.* Nowadays e-business scenarios always involve more than one organisation. An increase in competition forces businesses to make frequent changes to corporate policies and devise strategies in order to remain competitive. This means that changes have to be made fast in accordance to market changes. Changes to business rules and processes should take effect in an efficient way without the need for recoding or stopping systems. Rules can be applied when making decisions without the involvement of humans in business-to-consumer activities. It is unrealistic to expect that knowledgeable staff will always react in real-time to make decisions. These factors underline the need for business rules as external entities, thus managed by a Knowledge Management System.
2. *Gaps between Business and IT.* Managers and administrators, who work at the business level, tend to use abstract, high-level and human-readable policies to specify requirements or describe services. At the IT level, things work the other way around; policies are usually specified in a machine-readable way and enforced according to the specific environment. Knowledge exchange and sharing is necessary in this case to build an intelligent service.

To solve these problems, we proposed a security policy framework for business processes [50] [48] [51] [49], which described a preliminary concept of policy management and implementation in SOA. In a new line of work, this framework is further extended and a security gateway with semantic policy framework for service-oriented computing (SOC) is introduced in the following sections.

3.2 GRACIA - The Security Gateway Approach

A security gateway is a system made up of software components that is used to provide a security proxy for web services. For this purpose, the technically feasible

service interactions are confined to those specified by the services security policy. In the context of service-oriented computing, security means only allowing access or data flows between different web services that are positively wanted.

Security gateways are used as the central interface for web services with different levels of trust and security requirements. Web services can have different levels of trust even where the transition is not between internet and intranet. Instead, it is also possible for two internal services within an organisation to have different protection requirements.

In contrast to the more customary term ”firewall”, the use of the term ”security gateway” illustrates the protection of web services and service networks as opposed to the uptake of different tasks, such as packet filtering, anti-virus protection or monitoring of the network traffic (”intrusion detection”). Security gateway’s tasks often include service access control and security policy enforcement.

3.2.1 Threat Scenario

The following typical threats are assumed for a security gateway:

Organisational Shortcomings and Human Error
It is a relatively common occurrence that, due to negligence and insufficient checks, people fail to implement, either wholly or partially, legal and regulatory requirements that have been recommended or prescribed to them. This can cause damage which otherwise could have been prevented, or at least minimised. Depending on the function of the person in question and the importance of the compliance overlooked, the resulting damage could be quite serious. Compliance is frequently disregarded due to a lack of security awareness and domain knowledge.

Technical Failures
When programs and protocols are planned, mistakes that affect security can be made in the design. For instance, the developers of the protocols used in the internet surely did not expect that these protocols would one day become the basis for a world-wide computer network that is commercially extremely important.

Deliberate Acts
One of the deliberate acts is unauthorised access of web services. Without mechanisms for the identification and authentication of users, any control over unauthorised use of web services is virtually impossible. If the identification and authentication function can be abused, it is even possible to initiate automatic attempts by developing a program that systematically tests all conceivable passwords.

3.2.2 Countermeasures and Approaches

In order to successfully set up a security gateway for web services, a series of measures should be taken:

- **Determine the security objectives**
  To offer effective protection, the security gateway must be based on a comprehensive security policy and be integrated into the organisation's IT security concept. There are a number of different ways of implementing security gateways. To determine which concept is best suited to a particular instance, it is first necessary to clarify which security objectives are to be fulfilled by the security gateway. In our approach, we define the following objectives:
  * Protection of the trusted web services against unauthorised access from the non-trusted stakeholder;
  * Protection of the transmitted data against attacks on their confidentiality or integrity;
- **Establishing security policies**
  Based on the security objectives, security policies stipulating the tasks to be carried out by the security gateway must be created. These security policies must be embedded into the IT security strategy of the organisation concerned in accordance with regulatory and legal organisations. The security gateway policy determines the behaviour of the security gateway. It defines which information, services and protocols the security gateway handles, how it handles them and who may use them. The security requirements stem from the organisation-wide security policy and should be formulated into a specific security policy for the operation of security gateways in order to more precisely define and implement the high-level and generally formulated security policy in a given context.
- **Construction of the security gateway**
  In order to build up the security gateway, the security policy and rules ought to be implemented through the integration of different components. In the traditional firewall, these components could be packet filters. In our approach, a logic inference engine, a knowledge management system and an open source solution for security policy management are intergrated into the gateway.

The Gracia approach is based on the developement of the semantic web technology and knowledge representation methodology. The primary concept of this

approach is:

- The knowlege is mergered on a common ontology basis. The knowledge, which stems from a regulator organisation and the company's business policy, also has diffefrent ontology domains. The Virtual Knowledge Community and Corporate Knowledge are used here to formailise the common ontology basis for all of these knowledges. Because the knowledge is wriiten in natural language, in this approach the knowledge is represented in two forms: descripton logic form and horn logic form.
- This knowledge is interpreted with a semantic-rich language(OWL) and converted into a machine-readable format for the logic reasoning engine and security policy enforcement component. The logic component decides to permit or deny the access request to web services and this reasoing task takes place on the mergered knowledge basis (as mentioned above).

3.2.3 Case Study

A real-world regulator compliance scenario demonstrates the requirements of the security gateway. Let us consider the case of a security officer, Bob, who is working at the airport and checking the boarding passengers. The airport is equipped with several 802.11b hot spots providing travellers, airline employees and security officers with wireless connectivity for their portable devices, e.g., laptops, PDA and mobile phones. Airline companies may also provide additional services and resources to security officers, such as the possibility to control the CCTV Camera to focus on a person of particulary interest. Moreover, while working at the checking gate, officers may wish to share information with other officersby using the wireless connectivity available at the airport (considering the fact that some officers work without a fixed location within the airport.) Therefore, Bob may wish to access the CCTV service that is available in the area around him to keep an eye on a suspect person.
Additionally, the requirement for up-to-date information regarding the location of other officers within the airport means that he should be able to exchange location information with other officers. These activities need to be regulated by appropriate policies and enforced by specific facilities within the airport system. In particular, the following policies governing adequate access to services and resources may apply. We will use these policies as a running policy example throughout the rest of this thesis. Figure 3.1 illustrates this scenario.

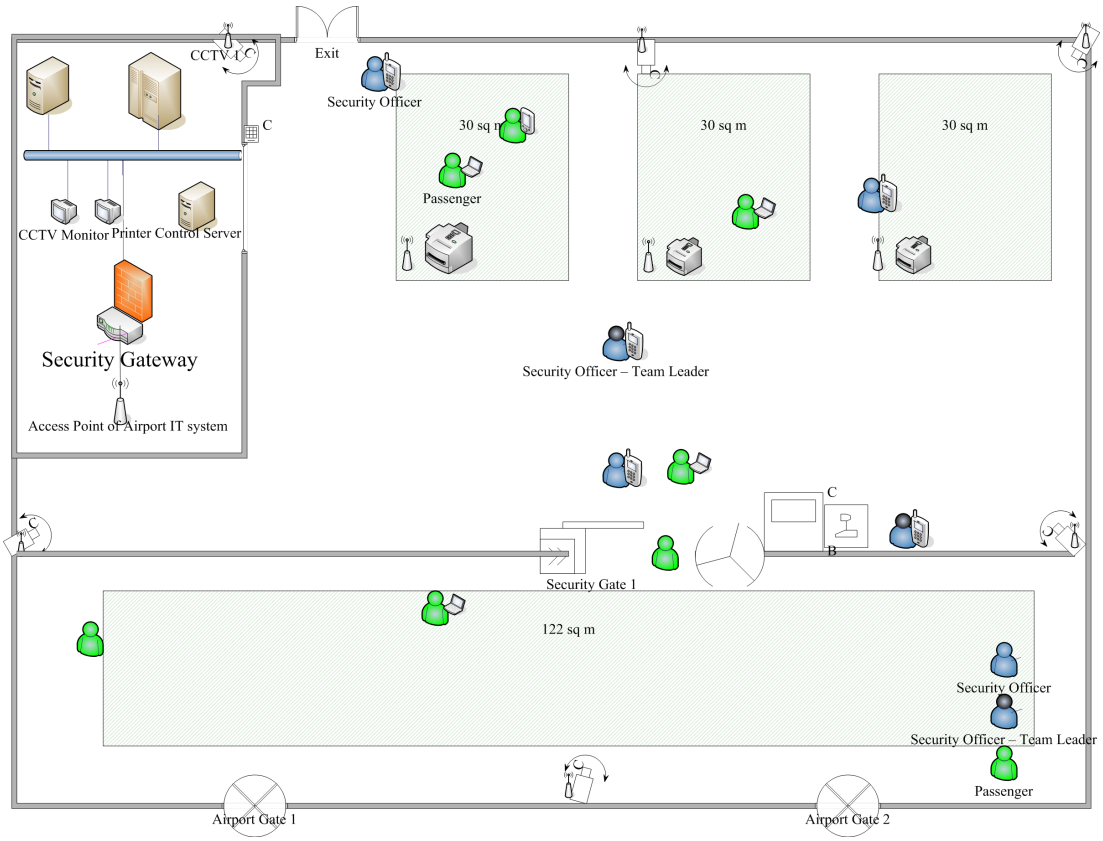

Figure 3.1: Use Case Scenario: Airport Security

– **Information Sharing Policy**
Security Officers that are currently co-located with the owner of the policy, i.e., with his device, are authorised to access the shared information stored on the owner's device.
This policy may be instantiated and enforced by Bob to share his information with co-located officers in a secure way, depending on context conditions at the moment in question.

– **CCTVAccess Policy**
Officers that are assigned to work as team leaders and are currently located in the boarding area are authorised to access the CCTV device.
This policy may be enforced by the provider of a CCTV service that is offered to security officers in some areas of the airport. The enforcement of this authorisation should ensure that officers having proper rights are able to access the service. Furthermore, if the default behavior of the system states that everything that is not explicitly permitted is prohibited, this policy also prevents unauthorised travellers from accessing the service.

3.3 GRACIA Architectural Overview

In this section, a security gateway with a semantic policy framework for service-oriented computing (SOC) is introduced for the specification and enforcement of policies to support security and trust management for SOC. The security gateway that intercepts all service requests is illustrated in Figure 3.2. It consists of the Gracia Policy Framework and the Gracia Knowledge Base.

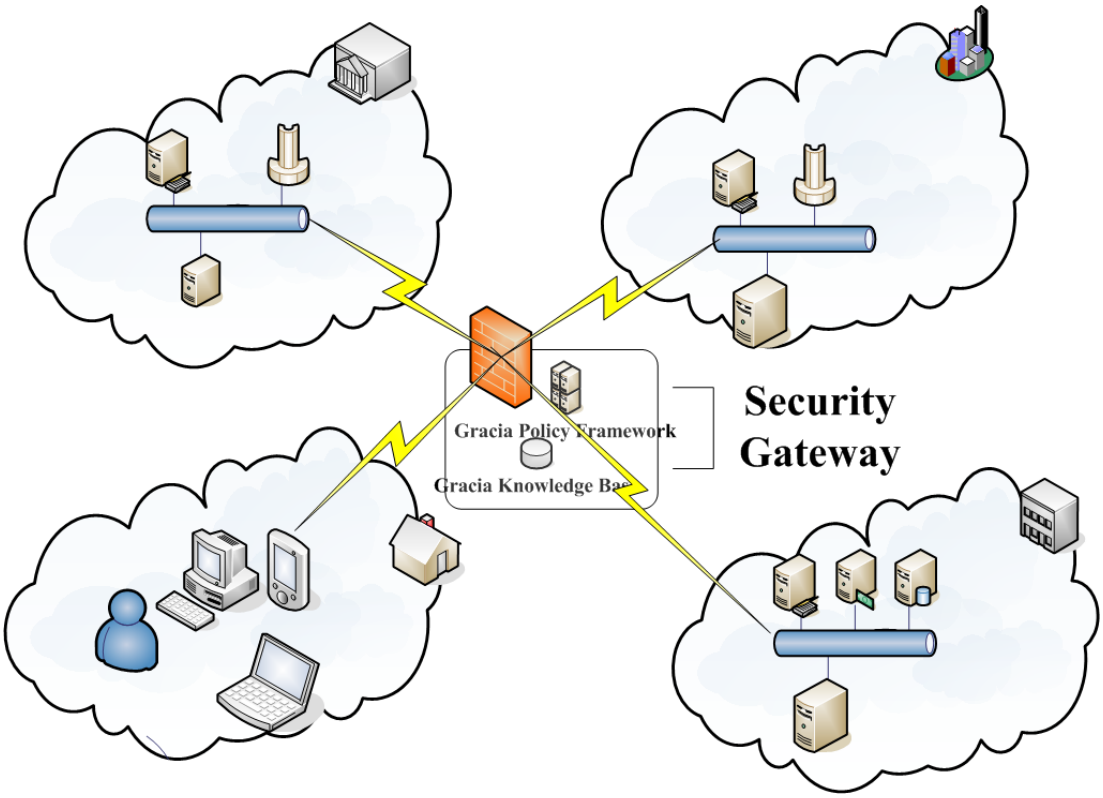

Figure 3.2: Security Gateway

In order to support conflict resolution and the life-cycle management of policies as well as to enable reasoning and decision making processes over the knowledge base, an architecture of the semantic policy framework is illustrated in Figure 3.3.

This includes two supporting services: a policy service and a knowledge service. Additional components include: reasoner, management tools and repository.

– **Policy Service.** The policy service acts as a Policy Decision Point [1](PDP) for web services policies(external) and Gracia Policy Language (internal),

[1]http://www.ietf.org/

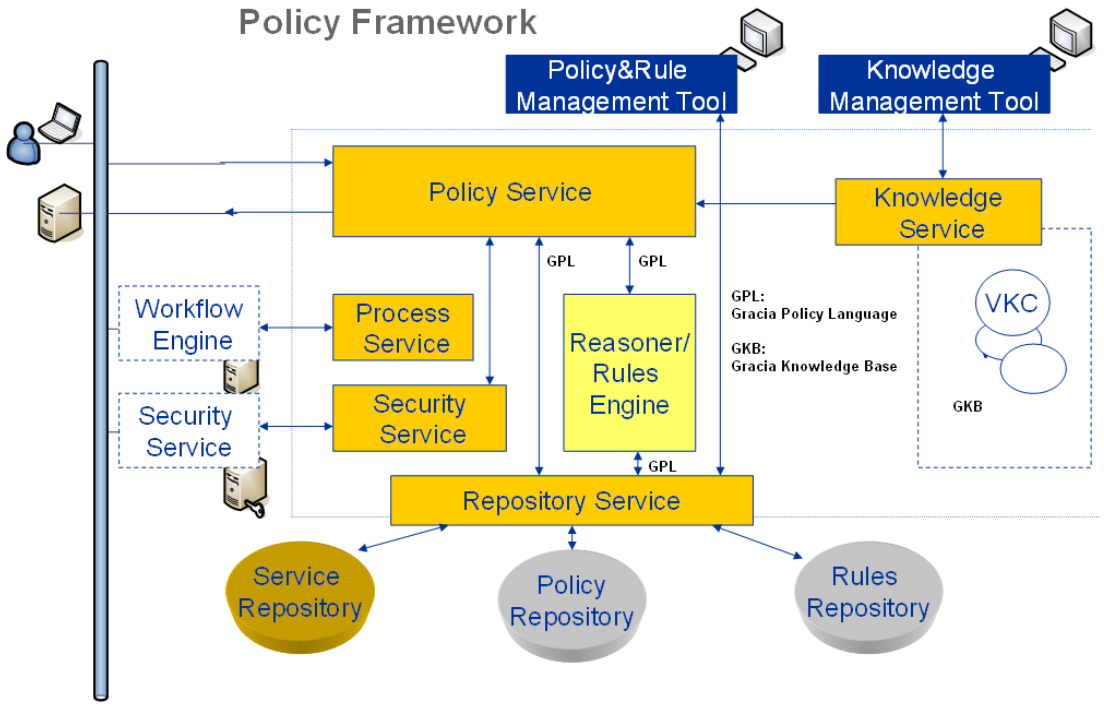

Figure 3.3: Framework architecture

including security and QoS requests. The policy service acts on service requests and renders a decision.

– **Knowledge Service.** The knowledge service manages the Gracia Knowledge Base and the repository of the static knowledge base: business rules, policies and service descriptions.

– **Repositories.** Three repositories are used for the storage of service descriptions, web service policies and business rules. This information can be managed through the policy and knowledge service.

– **Reasoner.** Reasoner is used to perform logical inference over Gracia Knowledge Base based on knowledge repositories and VKCs. In our implementation, we use KAON2 [85] [2] as the reasoner. The major advantage of KAON2 is that it is a very efficient reasoner when it comes to reasoning with Description Logics ontologies containing very large ABoxes and small TBoxes [86]. The terms Abox and Tbox are used to describe two different types of statements in ontologies. Together, Abox and Tbox statements make up a knowledge base.

[2] http://kaon2.semanticweb.org/

– **Management Tools.** The policy management tool acts as the interface to policy service; it manages the life-cycle of policies, creates and deploys new policies. The knowledge management tool provides an interface to manage the VKCs and knowledge repositories.

As the policy service acts as a PDP, it receives the service request from the Policy Enforcement Point (PEP)[3] at the service end. There are two possible options: a) It seeks the suitable policies within the policy repository, renders the decision and sends the result back to PEP, b) It cannot find the proper policy or there are conflicts in policies, the request will be converted to a request to the knowledge service. The knowledge service will analyse the request and prepare a knowledge base for the next reasoning step. The knowledge base is built on both static knowledge from repositories and dynamic knowledge from VKCs. VKCs build a special knowledge component outside the framework and are managed by the Knowledge Service Agent, which also acts as a leader agent on the knowledge service. Based on the result of reasoning on the knowledge base, the policy service can make its decision regarding the service request.

3.3.1 Gracia Policy Language

Web services interact with each other by exchanging SOAP messages. To provide for a robust development and operational environment, services are described using machine-readable metadata. This metadata serves several purposes, one of which being the description of the capabilities and requirements of a service - often called the web service policy.

Various approaches have been taken to achieve security policy specifications, including logic-based languages, role-based access control, various access controls and trust specification techniques [27]. However, a specification language, which can meet the following requirements, is still missing.

– **Support of semantic service descriptions.** How to model security properties into the policies and enable reasoning over them?

– **Integration of Domain Knowledge.** Domain Knowledge such as Business rules state core business policies. They control and influence business behavior. How is it possible to integrate them into the knowledge base and specify policy using rules?

[3] http://www.ietf.org

Various web services and semantic web services approaches such as UDDI[4], OWL-S [26], SWSF [10] and WSMO/WSML [32] have been investigated to describe the non-functional properties of a service. In [90] a set of the most relevant non-functional properties for Web services and their modelling are described. An overview of all these approaches is given in [108].

In this thesis, we define the Gracia policy specification language with an upper ontology in [51]. From the upper ontology, a domain-specific policy ontology describes the vocabularies for policies, business rule terms and service descriptions used within the domain. By using the domain-specific policy template, a policy specification can be generated automatically with Ontology language OWL [79] and Rules language SWRL [47]. In Chapter 4, the details regarding the specification of policies in this policy framework will be introduced.

3.3.2 Gracia Knowledge Base

In Chapter 5, it will be shown how knowledge sharing and exchange helped us model policies for services and how we define the Gracia Knowledge Base (GKB).

As Trust is nowadays considered to be a very important issue in knowledge management, we also defined a trust brokering solution inside the GKB, which we expect will be able to solve the anonymous trust management problem in SOC.

3.4 Summary

We have investigated a distributed knowledge management approach to aid the modelling of security policies for web services. This approach is based on the concept of corporate knowledge through the use of VKCs and the contribution from the Semantic Web Community (OWL and SWRL). By integrating the knowledge management service, the semantic policy framework is able to access external VKCs which can provide application-specific knowledge on transactions in SOC. After the knowledge integration, rich corporate knowledge can be used to fulfill the task of reasoning and enrich the policy with former unavailable information such as security requirements of web services and business rules affected by the transaction processes. In chapter 6, we will introduce the development details of

[4] http://www.uddi.org

the prototype of the semantic policy framework, in particular the development of Policy and knowledge management services.

The aim of this work is to create a security gateway with a semantic policy framework and knowledge management methodology that enables security, trust and QoS in service-oriented computing environments and provides a novel solution for fields such as e-business, telecommunication and enterprise application integration.

Chapter 4

Gracia Policy Language

4.1 Introduction

To address regulatory and security-related issues, policy approaches that specify policies in a way that is both context-based and semantically-rich are necessary. Two approaches have been used in our research: an ontology-based approach that relies heavily on the expressive features of Description Logic (DL) languages, and a rule-based approach that encodes policies as Logic Programming (LP) rules.

Explicit policies can help by dynamically regulating access rights to web services endpoints and maintaining an adequate level of security, predictability, and responsiveness to human control. By changing policies, service transactions can be continuously adjusted to accommodate variations in externally imposed constraints and environmental conditions without modifying the service implementation.

In order to capture the changes in the business logic and security-related requirements, our approach formalises two kinds of information: Access Control Policy and Services Security Constraints. This chaper will introduce the formalisation of this information with our semantic formal approach-Gracia Policy Language.

4.2 Related Work

Logic programming and rule-based formalisms are considered to be appealing policy specification languages. The most common type of policy is security policy,

which is used to pose constraints on a system's behavior. Recently, the notion of policy has been generalised to include other specifications of behavior and decisions, including business rules in all their forms. In the emerging area of service-oriented computing, the word "policy" is sometimes used to refer to the orchestration of elementary and compound services. Policies specify the interplay (dialogs, negotiations, etc.) between different entities and actors for the purpose of delivering services whilst enforcing desired application constraints and client requirements.

Policy complexity is increased by the interplay of multiple, heterogeneous requirements. These requirements must be harmonised and merged into a coherent policy. In complex organisations, different branches or departments may have direct control over their own data and establish their own security policy. At the organisation level, these different policies must be merged. National laws are also an external source for security constraints. The description of laws in different natural languages also increases the need for a semantic foundation for policy languages.

4.2.1 Logic-based Policy Specification

Logic-based policy specification languages aim at providing language constructs that enhance clarity, modularity and other desirable properties. Logic languages are particularly attractive as policy specification languages. One obvious advantage lies in their clean and unambiguous semantics, suitable for implementation validation, as well as formal policy verification. Secondly, logic languages can be expressive enough to formulate all the policies introduced in the literature. The declarative nature of logic languages yields a good compromise between expressiveness and simplicity. Their high level of abstraction, very close to the natural language formulation of the policies, makes them simpler to use than imperative programming languages, especially for people with little or no technical training. Such people are also not experts in formal logics, so generality is sometimes dismissed in favour of simplicity. For this reason, some languages do not adopt a first-order syntax, even if the policy language is then interpreted by embedding it into a first-order logic (FOL) [14] [17]. The embedding often isolates a fragment of the target logic with nice computational properties.

In a real system with hundreds of users and hundreds or thousands of data objects, the set of potential authorisations may have one hundred elements or more.

Moreover, if the policy is time-dependent, then the number of authorisations may increase significantly. Therefore, one can only afford policy languages with low polynomial complexity. In fact, most of the logic-based policy specification languages proposed so far are directly or indirectly mapped onto more or less extended forms of logic programs suitable for efficient, PTIME implementations.

The target logic is typically nonmonotonic, that is, the set of consequences of a theory do not increase monotonically with the set of axioms in the theory. Policy specification was proposed long ago as an application of nonmonotonic logics [78]. The reason behind this is that sometimes decisions need to be made in the absence of information. So, when new information is added to the theory, some decision may have to be retracted (because they have lost their justification), thereby inducing a nonmonotonic behavior. In the area of security, such default decisions arise naturally in real world policies. For example, open policies prescribe that authorisations are granted by default, whilst closed policies prescribe that authorasations should be denied unless otherwise stated.

For the first time ever, a nonmonotonic logic has been proposed as a policy specification language by Woo and Lam [116]. They show how default logic can be used to express a number of different policies. Woo and Lam also provide an axiomatisation of the Bell-LaPadula security model [12], refined by the need-to-know principle. In order to address complexity issues, Woo and Lam propose the use of the fragment of default logic corresponding to stratified, extended logic programs, that is, stratified logic programs with two negations (negation as failure and classical negation), whose unique stable model can be computed in quadratic time. Extended logic programs can be easily transformed into equivalent normal logic programs (with only negation-as-failure) by means of a straightforward predicate renaming. Default logic is a very flexible policy specification language. Different users and objects can be treated with different policies.

In the language of the security community, a fixed vocabulary is called a model, while in the AI community it would probably be regarded as an elementary ontology. In the next section, semantic approaches based on ontology are introduced.

4.2.2 Semantic Approaches

The word semantics is derived from the Greek word *sematikos* -which means significant meaning, derived from the sema sign. Semantics, which deals with meaning/content, is one of the three branches of semiotics. The other two branches

are syntax and pragmatics. Syntax is defined as ”the study of formal relationships between words” [59]. Pragmatics, deals with the ”study of the relation between language and context of use” [59]. Semantics differs from syntax (the formal structure or pattern) in pertaining to what something means. Semantics, the ”study of meaning”, can also be explained as the relationship between word symbols and their intended meaning. Formal logic semantics is important as it helps by defining a language in a methodical way. The formal representation of semantics helps to perform querying and reasoning.

The different types of semantics are:

- Denotational Semantics. Denotational meaning can be expressed as ”Knowing that”, which defines meaning as reference. In logic theory, denotational semantics formalises the meaning of languages/programs by means of mathematical functions.
- Operational Semantics. Operational meaning can be defined as ”Knowing How”. It is the valid algorithm for performing actions. This is enabled by defining a set of formal inference rules to define the valid transitions of the system.
- Axiomatic semantics. Axiomatic semantics is used to define assertions about properties of a system. It is based on mathematical logic to prove the correctness of the system. It describes how the state change of the system affects these assertions. In terms of programming, assertions are constraints such as pre/post-conditions and invariants of an operation, and axiomatic semantics deals with the state of these constraints after the execution of the operation.
- Model-Theoretic Semantics: Model-Theoretic Semantics [8] deals with the study of language meaning using mathematical and logical formalism. Semantics/Meaning of language is given by the interpretation of mathematical concepts of set-theory; this enables the explanation of the semantics via formal structures such as predicates. Formalism helps to bring about automated reasoning. Inferencing and reasoning are based on resolution theory.

The Semantic Web can be defined as ”an extended Web of machine-readable information and automated services that extends far beyond current capabilities” [106]. Adding explicit semantics to underlying content will transform the Web into a global knowledge source that can prove useful to a number of applications. As opposed to the information overload in the current Web, the Semantic Web will

help present the content in a more organised manner. The inherent nature of Semantic Web technology (semantically enriched) helps develop new and flexible approaches to Data Integration. This facilitates many applications, particularly Semantic Web Services, which are built using the infrastructure of the Semantic Web.

The core of the semantic knowledge present in the Semantic Web is presented through Ontologies. Ontology can be aptly described as a ”formal specification of conceptualisation shared in a community” [39]. It can also be looked upon as a vocabulary of terms and relations that is used to represent an unambiguous view of the world. Ontologies help to formalise the communication across the applications and extensions of the Semantic Web. The components of ontology can be briefly summarised as (i) Concepts (Vocabulary) (ii) Structure (hierarchy of concepts and their attributes) (iii) Specific characteristics of concepts and their attributes (e.g., Domain and Range Restrictions, Properties of relations). A clear perspective on the basics of ontology and its features can help enhance the semantic knowledge resident in the Semantic Web.

The Web Ontology Language -OWL [79] is a new formal language for the representation of ontologies in the semantic Web. OWL characteristics are obtained as extensions to RDF (Resource Description Framework) and RDF-S (RDF-Schema). OWL's semantics are grounded in DL expressivity. An ontology can be viewed as a Description Logic knowledge base. Description Logic (DL) is a knowledge representation formalism (KR) that represents a domain. The domain is described in terms of the concepts in the domain, properties used to define these concepts and the relationship between the concepts in the domain. Using DL as a foundation of ontologies helps us use its inherent reasoning to make implicit knowledge explicit. Based on the level of expression and time-complexity of reasoning, OWL comes in three flavours:

- OWL-Lite uses simple constraints and reasoning, and is computationally simple and efficient
- OWL-DL is computationally complete and decidable.
- OWL-Full offers the most maximum expressiveness, but offers no computational guarantees.

The control of access to web services in compliance scenarios raises novel requirements for the design of policy languages and policy run-time environments. In compliance scenarios, regulations typically change from one environment to

another, thus determining continuous variations in their physical position and in their execution context, including the set of entities and resources they may influence. Moreover, services can be accessed via the network in various processes, which exhibit different conditions before or after the service invocation. As it is not possible to exactly predict all the regulations an entity should be compliant to and the kind of resources it may wish to have access to, policy-based control cannot rely on any precise knowledge about the subjects/events/actions that need to be governed [109].

To deal with such characteristics, recent research efforts propose the adoption of semantically-rich representations for the expression of policy and domain knowledge. The adoption of Semantic Web languages for the specification and management of policies in regulator compliance scenarios brings several advantages. In fact, semantically-rich representations ensure that there is a common understanding between the service consumer and service provider, the current execution context and the actions they are permitted or obliged to perform. Moreover, modelling policies at a high level of abstraction simplifies their description and improves the analysability of the system [109]. Semantic Web languages also enable expressive querying and automated reasoning about policy representation.

Another emerging direction suggests that, in order to deal with the dynamic context changes that are typical in the service-oriented world, it may be advantageous to build policies directly over context conditions, i.e., to consider context as a primary element in the specification of policies [82]. The adoption of a semantic policy approach requires the definition of a policy model that can precisely identify the basic types of policies required to control services, specify how to express and represent related policies in a semantically expressive form, and how to enforce them. The following are considered as basic requirements for a semantic-based policy language [109]:

- the ability to model and represent the contexts in which services are executed and to which policies are associated, at a high level of abstraction.
- the ability to define what actions are permitted or forbidden to do on resources in specific contexts (authorisations or permission/prohibition policies);
- the ability to define the actions that must be performed on resources in specific contexts (obligations).

To illustrate the expressive capabilities of the two semantic policy frameworks, i.e.,

KAoS [19] and Rei [63], we consider the case scenario of security officers within an airport.

KAoS

KAoS is a framework that provides policy and domain management services for agent and other distributed computing platforms [19]. It has been deployed in a wide variety of multi-agent and distributed computing applications. KAoS policy services allow for the specification, management, conflict resolution and enforcement of policies within agent domains. KPAT, a powerful graphical user interface, allows non-specialists to specify and analyse complex policies without having to master the complexity of OWL.

KAoS adopts an ontology-based approach to semantic policy specification. In fact, policies are mainly represented in OWL as ontologies. The KAoS policy ontologies distinguish between authorisations and obligations. In KAoS, a policy constrains the actions that an agent is allowed or obliged to perform in a given context. In particular, each policy controls a well-defined action whose subject, target and other context conditions are defined as property restrictions on the action type. The Code below shows an example of KAoS authorisation, which represents the *CCTVAccess policy* previously described. The property *performedBy* is used to define the class to which the actor must belong for the policy to be satisfied.

Listing 4.1: KAoS Policy Ontology

```
<owl:Class rdf:ID="BoardingAreaCCTVAccessAction">
       <owl:intersectionOf rdf:parseType="Collection">
              <rdfsowl:Class
                            rdf:about="&action;AccessAction" />
       <owl:Restriction>
                     <owl:onProperty
                            rdf:resource="&action;performedBy" />
                     <owl:allValuesFrom
                            rdf:resource="#TeamLeaderOfSecurityOfficer" />
              </owl:Restriction>
              <owl:Restriction>
                     <owl:onProperty
                            rdf:resource="&action;accessedEntity" />
                     <owl:allValuesFrom
                            rdf:resource="#CCTV-Monitor392" />
```

```
            </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>
<policy:PosAuthorizationPolicy
        rdf:ID="BoardingAreaCCTVAccess">
            <policy:controls
                rdf:resource="#BoardingAreaCCTVAccessAction" />
            <policy:hasSiteOfEnforcement
                rdf:resource="&some-ontology;TargetSite" />
            <policy:hasPriority>10</policy:hasPriority>
</policy:PosAutihorisationPolicy>
```

In KAoS, context conditions that constrain a policy may be specified through the definition of appropriate classes defined via property restrictions. In particular, two main properties, i.e., the *hasDataContext* and the *hasObjectContext* properties, and their subproperties, are used for the characterisation of the action context. Some subproperties are defined in the KAoS ontology, like for instance the ones defining the actor (*performedBy*), the time and the target resource (*accessedEntity*) of an action, while others may be created within domain-specific ontologies. The following code shows the definition of a class, namely *TeamLeaderOfSecurityOfficer*, which represents all the Team Leaders that are working as security officers within the airport. This class is defined as a subclass of the Security Officer class, having the affiliation property restricted to the Airport.

As these examples show, KAoS is based on an ontological approach to policy specification, which exploits OWL, i.e., description logic, features to describe and specify policies and context conditions. In fact, contexts and related policies are expressed as ontologies. Therefore, KAoS is able to classify and reason about both domain and policy specification basing on ontological subsumption, and to detect policy conflicts statically, i.e., at policy definition time.

Listing 4.2: KAoS Policy Example

```
<owl:Class rdf:ID="TeamLeaderOfSecurityOfficer">
      <rdfs:subClassOf
      rdf:resource="&some-ontology;SecurityOfficer" />
      <rdfs:subClassOf>
                  <owl:Restriction>
                        <owl:onProperty
                        rdf:resource="&some-ontology;Teamleader" />
```

```
                    <owl:allValuesFrom                                    8
                    rdf:resource="&some-ontology;Airport" />
                </owl:Restriction>                                        10
        </rdfs:subClassOf>
```

However, a pure OWL approach encounters some difficulties with regards to the definition of certain kinds of policies - specifically those requiring the definition of variables. For instance, by relying purely on OWL, we could not define policies, which indicates property value constraints (in reference to statically unknown values.) Other examples include policies that contain parametric constraints, which are assigned a value only at deployment or run time. For this reason, KAoS developers have introduced role-value maps as OWL extensions, implementing them within the Java Theorem Prover, used by KAoS [83] [111].

The adoption of role value maps, description logic-based concept constructors that were originally introduced in the KL-ONE system [99], allows KAoS to specify constraints between property values expressed in OWL terms, and to define policy sets, i.e., groups of policies that share a common definition but can be singularly instantiated with different parameters. The proposed extensions add sufficient expressive flexibility to KAoS for the representation of the policies discussed. However, non-experienced users may have difficulties in writing and understanding these policies without the help of the KPAT graphical user interface.

Rei

Rei [63] is a policy framework that permits the specification, analysis and reasoning about declarative policies defined as norms of behavior. Rei adopts a rule-based approach to the specification of semantic policies. Rei policies restrict domain actions that an entity can/must perform on resources in the environment, allowing policies to be developed as contextually constrained deontic concepts, i.e., right, prohibition, obligation and dispensation.

The first version of Rei (Rei 1.0) is defined entirely in first order logic with logical specifications for introducing domain knowledge [60]. The current version of Rei (Rei 2.0), featured in this paper, adopts OWL-Lite to specify policies and can reason over any domain knowledge expressed in either RDF or OWL [61].
A policy basically consists of a list of rules and a context that is used to define the policy domain. Rules are expressed as OWL properties of the policy. In particular, the policy:grants property is used to associate a deontic object with a policy either directly or via a policy:Granting. The List4.3 shows the Rei 2.0 policy specification

of the InformationSharing policy.

Listing 4.3: Rei Policy

```
<policy:Policy rdf:ID="InformationSharingPolicy">
       <policy:actor rdf:resource="#requester" />
       <policy:grants rdf:resource="#Perm_InfoAccess" />
</policy:Policy>
```

In order to specify context conditions, one or more constraints need to be defined. A constraint, which may be simple or boolean, i.e., the Boolean combination of a pair of simple constraints, defines a set of actors or a set of actions that fulfill a certain property. A simple constraint, as shown in the following code, is modelled as a triple consisting of a subject, a predicate and an object, which defines the value of the property for the entity, following a pattern that is typical of logical languages like Prolog.

Listing 4.4: Rei policy-Contraints Example

```
<constraint:SimpleConstraint rdf:ID="LocationOfSecurtiyOfficer">
              <constraint:subject
                     rdf:resource="&some-ontology;SecurtiyOfficer" />
              <constraint:predicate
                     rdf:resource="&some-ontology;location" />
              <constraint:object
                     rdf:resource="#SecurtiyOfficer-location" />
</constraint:SimpleConstraint>
<constraint:SimpleConstraint
       rdf:ID="CoLocatedWithSecurtiyOfficer">
              <constraint:subject
                     rdf:resource="#requester" />
              <constraint:predicate
                     rdf:resource="&some-ontology;location" />
              <constraint:object
                     rdf:resource="#SecurtiyOfficer-location" />
</constraint:SimpleConstraint>
<constraint:And rdf:ID="Constraint_CoLocated">
              <constraint:first
                     rdf:resource="#LocationOfSecurtiyOfficer" />
              <constraint:second
```

```
                    rdf:resource="#CoLocatedWithSecurtiyOfficer" />
</constraint:And>
```

A constraint can be associated to a policy at three different levels. The first possibility is to impose a constraint within the definition of a deontic object, by means of the deontic:constraint property, as shown in the following code. In this case, the constraint can be expressed over the actor, the action to be controlled or over generic environmental states.

Additional constraints can be imposed within the Granting specification on the entity the granting is made to, the deontic object the granting is made over and, again, over generic environmental states. Finally, it is possible to express a set of constraints directly within the policy definition through the *policy:context* property. These constraints are generically defined as conditions over attributes of entities in the policy domain.

Listing 4.5: Deontic Logic Example

```
<deontic:Permission rdf:ID="Perm_InfoAccess">
       <deontic:actor rdf:resource="#requester" />
       <deontic:action
              rdf:resource="&some-ontology;AccessToSharedInfo" />
       <deontic:constraint
              rdf:resource="#Constraint_CoLocated" />
</deontic:Permission>
```

Rei 2.0 uses OWL-Lite for the specification of policies and domain-specific knowledge [61]. Though represented in OWL-Lite, Rei still allows the definition of variables that are used as placeholders as in Prolog. In fact, as shown in List 4.4, the definition of constraints follows the typical pattern of rule-based programming languages, like Prolog.

In this way, Rei overcomes one of the major limitations of the OWL language, and more generally of description logics. i.e., the inability to define variables. For example, Rei allows developers to express a policy stating that a user is allowed to access the shared files of another user if they are located in the same area, whereas pure OWL would not allow for the definition of the ”same as” concept.

Therefore, Rei’s rule-based approach enables the definition of policies that refer to a dynamically determined value, thus providing greater expressiveness and flexibility to policy specification. On the other hand, the choice of expressing Rei rules

similarly to declarative logic programs prevents it from exploiting the full potential of the OWL language. In fact, Rei rules knowledge is treated separately from OWL ontology knowledge due to its different syntactical form. OWL inference is essentially considered as an oracle. Hence, Rei rules cannot be exploited in the reasoning process that infers new conclusions from the OWL existing ontologies, which means that the Rei engine is able to reason about domain-specific knowledge, but not about policy specification. As a main consequence of this limitation, Rei policy statements cannot be classified by means of ontological reasoning. Therefore, in order to classify policies, the variables in the rules need to be instantiated, which boils down to a constraint satisfaction problem. Let us consider, for example, the previously described *CCTVAccess policy*.

Unlike KAoS, Rei does not allow for a policy disclosure process that retrieves policies controlling a specific type of action. Hence, the user wanting to use the printer could only try to access it and see what the Rei engine has answered in response to his/her single access attempt. For the same reason, Rei cannot statically detect conflicts, like KAoS does, but it can only discover them with respect to a particular situation.

4.3 Gracia Policy Language

In order to fill the gap between the Knowledge base layer and the Web Services Implementation layer, we specify a formal semantic language called *Gracia Policy Language*(GPL). GPL is based on the syntax and semantics of OWL and SWRL. This enables the GPL approach with the shared advantage of both description logic and logic programming.

The model-theoretic semantics of GPL is an extension of the direct model-theoretic semantics defined in the OWL and SWRL. The Gracia Policy Language is a result of introducing vocabularies and interpretations of specific security-related concepts inheriting all features of OWL and SWRL. The GPL has been expressed based on ontologies in Appendix C. Thus, Gracia policies are a combination of OWL ontologies and SWRL Rules. Similar to the traditional access control and security models, the GPL ontologies aim to support the formal specification of policies with respect to standards, legal regulations, domain practices, agreements, approaches and traditions, etc. Ontologies also define languages. The Gracia Policy ontologies define the Gracia policy language. Figure 4.1 shows the core part of the Gracia Policy ontologies.

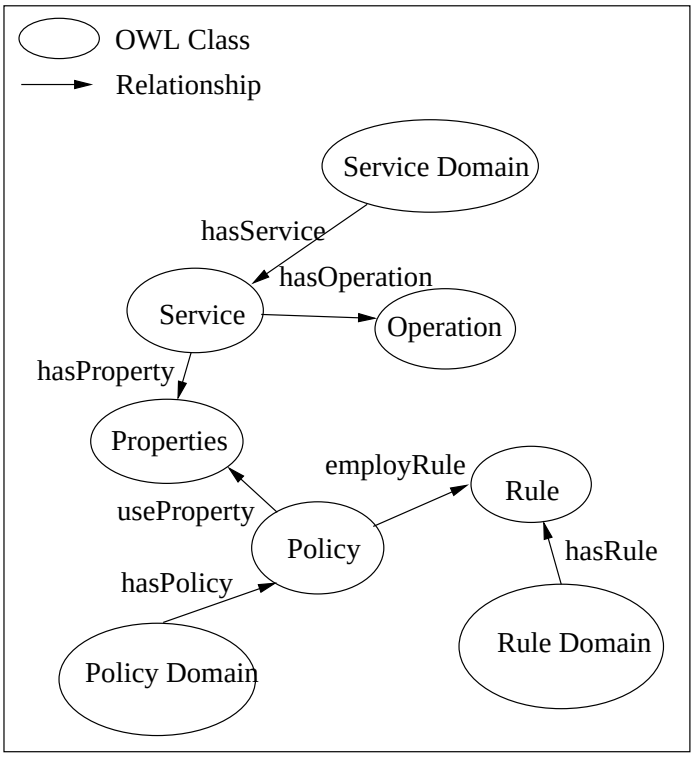

Figure 4.1: Gracia Ontology

The next step in describing the formal language is to determine the features of policy that need to be expressed in the formalism. Each type of policy contains a number of elements - *Subjects*, *Targets*, *Actions* and *Constraints*. In order to provide a complete formal representation of policies, the information contained in each of these elements must be included in the formalism.

In this section, we describe how each element in a policy is represented in the formalism and then go on to show how these clauses are combined to specify Gracia Policy.

Subjects and Targets

The subject and target elements of a policy refer to the ontology objects that are relevant to the policy. In the case of an authorisation policy, the Subject element is used to define what is being granted (or denied) the right to perform the operations specified in the Action clause. The target clause always refers to the objects on which the operations are being performed.

In our Gracia approach we specify subjects and targets as OWL Class or instances of OWL class, which in their more complex syntax allow for the combination of objects from different domains using set operations such as intersections

and unions. These are represented in the formal language of the form:

Listing 4.6: Formal Representation of Subjects and Targets

```
Class(namespace:SUBJECT partial)
Individual(namespace:Tom type(owl:Thing))
```

Operations

Every type of policy has an action element that specifies the operations that are relevant to it. An operation is defined using an identifier, some optional parameters and the name of the object upon which it will be performed (i.e. the target). This is represented in the formal language using the form:

Listing 4.7: Formal Representation of Operations

```
op(SubjObj, TargObj, OpName, Parms)
```

For authorisation policies, the action clause simply defines the set of one or more operations that are allowed (or prohibited) when the policy is applicable.

Constraints

Constraints are used in policy specifications to control the applicability of the policy based on the runtime state of the system. Although Gracia constraints are specified using the SWRL rules, typically only a subset of the features of SWRL are used when specifying policy constraints. In most situations, policy constraints are defined as Boolean expressions, either using comparison operators or attributes of objects.

The following section details the mapping of the GPL with current policies such as WS-Policy and XACML.

4.3.1 From GPL to WS-Policy

One shared characteristic of the current proposed web service policy languages is that they involve policy assertions and combinations of assertions. For example, a policy might assert that a particular service requires some form of reliable messaging or security, or it may require both reliable messaging and security. Several industrial proposals (e.g., WS-Policy) appear to restrict them to a kind of propositional logic with policy assertions being atomic propositions and the combinations being conjunction and disjunction. In our approach, a clear semantics

for the Gracia policy languages can be acquired by mapping the policy language constructs into a description logic (sub variant of first order logic).

The Web Ontology Language and the Resource Description Framework (RDF [73]) were chosen as the basis syntax and semantics for the Gracia Policy Language. Both RDF and OWL are strict subsets of first order logic, with the subspecies OWL-DL being a very expressive yet decidable subset. OWL-DL builds on the rich tradition of description logics where the tradeoff between computational complexity and logical expressivity has been precisely and extensively mapped out and practical, scalable reasoning algorithms and systems have been developed. In our implementation, the formalism underlying the WS-Policy grammar is captured and translated into OWL-DL class expressions.

Assertions are the building blocks for a Web service policy and satisfying them usually results in a behavior that satisfies the conditions for the service endpoints to communicate. A policy assertion is supported by a requestor if and only if the requestor satisfies the requirement, or accommodates the capability corresponding to the assertion. Policy assertions usually deal with domain-specific knowledge and they can be grouped into policy alternatives. An alternative is satisfied only if the requestor of the service satisfies all of the policy assertions contained in the alternative. Note that in our ontology, policy assertions and alternatives are represented with separate OWL classes related to the *containsAssertions* property. Determining whether a policy alternative is supported is done automatically using the results of the policy assertions. List 4.8 shows OWL Ontology for Policy Assertation.

Listing 4.8: OWL Ontology of WS Policy Assertation

```
<rdf:RDF xml:base="&WS-PolicyAssertion.owl;">

<!-- Ontology Information -->
<owl:Ontology/>

<!-- Classes -->
<owl:Class rdf:about="#Assertion"/>
 <owl:Class rdf:about="#Integrity">
    <rdfs:subClassOf rdf:resource="#Assertion"/>
</owl:Class>
<owl:Class rdf:about="#Language">
    <rdfs:subClassOf rdf:resource="#Assertion"/>
```

```
</owl:Class>
<owl:Class rdf:about="#SecurityToken">
  <rdfs:subClassOf rdf:resource="#Assertion" />
</owl:Class>
<owl:Class rdf:about="#TextEncoding">
  <rdfs:subClassOf rdf:resource="#Assertion" />
</owl:Class>
<owl:Class rdf:about="&owl;Thing" />

<!-- Datatypes -->
<rdfs:Datatype rdf:about="&xsd;int" />
<rdfs:Datatype rdf:about="&xsd;string" />
<!-- Datatype Properties -->
<owl:DatatypeProperty rdf:about="#hasEncoding">
  <rdfs:domain rdf:resource="#TextEncoding" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasLanguage">
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasPreference">
  <rdfs:domain rdf:resource="#Assertion" />
  <rdfs:range rdf:resource="&xsd;int" />
</owl:DatatypeProperty>
</rdf:RDF>
```

A policy is supported by the requestor of a service if the requestor satisfies at least one of the alternatives in the policy. Once the policy alternatives have been evaluated, it can be automatically determined whether a policy is supported by the requestor.

There are two operators used for the expression of relations between policies, alternatives and assertions: *All* and *ExactlyOne*. These operators are implemented as OWL classes *OperatorAll* and *OperatorExactlyOne* in Mindswap's ontology [69]. *OperatorAll* requires all the assertions to hold in order for the policy alternative to be satisfied. *OperatorExactlyOne* specifies that exactly one of the assertions has to hold in a collection of policy alternatives for the policy assertion to be satisfied.

To capture the semantics of WS-Policy, for example, to determine whether two policies are consistent with each other, is accomplished by translating the WS-Policy constructs from a normal form policy expression into OWL constructs. A normal form policy expression is a straightforward XML Infoset representation of a policy, enumerating each of its alternatives that in turn enumerate each of its assertions. The following List 4.9 is a schema outline of the normal form of a WS-Policy expression:

Listing 4.9: Normal form of a policy expression

```
<wsp: Policy>                                                          1
  <wsp:ExactlyOne>
    [ <wsp:All> [<Assertion>  </Assertion>]* </wsp:All> ]*             3
  </wsp:ExactlyOne>
</wsp:Policy>                                                          5
```

First, policy assertions are mapped directly into OWL-DL atomic classes (which correspond to atomic propositions). Though WS-Policy assertions often have some discernible substructure, it isn't the key to their logical status in WS-Policy. Or rather, that substructure is idiosyncratic to the assertion set, rather than being a feature of the background formalism. So a general WS-Policy engine must be adapted to deal with their structure.

Mapping *wsp:All* to an OWL construct is relatively straightforward because *wsp:All* means that all of the policy assertions enclosed by this operator have to be satisfied in order for communication to be initiated between the endpoints. Thus, it is a logical conjunction and can be represented as an intersection of OWL classes. Each of the members of the intersection is a policy assertion, and the resulting class expression (using the operator *owl:intersectionOf*) is a custom-made policy class that expresses the same semantics as the WS-Policy one. Handling *wsp:ExactlyOne* might be trickier, depending on the interpretation of the construct. There are two possible interpretations:

- *wsp:ExactlyOne* means that a policy is supported by a requester if and only if the requester supports at least one of the alternatives in the policy. In the previous version of WS-Policy there was a *wsp:OneOrMore* construct capturing this meaning. In such cases, the *wsp:ExactlyOne* is an inclusive *OR*, and can be mapped using *owl:unionOf*.
- The other interpretation is that *wsp:exactlyOne* means that only one, not more, of the alternatives should be supported in order for the requester to

support the policy. This is supported by Kagal'work [60], which states that although policy alternatives are meant to be mutually exclusive, it cannot be decided in general whether or not more than one alternative can be supported at the same time.

A complete formal representation of WS-Policy in Gracia Policy Language is listed in the Appendix A.1.

4.3.2 From GPL to XACML

The OASIS standard eXtensible Access Control Markup Language (XACML [84]) is a highly expressive access control language with significant deployment. XACML enables the use of arbitrary attributes in policies, allows for the expression of "deny" policies and enables the use of hierarchical RBAC. One of the many approaches that has recently emerged is the use of logic and formal reasoning techniques for analysis and verification of XACML policies. They support only a very small subset of the language.

We use Description Logics (DL) to provide a formalisation of XACML. At the root of all XACML policies is a *Policy* or a *PolicySet*. Each XACML policy document contains exactly one Policy or PolicySet root element. A *PolicySet* is a container that can hold other *Policies* or *PolicySets*, as well as references to policies found in remote locations. A *Policy* represents a single access control policy, expressed through a set of *Rules*. *Rules* are the most basic element of XACML that actually makes access decision. Essentially, a Rule is a function that takes an access request as input and yields an access decision (Permit, Deny, or Not-Applicable). To determine if a *Rule* is applicable to an access request, the *Target* element is used. A *Target* is a set of simplified conditions for the *Subject*, *Resource* and *Action* that must be met for a *Rule* to apply to a given request. These use boolean functions to compare values found in a request with those included in the *Target*.

Four main policy elements in XACML can be formalised with DL: *Rules*, *Requests*, *Policies* and *Policy Sets*. Whilst the Target element of Rules and Requests can be mapped to a DL class expression, the interaction of the access decision of various policy elements is difficult, if not impossible, to do using only description logics. This is due to the semantics of the combining algorithms which requires us to use a formalism that supports preferences. For this reason, Kolovski [68] proposed defeasible description logics to capture the semantics of the combining

algorithms, which is a formalism that allows for the expression of defeasible rules on top of description logics.

Reasoning services in DDL can be reduced to description logic reasoning, which makes it possible to use DL reasoners for the analysis of XACML policies. Representation of access control policies in description logic, for example OWL, has additional benefits for the policy author. The general applicability of OWL permits specification of policy using terminology that exists in a broader context than the XACML document itself.

A policy can adopt terminology present in enterprise data descriptions, common industry ontologies, and other OWL content. The benefits of reusing formalisms is twofold. Firstly, it expedites the policy creation process by avoiding repetition of already performed business process modelling work. Secondly, it improves quality by allowing the adoption of high-quality models already validated in other deployments. The application of OWL to policy creation and management is discussed in the work from NASA [105].

XACML has a profile targeted at representation of role based access control (RBAC). In our Gracia approach, we have developed an OWL Ontology for the RBAC profile in XACML as showed in List 4.10.

Listing 4.10: OWL Ontology for RBAC profile in XACML

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.ic3.ct.siemens.com/RBAC.owl#"
  xml:base="http://www.ic3.ct.siemens.com/RBAC.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Subject"/>
  <owl:Class rdf:ID="Resource"/>
  <owl:Class rdf:ID="Role"/>
  <owl:Class rdf:ID="User">
    <rdfs:subClassOf rdf:resource="#Subject"/>
  </owl:Class>
  <owl:Class rdf:ID="Contraints"/>
  <owl:Class rdf:ID="Group">
    <rdfs:subClassOf rdf:resource="#Subject"/>
```

```
</owl:Class>
<owl:Class rdf:ID="CompositeAction">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Action" />
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Permission" />
<owl:Class rdf:ID="AtomicAction">
  <rdfs:subClassOf rdf:resource="#Action" />
</owl:Class>
<owl:ObjectProperty rdf:ID="applyTo">
  <rdfs:domain rdf:resource="#Contraints" />
  <rdfs:range rdf:resource="#Permission" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="toRole">
  <rdfs:range rdf:resource="#Role" />
  <rdfs:domain rdf:resource="#Permission" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inheritFrom">
  <rdfs:domain rdf:resource="#Role" />
  <rdfs:range rdf:resource="#Role" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="SubjectToRole">
  <rdfs:domain rdf:resource="#Subject" />
  <rdfs:range rdf:resource="#Role" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="include">
  <rdfs:domain rdf:resource="#Group" />
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="partOf" />
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Group" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#partOf">
  <rdfs:range rdf:resource="#Group" />
  <owl:inverseOf rdf:resource="#include" />
  <rdfs:domain rdf:resource="#Group" />
```

```
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="toResouce">
    <rdfs:domain rdf:resource="#AtomicAction" />
    <rdfs:range rdf:resource="#Resource" />
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="forAction">
    <rdfs:range rdf:resource="#Action" />
    <rdfs:domain rdf:resource="#Permission" />
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="havePermission">
    <rdfs:range rdf:resource="#Permission" />
  </owl:ObjectProperty>
</rdf:RDF>
```

At the current level of maturity, many common RBAC patterns are supported by the translation, including separation of duties and role cardinality constraints. The DL translation has potential relevance in any role-based formalism and evaluation of its applicability to role-based workflow management. The List4.11 shows an example of XACML policy in the form of Gracia Policy Language. A more detailed example can be found in the appendix of this thesis.

Listing 4.11: XACML in Gracia Policy Format

```
<rdf:RDF
    xmlns="http://www.ic3.ct.siemens.com/rbac_example.owl#"
    xmlns:rbac-meta="http://www.ic3.ct.siemens.com/RBAC.owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="rbac.owl" />
  </owl:Ontology>
  <rdf:Property rdf:about="#toActionandResource" />
  <rdf:Property rdf:about="RBAC.owl#PermissionToRole" />
  <rbac-meta:Group rdf:ID="SecurityOfficer">
    <rbac-meta:SubjectToRole>
      <rbac-meta:Role rdf:ID="Controller" />
    </rbac-meta:SubjectToRole>
  </rbac-meta:Group>
  <rbac-meta:Resource rdf:ID="Monitor" />
  <rbac-meta:AtomicAction rdf:ID="checkStatus">
    <rbac-meta:toResouce rdf:resource="#Monitor" />
```

```
  </rbac-meta:AtomicAction>                                          17
  <rbac-meta:User rdf:ID="Bob">
    <rbac-meta:SubjectToRole rdf:resource="#Controller" />           19
  </rbac-meta:User>
  <rbac-meta:Permission rdf:ID="toViewMonitor">                      21
    <rbac-meta:toActionandResource rdf:resource="#checkStatus" />
    <rbac-meta:forAction rdf:resource="#checkStatus" />              23
    <rbac-meta:toRole rdf:resource="#Controller" />
    <rbac-meta:PermissionToRole rdf:resource="#Controller" />        25
  </rbac-meta:Permission>
</rdf:RDF>                                                           27
```

4.4 Summary

In this chapter, we introduce our Gracia Policy Language, which shares syntax and semantics with OWL and SWRL.

We emphasise that, at present, we intend the Gracia Policy Language to be used not only at design time, but also in a policy enforcement point (PEP) for the enforcement of policies. Firstly, it is still not entirely clear how efficient the GPL would be for enforcement in today's SOA set-ups. They can be computationally expensive. Given the fact that one requirement for PEP is that it can handle the response requirements of applications under load, we need to take care not to introduce too much overhead.

The full expressivity of XACML is not yet accessible in the DL representation. Kolovski notes some constructions in XACML that are not readily translatable [68], XACML uses XML Schema and additional datatypes, which yields a datatype expressivity that is beyond its counterpart in OWL-DL. For this reason, we may develope user-defined datatype extensions of OWL and enhance the utility of this approach by enabling translation of access control policies which specify constraints with respect to ranges of values. Future research is required to determine the extent of the overlap between datatype reasoning in the OWL and XACML representations.

Chapter 5

Gracia Knowledge Base

5.1 Introduction

In our work, Gracia Policy Language is used to represent security constraints and other regulatory requirements. This kind of security-related information will be distributed across the whole SOA environment. Even unknown entities could possibly share their knowledge during the service transaction process and business process. We then developed the concept of **Gracia Knowledge Base**, which is based on the multiagent system approach of the Virtual Knowledge Community [21]. In this chapter we will introduce both the hybrid approach to the Gracia Knowledge Base and discuss trust issues related to knowledge-sharing. A series of trust-related protocols are designed to transfer the evidence between agents enabling evidence-based trust management, which is now popular in pervasive computing and service-oriented computing.

5.2 Related Work

5.2.1 Knowledge, Constraints and Rule Interchange Format

Knowledge Interchange Format (KIF) [37] is a computer-oriented language for the interchange of knowledge between disparate programs. It has declarative semantics (i.e. the meaning of expressions in the representation can be understood without the use of an interpreter for the manipulation of the expressions); it is logically comprehensive (i.e. it permits the expression of arbitrary sentences in the

first-order predicate calculus); it provides for the representation of knowledge and nonmonotonic reasoning rules; and it provides for the definition of objects, functions, and relations. However, KIF is not well designed and has a lack of support for internet and web environment [95].

Constraint Interchange Format (CIF) is based on the Colan [9] constraint language, which itself is based on range restricted first order logic (FOL). Earlier versions of the language were aligned with Resource Description Framework (RDF) [54] and SWRL. CIF constraints are essentially defined as quantified implications, meaning one can re-use the implication structure from SWRL but allow for nested quantified implications within the consequence of an implication. An example CIF constraint is shown in human-readable SWRL-style syntax below:

$$(\forall ?x \in X, ?y \in Y) p(?x, ?y) \wedge Q(?x) \Rightarrow$$
$$(\exists ?z \in Z) q(?x, ?z) \wedge R(?z) \Rightarrow$$
$$(\forall ?v \in V) s(?y, ?v)$$

In [80], an RDF/XML syntax is provided as an extension of the SWRL syntax to support the publishing and interchange of CIF constraints. A new **rdfs:Class** Constraint, with properties hasQuantifiers and hasImplication is defined. For example, if one were to introduce a business requirement such as ”every delegation group must contain at least one participant from the government”, the following code5.1 applies RDF/XML for this constraint.

Listing 5.1: RDF/XML for the constraints

```
<cif:Constraint>
 <cif:hasQuantifiers
     rdf:parseType="Collection">
  <cif:Forall>
   <cif:var rdf:resource="#g"/>
   <cif:set rdf:resource="#Delegationgroup"/>
  </cif:Forall>
  <cif:Exists>
   <cif:var rdf:resource="#p"/>
   <cif:set rdf:resource="#Government"/>
  </cif:Exists>
 </cif:hasQuantifiers>
 <cif:hasImplication>
   <swrl:Imp>
```

```
    <swrl:body rdf:parseType="Collection" />
    <swrl:head rdf:parseType="Collection">
     <swrl:IndividualPropertyAtom>
      <swrl:classPredicate
           rdf:resource="#has-member" />
      <swrl:argument1 rdf:resource="#g" />
      <swrl:argument2 rdf:resource="#p" />
     </swrl:IndividualPropertyAtom>
    </swrl:head>
   </swrl:Imp>
  </cif:hasImplication>
 </cif:Constraint>
```

Rule Interchange Format (RIF)[1] is another interchange format used for logic expressions on the Web. The W3C RIF WG[2] has the mission of producing a rule interchange format in order to translate rules between rule languages and thus be able to use them for different rule systems. As the interest in rule languages and systems is noticeable in different communities such as the Business Rules and the Semantic Web communities, the W3C RIF WG tries to find consensus in having a core rule interchange format combined with a set of extensions. The charter talks about two phases. The first phase addresses a simple, useful and extensible interchange format for rules. This first phase runs until the end of year 2007. The second phase addresses extensions of the core rule interchange format developed as a result of the first phase. As RIF is still under developement, we consider CIF as the basis of our approach. It is expected that CIF will evolve to use RIF in place of SWRL as the new format takes shape. As it is currently planned, Phase 1 RIF is essentially Horn Logic. If Phase 2 RIF includes full FOL it is possible that this format will fully subsume CIF. At this point it is conceivable to simply define CIF as a subset of RIF: constraints would be interchanged in RIF itself [92].

5.2.2 Integration of Rules and Ontology

The management of security and regulator constraints requires the appropriate description of domain knowledge. The current approaches have outlined two main

[1] http://www.w3.org/2005/rules/
[2] http://www.w3.org/2005/rules/wg

research directions. On one side, a purely ontology-based approach exploits description logic, e.g., OWL, to describe policies at a high level of abstraction, in a form that allows their classification and comparison. This feature is necessary, for instance, in order to detect conflicts between policies before they are actually enforced, thus granting interoperability between entities belonging to different domains that adopt different policies. On the other side, a rule-based approach relies on the features of logic programming languages, e.g., Prolog, to enable the evaluation of policy instances.

Description Logics (DLs) are currently the most commonly used formalisms for building ontologies and have been proposed as standard languages for the specification of ontologies in the Semantic Web [79]. In particular, the OWL family of languages, based on Description Logics, has been explicitly designed for this purpose. DLs are a family of knowledge representation formalisms based on first-order logic (FOL). In fact, almost all DLs coincide with decidable fragments of function-free first-order logic with equality, and the language of a DL can be seen as a restricted FOL language over unary and binary predicates and with a controlled form of quantification (actually, DLs are equipped with a special, variable free syntax). Notably, DLs have been designed to optimise the trade-off between expressive abilities and complexity of reasoning, hence the in-depth research into the computational properties of DLs [5]. However, the typical expressiveness of DLs doesn't address the following aspects [95]:

- the possibility of defining predicates of arbitrary arity (not just unary and binary)
- the use of variable quantification beyond the tree-like structure of DL concepts (many DLs actually correspond to subsets of the two-variable fragment of first order logic)
- the possibility of formulating expressive queries over DL knowledge bases (beyond concept subsumption and instance checking)
- the possibility of formalising various forms of closed world reasoning over DL knowledge bases
- in general, the possibility of expressing forms of nonmonotonic knowledge, like default rules [94]

The task of overcoming the limitations of OWL and DLs is currently receiving a lot of attention in the Semantic Web community. In order to fully overcome the above

limitations, the realm of classical first-order logic would have to be abandoned in favour of nonmonotonic logic.

Almost all the kinds of knowledge that cannot be formally addressed in a classical, first-order logic setting have a ”rule-like” form, i.e., they can be expressed by statements in the form ”if the precondition α holds then the conclusion β holds”, where the precondition and the conclusion are logical properties. However, such a piece of knowledge cannot simply be formalised through the standard material implication of classical logic: in other words, it is not possible to capture the intended meaning of the above statement by an implication in classical first-order logic of the form $\alpha \rightarrow \beta$.

Therefore, rule-based formalisms grounded in logic programming have repeatedly been proposed as a possible solution for the above limitations. Therefore, the addition of a rule layer on top of OWL is nowadays seen as the most important task in the development of the Semantic Web language stack. The Rule Interchange Format (RIF) working group of the World Wide Web Consortium (W3C), as mentioned previously, is currently working on standardising such a language.

Most of the proposals in this field focus on logic programs expressed in Datalog (and Datalog’s nonmonotonic extensions) [31]. With respect to DLs, Datalog allows for the use of predicates of arbitrary arity, the explicit use of variables and the ability to express more powerful queries. Moreover, its nonmonotonic features allow for the expression of default rules and forms of closed-world reasoning [95].

5.3 Gracia Knowledge Base

In this section, we will first provide a formal definition of the Gracia knowldege base and then propose an approach aimed at integrating the process information by mapping it to the ontology language OWL.

With the help of an e-payment example, we will demonstrate the use of VKC and a multiagent system to enable knowledge sharing in the web services environment. We also consider the trust of knowledge as the primary security issue in the knowledge integration process. A series of so-called trust brokering protocols are introduced to enable the anonymous trust evaluation of agents in the Gracia knowledge Base.

5.3.1 Hybrid Knowledge Base Approach

The Gracia knowledge base consists of the security and domain ontology enriched with rules. For the moment, the ontology about the security constraints and domain knowledge is represented in OWL DL and the rules in SWRL.

All of the knowledge, the ontology in OWL DL (Tbox), the Horn rules (Rule-box) and the facts (Abox) are gathered within a single file provided as input to the reasoner.

- **Tbox**: the Tbox provides logical definitions of concepts (classes), roles (properties) and asserted axioms. A TBox stores the conceptual knowledge of an application domain. It defines the intentional knowledge in the form of a terminology (hence the term ”TBox”). The terminology consists of concepts, which denote sets of individuals and roles, which in turndenote binary relations between individuals. The Gracia Knowledge Base can build atomic concepts and roles (concept and role names) and can also build complex descriptions of concepts and roles. For example, the OWL definition implies that all Security Officers are subclasses of Airport Officers
- Rule-box: The Rule-box contains all the rules extending the ontology, for example the CCTVAccessPolicy can be decribed as: *inTeamleader(?x)* $\wedge$ *inBoardingZone(?x,?y)* $\rightarrow$ *allowtoAccess(?x, ?z)*. If a security officer *x* is also a Teamleader, and *x* is in theBoarding Zone *y*, then *x* is allowed to access a CCTV device *z*. Such rules are needed to infer the missing knowledge of the classes definitions for instance retrieval. Rules are also useful for expressing queries. For example, to find all of the possible instances of officers in possession of mobile devices with digital signatures d_i installed: *Q*(?x_i.... x_n)$\leftarrow$ $\wedge$ $_{i=1ton}$(*SE*(?x$_i$)) $\wedge$*DigiSignatureInstalled*($\wedge$,?x_i).
- Abox: The Abox contains the individuals (instances of classes) and the instances of relations between them. An ABox contains extensional knowledge regarding the domain of interest. It introduces the assertional knowledge (hence the term ”ABox”) (world description). Whereas TBoxes restricts the set of possible words, ABoxes allows the user to describe a specific state of the world by introducing individuals (or instances) together with their properties. In the Abox, knowledge can be divided into a concept assertion, which states that an individual is a member of concept, and a role assertion with a pair of individuals. When we say an ABox A is defined with respect to a Tbox, the concept description in A may contain defined names of TBox.

Notation. A Gracia Knowledge Base is denoted by Σ=(T, A, R) where T is the TBox, A the ABox and R the Rule-Box. Combining what has been said so far, a Knowledge Base Σ is a tuple Σ=(T, A, R), which has a model if there exists an interpretation I that satisfies both T and A. If an axiom α in T or A is true in every model of Σ, it can then be said that Σ logically implies α.

A complete formal representation example of Gracia Knowledge Base is listed in the appendix of this thesis B.1.

5.3.2 Agent-based Knowledge Integration

Multiagent systems have been invented to tackle Knowledge Integration problems within artificial intelligence. The Agent Oriented Abstraction has been proposed to describe a society of agents in a fully generic way. In this section, the application of these models to the abstract modelling of the *Gracia Knowledge Base* will be discussed.

Agents and Web Services

In the cyber world, agent-based approaches are more powerful when running agents in a distributed and dynamic environment (potentially on a web-wide scale) to perform complex actions for their users [43]. Uniting agents and web services can enhance the construction and flexibility of web service applications [112].

Before discussing some of the other agent-based approaches in the following sections, it is first necessary to acquaint the reader with some agent related definitions. An agent is capable of carrying out autonomous actions in an assigned environment in order to meet its design objectives [117]. Based on this definition, an intelligent agent can be assigned three additional characteristics: reactivity, proactivity and social ability. The concept of the multiagent has emerged as a paradigm for designing complex software systems. It is mainly used for the more enhanced formalisation of problems in Distributed Artificial Intelligence (DAI) [114].

The world of web services is characterised as loosely-coupled distributed systems based on SOC. The use of web services could be described as actions that the agent might execute to meet its goals. In [53] four major trends that have driven SOC and Multi Agent Systems (MAS) into the future were analysed. They are among emerging approaches with MAS-like characteristics in SOC, such as ubiquitous computing, ontologies, service-level agreements and quality-of-service

measures for instance. All of them can be suitably tackled with MAS concepts and techniques.

Agent Oriented Abstraction

In the AOA approach, agents are viewed as objects which, through their knowledge contents, are organised into annotations that gather classes. Encapsulation, inheritance and polymorphism are features that can be adequately defined. The AOA model can be abstractly summarised into a number of basic definitions. A detailed description of these definitions can be found in [22].

Chiefly, in AOA all agents have two parts: a decision mechanism and knowledge. For the former, a scope of possible classification of utility was given: expected utility function, the common sense measure of usefulness, the class of models and the class arising from logical modelling. For the latter, there are also some associated classes: ontology, communication, cognition and safety. Based on the knowledge annotations, agents can generate utility related to their tasks and goals.

Within the AOA approach, web services and their related policies can be abstractly modelled in the knowledge part of agents running around the semantic web. Section 5.3.3 will illustrate this approach with the aid of a working example.

Virtual Knowledge Communities

In [75] the application of the AOA model to the abstract modelling of corporate knowledge is investigated. To avoid the separation of agents and knowledge, it was considered that agents explicitly represent knowledge and communication ability.

Traditionally, information is mostly centralised within a uniform information structure. This viewpoint is not truly compliant with the nature of knowledge that is subjective, distributed and contextual [18]. From the perspective of the knowledge information society, modern knowledge management often focusses on the constitution of communities of practice and communities of interest [34].

The concept of a community of practice or a community of interest can be supported in a virtual community in order to bring the concerned agents together to share their knowledge with each other. A community is a place where agents can meet and share knowledge with other agents who share a similar domain of

interest. The concept of a VKC was introduced as a means for agents to share knowledge about a topic [76]. It aims to increase the efficiency with which information is made available throughout the society of agents.

From the point of view of corporate knowledge management, agents can be individuals, software assistants or automata. Agents possess knowledge and processes within the society tend to make agents produce and exchange knowledge with each other. These processes are distributed throughout the society and, through their own intrinsic goals, contribute to the resolution of a unique high-level challenge. This acts as the link between corporate knowledge and VKC [75].

In the implemented model [42] there are two main modelling-types for VKC: agent modelling and community modelling. The former has four key aspects: personal ontology, knowledge instances, knowledge cluster and mapper. Personal ontology represents the knowledge of an agent. It describes the taxonomy of the relationships between the concepts and predicates that an agent understands. The knowledge instances are instances of objects defined by the personal ontology. It was assumed that an agent's knowledge consists of both its personal ontology and knowledge instances according to its personal ontology. The knowledge cluster is the sub-part of an ontology that can be shared between agents. Clusters are defined by their head concept, an indicator of the different parts of knowledge existing within a cluster. The mapper chiefly contains a set of mapping from personal terms to mapped terms, and allows an agent to add such mappings, and use the mapper to normalise or personalise a given knowledge cluster or instance using these mappings. It facilitates knowledge-sharing among agents with regards to the heterogeneity of knowledge.

Community modelling also has some key aspects: domain of interest, community pack and community buffer. A domain of interest exists in each VKC and is similar to the concept of ontology for an agent. It is given by the creator of the community, the community leader. The community pack is what defines the community. It consists of a community knowledge cluster, a normalised ontology which contains at least the head of the community cluster, and the identification of the leaders of the community. The community buffer can record messages which are used by the member of a community to share their knowledge. This approach is compatible with blackboard systems but differs in the sense that agents cooperate to find a solution to their respective problem, not for a unique goal.

The VKC approach has been designed and partially implemented as a prototype system. The implementation is based on Java Agent Development Framework

(JADE) and Java Runtime Environment (JRE) platform. It was tested and evaluated. A component of the system enables us to simulate virtual knowledge communities (VKCs).

5.3.3 Example: e-Payment

To demonstrate the need for modelling policies and knowledge base of these policies within the Agent Oriented Abstraction, a scenario of the payment problem in the e-money system is as follows:

A mobile network operator has developed a service platform to allow customers to pay bills and buy things both online- and offline with their PDA, cell phone or laptop. The other customers of the service platform, for example a bank, will accept the payment request and decide if the transaction should be processed. Due to the dynamically changing business rules, legal framework for e-business and customer preferences, policies for these issues are distributed and managed by different stakeholders in the transaction.

- A customer, Bob, buys a video camera in shop A at airport B, issues a payment request via his PDA to a payment service of his bank on the service platform, specifying payment details including the amount to be paid and the payment method. The payment amount is $1,000 and the payment method is Visa card.
- The payment service of the bank receives the transaction request. First, the bank handles the request. The airport only allows transactions in a foreign currency (such as US Dollars) and the requestor has a good credit history. Next, based on the preferred payment method and physical connection type of the customer's PDA (which could be WLAN, GPRS or Ethernet), an encryption algorithm is chosen. This protects the integrity of the transferred data due to the fact that physical connection types can support different bandwidths. The bank has an agreement with the shop: requests from customers in the shop at the airport will be handled with a higher priority. The priority of policies are ranked in the following order: 1) legality, 2) security and 3) convenience.
- The network operator knows that Bob's PDA is connected via GPRS and it is connected to a base station at airport B.
- The local government allows trading in US dollars and the bank has determined Bob's credit history to be good.

In this scenario, we have one authorisation policy of payment service, one obligation policy of PDA, three rules and one non-functional description of payment service. The concepts of authorisation policy and obligation policy are introduced in [104]. Three policies of the payment service are modelled: Legality Policy (Access control), Security Policy (Data Integrity) and Convenience Policy (QoS). These kinds of services will be different domains of interest in VKCs.

Agents and Their Knowledge

In this scenario, there are five agents categorised as follows: P-agent: “Bank”; R-agent: “PDA”; M-agents: “Network Operator”, “Local Government” and “Airport”. It is assumed that any of these agents are modelled as a VKC. This implies that an agent’s knowledge is ontologically structured. Figure 5.1 provides a simplified view of knowledge of agents in the scenario. The dashed line indicates instances of concept, while the solid line with a label and arrow indicates the relationship between concepts. The VKC approach The basic concepts of VKC were introduced in Section 5.3.2. Figure 5.2 will now illustrate how to build corporate knowledge with VKC based on the individual knowledge base shown in Figure 5.1.

All stakeholders in this scenario are defined as service providers (P-agent), service requesters (R-agent) and middle-agents (M-agent) [28]. Figure 5.2 depicts a simplified community of communities for the e-business scenario consisting of several possible member communities. Agents can create as many communities as they like. In this picture there are three virtual knowledge communities for different domains of interest: legality, security and convenience. As mentioned above, an agent can join more than one community, agents ”PDA” and ”Bank” participate in each of these three communities, respectively as service requester and service provider. The M-agents ”Local Government”, ”Network Operator” and ”Airport” join the communities “Legality”, “Security” and “Convenience” respectively.

In order to explain the ways in which a community generally works towards the facilitation of knowledge sharing, the community “Security” is illustrated in Figure 5.3. In order to share their knowledge clusters and instances´, Agents “Bank”, “Network Operator” and “PDA” can read and write messages in the community buffer. The dashed line leading from an agent to the community buffer implies that an agent writes messages into the buffer, whilst the dashed line in the opposite direction depicts the reading of messages.

In Figure 5.3 messages are structured through simplified notations of ontologies

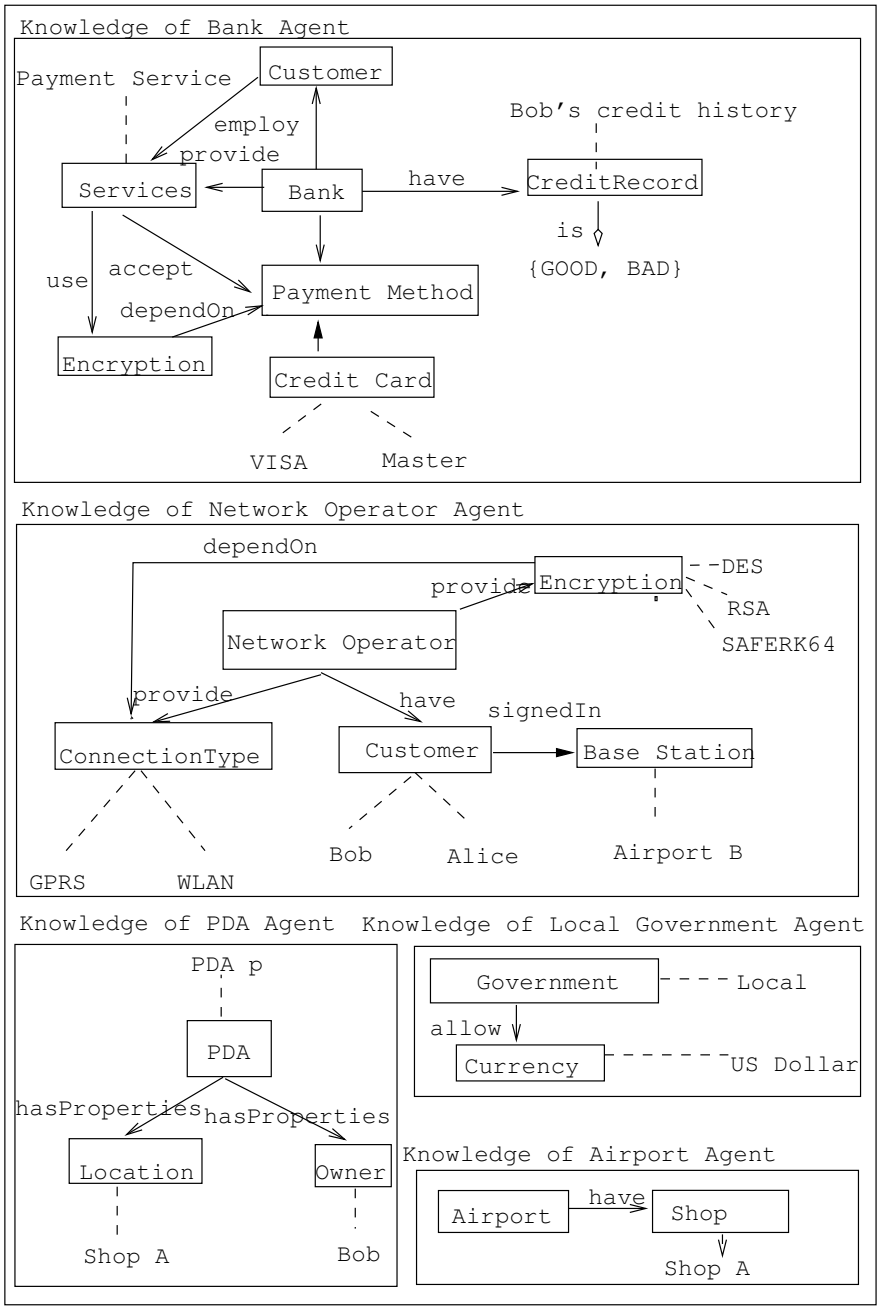

Figure 5.1: Agent's knowledge with their personal ontology and instances.

with concepts, their instances and relations between them. For example, the notation `PaymentMethod:Visa,Master` denotes that the concept PaymentMethod has two instances: Visa and Master.

The agent "Bank" writes one message into the buffer and reads two messages which are coming from "Network Operator" and from "PDA" respectively. The

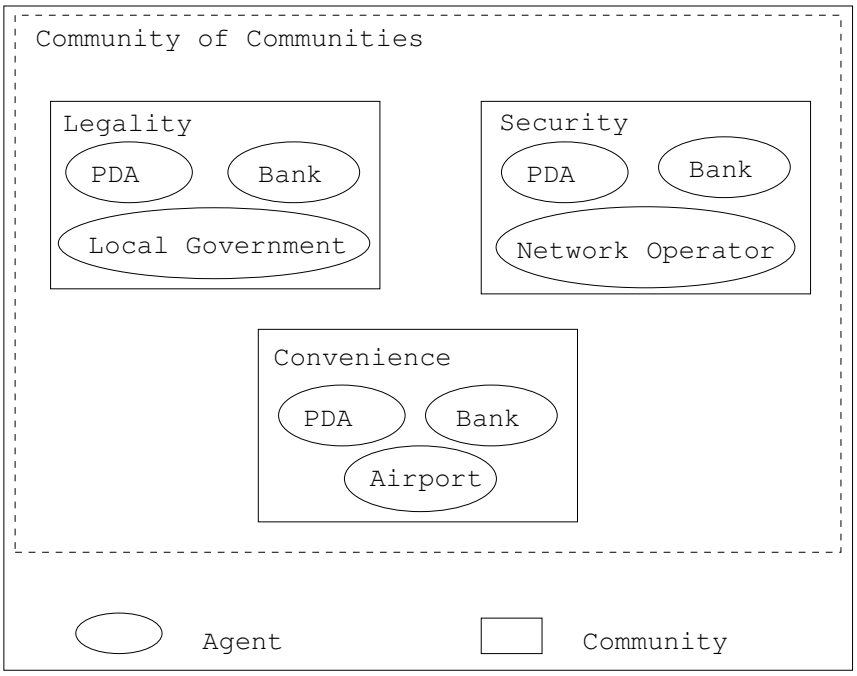

Figure 5.2: The simplified view of virtual knowledge communities of the e-business scenario.

agent “Network Operator” writes two messages yet only has interest in the message from “PDA”. “PDA” inputs just one simple message but benefits from the contribution of other agents in the community. After the knowledge exchange step, each agent use the new information to update its knowledge base respectively to its knowledge cluster and personal ontology. The feature introduced in the above example (especially for agent “PDA”) is illustrated in Figure 5.4. The “PDA” now knows that it can sign in to the base station at “Airport B” and employs the “Payment Service” with Visa or Master card.

Knowledge Integration

The aim of knowledge integration is to build corporate knowledge with VKCs and knowledge from repositories defined in the semantic policy framework. Because OWL is closely related to Description Logic, it has many features that are derived from the family of knowledge representation system [79]. OWL is therefore adopted for the representation of knowledge in this approach. Although OWL is supposed to be effectively implementable, its expressive power is limited. To solve this problem, SWRL extends the set of OWL axioms.

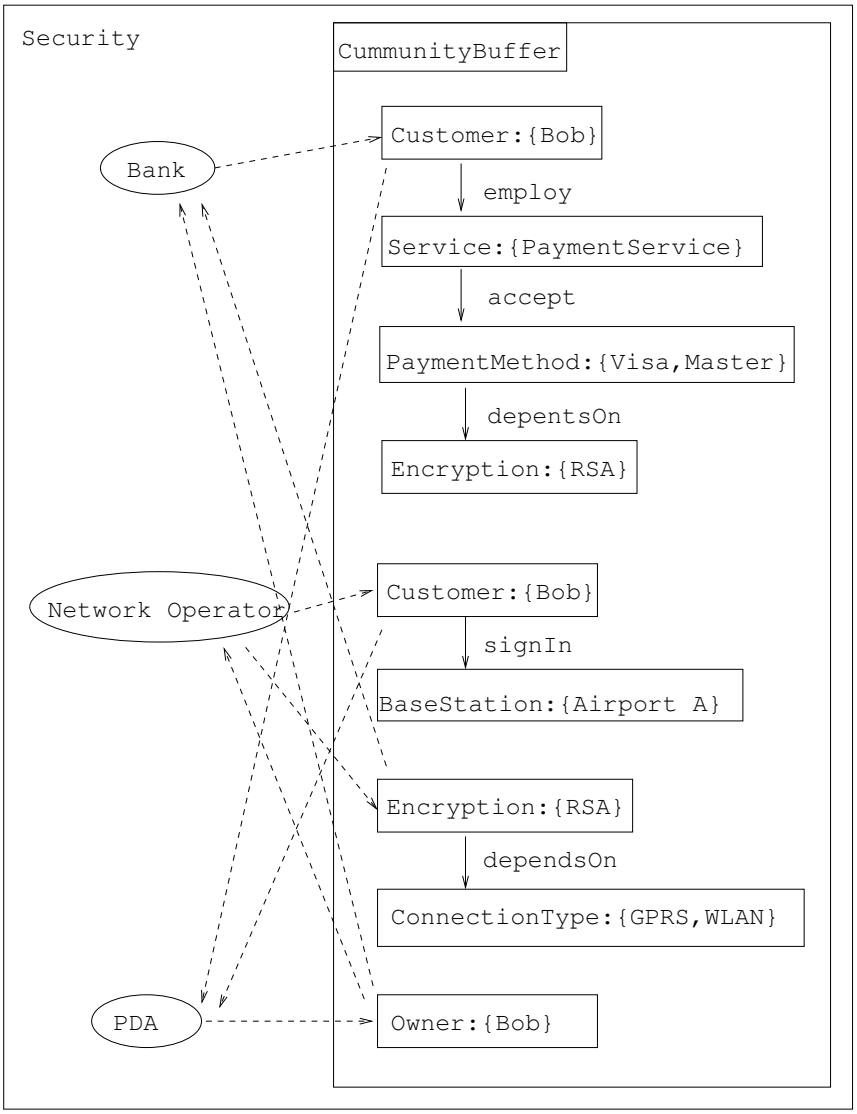

Figure 5.3: Agents who shared their knowledge within the community “Security”.

The following List 5.2 represents a simplified version of the PDA Agent’s knowledge after the Knowledge Integration.

Listing 5.2: Knowledge Integration of PDA agent

```
  <owl:Class rdf:ID="Encryption" />
  <owl:Class rdf:ID="PDA" />                                    2
  <owl:Class rdf:ID="Owner" />
  <owl:Class rdf:ID="ConnectionType" />                         4
  <owl:Class rdf:ID="Location" />
+ <owl:ObjectProperty rdf:ID="dependOn">                        6
```

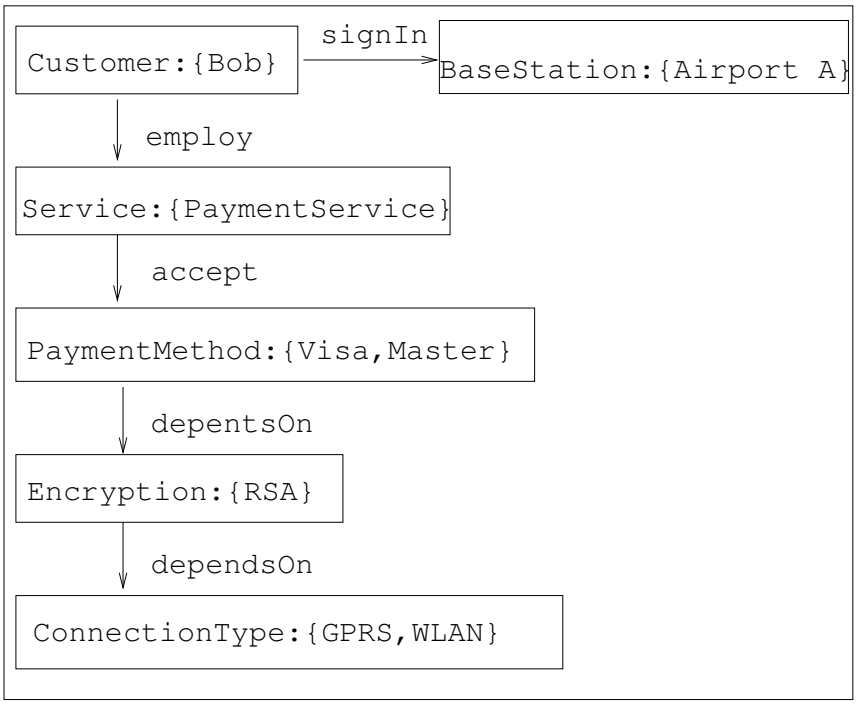

Figure 5.4: Knowledge evolvement of the agent “PDA” following the knowledge exchange in the community “Security”.

```
+ <owl:ObjectProperty rdf:ID="hasProperties">
+ <owl:ObjectProperty rdf:ID="hasConnectionType">                    8
+ <owl:ObjectProperty rdf:ID="useEncryptAlgorithm">
  <swrl:Imp rdf:ID="Rule-1">                                        10
  <swrl:body>
  <swrl:AtomList>                                                   12
  <swrl:IndividualPropertyAtom>
  <swrl:argument2>                                                  14
  <ConnectionType rdf:ID="GPRS" />
  </swrl:argument2>                                                 16
  <swrl:argument1 rdf:resource="#PDA" />
  <swrl:propertyPredicate                                           18
                    rdf:resource="#hasConnectionType" />
  </swrl:IndividualPropertyAtom>                                    20
  </swrl:AtomList>
  </swrl:body>                                                      22
  <swrl:head>
  <swrl:AtomList>                                                   24
```

```
<swrl:IndividualPropertyAtom>
<swrl:argument2>
<Encryption rdf:ID="DES" />
</swrl:argument2>
<swrl:propertyPredicate
                    rdf:resource="#useEncryptAlgorithm" />
<swrl:argument1>
<swrl:Variable rdf:ID="#PDA" />
</swrl:argument1>
</swrl:IndividualPropertyAtom>
</swrl:AtomList>
</swrl:head>
</swrl:Imp>
<PDA rdf:ID="FSC-PDA" />
<ConnectionType rdf:ID="WLAN" />
<Encryption rdf:ID="RSA" />
```

A prototype system that can generate XACML[3] policy based on the corporate knowledge has been implemented. Through the use of a predefined access control policy template with essential information of subjects and actions to be taken, XACML policy can be generated from the results of reasoning over corporate knowledge.

5.3.4 Trust Issues in Virtual Knowledge Communities

Recent research in the field of trust management shows new possibilities for achieving direct authorisation of security-critical actions. It is especially meaningful for Virtual Knowledge Communities that are characterised by a large number of agents operating in the absence of a globally available fixed infrastructure. Traditional security models depend on centralised administrative authorities such as the discretionary and mandatory models for access control [70, 98]. In Virtual Knowledge Communities, it is not feasible to adopt a centralised approach for the authorisation of actions due to the large scale existence of trust domains [107].

Two types of trust management approaches have been proposed as the solution to this problem: credential-based and evidence-based trust management. The credential-based trust management model and the concept of trust management

[3]OASIS eXtensible Access Control Markup Language, http://www.oasis-open.org

itself were first introduced by Matt Blaze et al. [15]. With this approach, trust is established through the delegation of privileges to trusted entities via the use of credentials and certificates. There are many models developed on the basis of the credential-based model [25,62,71,100]. However, a shortcoming of this approach is that it neglects the problem of risk and uncertainty.

In order to solve these problems, the first version of an evidence-based model for the SECURE project ([1] contains the final version of SECURE) was developed based on several research projects [2, 66, 77, 102, 119]. The work of the SECURE project defines a collaboration model based on previously defined trust and risk models. This collaboration model addresses issues related to the trust management lifecycle - in particular, the process of trust formation, evaluation and exploitation. It enables evidence-based, autonomous decision making and is capable of performing this without prior knowledge of the operating environment. Consequently, this enables collaboration between little known or completely unknown agents in VKC.

With the evidence-based model, trust relationships are established according to a pre-defined risk model and trust model. The risk model evaluates the risk that an agent's action may pose to the local domain whereas the trust model calculates the trust value of the agent. This value is primarily based on three aspects: observation, recommendation and reputation.

Obviously, the proof of an agent's trustworthiness is relies heavily on the use of evidence. The work of SECURE [107] has proposed some possible solutions for gathering evidence to establish the trust levels of agents. However, in the real world, evidence is usually distributed across multiple networks making it difficult to find. An agent may interact with other agents in different VKCs. Furthermore, the other agents can save the historical records of the interactions with the visiting agent and keep them for a certain period of time. Therefore, when the agent enters a new VKC, it can prove its trustworthiness to the new VKC peers by querying its old VKC peers for evidence based on its historical behavior and past interactions. Some difficulties, though, which may restrict or even prevent this include:

- Technical limitations of agents. For example, agents could range from a sensor to an intelligent robot equipped with a high-end CPU and huge memory. Technical characteristics like transmission range, processing power and storage capabilities could also affect the evidence gathering process.
- Technical impediments existing in the VKCs. Firewalls, for instance, can prove to be an obstacle when they prevent the creation or transmission of

evidence between internal and external entities.

– Privacy policies of witness agents. These policies have the potential to restrict the release of evidence information.

This section defines an evidence gathering model which is referred to as *trust brokering* [52]. This model is based on the work of the SECURE project. In this model, a new component named rust broker is added to the trust collaboration architecture. The trust broker works in conjunction with the other architectural components to facilitate the secure gathering of evidence across multiple VKCs. A series of trust brokering protocols are designed to incorporate the trust broker. The following security requirements are incorporated into the proposed evidence gathering model:

– Privacy - How can the recommender be prevented from revealing personal information?

– Data Integrity - How can recommendations (evidence) be protected from tampering when transmitted between agents?

The proposed “trust broker” model aims at providing secure evidence gathering techniques whilst, at the same time, ensuring that the privacy of agents is maintained.

Previous Work

The SECURE project’s work towards a collaboration model based on the trust management lifecycle was a major source of inspiration for our research.

In the first version of the SECURE collaboration model, the decision-making process is triggered by a collaboration request. This request includes the requester’s information and the context information. Firstly, the Entity Recognition module (ER [101]) of the SECURE trust engine recognises the requester, which may consist of a pseudonym for privacy protection. The ER module also plays an important role in the trade of privacy for trust [101] and decentralised mitigation of Sybil-like attacks [29] thanks to trust transfer [101]. The other modules of the SECURE trust engine evaluate the trust value with the help of two different resources: observation and recommendation. Recommendations should also be adjusted before they are evaluated. Both resources produce evidence for the evaluation of the trust value.

The following evidence-gathering options exist [107]:

- Ask known or unknown agents for possible recommenders of evidence.
- Ask neighbors, who may be easily accessible, for possible recommenders of evidence.
- Broadcast requests for recommendations.
- Ask brokers to suggest good recommenders.

The first approach cannot ensure the trustworthiness of recommenders. Consequently, this can lead to multiple security threats such as recommendation spamming. The second option has the disadvantage of not being able to guarantee the quality of a recommender. For the third option of broadcasting, this is often used as a last resort as it results in network flooding and intensifies the threat of eavesdropping.

EigenTrust [64] presents a distributed and secure method for computing global trust values based on Power iteration. The global reputation of an agent is determined by the local trust values assigned to the agent by other entities, weighed by the global reputations of the assigning entities. However, these distributed trust algorithms have not considered the technical limitations of entities and are only designed for use in P2P networks.

ENTRAPPED [55] is a novel approach that provides a mechanism for the distribution of trust information using Distributed Hash Tables to store the trust information of P2P environments. It also uses Certification Agencies to make Sybil attacks [29] expensive. This approach is designed to work in overlay networks, which enables the collection of all of the evidence. However, for networks such as ad hoc networks and sensor networks - which have different network topologies and cannot guarantee the requirements of overlay networks in P2P environments - an enhanced model is required.

Trust Broker Model

The Trust Broker Model provides an infrastructure for the secure access of trust information across different system boundaries.

The entire Gracia knowledge base is considered to be a *global domain* and categorise individual domains with domain of interest as *local domains*. The first reason for this is that the Gracia Knowledge Base consists of multiple diverse knowledge sources. This characteristic renders a unique approach infeasible. The second reason is that the sheer number of agents makes it impossible to maintain

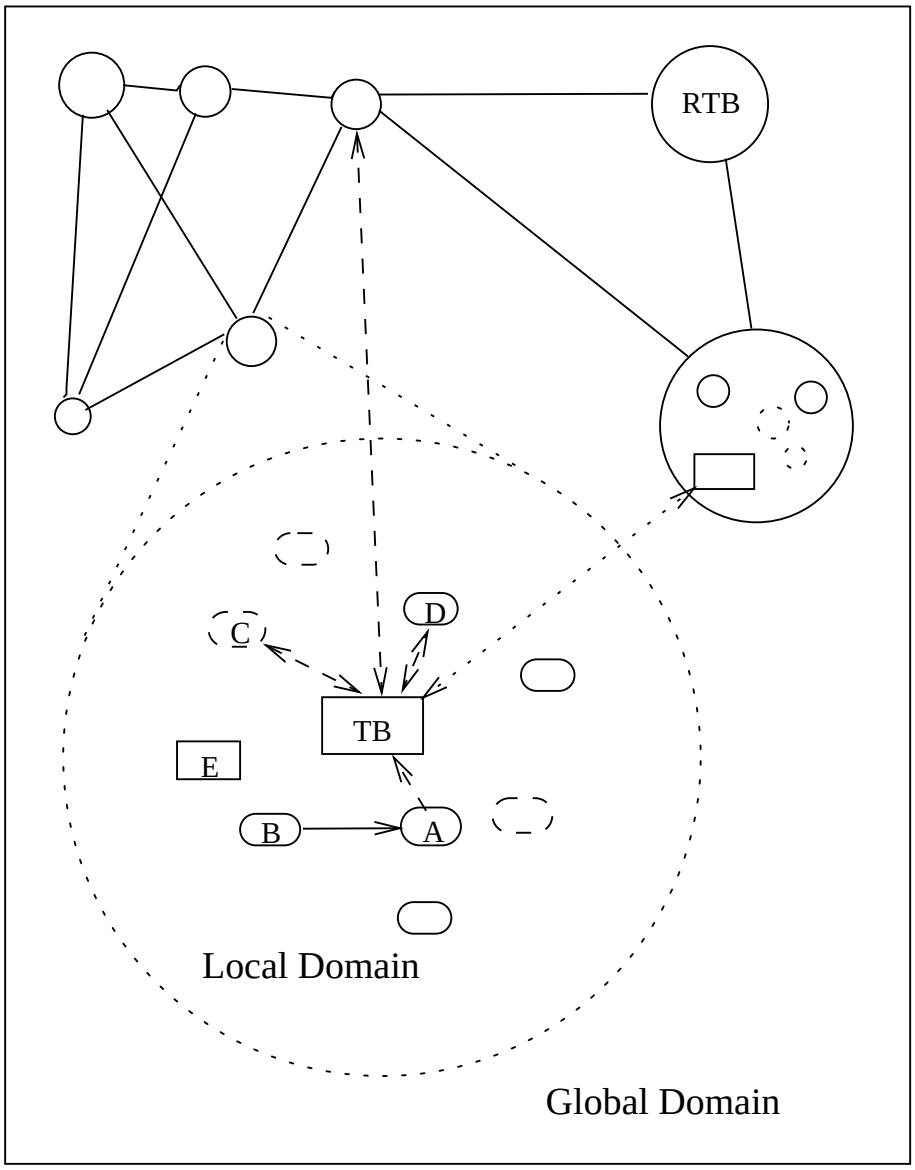

Figure 5.5: Global Domain and Local Domain

information in relation to every single one of them. However, grouping agents into local domains reduces these costs. All agents should be able to identify other agents within their local domain. An agent in a local domain views other local domains as remote domains. The identity of an agent may change when it moves to another domain. This concept of local domains follows real life occurrences. For instance, agents are commonly divided into groups on an organisational or commercial basis. In the context of our Trust Broker Model, such groupings can be represented as local domains. It is assumed that all local domains are connected

to each other and that no “isolated isles” are created by the division of the global domain.

As mentioned before, all agents differ in their capabilities such as their communication ranges or storage capabilities. We distinguish them using two criteria: (1) if they can communicate with other domains and (2) if the agent can perform the basic function of trust brokering. The agents that can fulfill both of these requirements are denoted as *strong agents*. All other agents are denoted as *weak agents*.

Trust Broker and Trust Broker Unit

Within each local domain there is an active trust broker which acts as a broker agent. The trust broker tracks all information relating to interactions between agents in the local domain. As an explanation, consider the following scenario: agent A, who wants to find a possible source of recommendation for agent B, sends a request with the name of agent B to the trust broker and the trust broker queries its repository for all agents that have performed the same action with agent B in the past and requests that these agents provide evidence about B.

Only strong agents in the domain can act as trust brokers. At any single moment, there is only ever one active trust broker. All other “potential” trust brokers in a local domain (and the active trust broker itself) are known collectively as the *Trust Broker Pool*. Should the active trust broker disconnect or fail to serve the request, a process called *Trust Broker Election* is carried out within the Trust Broker Pool to select a new trust broker. Each time an agent fails to communicate with the trust broker, it sends a signal for an election. When the number of calls in the domain exceed a predefined amount, an election takes place. Possible criteria for the delection of a new trust broker include:

- trustworthiness of a strong agent.
- strong agent’s policy (its own opinion about being a trust broker).
- strong agent’s ability (memory, CPU, network bandwidth, etc.).
- mobility of the agent (embedded device or mobile device).

The repository of all recorded actions is kept up-to-date through the use of the trust brokering protocols and a backup service operating within the Trust Broker Pool.

It is assumed that Trust brokers are trusted by all agents in the local domain. To provide privacy protection, all requests for evidence go through the trust broker in order to prevent the disclosure of the identity of the agent providing the evidence to the agent requesting the evidence.

The *Trust Broker Unit* (TBU) is a component that would be integrated into the SECURE trust engine [1]. The TBU enables: (1) strong agents to provide a brokering service to others and (2) weak agents to collaborate with the trust broker.

There are two types of TBU - weak TBU for weak agents and strong TBU for strong agents.

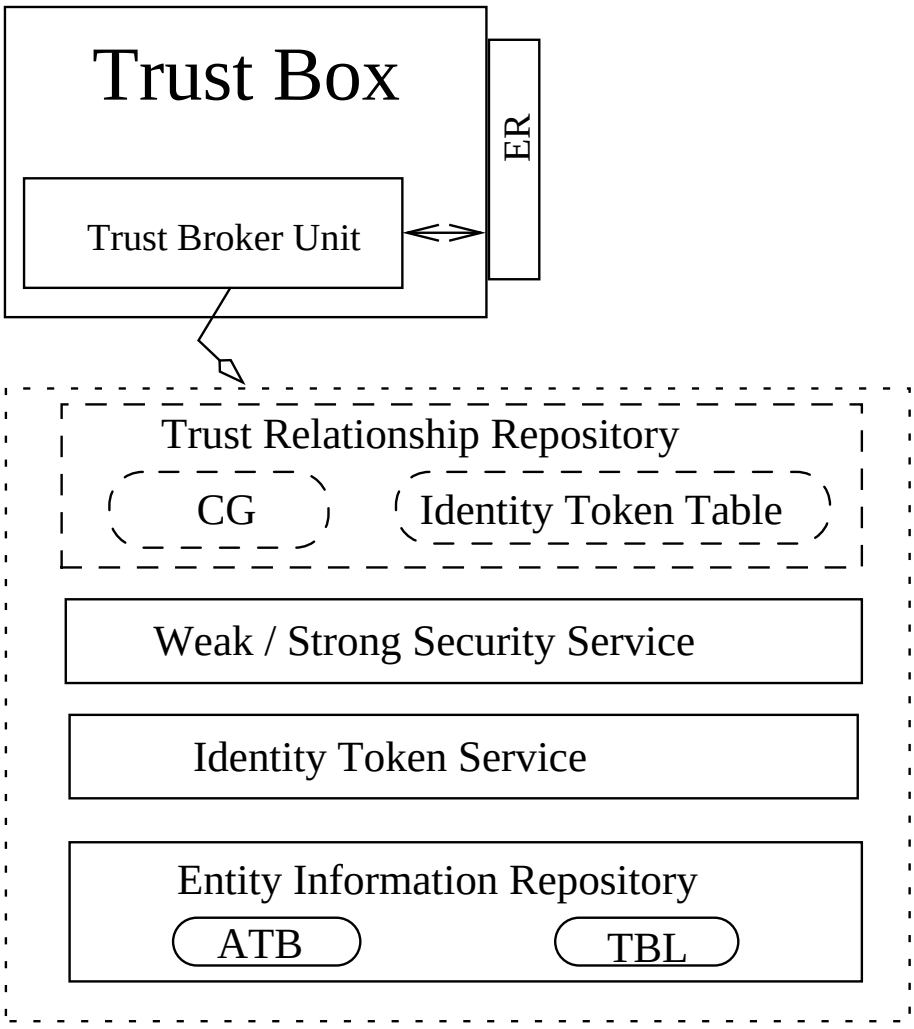

Figure 5.6: Trust Broker Unit

As illustrated in Figure 5.6, the weak TBU consists of three sub-components: *Agent Information Repository* (AIR), *Identity Token Service* (ITS) and *Weak Se-*

curity Service (WSS). The WSS provides basic security functions (such as performing hash functions, encryption, etc.) to agents whereas the ITS enables the creation of identity tokens and pseudonyms for agents. Pseudonyms are virtual identities for the identification of agents within a local domain and may change whenever an agent moves into other local domains. An identity token acts as a shared secret between the trust broker and the agent itself. The AIR exists in every agent and contains the following:

- Active Trust Broker (ATB) - The information from the active trust broker in the domain.
- Trust Brokers List (TBL) - A list of trust brokers that the agent has known or used (e.g. in cases such as when the agent comes from another domain or when the agent moves frequently from one local domain to another). It contains information regarding former trust brokers and the identity tokens which were used by the agent to interact with all of the trust brokers.

The strong TBU extends the functions of the weak TBU and consists of four subcomponents: *Trust Relationship Repository* (TRR), *Agent Information Repository* (AIR), *Identity Token Service* (ITS) and *Strong Security Service* (SSS). The SSS enables trust brokers to manage their keypairs. Furthermore, it provides services for the signature of evidence and validation of signatures on incoming evidence. This mechanism ensures that the integrity of evidence is maintained. The TRR provides recommendation sources and manages the identity tokens. More specifically, it contains two types of information:

- Collaboration Graph (CG) - This is a directed graph. The agents which have collaborated with other agents in the same local domain are referred to as the knots in the graph. The trust relationship, which is, for example, the result of a collaboration request from B to A, is referred to as a directed trust arrow from A to B. In the active TBU, the CG is updated each time a collaboration interaction occurs. However, in a deactivated TBU the collaboration graph is either empty or contains only redundant information for backup purposes.
- Identity Token Table (ITT) - This table keeps all records of known identity tokens in the local domain. The trust broker uses the table to verify requests for the signature of evidence.

The TBU is the technical foundation of the trust brokering model. The process of trust brokering is defined by the trust brokering protocols which are discussed in the next section.

Trust Brokering Protocols

A series of trust brokering protocols have been proposed in conjunction with the trust broker model to implement secure evidence gathering procedures. These procedures are as follows:

- Registration of a new agent to a local domain.
- Registration of an existing agent to a newly elected trust broker.
- Creating signed evidence.
- Gathering evidence from within the local domain.
- Gathering evidence originating from remote domains.

In all of the protocols, it is assumed that every trust broker possesses an embedded digital certificate issued by a certification authority. Additionally, it is assumed that trust brokers are somewhat more reliable than ordinary agents and that all agents in a local domain trust their trust broker.

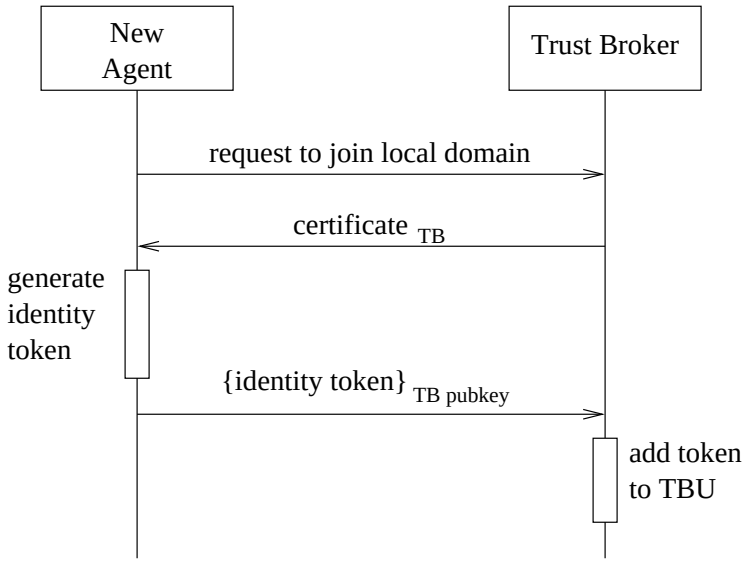

Figure 5.7: Registration of a new agent to a local domain

Registration of a new agent to a local domain

This process is executed when an agent enters a new local domain. The purpose of this protocol is to allow the new agent to register a new identity token with

its new trust broker. The identity token simply acts as a shared secret between the agent and the trust broker. It contains no personal identification information about the agent but instead, contains the agent's chosen pseudonym and a secret nonce. For privacy protection, whenever an agent enters a new local domain and creates a new identity token, both the pseudonym and the nonce should be changed.

Figure 5.7 shows the sequence of events performed in this protocol. To begin, the new agent sends the trust broker a request to register itself in the local domain. The trust broker replies to this request by sending its certificate to the new agent. On receipt of this certificate, the agent creates a new identity token before sending this token (encrypted with the trust broker's public key) to the trust broker. Once received, the trust broker verifies that the selected pseudonym is non-conflicting (it is assumed that pseudonyms within a local domain are unique) and then adds the token to its TBU.

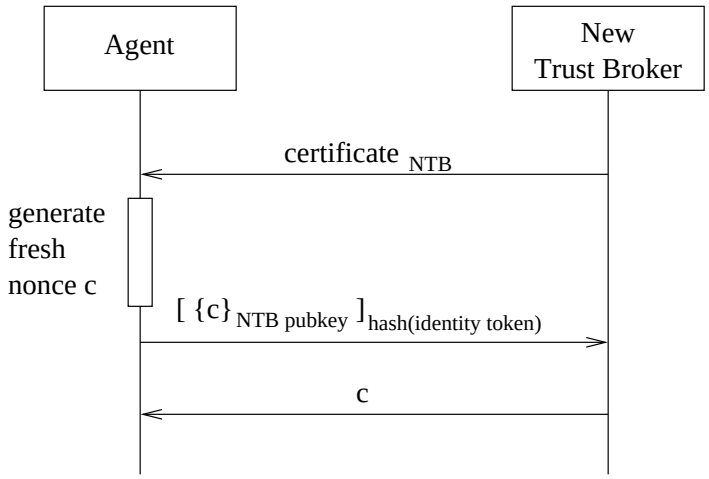

Figure 5.8: Registration of an existing agent to a newly elected trust broker

Registration of an existing agent to a newly elected trust broker

The objective of this protocol is to confirm the identity of a newly elected trust broker to an existing agent in the local domain. To achieve this, the agent issues a challenge to the new trust broker and then, based upon its response, the agent can determine whether or not the new trust broker is indeed the trust broker and not an attacker.

The sequence of message exchanges for this protocol is shown in Figure 5.8. It begins with the new trust broker sending its certificate to the agent. When this certificate is received, the agent creates a challenge (a fresh nonce), encrypts this using the supplied public key and then encrypts this encrypted data with the hash of the identity token. To respond to this challenge, the new trust broker must be capable of informing the agent of the value of the fresh nonce that it created. If the trust broker is able to successfully decrypt the challenge, then the agent can be sure that the new trust broker is indeed the true trust broker.

Creating signed evidence

To attribute data integrity protection to evidence when transported between agents, we have incorporated the use of signed evidence into the trust broker model. Although such evidence can still be read by everyone, this approach makes it tamper-resistant.

To create signed evidence, the agent first sends its evidence (encrypted, together with its identity token, the trust broker's public key) to the trust broker. The reason for the inclusion on the agent's identity token is to prevent other agents from pretending to be the authentic agent. Once received, the trust broker then adds this information to its collaboration graph before signing the evidence and sending it back to the agent. Figure 5.9 illustrates this process.

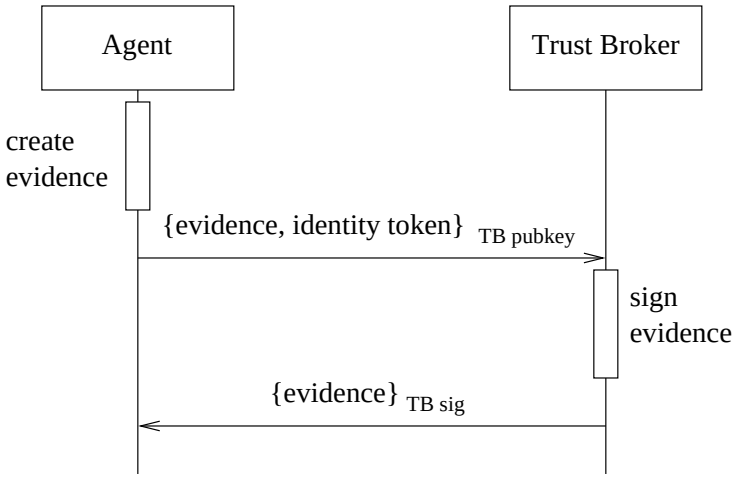

Figure 5.9: Creating signed evidence

Gathering evidence from within the local domain

This protocol, otherwise known as the local trust brokering protocol, is designed for operation solely within the local domain. It aims at locating possible recommendation sources within the local domain and retrieving evidence from these sources in a secure manner. Information on locating sources is provided in the CG managed by the trust broker. To explain this protocol, Figure 5.10 provides a visual representation of its message sequence. It depicts four participants - specifically, three agents and the trust broker. Agent 1 (E1) wants to perform an action with Agent 2 (E2). However, E2 first wants to ensure that E1 is trustworthy and therefore contacts the trust broker (TB) to gather evidence in regards to E1. Agent 3 (E3) is the evidence provider in this example.

The protocol follows the sequence below:

1. E1 sends a message to E2 requesting permission to perform action A.
2. To make a decision as to whether or not this action should be executed, E2 sends a request to TB asking for evidence on E1 based on action A.
3. TB searches the CG to find possible sources which could provide appropriate evidence.

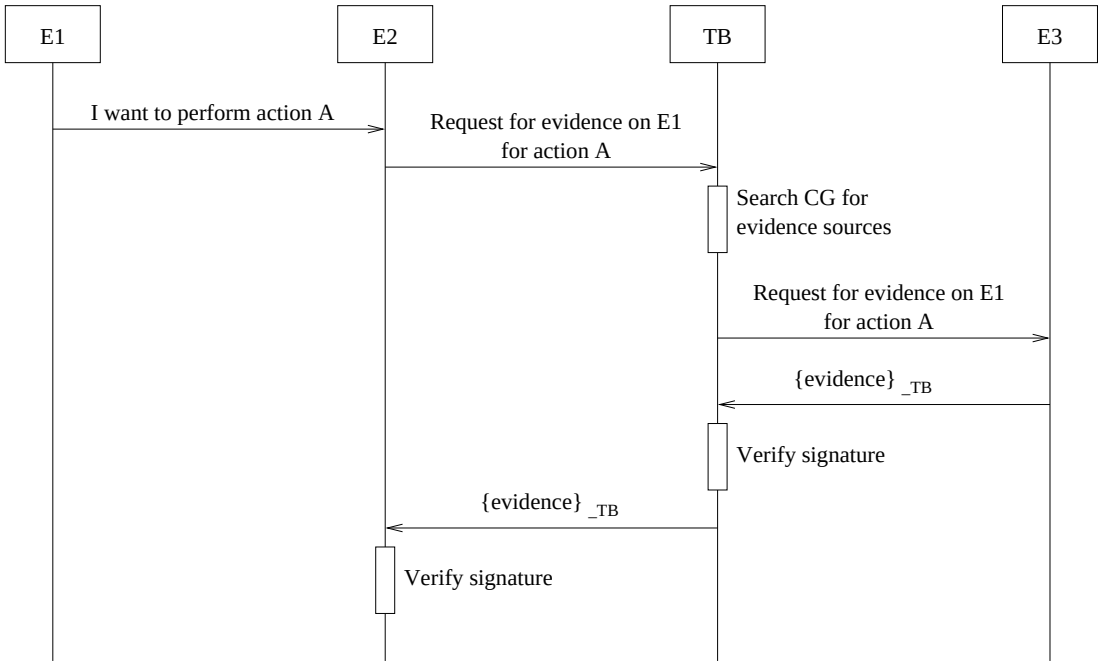

Figure 5.10: Gathering evidence from within the local domain

4. TB sends a request to E3 (found in the CG) to provide evidence on E1 for action A.
5. E3 sends the evidence requested (signed by TB).
6. TB verifies the signature on the evidence and forwards it to E2.
7. E2 verifies the signature on the evidence and uses it to make a decision on E1's request to perform action A.

The advantage of this protocol is that it provides privacy protection (by hiding E3's identity from E2) and data integrity protection for the evidence (preventing evidence from being tampered with by malicious agents).

Gathering evidence originating from remote domains

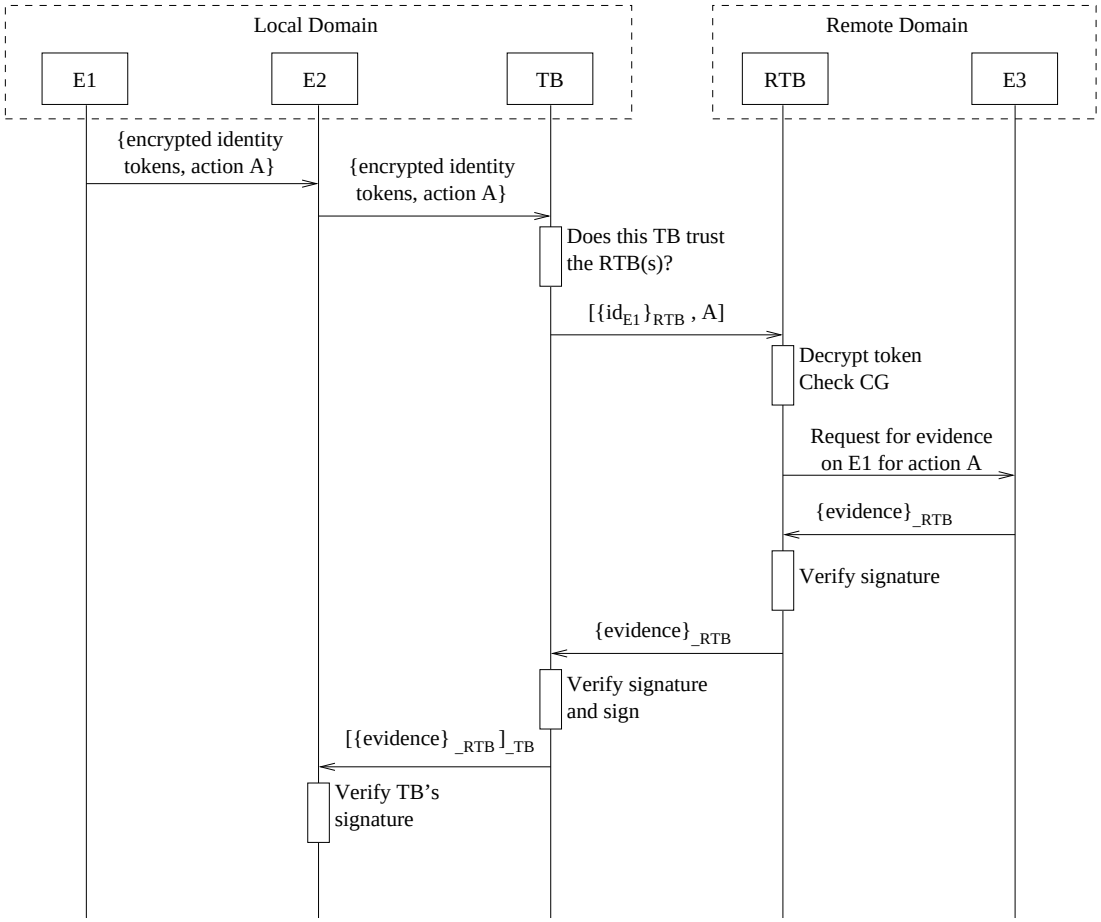

Figure 5.11: Gathering evidence originating from remote domains

This protocol, also called the remote trust brokering protocol, extends on the previous protocol to cater for the collection of evidence from remote domains. As before, the aim of the protocol is still the location of possible recommendation sources and the secure retrieval of evidence. However, the main complicating issue here is the involvement of multiple trust domains.

To show how this protocol works, Figure 5.11 illustrates the message flow between five participants. Agent 1 (E1), Agent 2 (E2) and Trust Broker (TB) are all within the same local domain whereas Remote Trust Broker (RTB) and Agent 3 (E3) are located in another trust domain. As with the previous protocol, E1 wants to perform an action with E2. However, in this scenario, E1 provides E2 with a list of possible remote sources to gather evidence from. Here, we assume that E1 still knows its old identity tokens with trust brokers from other local domains as well as the certificates from these trust brokers. In this example, RTB represents a former trust broker of E1.

The following is an explanation of each step in this protocol:

1. E1 sends a request to E2 to perform action A. Additionally, E1 sends a set of tokens for E2 to use to locate remote sources of evidence. This set of tokens basically consists of E1's old identity tokens encrypted with the public keys of the respective former trust brokers. The identities of these remote trust brokers are also sent with these tokens.
2. E2 receives E1's request and forwards the received tokens and information about the desired action to TB so that evidence can be gathered.
3. TB receives the encrypted tokens and decides whether or not to pass them onto their respective remote trust brokers. TB makes this decision based on its view of the web of trust. In other words, TB must decide if it trusts a particular remote trust broker. Following this, TB forwards only the tokens destined to trustworthy remote trust brokers (along with the action).
4. RTB, a trusted remote trust broker, decrypts the received token and checks its CG for possible recommendation sources for E1. Note that in this remote trust domain, E1 here denotes an old pseudonym. RTB then sends a request for evidence to E3 - the agent found in RTB's CG possessing evidence on E1.
5. E3 replies to RTB's request by sending evidence signed by RTB.
6. RTB verifies the signature on this evidence and forwards it to TB.
7. Once received, TB also verifies the signature on the evidence from RTB and then signs it again. The reason for TB also signing the evidence is be-

cause we assume that E2 does not possess RTB's certificate. Furthermore, it proves to E2 that TB considers the external evidence to be trustworthy. After signing the evidence, TB forwards it to E2.

8. E2 verifies TB's signature on the evidence and uses it to make a decision regarding E1's request to perform action A.

Evaluation Process

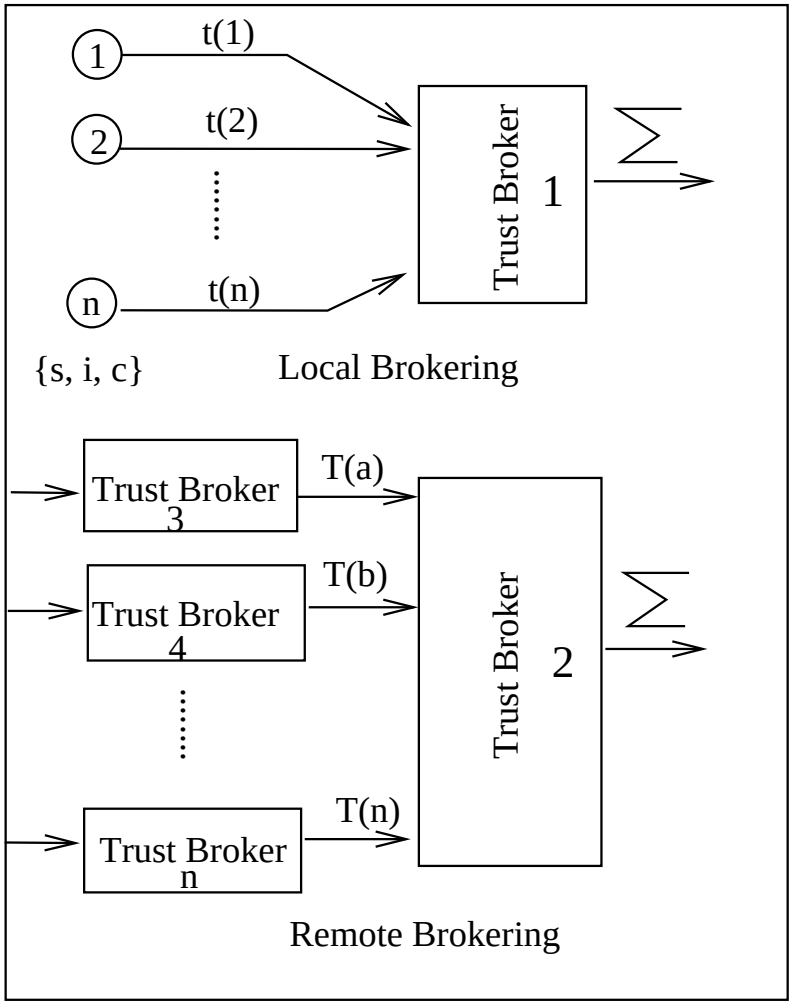

Figure 5.12: Evaluation Process

The recommendation, one kind of evidence, is transferred through the trust brokers. In this model, the trust broker can evaluate multiple recommendations to

derive (1) a combined result, (2) "attraction" for the trust value [107] or (3) recommendations representing a local domain. The output of the evaluation depends on the brokering protocol. Figure 5.12 shows the hierarchical architecture of the evaluation process.

The work of the SECURE project presents a $\{s, i, c\}$-triple value which can be used to represent evidence - the historical record of an interaction. The value s is the number of positive interactions(shared knowledge is true) and c is the number of negative interactions(shared knowledge is fake). The uncertainty is taken into account using i. The evidence collected by the trust broker in the local trust brokering protocol can be defined as:

$$R_{local} = \frac{\sum_{i \in N} \frac{s_i \times t_i}{s_i + i_i + c_i}}{\sum_{i \in N} t_i} \tag{5.1}$$

where t_i is the trust value representing the trustworthiness of agent i to the trust broker and it weighs the recommendation. It is used to measure the accuracy of the recommendation. This concept is similar to Meta-Trust [55]. In the remote trust brokering protocol, the evidence collected from other domains can be defined as:

$$R_{global} = \frac{\sum_{j \in M} \frac{\sum_{i \in N} \frac{s_i \times t_i}{s_i + i_i + c_i}}{\sum_{i \in N} t_i} \times T_j}{\sum_{j \in M} T_j} \tag{5.2}$$

$$= \frac{\sum_{j \in M} R_{local} \times T_j}{\sum_{j \in M} T_j} \tag{5.3}$$

where T_j here represents the trustworthiness of the remote trust brokers to the local trust broker.

The evaluation of recommendations is a hard process and requires more consideration in regards to performance and feasibility. We only propose a simple solution here. Further research is required to derive more efficient and feasible mechanisms to perform these evaluations.

5.4 Summary

This Chapter introduced the hybrid approach to knowledge representation in our framework. By combining the power of description logic and Horn-like Rules, we can do more complex reasoning above the Gracia Knowledge Base, which

enables the implementation of the knowledge-based access control in the next chapter.

This chapter also presented trust brokering protocols for the sharing of knowledge within the Gracia Knowledge Base. A Trust Broker Unit is installed to manage the evidence of knowledge sharing transactions. Additionally, protocols are designed to facilitate the evolution of the trust value/reputation by collecting evidence from both local and remote domains. The protocols also demonstrate the possibility of using the trust-based knowledge management method in setting up the Gracia Knowledge Base.

The next chapter describes the implementation of the Security Gateway and the Gracia Policy Framework within the Security Gateway. We also present results in terms of scalability, its robustness against context changes, the efficiency of the access control model and the communication overheads incurred.

Chapter 6

Implementation of the Security Gateway

6.1 Introduction

Managing and enforcing security requirements are key principles for successful implementation of web service-based system. A vast number of different technologies for modelling and representing security requirements exist, starting from the object-oriented model to the Semantic Web approach. Challenges arise when security-related business logic is directly imbedded into the application logic, which is a quite common practice. This has a highly negative impact on the maintainability of security requirements due to the fact that business logic tends to change quite frequently.

The usage of an inference engine using knowledge bases can greatly improve the maintainability in such cases. Such engines handle the execution and management of business rules and regulatory constraints from the business domain. The administration of this kind of knowledge should ideally be performed by domain experts or business analysts.

In this chapter, we provide an approach to the integration of the Gracia Knowledge Base (GKB) with an ontology inference engine-KAON2, accessible through a series of Web service based components. All the knowledge, stored in GKB and managed by a graphic user interface, is accessible via customised Web services on the top of GKB. The distinguishing feature of our approach is:

– the automated generation of security web service policies like WS-Policy and XACML from the knowledge base.

– the automated policy conflicts detection for a changing service-oriented system in the run-time.

The approach is a security gateway called **Gracia**, which consists of a policy framework and a set of web service-based interfaces for external components such as Process Engine, Rule Engine and Web Service Repository.

6.2 Reasoning and Querying in Rules and Knowledge Base

We use an inference engine for Rules and a Knowledge Base in our approach. There are a lot of choices in both the academic and industrial world. Some of these choices are mentioned in the following paragraphs.

Protégé[1] provides different ways in which the user can query the content of an ontology. The **QueryTab** is part of the Protégé system and allows the user to retrieve instances from the ontology that match certain criteria. The queries are defined following the pattern *(class, slot, operator, value)* and can be composed in a conjunction or disjunction of queries. The operator can be set according to the type of values that the slot may take. For instance, a slot of type *String*, besides the *equals*, allows other lexical operators such as *contains* or *begins with*. The query engine is built-into Protégé frames and uses the closed world assumption, i.e. it will only return the things that have been explicitly asserted in the knowledge base as true.

Another powerful way of executing queries in Protégé is the **Flora plug-in** which translates a Protégé-frames ontology together with the user-defined axioms into an F-Logic (Frame-logic) program. The queries are executed in a different process and the results are displayed as a list in the user interface of the plugin. The vision of Flora was the realisation of a system that offers a logic-based knowledge representation for frames with meta-modeling facilities and side-effects that can be used not only for knowledge representation but also as a complete programming language. Flora is implemented on top of XSB Prolog [2] - a Logic Programming and Deductive Database system - and is founded on F-Logic, HiLog and Transaction Logic [118]. Flora has special language constructs for representing frames

[1] http://www.protege.stanford.edu
[2] http://xsb.sourceforge.net/

with complex structures by allowing the nesting of definitions. Flora also supports a natural way of meta-modeling in the style of HiLog. One use of meta-modeling is to query the signatures of classes. The Flora plug-in supports the definition of F-Logic axioms as part of the ontology, which are then used in the execution of queries.

Mandarax[3] is an open source java library for inference rules. This includes the representation, persistence, exchange, management and processing (querying) of rule bases. The main objective of mandarax is to provide a pure object oriented platform for rule based systems. Mandarax is a pure Java implementation of a rule engine. It supports mutiple types of facts and rules based on reflection, databases, EJB etc, supporting XML standards (RuleML 0.8). It provides a J2EE compliant inference engine using backward chaining.

JBoss Rules [57] (aka Drools 3.0) is a Rule Engine implementation based on Charles Forgy's Rete algorithm tailored for the Java language. Drools is written in Java, but is able to run on Java and .Net. JBoss Rules is designed to allow pluggeable language implementations. The language supported by JBoss Rules is called Drools Rule Language (DRL). The rule engine supported in JBoss Rules complies with the standard Java Rule Engine API (known as JSR94 [56]). Analogously to the approaches presented so far in this chapter, in this approach rules are also triggered by means of invoking methods on the rule engine from Java code. Therefore, it suffers again from the problems imposed by low-level and crosscutting rule connections.

In the context of reasoning with OWL, four different approaches can be roughly distinguished [115]:

- The inference calculus implemented in tableaux-based provers for DLs: They are available via systems like Pellet [103], RacerPro [41], or FaCT++ [110]. They implement a conceptually sound as well as complete approach for which many optimisations are so far known. Unfortunately, complete instance reasoning still requires expensive computations.
- The approaches to transform an OWL ontology into a disjunctive datalog program and to utilise a disjunctive datalog engine for reasoning: It is implemented in KAON2 [85]. This allows for fast query answering due to well-known optimisation techniques from deductive databases such as magic set transformation.

[3] http://mandarax.sourceforge.net

- Other systems using a standard rule engine to reason with OWL: for example OWLIM [67] or OWLJessKB. This is fast and easily tunable to different language fragments by simply manipulating the rule set. However, this procedure is known be incomplete and a drain on resources when filled with large amounts of implicit knowledge (due to their materialisation strategy.)
- Hybrid approaches: such as QuOnto [3], Minerva [120], Instance Store [11], or LAS [23] combine an external reasoner (often a tableaux-based system) with a Database system. This enables the processing of large data volumes due to secondary storage mechanisms. On the other hand, this combination only allows for a very limited language expressivity.

Table 6.1 shows a detailed comparison of the reasoners.

	Ontology Language	Supported Logic	RDF Query	Rules
KAON	a proprietary extension of RDFS	DL	N/A	N/A
KAON2	OWL-DL, OWL-Lite	DL	SPARQL	SWRL
FACT	OWL-DL	DL	N/A	N/A
FACT++	OWL-DL	DL	N/A	N/A
Racer	OWL-DL	DL	nRQL	SWRL
Cerebra	OWL-DL	DL	Xquery	SWRL
cwm	OWL-Full	DL	N/A	N3
Euler	OWL-Full	DL	N/A	N3
Surnia	OWL-Full	FOL/DL	N/A	N/A
Jena/HP	OWL-Full	DL	ARQ/SPARQL	N3
Pellet	OWL-DL,OWL-Lite	DL	SPARQL	SWRL
F-OWL	OWL	F-logic,DL ,Transaction Logic	N/A	N3
E-Wallet	N/A	N/A	N/A	N/A
Sesame	N/A	N/A	SPARQL	N/A
DLP	N/A	DL	N/A	N/A

Table 6.1: Reasoners

Considering the performance and ability in dealing with the large ABox, we have chosen KAON2 as the inference engine in our approach. KAON2 provides an integrated API for the reading, writing, and management of OWL DL ontologies extended with SWRL rules. Currently, OWL RDF and OWL XML file formats are sup-

ported. A built-in reasoner for OWL DL (except nominals and datatypes) extended with a DL-safe subset of SWRL is inside the KAON2 framework and reasoning is based on novel algorithms, which reduce an OWL ontology to a (disjunctive) datalog program. These algorithms allow KAON2 to handle relatively large ontologies with high efficiency. Its performance compares favourably to other state-of-the-art OWL DL reasoners. KAON2 supports the answering of conjunctive queries expressed in SPARQL. SPARQL is a query language for the RDF developed by W3C [93].

6.3 Security Gateway

The functionality of the Security Gateway is naturally divided into two parts: the run-time security function, also called policy enforcement, and the management function. The enforcement function controls run-time access of requestors to protected web services and also SOAP messenge-level security (Integrity and Encryption) , according to ontology-based Gracia policies, attributes of subjects, objects and operations. The management function provides mechanisms for the manipulation of Gracia Policy and Gracia Knowledge Base e.g. semantic annotations of rules and domain ontologies.

The enforcement function involves several architectural components and nodes. A subject of access is a web service client that invokes protected web services to access data or to affect the real world. The client accesses the web services through networks. A Security Gateway mediates access to the protected web service and enforces rules of corresponding security policies and external regulatory rules. The Security Gateway must evaluate all requests, correctly evaluate semantic profiles and policies an be incorruptible and nonbypassable. A Gracia policy is a combination of ontology and rules. Subject and object descriptors are well known patterns that provide access to attributes of subjects and objects of access. In Gracia, we specialise the descriptor pattern into semantic profiles for users, data, web services, policies and context. Figure 6.1 illustrates a control flow for the enforcement function that is driven by the Security Gateway.

Let us consider a simple example in order to reveal the roles of the above described components and activities that comprise the algorithm of the enforcement function. In our end-user scenario in section 3.2.3 passengers are able to access to the printer in their waiting zone using a PDA. A client sends a SOAP/XML request via Bluetooth to a web service in the control room, which controls a printer

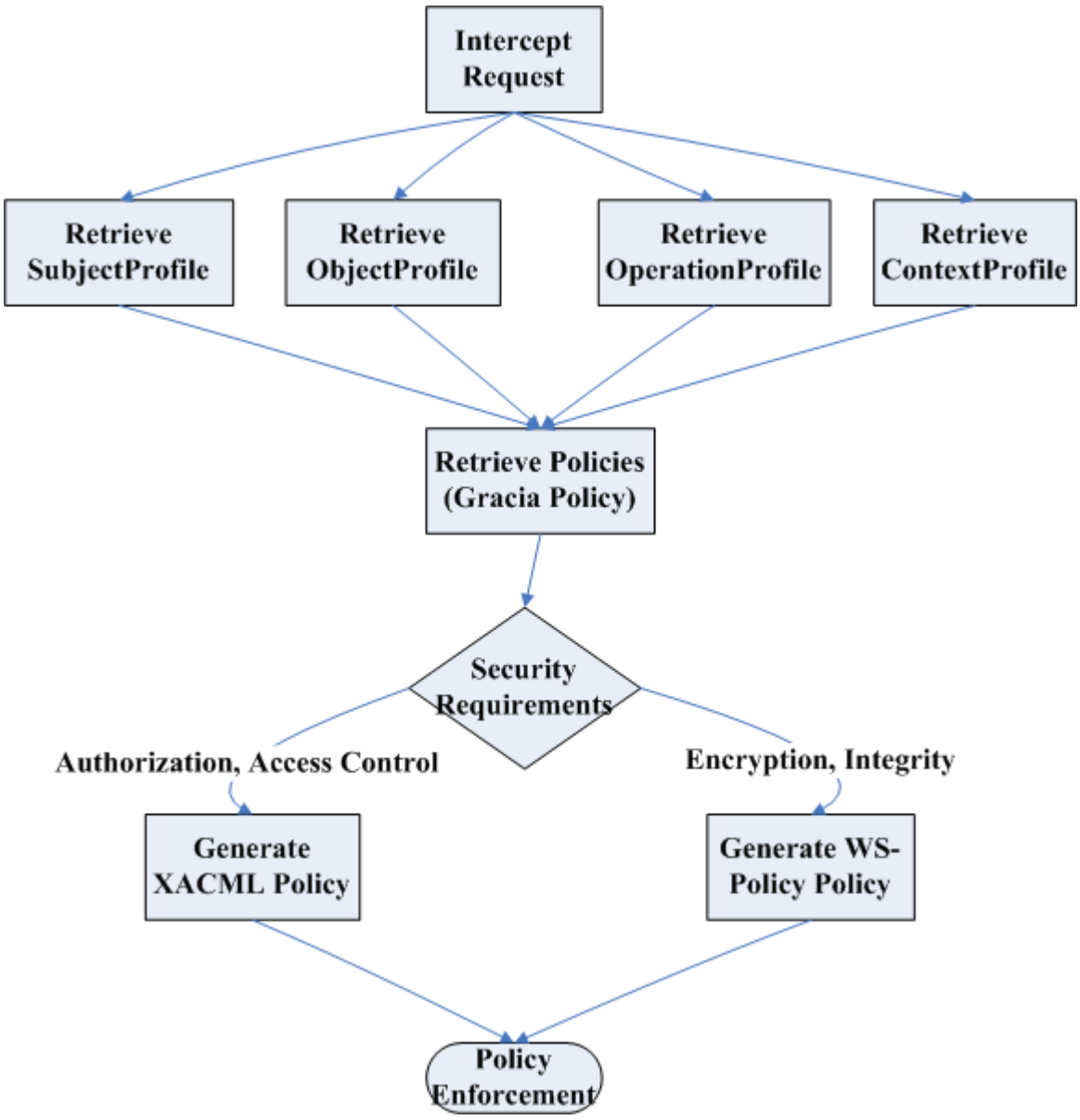

Figure 6.1: Workflow of Security Gateway

in the airpot. A Security Gateway intercepts this request. After that, the Security Gateway initiates the process of request evaluation. The Security Gateway extracts a URI of the web service's operation, supplied credentials and passenger context information from request's header. These are input parameters for the retrieval or creation of semantic profiles for the passenger and operation. The Security Gateway can create or retrieve the semantic profile for the object based on the input SOAP/XML message and its WSDL description. After collecting semantic profiles of the user and operation, the Security Gateway retrieves applicable

Gracia policies. In our example, this retrieval uses information regarding location within the airport and the airport printer service policy from the Gracia Knowledge Base. According to different security requirements, XACML or WS-policy policies will be generated and enforced by the Gracia policy framework. In the following section, we will introduce these components in detail.

6.3.1 Design of Gracia Policy Framework

The knowledge base which will be used in the Gracia Policy Framework will be defined in OWL and SWRL. The decidable subset of OWL, OWL-DL, is used to represent the information in the system. The rules used to define the access rights of individuals to web services endpoints are written in a semantically aware language. SWRL was chosen as the rule language for our system. The main advantage of using SWRL is its ability to provide support for complex relationships between properties, therefore extending the expressiveness of what can be defined in OWL-DL. The subjects of the rules will be defined in the knowledge base, as described in Section 5.3.1. Rules may be written to protect access to two specific resources; the web service endpoints and the information they return.

Since the Gracia knowledge base is essentially written in OWL-DL, we can use an OWL-DL reasoning engine to evaluate the rules. To enforce authorisation rules in our approach, we use a combination of SWRL and OWL-DL. The main advantage of SWRL, which was discussed in section 2.3.2, may also present a new problem domain since it extends the expressiveness of OWL-DL beyond the decidable subset of OWL. There are two ways to overcome this problem. Either restrict the expressiveness of SWRL and use an existing reasoning engine such as Pellet or Racer, or use a reasoner such as KAON2 which has been extended to handle SWRL rules. We have chosen to use the extended reasoner KAON2. To ensure a decidable result from our authorisation we will restrict how the user writes the rules rather than restricting the language itself.

Authorisation decisions will be made with respect to a Web Service endpoint. This may lead to one of three results: 1)full access, 2)no access and 3)limited access. A requester being granted limited access implies that they can access the endpoint but they will potentially return sensitive information that they do not have access to. When this happens, the response associated with the initial request is examined and the information returned must be legally accessible by the requester. Any information that is defined as illegal for the requester will be pruned

before the response is sent. The level of pruning is defined by the user who is responsible for the update of the rules and knowledge base.

6.3.2 Implementation of Gracia Policy Framework

The UML component diagram (Figure 6.2) depicts the architecture of the prototype for the policy enforcement mechanism. The internal structure of the security gateway consists of the Gracia knowledge Base, Gracia policy framework and the reasoner (query processor) provided by the KAON2 of SPARQL queries. The Gracia knowledge Base has in-memory knowledge base (rule sets) in the form of an ontology model. All ontologies are accessible via HTTP.

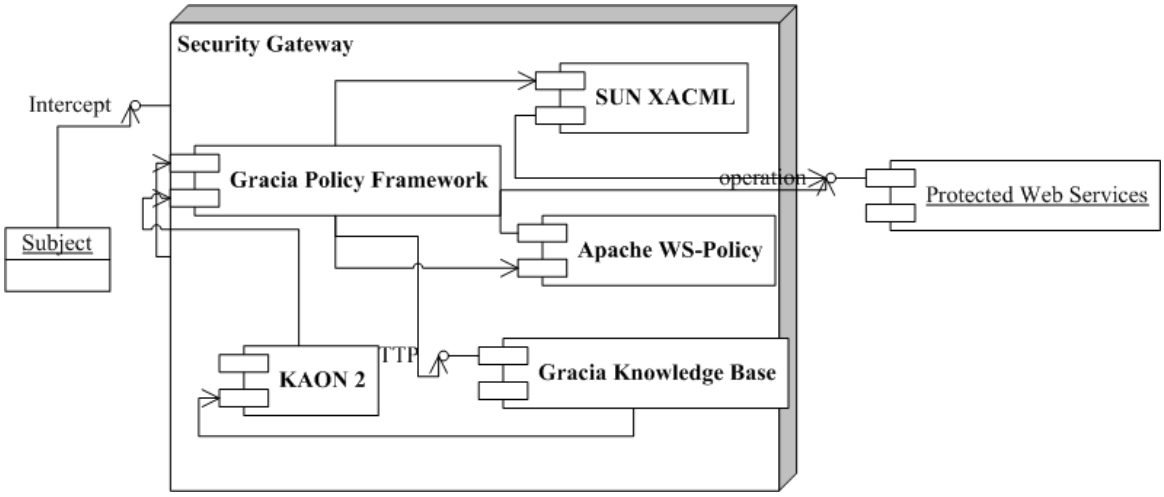

Figure 6.2: Component Diagram of Security Gateway

The development environment consists of several interrelated elements. Java 2 standard edition development kit version 1.5[4] is a programming language and platform that was chosen for the prototyping of research ideas. KAON2 is a semantic web framework with an SPARQL processor. Eclipse[5] is an open source community that produces extensions with a myriad of plugins, and it is also called integrated development environment (IDE). The latest version of the IDE is 3.2. The Eclipse Test & Performance Tools Platform (TPTP)[6] project consists of four subprojects, one of which provides tools for tracing and profiling java applications

[4]http://java.sun.com/j2se/1.5.0/
[5]http://www.eclipse.org
[6]http://www.eclipse.org/tptp/

for further analysis of performance. The Protégé is the most appropriate tool for the creation of defined Gracia ontologies in the RDF/XML exchange syntax of OWL. The Protégé is an open source and free ontology editor with a number of plugins for editing (Protege-OWL) and visualising OWL ontologies. In this approach, Tomcat 6.0 [7] is used as a Web and Application server. Apache Axis 1.4 [8] is a SOAP engine used as a Tomcat container for web services transactions. SUN XACML[9] is used to enforce the XACML policy generated by the policy framework. Apache WS-Commons Policy 1.0[10] is an implementation of the WS-Policy specification and is used to enforce WS-Policy specification in our approach.

The security functions designed and implemented as part of the security gateway are built in Java using Apache Axis as the SOAP implementation. The core encryption and decryption engine is developed using Apache's Web Service Security for Java (WSS4J)[11] implementation of the WS-Security specification from OASIS. It adheres to the W3C specification from XML-Encryption. The signing and signature verification engine is also developed using WSS4J and adheres to the W3C specification for XML-Signature. The key management is built according to the XML Key Management specification(XKMS)[12].

6.3.3 Design and Implementation of Management Tools

Management Tools consists of two tools that allow security responsible people to define and manage security policies and the knowledge base at the level of the business domain. The policies regarding access control for data, record retention (archiving of data, exporting/importing of data, deletion/insertion/modification of data), and transmission of data between systems are supported by our framework. The knowledge base is based on the VKC approach.

Management Tool comes with two user interfaces, one for knowledge definition and the other for policy definition. The knowledge definition GUI 6.3 is used by a domain expert who understands the underlying data and entities and abstracts attributes and methods out of data entities.

[7] http://tomcat.apache.org/
[8] http://ws.apache.org/axis/
[9] http://research.sun.com/projects/xacml/
[10] http://ws.apache.org/commons/policy/index.html
[11] http://ws.apache.org/wss4j
[12] http://www.w3.org/TR/xkms/

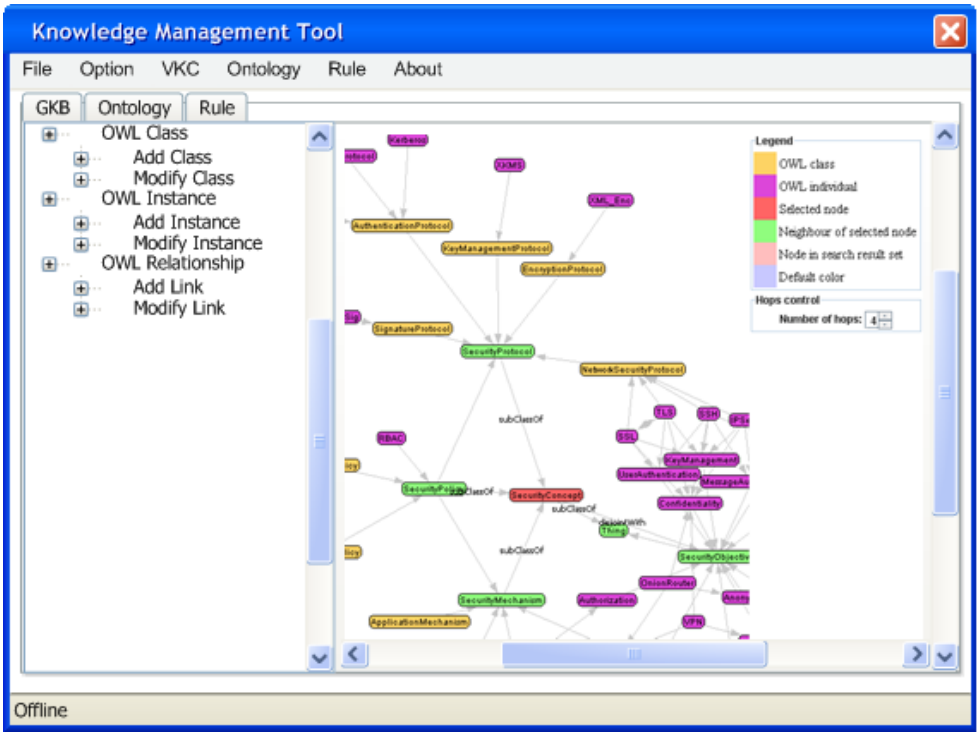

Figure 6.3: Knowledge Management Tool

To visualise the Gracia Knowledge Base, OWL2Prefuse[13] is embedded into the tool. OWL2Prefuse is a Java package that creates Prefuse[14] graphs and trees from OWL files. It takes care of converting the OWL data structure into the Prefuse datastructure. This makes it easy to use the Prefuse graphs and trees in our management tools. The following code helps with the integration of Prefuse graphs into our approach.

Listing 6.1: Prefuse Code Example

```
public GraphPanel createGraphPanel(String p_OWLFile){
          OWLGraphConverter graphConverter =                    2
                        new OWLGraphConverter(p_OWLFile,
                           true);
          Graph graph = graphConverter.getGraph();              4
          GraphDisplay graphDisp = new GraphDisplay(graph, true);
          GraphPanel graphPanel =                               6
              new GraphPanel(graphDisp, true, true);
          return graphPanel;                                    8
```

[13] http://owl2prefuse.sourceforge.net
[14] http://prefuse.org

```
        }
```

The policy definition GUI6.4 closely resembles that of one based on natural language; it can be used by any person for the definition of policies, without any knowledge regarding the idiosyncrasies of the technical domain. It automatically detects the conflicts, if any, between polices whilst defining them.

The tool automatically ensures the accurate and timely execution of the business rules and policies as and when the situation arises (based on events, such as database events, transactional events, external events and temporal events). If the policy demands it, the policy execution engine can even intelligently defer the execution of the policy to a more opportune moment.

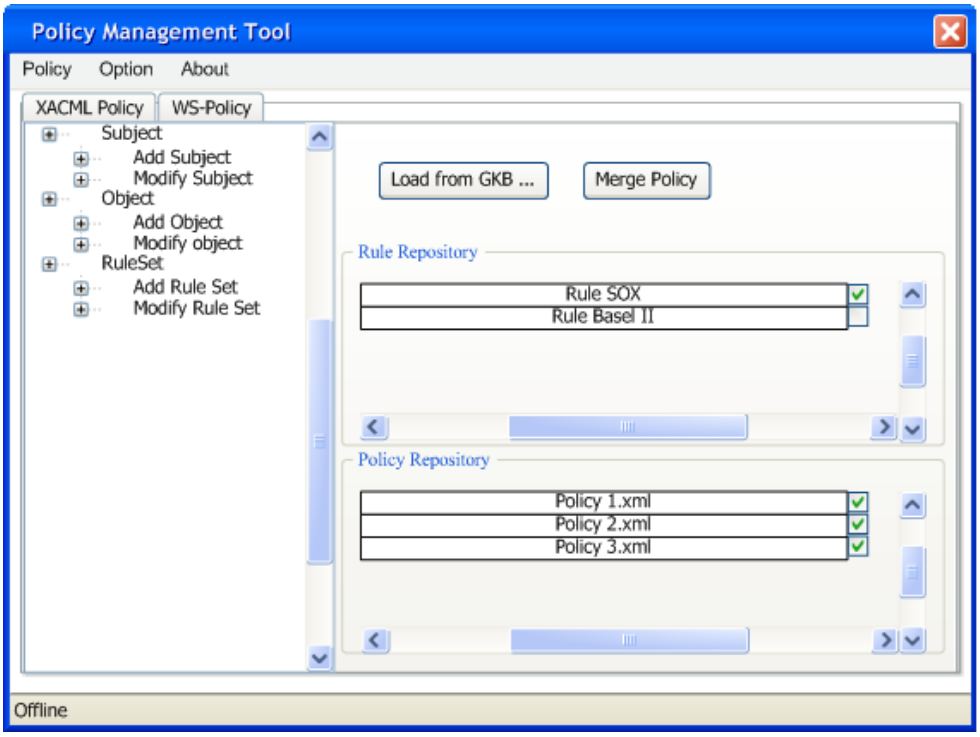

Figure 6.4: Policy Management Tool

6.4 Evaluation

We piloted the reasoning procedure to test the performance of our security gateway. For the programming and testing performance we used tools from Eclipse. The Eclipse Test & Performance Tools Platform (TPTP) project consists of four

subprojects, one of which provides tools for tracing and profiling java applications for further analysis of their performance.

The internal structure of the security gateway has an in-memory knowledge base (GKB) in the form of an ontology model. All ontologies were placed on the web server and were accessible via the HTTP protocol. The fastest response time of the reasoning process corresponded to the simplest policy and domain ontologies. The policy ontology consisted of one class for web service clients with one user, one class for protected objects with one individual and one class for web service operations with one operation. All RDF statements were loaded into the KAON2 workspace during the start-up process. The SPARQL query corresponded to the authorisation rule for policies defined using privilege statements only.

The average cumulative CPU time for the process was 0.57 seconds. The hosting computer was an FSC Laptop with an Intel Centrino 1.60GHz CPU, 512MB of RAM and OS Microsoft Windows XP Professional version 2002 with Service Pack 2.

6.5 Summary

We implemented a research prototype of the security gateway for web service protocols and are currently in the process of field testing it using a knowledge base comprising of a small set of rules and limited ontology models. We are currently working on real-world experiments and on further simulation studies to check the behavior of our security gateway. These experiments additionally simulate web services transaction events and work loads which do not occur on a daily basis. Additionally, we are investigating some ideas regarding the ways in which management tools can exercise a more comprehensive management of the knowledge base. Firstly, we will enhance the trust management in such a way that it can manage the trust value of every stakeholder in the SOC. Lastly, we will work on handling implicit knowledge and complex rules.

This prototype demonstrates the possibile use of a DL reasoner like KAON2 to support the decision making process in both the access control scenario and web services policy conflicts detection scenario.

Chapter 7

Conclusion

This chapter provides a summary of the work presented in this thesis and discusses what has been achieved. We conclude by indicating future work stemming from the results of the current approach presented in the previous chapter.

7.1 Review of Achievements

The introduction to this thesis identified the contributions this work has made to the field of policy based security management. In this section we will re-iterate and summarise this evidence.

- **Formal representation of policies and domain knowledge**: In Chapter 4 we have presented a formal semantic language called Gracia Policy (GPL) Language that can be used to represent security policies of web services, business rules and domain knowledge. The GPL is a combination of OWL and SWRL, thus, it is based on both Description Logic and Logic Programming. We have also shown how the GPL can be mapped to other standardised web service languages like XACML and WS-Policy. The mapping also enables the automated generation of a low-level execution policy(XACML, WS-Policy) from the high-level semantic policy (GPL).
- **Hybrid knowledge representation and integration of security information**: In Chapter 5, we presented a hybrid approach to knowledge representation called Gracia Knowledge Base (GKB). The GKB uses the Virtual Knowledge Community (VKC) concept for the integration of knowledge from different stakeholders into SOC. This knowledge sharing process was demonstrated by an e-payment example.

- **Trust Brokering Protocols for Trust Management in the Virtual Knowledge Community**: In Chapter 5, we also defined a series of trust brokering protocols designed to broker the evidence of knowledge sharing activities in the VKCs.
- **Tool Support**: As described in Chapter 6, we developed a prototype to prove the possibility of building security functions with semantic web technology. This prototype is a security gateway for web service protocols. We use a KAON2 reasoner to query the GKB and policies in GPL can be converted into XACML or WS-Policy by a policy framework, which works as a web service component inside the security gateway. Management tools have also been designed to enable easy administration of policies and knowledge.

7.2 Future Work

The formal semantic approach to the security gateway in this thesis is by no means complete. This section presents the work that we feel is fundamental to the future development of semantic policy-based security management. This work can be categorised into the following fields:

- **computational complexity**: The computional complexity of our approach increased rapidly whilst the Rule-Box and ABox of the GKB grew bigger. The efficiency of the system should be optimised by using efficient algorithms.
- **Policy analysis**: One of the main shortcomings in our policy framework is the ommision of a method for detecting and resolving policy conflicts. We hope to extend the policy validation technique to support this vision.
- **Tool Implementation**: In addition to improving the current user interface, we also propose to extend the tool to include support for adapting the VKC-based knowledge management system.

Appendix A

Gracia Policy Language

Listing A.1: WS-Policy in Gracia Policy Format

```
<rdf:RDF>
<!-- Ontology Information -->
  <owl:Ontology rdf:about="">
    <owl:imports>
      <owl:Ontology rdf:about="&WS-PolicyAssertion.owl;" />
    </owl:imports>
  </owl:Ontology>

<!-- Classes -->
  <owl:Class rdf:about="#ExamplePolicy">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:complementOf>
              <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                  <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                      <rdf:Description rdf:about="#policyClass1" />
                      <rdf:Description rdf:about="#policyClass2" />
                      <rdf:Description rdf:about="#policyClass3" />
                      <rdf:Description rdf:about="#policyClass4" />
                    </owl:intersectionOf>
```

```
              </owl:Class>
            </owl:unionOf>
          </owl:Class>
        </owl:complementOf>
      </owl:Class>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
              <rdf:Description rdf:about="#policyClass3" />
              <rdf:Description rdf:about="#policyClass4" />
            </owl:intersectionOf>
          </owl:Class>
          <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
              <rdf:Description rdf:about="#policyClass1" />
              <rdf:Description rdf:about="#policyClass2" />
            </owl:intersectionOf>
          </owl:Class>
        </owl:unionOf>
      </owl:Class>
    </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#elementClass0">
 <rdfs:subClassOf
    rdf:resource="&WS-PolicyAssertion.owl;#SecurityToken" />
 <owl:equivalentClass>
   <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
       <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
           <owl:Restriction>
             <owl:hasValue
                rdf:language="EN">wsse:Kerberosv5TGT</owl:hasValue>
```

```
            <owl:onProperty rdf:resource="test#hasTokenType" />
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#elementClass2">
  <rdfs:subClassOf
     rdf:resource="&WS-PolicyAssertion.owl;#Algorithm" />
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:hasValue
                 rdf:language="EN">wsse:AlgSignature</owl:hasValue>
              <owl:onProperty rdf:resource="test#hasType" />
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:hasValue rdf:language="EN">
              http://www.w3.org/2000/09/xmlenc#aes</owl:hasValue>
              <owl:onProperty rdf:resource="test#hasURI" />
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

```
<owl:Class rdf:about="#elementClass3">
  <rdfs:subClassOf
     rdf:resource="&WS-PolicyAssertion.owl;#SecurityToken" />
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:hasValue
                 rdf:language="EN">wsse:X509v3</owl:hasValue>
              <owl:onProperty rdf:resource="test#hasTokenType" />
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#elementClass4">
  <rdfs:subClassOf
     rdf:resource="&WS-PolicyAssertion.owl;#Algorithm" />
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:hasValue rdf:language="EN">
              http://www.w3.org/2001/04/xmlenc#3des-cbc</owl:hasValue>
              <owl:onProperty rdf:resource="test#hasURI" />
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
        <owl:Class>
```

```
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:hasValue
                 rdf:language="EN">wsse:AlgEncryption</owl:hasValue>
              <owl:onProperty rdf:resource="test#hasType" />
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="#errorType" />
<owl:Class rdf:about="#policyClass0">
  <rdfs:comment>&lt;SecurityToken>
                wsse:Kerberosv5TGT
            &lt;/SecurityToken></rdfs:comment>
  <rdfs:subClassOf rdf:resource="#errorType" />
</owl:Class>

<owl:Class rdf:about="#policyClass2">
  <rdfs:subClassOf
     rdf:resource="&WS-PolicyAssertion.owl;#Integrity" />
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:allValuesFrom rdf:resource="#elementClass1" />
              <owl:onProperty rdf:resource="test#hasAlgorithm" />
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
```

```
    </owl:equivalentClass>
  </owl:Class>

  <owl:Class rdf:about="#policyClass3">
    <rdfs:comment>&lt;SecurityToken>
                  wsse:X509v3
              &lt;/SecurityToken></rdfs:comment>
    <rdfs:subClassOf rdf:resource="#errorType" />
  </owl:Class>

  <owl:Class rdf:about="#policyClass11">
    <rdfs:subClassOf
       rdf:resource="&WS-PolicyAssertion.owl;#Integrity" />
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
              <owl:Restriction>
                <owl:allValuesFrom rdf:resource="#elementClass11" />
                <owl:onProperty rdf:resource="test#hasAlgorithm" />
              </owl:Restriction>
            </owl:intersectionOf>
          </owl:Class>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>

  <owl:Class rdf:about="&WS-PolicyAssertion.owl;#Algorithm" />
  <owl:Class rdf:about="&WS-PolicyAssertion.owl;#Integrity" />
  <owl:Class rdf:about="&WS-PolicyAssertion.owl;#SecurityToken" />

<!-- Annotation Properties -->
  <owl:AnnotationProperty rdf:about="&rdfs;comment" />

<!-- Datatype Properties -->
```

```
  <owl:DatatypeProperty rdf:about="#hasAssertion">
    <rdfs:domain rdf:resource="#errorType" />
    <rdfs:range rdf:resource="&xsd;string" />
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:about="test#hasTokenType" />
  <owl:DatatypeProperty rdf:about="test#hasType" />
  <owl:DatatypeProperty rdf:about="test#hasURI" />

<!-- Object Properties -->
  <owl:ObjectProperty rdf:about="#hasSecurityToken" />
  <owl:ObjectProperty rdf:about="test#hasAlgorithm" />

<!-- Instances -->
  <rdfs:Datatype rdf:about="&xsd;string" />
</rdf:RDF>
```

Listing A.2: XACML in Gracia Policy Format

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.ic3.ct.siemens.com/rbac_example.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rbac-meta="http://www.ic3.ct.siemens.com/RBAC.owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://www.ic3.ct.siemens.com/rbac_example.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.ic3.ct.siemens.com/rbac.owl" />
  </owl:Ontology>

  <rdf:Property
     rdf:about="http://www.ic3.ct.siemens.com/RBAC.owl#toActionandResource" />
  <rdf:Property
     rdf:about="http://www.ic3.ct.siemens.com/RBAC.owl#PermissionToRole" />

  <rbac-meta:Group rdf:ID="DepartmentHead">
```

```
    <rbac-meta:SubjectToRole>
      <rbac-meta:Role rdf:ID="Managemers" />
    </rbac-meta:SubjectToRole>
  </rbac-meta:Group>
  <rbac-meta:Resource rdf:ID="Invoice" />
  <rbac-meta:AtomicAction rdf:ID="signDocument">
    <rbac-meta:toResouce rdf:resource="#Invoice" />
  </rbac-meta:AtomicAction>
  <rbac-meta:User rdf:ID="Bob">
    <rbac-meta:SubjectToRole rdf:resource="#Managemers" />
  </rbac-meta:User>
  <rbac-meta:Permission rdf:ID="toSignInvoice">
    <rbac-meta:toActionandResource rdf:resource="#signDocument" />
    <rbac-meta:forAction rdf:resource="#signDocument" />
    <rbac-meta:toRole rdf:resource="#Managemers" />
    <rbac-meta:PermissionToRole rdf:resource="#Managemers" />
  </rbac-meta:Permission>
</rdf:RDF>
```

Appendix B

Gracia Knowledge Base

Listing B.1: Gracia Knowledge Base in OWL/SWRL

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="&rdf;#"
         xmlns:rdfs="&rdfs;#"
         xmlns:owl="&owl;#"
         xmlns:xsd="&xsd;#"
         xmlns:swrl="&swrl;#"
         xmlns:gkb="&gkb;#"
>


<!--
=============================================
 Ontology CLASSES and PROPOERTIES
=============================================
-->

 <owl:Class rdf:ID="Regulation">
   <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
         <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
             <owl:Class rdf:about="#Norm" />
             <owl:Restriction>
```

```
                <owl:someValuesFrom rdf:resource="#Legislative_Body"
                    />
                <owl:onProperty rdf:resource="#utterer" />
              </owl:Restriction>
            </owl:intersectionOf>
          </owl:Class>
        </owl:someValuesFrom>
        <owl:onProperty rdf:resource="#bears" />
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Legal_Document" />
    </rdfs:subClassOf>
    </owl:Class>

<!--
////////////////////////////////////////////
 Ontology INSTANCES
////////////////////////////////////////////
-->
<Regulation rdf:ID="SOX">
   ...
</Regulation>

<!--
////////////////////////////////////////////
 SWRL Rules
////////////////////////////////////////////
-->
<swrl:Variable rdf:ID="X"/>
<swrl:Variable rdf:ID="Y"/>


<swrl:Imp>
  <swrl:body rdf:parseType="Collection">
    <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="&gkb;#Solution"/>
```

```
      <swrl:argument1 rdf:resource="#X"/>
    </swrl:ClassAtom>
    <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="&gkb;#ValuedConstraint"/>
      <swrl:argument1 rdf:resource="#Y"/>
    </swrl:ClassAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&gkb;#satisfies"/>
      <swrl:argument1 rdf:resource="#X"/>
      <swrl:argument2 rdf:resource="#Y"/>
    </swrl:IndividualPropertyAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&gkb;#violates"/>
      <swrl:argument1 rdf:resource="#X"/>
      <swrl:argument2 rdf:resource="#Y"/>
    </swrl:IndividualPropertyAtom>
  </swrl:body>

  <swrl:head />

</swrl:Imp>
</rdf:RDF>
```

Appendix C

Security Ontology

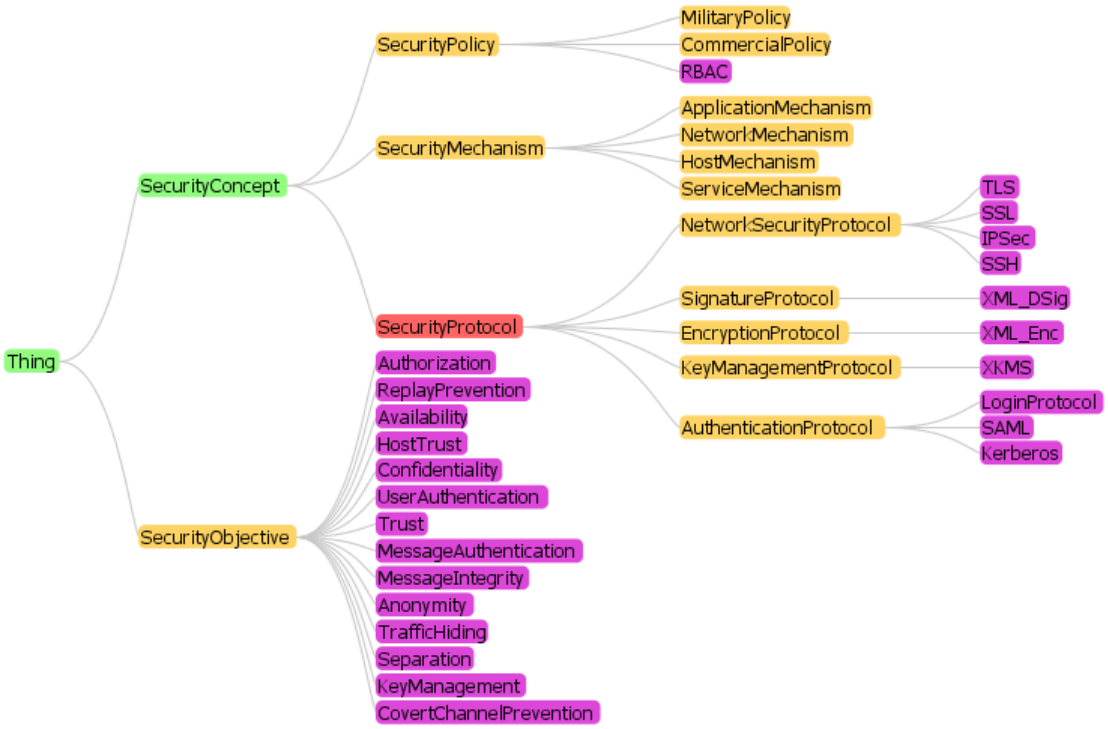

Figure C.1: Security Ontology

Appendix D

UML Diagrams

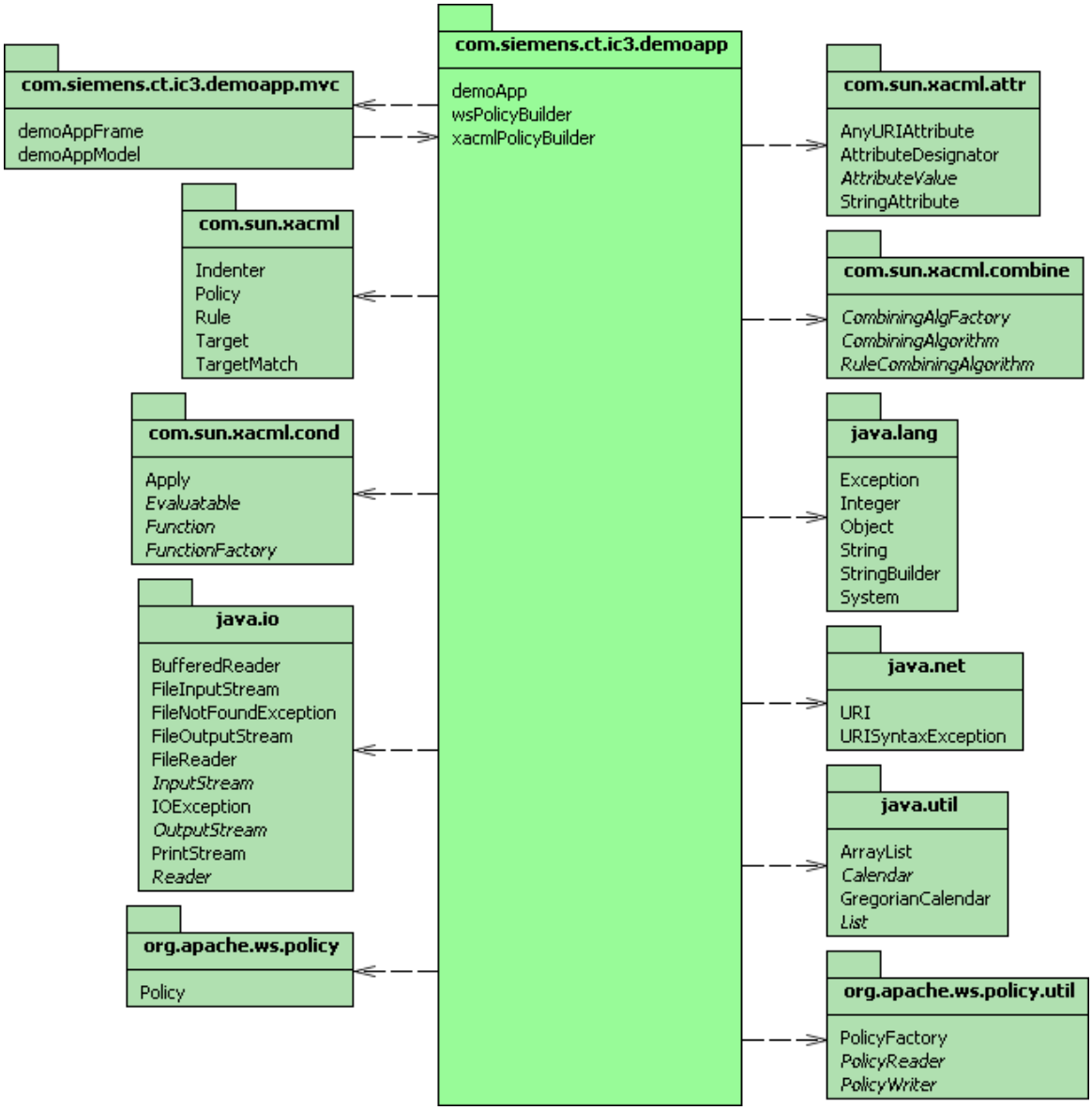

Figure D.1: Demo Application

Bibliography

[1] The SECURE project, http://secure.dsg.cs.tcd.ie/.

[2] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, page 6007, Washington, DC, USA, 2000. IEEE Computer Society.

[3] A. Acciarri, D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. Quonto: Querying ontologies. In *20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, number 1670-1671, Pittsburgh, USA, 2005. The MIT Press.

[4] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[5] F. Baader, D. Calvanese, D.McGuinness, D. Nard, and P. F.Patel-Schneider. *The Description Logic Handbook:Theory, Implementation and Applications.* Cambridge University Press, 2003.

[6] S. Bajaj. *Web Services Policy Framework (WS-Policy)*. http://schemas.xmlsoap.org/ws/2004/09/policy, March 2006.

[7] M. Ball. *OO jDREW: Design and Implementation of a Reasoning Engine for the Semantic Web*, 2005. Honours Thesis Project Report.

[8] Jon Barwise and John Etchemendy. Model-theoretic semantics. *Foundations of cognitive science*, pages 207–243, 1989.

[9] N. Bassiliades and P. Gray. CoLan. A functional constraint language and its implementation. *Data and Knowledge Engineering*, pages 203–249, 1994.

[10] Steve Battle, Abraham Bernstein, Harold Boley, Michael Gruninger, and Richard Hull. Semantic web services framework (SWSF) overview version 1.0. *Semantic Web Services Initiative (SWSI)*, 2005.

[11] S. Bechhofer, I. Horrocks, and D. Turi. The owl instance store: System description. In *20th International Conference on Automated Deduction (CADE 2005)*, number 177-181, Tallinn, Estonia, 2005. Springer.

[12] D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, Mitre C., 1976.

[13] T. Bellwood, S. Capell., et al. *Universal Discovery, Discovery, and Integration (UDDI)*, 2004. Specification version 3.02.

[14] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM TODS*, 23(3), 1998.

[15] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.

[16] H. Boley. Object-Oriented RuleML: User-level roles, URI-grounded clauses, and order-sorted terms. In *Second International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML'03)*, pages 1–16, 2003.

[17] P.A. Bonatti, S. De Capitani Di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, 2002.

[18] M. Bonifacio, P. Bouquet, and R. Cuel. Knowledge nodes: the building blocks of a distributed approach to knowledge management. *Journal of Universal Computer Science*, 8(6):652–661, 2002.

[19] J. Bradshaw and A. Uszok. Representation and reasoning for daml-based policy and domain services in kaos and nomads. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 835–842, New York, NY, USA, 2003. ACM Press.

[20] Paul Buhler, José M. Vidal, and Harko Verhagen. Adaptive workflow = web services + agents. In *Proceedings of the International Conference on Web Services*, pages 131–137. CSREA Press, 2003.

[21] J. Calmet, A. Daemi, R. Endsuleit, and T. Mie. A liberal approach to openness in societies of agents. In *Fourth International Workshop on Engineering Societies in the Agents World (ESAW)*, London, 2003.

[22] J. Calmet, P. Maret, and R. Endsuleit. Agent-oriented abstraction. *Revista (Real Academia de Ciencias, Serie A de Matematicas)*, 98(1):77–84, 2004. Special Issue on Symbolic Computation and Artificial Intelligence.

[23] C. Chen, V. Haarslev, and J. Wang. Las: Extending racer by a large abox store. In *Int.Workshop on Description Logics (DL05)*, number 200-207, Edinburgh, Scotland,UK, 2005.

[24] E. Christensen, F. Curbera, et al. *Web Services Description Language (WSDL)*. W3C Web Services Activity, 2003. Specification version 1.2., W3C Technical Document.

[25] Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. Referee: trust management for web applications. In *Selected papers from the sixth international conference on World Wide Web*, pages 953–964, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[26] OWL Services Coalition. OWL-S: Semantic markup for web services. *Whitepaper Version 1.0.*, 2004.

[27] N. Damianou, A. Bandara, M. Sloman, and E. Lupu. A survey of policy specification approaches. Technical report, Department of Computing, Imperial College of Science Technology and Medicine, 2002.

[28] K. Decker, M. Williamson, and K. Sycara. Matchmaking and brokering. In *the Second International Conference in Multi-Agent Systems (ICMAS'96)*, Kyoto, Japan, 1996.

[29] J. Douceur. The Sybil Attack. In *Proceedings of the 1st Intl Workshop on Peer-to-Peer Systems(IPTPS¡¯02)*, 2002.

[30] SPI Dynamics. Webinspect, January 2003.

[31] T. Eiter, G. Gottlob, and H. Mannilla. Disjunctive datalog. *ACM Trans. on Database Systems*, 22(3):364418, 1997.

[32] D. Fensel and C. Bussler. The web service modeling framework WSMF. Technical report, Vrije Universiteit Amsterdam, 2002.

[33] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, S. Shapiro, and C. Beck. Specification of the kqml agent-communication language. *The DARPA Knowledge Sharing Initiative External Interfaces Working Group*, 1992.

[34] Gerhard Fischer and Jonathan Ostwald. Knowledge management: Problems, promises, realities, and challenges. *IEEE Intelligent Systems*, 16(1):60–72, 2001.

[35] Charles Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[36] N. E. Fuchs, S. Höfler, K. Kaljurand, F. Rinaldi, and G. Schneider. Attempto controlled english: A knowledge representation language readable by humans and machines. In *Reasoning Web*, pages 213–250, 2005.

[37] M. Genesereth and R. Fikes. Kif: Knowledge interchange format version 3.0 reference manual. Technical report, Stanford University, Knowledge Systems Laboratory, 1992.

[38] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[39] Thomas R. Gruber. *Model formulation as a problem-solving task: computer-assisted engineering modeling*, chapter Knowledge Acquisition as Modeling, pages 105–127. John Wiley & Sons, Inc, NY, USA, 1993.

[40] M. Gudgin, M. Hadley, et al. *SOAP Version 1.2 Part 1: Messaging Framework*, 2003. W3C Recommendation.

[41] V. Haarslev and R. Möller. Racer: A core inference engine for the semantic web ontology language (owl). In *the 2nd Int. Workshop on Evaluation of Ontology-based Tools*, pages 27–36, 2003.

[42] Marc Hammond. Virtual knowledge communities for distributed knowledge management: A multi-agent-based approach using jade. Master's thesis, University of Karlsruhe and Imperial College London, 2004.

[43] James Hendler. Agents and the semantic web. *Intelligent Systems*, 21(2):30–37, 2001.

[44] D. Hirtle. TRANSLATOR: A TRANSlator from LAnguage TO Rules. In *Proceedings of the Canadian Symposium on Text Analysis (CaSTA'06)*, New Brunswick, Canada, 2006.

[45] Jason Hogg, Don Smith, Fred Chong, Dwayne Taylor, Lonnie Wall, and Paul Slater. Web service security. scenarios, patterns, and im-plementation guidance for web services enhancements (wse) 3.0. *Microsoft Press*, 2005.

[46] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3c member submission, World Wide Web Consortium, 2004.

[47] Ian Horrocks and Peter F. Patel-Schneider. SWRL: A semantic web rule language combining OWL and RuleML. Technical report, The Rule Markup Initiative, May 2004.

[48] D. Huang, Y. Yang, and J. Calmet. A knowledge-based security policy framework for business process management. In *IEEE International Conference on Intelligent Agents, Web Technologies and Internet Commerce(IAWTIC06)*, pages 154–161, Sydney, Australia, 2006.

[49] D. Huang, Y. Yang, and J. Calmet. Modeling web services policy with corporate knowledge. In *IEEE International Conference on E-Business Engineering(ICEBE 2006)*, pages 216–223, Shanghai, China, 2006.

[50] Dong Huang. Semantic policy-based security framework for business processes. In *Proc. of the Semantic Web and Policy Workshop*, Galway, Ireland, November 2005.

[51] Dong Huang. Semantic descriptions of web services security constraints. In *International Symposium on Service-Oriented System Engineering (SOSE2006)*, number 81-84, Shanghai, China, October 2006.

[52] Dong Huang and Shane Bracher. Towards evidence-based trust brokering. In *Proc. of the SECOVAL Workshop, held in conjunction with the 1st IEEE/CREATE-NET SecureComm*, Athens, Greece, September 2005.

[53] Michael N. Huhns, Munindar P. Singh, Mark Burstein, Keith Decker, Ed Durfee, Tim Finin, Les Gasser, Hrishikesh Goradia, Nick Jennings, Kiran Lakkaraju, Hideyuki Nakashima, Van Parunak, Jeffrey S. Rosenschein, Alicia Ruvinsky, Gita Sukthankarand Samarth Swarup, Katia Sycara, Milind Tambe, Tom Wagner, and Laura Zavala. Research directions for service-oriented multiagent systems. *Internet Computing*, 9(6):65–70, November/December 2005.

[54] K. Hui, S. Chalmers, P. Gray, and A. Preece. Experience in using rdf in agent-mediated knowledge architectures. *Agent-Mediated Knowledge Management (LNAI 2926). Springer-Verlag*, pages 177–192, 2004.

[55] David Ingram. An Evidence Based Architecture for Efficient, Attack-Resistant Computational Trust Dissemination in Peer-to-Peer Networks. In *Proceedings of the Third International Conference on Trust Management (iTrust '05)*, May 2005.

[56] Java Community Process. *JSR 94: JavaTM Rule Engine API*. Specification available at: http://jcp.org/en/jsr/detail?id=94.

[57] JBoss. *JBoss Rules*. http://www.jboss.com/products/rules.

[58] Simon Johnston. Modeling security concerns in serviceoriented architectures. *Developworks*, July 2005.

[59] Daniel Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

[60] L. Kagal. Rei: a policy language for the me-centric project. Technical Report HPL-2002-270, HP Labs, 2002.

[61] L Kagal. *A Policy Based Approach to Governing Autonomous Behavior in Distributed Environments*. PhD thesis, Faculty of the Graduate School of the University of Maryland, Baltimore County, USA, 2004.

[62] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, and T. Finin. A security architecture based on trust management for pervasive computing systems, 2002.

[63] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003.

[64] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, 2003.

[65] Steve Anderson Kelvin Lawrence, Chris Kaler. Web services security. Technical report, OASIS, 2004.

[66] M. Kinateder and K. Rothermel. Architecture and Algorithms for a Distributed Reputation System. In P. Nixon and S. Terzis, editors, *Proceedings of the First International Conference on Trust Management*, volume 2692 of *LNCS*, pages 1–16, Crete, Greece, May 2003. Springer-Verlag.

[67] A. Kiryakov, D. Ognyanov, and D. Manov. Owlima pragmatic semantic repository for owl. In *Int.Workshop on Scalable SemanticWeb Knowledge Base Systems (SSWS05)*, number 182-192 in 1, New York City, USA, 2005. Springer.

[68] Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *the 16th International World Wide Web Conference*, 2007.

[69] Vladimir Kolovski and Bijan Parsia. Ws-policy and beyond: application of owl defaults to web service policies. In *2nd international semantic web policy workshop (swpw'06)*, 2006.

[70] R. Kraft. Research and design issues of access control for network services on the web. In *The 3rd International Conference on Internet Computing (IC 2002)*, pages 542–548, 2002.

[71] N. Li and J. Mitchell. Rt: A role-based trust-management framework, 2003.

[72] Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, November/December 1999.

[73] Frank Manola and Eric Miller. Rdf primer. Technical report, W3C Recommendation, February 2004.

[74] Dragos Manolescu and Boris Lublinsky. Soa security. http://orchestrationpatterns.com/files/SOASecurity.pdf, 2007.

[75] Pierre Maret and Jacques Calmet. Modeling corporate knowledge within the agent oriented abstraction. In *Proc. International Conference on Cyberworlds (CW'04)*, pages 224–231. IEEE Computer Society, 2004.

[76] Pierre Maret, Mark Hammond, and Jacques Calmet. Virtual knowledge communities for corporate knowledge issues. In *Proceedings 5th International Workshop on Engineering Societies in the Agents World (ESAW'04)*, volume 3451 of *Lecture Notes in Computer Science*, pages 33–44, Toulouse, France, October 22-24 2004. Springer.

[77] S. Marsh. Formalising trust as a computational concept, 1994.

[78] J. McCarthy. Applications of circumscription in formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[79] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. Technical report, W3C Recommendation, February 2004.

[80] Craig McKenzie, Peter Gray, and Alun Preece. Extending SWRL to express fully-quantified constraints. In *Proc. of RuleML 2004 Workshop at ISWC 2004*, Hiroshima, Japan, November 2004.

[81] M. Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, pages 95–128. MIT Press, Cambridge, MA, 1981.

[82] R. Montanari, A. Toninelli, and J.M. Bradshaw. Context-based security management for multi-agent systems. In *Second IEEE Symposium on Multi-Agent Security and Survivability*, Philadelphia, USA, August 2005. IEEE Press.

[83] L. Moreau, J. Bradshaw, M. Breedy, L. Bunch, M. Johnson, Kulkarni S., Lott J., Suri N., and Uszok A. Behavioural specification of grid services with the kaos policy language. In *Cluster Computing and Grid 2005*, Cardiff, UK, 2005.

[84] Tim Moses. extensible access control markup language (xacml) version 2.0 3. OASIS Standard, Feb 2005.

[85] B. Motik and R. Studer. Kaon2 a scalable reasoning tool for the semantic web. In *the 2nd European Semantic Web Conference (ESWC05)*, Heraklion, Greece, 2005.

[86] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. In *Proc. of International Semantic Web Conference 2004*, pages 549–563, Hiroshima, Japan, November 2004.

[87] A. Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[88] OASIS. Oasis security assertion markup language (saml). Technical report, http://www.oasis-open.org/committees/security, 2004.

[89] Mark ONeill. Architecting security for web services. *JavaPro*, August 2003. Available from http://www.ftponline.com/channels/security/javapro/2003 08/magazine/features/moneill.

[90] Justin O'Sullivan, David Edmond, and Arthur H. M. ter Hofstede. Formal description of non-functional service properties, queensland university of technology. Technical report, http://www.service-description.com, 2005.

[91] J Pescatore. Web services: Application-level firewalls required. Technical report, Gartner, Standford, 2002.

[92] Alun Preece, Stuart Chalmers, Craig McKenzie, Jeff Pan, and Peter Gray. Handling soft constraints in the semantic web architecture. In *Proc. of RoW2006 Reasoning on the Web at WWW2006*, Edinburgh, UK, 2006.

[93] Eric Prudhommeaux and Andy Seaborne. Sparql query language for rdf. *W3C Candidate Recommendation*, 2006.

[94] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81132, 1980.

[95] Riccardo Rosati. The limits and possibilities of combining description logics and Datalog. In *Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML2006)*. IEEE Computer Society Press, 2006.

[96] The Rule Markup Initiative. *RuleML*. http://www.ruleml.org/.

[97] S.Anderson. Web services trust language (ws-trust). Technical report, http://schemas.xmlsoap.org/ws/2005/02/trust, February 2005.

[98] Ravi S. Sandhu and Pierrangela Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[99] M Schmidt-Schauss. Subsumption in kl-one is undecidable. In *the First Intl Conference on the Principles of Knowledge Representation and Reasoning (KR 1989)*, 1989.

[100] K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for policy languages for trust negotiation. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, page 68, Washington, DC, USA, 2002. IEEE Computer Society.

[101] J.-M. Seigneur, A. Gray, and C. D. Jensen. Trust Transfer: Encouraging Self-Recommendations without Sybil Attack. In *Proc. of the Third International Conference on Trust Management*, LNCS. Springer, 2005.

[102] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[103] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl dl reasoner. *Journal of Web Semantics*, 2006.

[104] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2:333–360, 1994.

[105] Michael A. Smith, Andrew J. Schain, Kendall Grant Clark, Arlen Griffey, and Vladimir Kolovski. Mother, may i? owl-based policy managementat at nasa. In *the OWLED 2007*, 2007.

[106] Berners-Lee T., Hendler J., and Lassila O. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2001.

[107] Sotirios Terzis, Waleed Wagealla, Colin English, and Paddy Nixon. The SECURE collaboration model. Technical report, Dept. of Computer and Information Sciences, University of Strathclyde, 2003.

[108] Ioan Toma and Douglas Foxvog. Non-functional properties in web services. Technical report, DERI, 2006.

[109] Alessandra Toninelli, Jeffrey Bradshaw, Lalana Kagal, and Rebecca Montanari. Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments. In *Proceedings of the Semantic Web and Policy Workshop*, November 2005.

[110] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *Int. Joint Conference on Automated Reasoning (IJCAR06).*, 2006.

[111] A. Uszok, J. M. Bradshaw, R. Jeffers, A. Tate, and J. Dalton. Applying kaos services to ensure policy compliance for semantic web services workflow composition and enactment. In *Third International Semantic Web Conference*, pages 425–440, Hiroshima, Japan, 2004.

[112] Christopher D. Walton. Uniting agents and web services. *AgentLink News*, 1(18):26–28, August 2005.

[113] Max Weber. *Economy and society*. University of California Press, 1986.

[114] Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[115] Timo Weithöner, Thorsten Liebig, Marko Luther, Sebastian Böhm, Friedrich von Henke, and Olaf Noppens. Real-world reasoning with owl. In *4th European Semantic Web Conference 2007*, 2007.

[116] T. Y. C. Woo and S. S. Lam. Authorizations in distributed systems: A new approach. *Journal of Computer Security*, 2(2,3):107–136, 1993.

[117] Michael Wooldridge. Intelligent agents. In Weiss Gerhard, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 1, pages 27–78. The MIT Press, 1999.

[118] G. Yang and M. Kifer. Flora-2: User's manual, 2002.

[119] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 294–301, New York, NY, USA, 2002. ACM Press.

[120] J. Zhou, L. Ma, Q. Liu, L. Zhang, Y. Yu, and Y. Pan. Minerva: A scalable owl ontology storage and inference system. In *1st Asian Semantic Web Conference (ASWC 2006)*, number 429-443 in 1, Beijing, China,, 2006. Springer.

List of Figures

List of Tables

Listings

Index

List of Publications

D. Huang, Y. Yang, J. Calmet: **A Knowledge-based Security Policy Framework for Business Process Management**, in Proc. of IEEE International Conference on Intelligent Agents, Web Technologies and Internet Commerce(IAWTIC06), Sydney, Australia, pp 154-161, 29 November-1 December 2006.

D. Huang: **Semantic Descriptions of Web Services Security Constraints**, In Proc. of International Symposium on Service-Oriented System Engineering (SOSE2006), Shanghai, China, pp 81-84, 25-27 October 2006.

D. Huang, Y. Yang, J. Calmet: **Modeling Web Services Policy with Corporate Knowledge**, In Proc. of IEEE International Conference on E-Business Engineering(ICEBE 2006), Shanghai, China, pp 216-223. October 2006. (Best paper candidate)

D. Huang: **Semantic Policy-based Security Framework for Business Processes**. In Proc. of International Semantic Web Conference (ISWC2005), the Semantic Web and Policy Workshop, Galway, Ireland, November 2005.

D. Huang, S. Bracher: **Towards Evidence-based Trust Brokering**, In Proc. of the IEEE/CREATE-NET SECOVAL Workshop(SECOVAL 2005), Athens, Greece, September 2005