DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
TELEMÁTICOS

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

APPLICATION OF MODEL-DRIVEN TECHNIQUES
TO THE DESIGN OF
NON-FUNCTIONAL CONCERNS OF
SERVICE-ORIENTED SOFTWARE SYSTEMS

Autor: Juan Pedro Silva Gallino
Ingeniero de Telecomunicación

Director: Miguel Ángel de Miguel Cabello
Doctor Ingeniero Informático

2012

POLITÉCNICA

Tribunal nombrado por el Magfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día ____ de _____ de 201__.

Presidente: _____.

Vocal: _____.

Vocal: _____.

Vocal: _____.

Secretario: _____.

Suplente: _____.

Suplente: _____.

Realizado el acto de defensa y lectura de la Tesis el día ____ de _____
de 201__ en la E.T.S.I. /Facultad _____
_____.

Calificación:_____.

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

Resumen

Internet se ha convertido en la herramienta por excelencia para el intercambio de servicios de negocio e información entre empresas. En este contexto, las arquitecturas orientadas a servicios (SOA) y los servicios web (WS) han surgido como la plataforma más apropiada para las interacciones aplicación-aplicación.

Por otra parte, el desarrollo guiado por modelos (MDD), y Model-Driven Architecture (MDA) en particular, son nuevos paradigmas que promueven el uso de modelos del sistema como elementos fundamentales en el proceso de desarrollo. Estos nuevos enfoques soportan la aplicación de mejores prácticas, patrones, y la reutilización en el desarrollo de familias de sistemas. Sin embargo, a las propiedades o características no funcionales (NF) (tales como seguridad, adaptabilidad, calidad de servicio, etc.) no se les ha prestado suficiente atención por parte de estos enfoques. Recientemente, han surgido nuevas propuestas que estudian específicamente el tema de las propiedades no-funcionales del software, permitiendo una separación de las características funcionales y no funcionales de los sistemas *en la etapa de diseño*. Entre los ejemplos de dichas propuestas encontramos Viewpoints, Multi-Dimensional Separation of Concerns (MDSOC), Aspect-Oriented Modeling (AOM) or Early Aspects (EA).

Actualmente, las organizaciones de estándares, tales como el Object Management Group (OMG), se encuentran proponiendo soluciones que ofrezcan alternativas estándar al modelado de servicios. Herramientas y descripciones sobre la experiencia del uso de dichos estándares se esperan en el corto plazo. Sin embargo, no ha sido hasta muy recientemente que los aspectos no funcionales de los servicios han sido considerados para estos procesos de estandarización. Por ende, soluciones en este área no son esperadas en los años venideros.

Dentro de las carencias de la computación orientada a servicios se destaca la necesidad de metodologías que soporten la especificación y el diseño de servicios y composiciones de servicios, asociando de esta forma las metodologías de diseño con las técnicas de modelado de procesos de negocio. La *conciencia de calidad de servicio* es también destacada como otro gran desafío en el futuro de las arquitecturas orientadas a servicios.

Estos desafíos, nuevos enfoques y metodologías que han sido enumerados previamente, los cuales brindan soporte para la inclusión de características no funcionales en el desarrollo de sistemas SOA, son los motores que impulsan este trabajo. Aunque existen múltiples tecnologías de implementación que buscan facilitar el desarrollo de servicios web y sistemas SOA, la falta de una sólida base metodológica para el desarrollo de tales aplicaciones acentúa la necesidad de nuevas técnicas o métodos de modelado que pudieran garantizar la calidad del desarrollo de este tipo de sistemas.

Esta tesis presenta un enfoque para lograr una solución para el desarrollo de arquitecturas orientadas a servicios con conciencia de características no funcionales, integrada, guiada por modelos, y que engloba múltiples áreas de investigación. La solución propuesta brinda, entre otros beneficios, una mejor comprensión del sistema al completo, y un desarrollo independiente de la plataforma objetivo, mejorando así la reutilización de diseño, y simplificando la evolución del sistema, con el consecuente aumento de la productividad.

Abstract

Internet has become the tool “par excellence” for business and information exchange between companies. In this context, service-oriented architectures (SOA) and web services (WS) have emerged as the most suitable platform for application-to-application interactions.

On a different token, model-driven development (MDD), and Model-Driven Architecture (MDA) in particular, are new paradigms that promote the use of system models as primary artifacts in the development process. These approaches support the application of best practices, patterns, and reuse in the development of families of systems. However, non-functional (NF) properties or concerns (such as security, adaptability, quality of service, etc.) have not been sufficiently addressed by these approaches. Recently, new proposals have risen that specifically focus on the area of non-functional concerns, allowing for the separation of functional and extra-functional characteristics of the systems *at design time*. Examples of those are Viewpoints, Multi-Dimensional Separation of Concerns (MDSoc), Aspect-Oriented Modeling (AOM) or Early Aspects (EA).

Currently, standardization organizations, such as the OMG, are proposing standard solutions for service modeling. Tools and experiences on the use of such standards are expected in the short term. However, it has not been until very recently that non-functional aspects of services have been considered for standardization processes. Thus, no solution in this area is expected within the next few years.

In the research roadmap for service-oriented computing, the need for methodologies supporting the specification and design of services and service compositions is remarked, in that way associating the design methodology with business process modeling techniques. *QoS-awareness* is also stressed as another grand challenge in the future of service-oriented architectures.

These challenges and new approaches and methodologies enumerated above, supporting the inclusion of non-functional concerns development into SOA systems, are the engines propelling this research. Although multiple implementation technologies exist to facilitate the development of web services and SOA systems, the lack of a sound methodological base for the development of such applications stresses the need for new modeling methods or techniques that could guarantee the quality of the development of this type of systems.

This dissertation presents an approach for the achievement of an integrated, model-driven solution for the development of service-oriented architectures with non-functional awareness, spanning over multiple areas of research. The proposed approach provides, among other benefits, a greater understanding of the system as a whole and a platform-independent development, improving reusability of designs, and simplifying the evolution of the system, thus increasing productivity.

Agradecimientos

Sería muy crédulo suponer que una persona es capaz de llegar hasta este punto por sí solo. A lo largo del camino, muchas otras personas le han llevado de la mano de una u otra forma, hasta que es capaz de lograr su objetivo. Me gustaría recordar aquí a algunos de aquellos que me han ayudado en la travesía.

Aunque el orden aquí no pretenda indicar mayor o menor relevancia, reservo un lugar destacado a mi director de tesis, Miguel Ángel de Miguel, que me ha guiado durante estos años por el camino de la investigación, y me ha indicado la dirección correcta en las horas más oscuras de mi trabajo. Sin él, esta tesis no hubiera sido posible. También guardo un lugar muy importante para Alejandro Alonso, quién me encontró un lugar en el grupo y, desde el comienzo y a lo largo de este tiempo, y muy a pesar de mi cabeza dura, ha logrado enseñarme algo de lo tanto que sabe, lo que para mí ha sido mucho. No me puedo olvidar de los demás profesores, Juan Zamorano y Juan Antonio de la Puente, que han estado siempre ahí para lo que necesitase durante todo este tiempo. Todos ellos son, además de mis maestros, mis amigos.

En mi trabajo en todo este tiempo me he sentido como en casa, y para ello ha sido fundamental, al igual que los anteriormente mencionados, la presencia de excelentes personas y compañeros de grupo como Javier Fernández Briones, Emilio Salazar, Daniel Tejera, José Pulido, Santiago Urueña, Daniel Berjón, y todos aquellos en algún momento han formado parte del él. A todos ellos, muchas gracias por estos años compartidos.

No puedo olvidarme de Angelines, la secretaria del departamento. Sin ella las innumerables barreras burocráticas hubieran sido insalvables!. A tí, Angelines, muchas gracias.

Un recuerdo especial lo tengo para mi familia de acogida en España: el Club de Rugby Tres Cantos, y cada uno de los que en él han compartido conmigo tantos buenos momentos. De allí, como de mi trabajo, han surgido excelentes amistades, que perdurarán en el tiempo.

Por último, y no por ello menos importante, agradezco a mi familia. No solo por permitirme estar aquí (en el mundo!), sino por haber sido mi apoyo constante y una fuente inagotable de cariño durante toda mi vida. Con ellos, agradezco en especial a Lorena, mi compañera de ruta, a quien le toca soportarme todos los días, y sin la cual este camino hubiera sido intolerable. A ella y a su familia, que ha sido como la mía propia, los llevo en el corazón.

A todos aquellos a los que aquí menciono, y a aquellos a los que no menciono, pero que no olvido, gracias... totales!¹.

---

¹Parafraseando a Gustavo Cerati.

Glossary

AO Aspect-Oriented	MDA Model Driven Architecture
AOM Aspect-Oriented Modeling	MDD Model Driven Development
AOP Aspect-Oriented Programing	MDE Model Driven Engineering
AOSD Aspect-Oriented Software Development	MDSoC Multi-Dimensional Separation of Concerns
BPMN Business Process Model and Notation	MOF Meta Object Facility
CBSD Component-Based Software Development	NF Non-Functional
CIM Computation Independent Model	NFP Non-Functional Property
DSM Domain Specific Modeling	OCL Object Constraint Language
DSMM Domain Specific Multimodeling	OMG Object Management Group
DSML Domain Specific Modeling Language	PIM Platform Independent Modeling
DSL Domain Specific Language	PSM Platform Specific Model
EA Early Aspects	QoS Quality of Service
EMF Eclipse Modeling Framework	RAS Reusable Asset Specification
GEF Graphical Editing Framework	RBAC Role-Based Access Control
GMF Graphical Modeling Framework	SOA Service-Oriented Architectures
IT Information Technology	TMF Textual Modeling Framework
iMM intermediate MetaModel	UML Unified Modeling Language
M2M Model-to-Model Transformation	WS Web Services
M2T Model-to-Text Transformation	WSDL Web Services Description Languages
MD Model Driven	XMI XML Metadata Interchange
	XML eXtensible Markup Language

Contents

Resumen	i
Abstract	ii
Agradecimientos	iii
Glossary	iv
Contents	v
List of Figures	ix
List of Tables	xi
List of Listings	xii
1 Introduction	1
1.1 Motivation	1
1.2 Research Problem	2
1.3 Covered Areas	3
1.3.1 SOA/WS	3
1.3.2 MDD, MDA, DSM	4
1.3.3 NF Concerns Modeling	5
1.4 Proposed Approach	6
1.5 Contributions	6
1.5.1 Theoretical Contributions	6
1.5.2 Methodological Contributions	7
1.5.3 Practical Contributions	7
1.5.4 Funding Acknowledgements	7
1.6 Dissertation Organization	8
2 Background	9
2.1 Model-Driven Approaches	9
2.1.1 Setting the Cornerstone	9
2.1.2 Terminology	11
2.1.3 Model-Driven Initiatives	17
2.1.4 Comparing UML and DSML Approaches	27
2.1.5 State of the Practice in Modeling Transformation Tools	29
2.2 Modeling Service-Oriented Architectures	29

2.2.1	Service Oriented Architectures	29
2.2.2	Modeling Service-Oriented Architectures	30
2.3	Summary	32
3	MDD of a WSOA and NF Policies	35
3.1	Introduction	35
3.2	In Search Of The Integrated Solution	36
3.2.1	Requirements and Objectives	36
3.2.2	Defined Architecture	37
3.2.3	Development Process	38
3.3	NF-Concern Independent Metamodels	39
3.3.1	Functional CIM Input Metamodel: BPMN	39
3.3.2	Functional PIM Input Metamodel: UML	41
3.3.3	WSDL Metamodel	42
3.3.4	WS-Policy Metamodel	42
3.4	Measurement and Analysis of Models	43
3.5	Discussion of the Approach	44
3.5.1	Comparison with Domain-Specific Multimodeling	44
3.5.2	Comparison with Unique Metamodel	44
3.6	Summary of the Chapter	45
4	Security and AC as NFPs in SOAs	47
4.1	Introduction	47
4.2	Integration, Transformation, and Composition of Metamodels	48
4.3	NF Metamodels	49
4.3.1	Modeling Security Intents at the CIM level	49
4.3.2	Access Control Input Metamodel: SecureUML	53
4.3.3	Security Input: QoS Framework Metamodel	55
4.3.4	Weaving Metamodel	56
4.3.5	WS-SecurityPolicy Metamodel	56
4.3.6	iMM Metamodel	58
4.4	Definition of the iMM MM for Security and AC	59
4.4.1	Functional Metamodel	60
4.4.2	Generic Access Control Metamodel	61
4.4.3	Policy Package of the CBDI-SAE Metamodel for SOA	62
4.4.4	Identification of Correspondences	65
4.4.5	Resulting iMM Metamodel	66
4.5	Metamodels' Mappings	68
4.5.1	CIM to PIM	68
4.5.2	SecureUML to iMM	69
4.5.3	QoS Framework Metamodel to iMM	71
4.5.4	UML with SoaML Profile to iMM	71
4.5.5	iMM to XML Schema	71
4.5.6	iMM to WSDL	72
4.5.7	iMM to WS-Policy	72
4.6	Analyses in the iMM Metamodel	73
4.6.1	Requirements Addressing Metric	73
4.6.2	Access Control Conflict Analysis	73
4.7	Implementation of a Prototype	74
4.8	Summary of the Chapter	76

5	The WS-I SCM Use Case	77
5.1	Introduction	77
5.2	The SCM Use Case, Functional Specification	79
5.2.1	Warehouse Service	80
5.2.2	Warehouse Callback Service	80
5.3	Services Security Specification	84
5.4	Access Control Specification	84
5.5	MDD of WS-I's SCM Use Case with NFPs	84
5.5.1	Business Process Model	84
5.5.2	PIM Input Models	87
5.5.3	Matching Non-Functional Models	90
5.5.4	Transformation Into the iMM Model	91
5.5.5	Generation of Resulting Artifacts	92
5.6	Summary of the Chapter	98
6	Related Work	101
6.1	Modeling Non-Functional Properties	101
6.1.1	UML Non-Functional-Aware Models	102
6.1.2	Domain Specific Languages	103
6.1.3	Author's Publication on the Subject	106
6.2	Methodologies and Development Processes	106
6.2.1	Author's Publication on the Subject	109
6.3	Analyses, Metrics, and Model Measuring	109
6.3.1	Model Metrics	109
6.3.2	Model-Driven Analyses	110
6.3.3	Author's Publication on the Subject	112
6.4	Modeling SO and WS Systems	112
6.4.1	Proposals Addressing the Functional Aspects of SOA Systems	112
6.4.2	Non-Functional Approaches Based on UML Profiles	113
6.4.3	Other Approaches Addressing Non-Functional Properties	114
6.4.4	Author's Publication on the Subject	118
6.5	Summary of the Chapter	118
7	Summary & Conclusions	119
7.1	Summary and Major Contributions	119
7.1.1	Contributions	120
7.2	Discussion	121
7.3	Future Lines of Work	123
Appendices		
A	Input and Composition Models for WS-I's SCM Use Case	127
B	Transformation Templates	133
B.1	iMM2WSDL.AtlanZoo.doc.lit.wrapped.atl	134
B.2	appContext.mt	143

C	Generated Files for WS-I's SCM Use Case	147
C.1	SecurityConnector.java	148
C.2	WeekDaysConstraintVoter.java	151
C.3	SCM Use Case Application Context (Security)	152
C.4	SCM Use Case Access Control Context	159
C.5	Warehouse Service Generated Policy Document	160
	Bibliography	163

List of Figures

1.1	A visualization of MDSoc	5
2.1	Evolution of the Abstraction Level Through the Years	10
2.2	Relation between abstract and concrete syntaxes.	14
2.3	A view of the Different Model-Driven Approaches	18
3.1	Structure of the proposed solution.	38
3.2	Proposed Development Process.	40
3.3	Conceptual view of the SoaML profile. (<i>OMG, 2009</i>)	42
3.4	WS-Policy Metamodel.	43
4.1	CIM and PIM Level Metamodels.	49
4.2	PIM and PSM Level Metamodels.	50
4.3	Relations Between All Metamodels.	50
4.4	SecureUML Metamodel.	54
4.5	QoS Framework Metamodel.	55
4.6	Weaving Metamodel.	57
4.7	WS-SecurityPolicy Metamodel.	57
4.8	Composition of the iMM Metamodel.	58
4.9	Functional Metamodel, Packages View.	61
4.10	Functional Metamodel, Services View.	62
4.11	Generic Access Control Metamodel	63
4.12	CBDI-SAE Metamodel for SOA's Policy Package	64
4.13	iMM Metamodel (conceptual partial view).	67
4.14	Implementation of the Approach in Figure 3.1 for Security and Access Control.	68
4.15	SecureUML to iMM Mapping.	70
4.16	Structure of the proposed solution.	75
5.1	Supply Chain Management Use Case	78
5.2	Structure of the proposed solution.	79
5.3	Retailer System Services Interaction	82
5.4	Annotated BPMN Model Input.	85
5.5	<i>Retailer's Collaboration Business Process Model</i>	86
5.6	SoaML, QoS, and SecureUML Inputs.	87
5.7	<i>SecureUML Model</i>	90
5.8	PIM Inputs to iMM (PIM 2) Stage.	91
5.9	Outputs of the Development Process.	93
5.10	<i>Resulting Warehouse Service's WSDL document</i>	95

5.11	Warehouse Service's WS-Policy document	96
5.12	Warehouse Service's Security Configuration	99
5.13	Warehouse Service's Authorization Configuration	100
A.1	Warehouse Service's SoaML Model	128
A.2	Composition of Retailer's QoS and UML Models	129
A.3	Composition of Retailer's SecureUML and UML Models	130
A.4	QoS Policies to WS-Policy Mapping	131
A.5	Resulting Retailer's IMM Model	132

List of Tables

2.1	OMG's Standards related to model-driven technologies.	17
2.2	Comparison of UML and DSL	28
4.1	Security Sub-Characteristics in Literature.	52
4.2	Security Intents Common Vocabulary.	53
4.3	Parts of the IMM metamodel.	59
4.4	CIM to PIM Mappings	69
4.5	QoS to iMM Mapping.	71
4.6	UML's SoaML Profile to iMM Mapping.	71
4.7	iMM to WSDL Mapping.	72
4.8	iMM to WS-Policy Mapping.	72
4.9	Access Control Propagation Conflicts	74
5.1	Warehouse & WarehouseCallback Services	81
5.2	Security Requirements	83
5.3	Access Control Requirements	84
5.4	Warehouse Service's SoaML Model Elements	88
5.5	QoS Catalog for the QoS Category "Security"	89
5.6	Annotated Non-Functional Models	90
5.7	Warehouse Service and QoS Models Composition	92

Listings

B.1	iMM2WSDL.AtlanZoo.doc.lit.wrapped.atl	134
B.2	appContext.mt	143
C.1	SecurityConnector.java	148
C.2	WeekDaysConstraintVoter.java	151
C.3	Application Context (Security)	152
C.4	Access Control Context	159
C.5	Warehouse Policy	160

Chapter 1

Introduction

1.1 Motivation

Model-driven development (MDD), and Model-Driven Architecture (MDA) (OMG, 2003) (the Object Management Group’s (OMG) initiative) in particular, are new paradigms that promote the use of system models as primary artifacts in the development process. These approaches support the application of best practices, patterns, and reuse in the development of families of systems, leveraging the development of software systems of much further complexity than before.

Increments in productivity, a reduction in cost and development time, and improvements in the resulting quality are counted among the benefits of MDD/MDA. However, non-functional (NF) properties or concerns (such as security, adaptability, quality of service, etc.) have not been sufficiently addressed by these approaches.

On a different token, the Internet has become the tool “par excellence” for business and information exchange between companies. In this context, service-oriented architectures (SOA) and web services (WS) (IBM, 2007) have emerged as the most suitable platform for application-to-application interactions.

SOA has become popular because it supports the reuse of pre-existent applications, and allows for interoperability between heterogeneous technologies and systems. This generates benefits, such as a rise in productivity, greater agility and flexibility for developing systems, and a better alignment between business objectives and the infrastructures supporting them. In their research roadmap (Papazoglou et al., 2008) for service-oriented computing, Papazoglou et al. remark the need for methodologies supporting the specification and design of services and service compositions, associating the design methodology with business process modeling techniques. *QoS-awareness* is also stressed as another grand challenge in the future of service-oriented architectures.

According to (Toma et al., 2008), three different aspects must be considered when talking about services:

1. *functional*, containing the formal specification of what exactly can the service do,
2. *behavioral*, describing how the functionality of the service can be achieved,
3. and *non-functional*, capturing constraints over the previous two.

Among these three aspects of a service description, the functional and behavioral aspects are the most investigated ones (Toma et al., 2008, page 4), even though the high relevance of NFPs for all service related tasks. This thesis focuses on precisely this area, the consideration of non-functional aspects of services, addressing them from the early stages of development on. Security and access

control, two NF concerns of growing importance, have been selected as the two properties employed to illustrate on the usefulness of the approach presented in this dissertation.

Recently, new proposals have risen that specifically focus on the area of non (or “extra”) functional concerns. Many of these proposals are based in technologies pre-existent at programming level, such as aspect-oriented programming (AOP) (Gregor Kiczales et al., 1997). This is the origin of aspect-oriented modeling (AOM) or early aspects (EA) (Elrad et al., 2002; Rashid et al., 2002), where cross-cutting characteristics of software are modeled as elements that intercept or infiltrate the base code. Other, more generic approaches also exist. An example of those are Viewpoints (Finkelsetin et al., 1992) or Multi-Dimensional Separation of Concerns (MDSoc) (Tarr et al., 1999), where the system is the result of a set of views or diagrams that describe its different characteristics from different perspectives (e.g., behavioral perspective, non-functional perspective, etc.). All these approaches share common ideas, and allow for the separation of functional and extra-functional characteristics of the systems *at design time*.

Model-driven initiatives tackle problems caused by system size or complexity by raising the level of abstraction from code-level to model-level. For (Schmidt, 2006), instead of general-purpose notations (which are claimed to rarely express application domain concepts and design intent), Domain Specific Modeling Languages (DSMLs, a.k.a. metamodels) can be tailored to express with great precision the domain’s semantic and syntax. These languages reduce the available modeling elements to key, familiar problem-domain concepts (and rules), narrowing down the design space (Kelly and Tolvanen, 2008), to achieve higher levels of productivity. Developers can then concentrate on defining the problem instead of implementation-level details (Pitkänen and Mikkonen, 2006).

Even more recent is the emergence of middleware platforms with aspect-orientation support. Such platforms allow for the separation of NF concerns from the functional bulk of the system. The more advanced middleware software has been designed specifically to leverage characteristics such as adaptability, flexibility, robustness, and security. Declarative configuration files indicate the way in which such platforms integrate functional and extra-functional pieces of code, allowing for an independent design and development of such aspects.

The challenges, and the approaches and methodologies enumerated above, adapted to supporting the inclusion of non-functional concerns development into SOA systems, are the engines propelling this research.

1.2 Research Problem

Service-oriented architecture’s modeling is an area in which, although multiple proposals have been put forward by tools vendors, a consensus has yet to be achieved. SOA is the common reference for many in-development standards and execution frameworks and, therefore, special attention is being paid to it. It has not been until very recently (OMG, 2008c) that non-functional aspects of services have been considered for standardization processes. Thus, no solution in this area is expected within the next few years.

Although multiple implementation technologies exist to facilitate the development of web services and SOA systems, the lack of a sound methodological base for the development of such applications stresses the need for new modeling methods or techniques that could guarantee the quality of the development of this type of systems. For instance, as presented in (Juric et al., 2010, Chapter 6, “Securing BPEL Processes”), 16 steps which include more or less technical questions (e.g., Security header layout) are necessary in IBM’s Websphere (IBM, 2011a) latest version (at the time of writing) to generate a policy requiring a UsernameToken for a service request. Furthermore, the average web services user/programmer doesn’t know if a UsernameToken is the appropriate kind of identification token to require, or if it represents a suitable security configuration. Service modeling standards, such as (OMG, 2009), make no suggestions on this respect. WS-Policy is defined at

a technical level, for a developer role. The same approach may not be appropriate for business analysts.

In particular, it was identified that there exists a lack of a design solution that:

- Allows an independent design of the functional and non-functional concerns of service-oriented architectures, fostering reuse.
- Permits that each concern be addressed in a convenient manner.
- Is modular enough to avoid the pollution between functional and non-functional information, and to maintain confidentiality of design at the architectural level.
- Provides a practical way of measuring and analyzing the system for non-functional characteristics in early stages of the development.

The independent design and development of functional and NF concerns leverages variability, such as in software product lines (Clements and Northrop, 2001), and allows for the reuse of both functional and non-functional designs. Domain-specific models, much closer to the problem domain, provide an appropriate mechanism to describe an specific concern conveniently. At the architectural level, the use of separate design models eliminates the need of non-functional expertise for system designers, and facilitates the maintenance of the secrecy of the NF design, if desired. Finally, integrated models support the execution of analysis and measurements of both functional and NF aspects of the system (for, among other things, validation of design). This thesis proposes that, at each level, an appropriate technicality be used, and for each concern, an appropriate metamodel.

1.3 Covered Areas

1.3.1 SOA/WS

In (OMG, 2009), the standard defines SOA as “a way of describing and understanding organizations, communities and systems to maximize agility, scale and interoperability”.

According to IBM (IBM, 2007) “Service-Oriented Architecture (SOA) is an IT architectural style that supports the transformation of your business into a set of linked services, or repeatable business tasks, that can be accessed when needed over a network”. These services can automatically interconnect between themselves, if needed, forming “on demand” applications, connecting groups of service providers and consumers in a way that allows business to adapt to the changing conditions and requirements. In some occasions, this can even happen without human intervention.

Among the benefits of SOA one can find a better business-IT alignment, and loosely-coupled component-based software systems (IBM, 2007). Component-Based Software Development (CBSD) is the modularization strategy on which the implementation of services is commonly based. Components are behind these services, providing its functionality; but components were already there before. Service-orientation incorporates the great benefits for which the IT industry had been searching a long time.

Moreover, service-oriented architectures offer a network-based infrastructure, permitting geographically disperse and technologically diverse resources to work together as one, creating “on-the-fly” applications on demand. SOA also allows for greater code reuse, a better standardization of company’s processes, and a simpler centralized corporate control (IBM, 2007) .

Web services is the technology that has risen to the challenge of supporting service-oriented architectures. It provides an standard-based framework that makes the interaction between applications through the web possible. In the last few years, most enterprises and organizations have been using different approaches that would allow them to connect to other organizations

through the infrastructures provided by the Internet. In this environment, web services emerged, offering a systematic and extensible XML-based framework, and built on top of existing Web protocols. Web services standards define mechanisms for the description, location, reliability, transactions, and communication between on-line applications.

When applying MDA principles to the development of Web Services, the outcome of the resulting development process would be the automatic generation of (in the particular case under research) Web Services Description Languages (WSDL) descriptors and WS-Policy documents, as well as the (skeleton) code implementing the services. The use of WSDL supports the description of the service in a platform-neutral fashion, freeing the service itself from the platform in which it will be deployed. Policies (WS-Policy documents), in turn, provide an standard mechanism for expressing the non-functional characteristics of the service. As (Arsanjani, 2004), we believe that services should ideally be governed by declarative policies and thus support a dynamically re-configurable architectural style.

The concepts that conform the MD development of service-oriented architectures, i.e.:

- Business Processes
- Components
- Web Services
- Non-Functional Policies

are each one addressed in this dissertation.

1.3.2 MDD, MDA, DSM

Model Driven Architecture (MDA) is a relatively new approach towards software systems design and development. It promotes the efficient and effective use of system models in the development process. In addition, it allows for best practices and patterns reuse in the development of families of systems.

The OMG defines MDA as a form of organizing and managing enterprise architectures, supported by services, and tools to define and transform models. MDA is a means for augmenting software development productivity, and improving the quality and longevity of the resulting system. The primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns (OMG, 2003; Miller, 2009; Asadi and Ramsin, 2008).

MDD, and particularly MDA, carry with them many desirable properties. Among these properties, three could be considered fundamental:

- increase in productivity,
- portability (reusability),
- interoperability.

These three properties are of great importance for many systems, particularly for service-oriented systems, which are the focus of this research. SOA systems' implementations have major differences depending on their target platforms, for instance. MDA allows the system developers to focus on its functionality, rather than in the platform-dependent details, freeing them from this burden. Consequently, a more precise solution, finely adjusted to the system's requirements may be achieved in a shorter time. Likewise, the resulting system's design could be reused in the future, as technology progresses, thanks to its platform and implementation independence.

Domain-specific modeling (DSM) (Kelly and Tolvanen, 2008) is another paradigm that fits under the general umbrella term "model-driven engineering". DSM is conventionally regarded as a means to raise the level of abstraction in software development (Pitkänen and Mikkonen, 2006; Gray, 2002; Kelly and Tolvanen, 2008). DSM languages reduce the available modeling elements to key, familiar problem-domain concepts (and rules), narrowing down the design space (Kelly and Tolvanen, 2008), to achieve higher levels of productivity.

Figure 1.1: A system may be visualized as a polyhedron, each face showing a different view.

1.3.3 Non-Functional Concerns Modeling

A system can be visualized as the composition of multiple functional and non-functional concerns, each one composing a different view of a whole. Imagine them as forming a polyhedron, where each face shows a different view. Each concern may be described in a different view model, by means of a particular notation that helps to more clearly express the concepts which are of importance in this particular perspective. These perspectives, or views, present the properties of a system which are considered relevant at any point in time (Brown, 2004). The criteria applied in this thesis is that some of these views/perspectives could describe non-functional concerns of the system (different software development paradigms share this idea), in this case to, among other things, avoid the need of non-functional expertise on the system modeler, allow for privacy/secretcy of designs (if desired), and improve productivity by reuse of functional and non-functional designs.

Multidimensional separation of concerns (MDSoc) (Tarr et al., 1999) introduces the concept of “hyperslice”, described as “intended to encapsulate concerns in dimensions other than the dominant one”. A system is a collection of hyperslices that are to be composed together to conform the complete system. This type of separation of concerns provides the capacity to capture several views of a single class, permitting a better understanding of the implementation of each view in isolation (Gray, 2002).

Aspect-oriented programming (Gregor Kiczales et al., 1997), which has been around for a while, has recently redounded, thanks to its maturity, in attempts to make the most of this approach in earlier stages of software development. These attempts (e.g., AOM or EA (Elrad et al., 2002; Rashid et al., 2002)) represent currently very active research areas.

(Waddington and Lardieri, 2006; Hessellund, 2009; Brown et al., 2005) predict the use of several views to represent the different concerns that form a system, or to fulfill different roles within the development. Furthermore, the use of different, more suitable domain-specific languages for each concern or view has been proposed. (Hessellund, 2009) argues that multiple domain-specific modeling languages are needed to better suit each individual view, and that the challenge is “integrating these different views and avoiding the potential cacophony of multiple different languages”. This assembly of multiple domain-specific views is also anticipated by (Kelly and

Tolvanen, 2008), (Vallecillo, 2010), and (Oberortner et al., 2008).

Integrated models not only provide great means for the inclusion of non-functional concerns into systems' design, but also present themselves as the perfect subjects for the analysis and measuring of such characteristics. A sound model-driven development strategy must not leave out the early evaluation of designs with respect to extra-functional properties.

1.4 Proposed Approach

From within the technologies mentioned above, model-driven development and service-oriented architectures are currently very much accepted both in academics and industry (in the latter, SOA has much greater penetration than MDD/MDA). However, the development of extra-functional aspects as system's independent constituent blocks is slowly migrating from an exclusively academic subject towards corporate/industrial evaluation of its usefulness. Research on modeling/design of such independent non-functionality blocks is relatively recent.

The application of model-driven techniques on SOAs is anything but novel: standards and its implementations are available, and research and industrial experiences have been published on the subject. Nonetheless, the communion between the model driven development of non-functional aspects as independent building blocks, which are later combined to result in a complete SOA system is quite innovative. This is the major research subject of this thesis.

This thesis proposes the application of the separation of the design into multiple concerns (may them be called views, aspects, or any other alternative name) to integrate the development of non-functional properties of service-oriented systems in a modular, reusable, and maintainable manner. Such a combination of technologies generates really interesting benefits. For instance, model-driven development provides platform and implementation independence, and facilitates design reuse. On the other hand, service-oriented architectures allow for the creation of loosely-coupled, geographically disperse, on-demand systems. Finally, MDSoC supports greater code/design modularization, leveraging understandability and maintainability of the design.

This dissertation sets out the study of a possible approach for the modeling of software services, with regard of extra-functional aspects. The most prominent state of the art proposals for incorporating non-functional concerns into systems' design are considered in this work. Analysis and measuring possibilities are evaluated for model-driven development approaches. The use of standard modeling languages, in particular extensions to the Unified Modeling Language (UML) (OMG, 2010e) in the form of profiles, is compared to alternative modeling approaches, such as domain-specific modeling. Strengths and weaknesses of both are weighted up, resulting on the use of one or the other approach where most suitable. Comparatively, available open source modeling tools are evaluated for its usefulness in the area. A prototype of a development framework, and its integration with execution environments is part of the contributions of this thesis.

1.5 Contributions

This thesis provides a variety of contributions to the field of the model-driven development of the non-functional concerns of service-oriented architectures. Such contributions may be categorized as theoretical (adds to the theory of the research field), practical (provides tools or mechanisms that may be applied), or methodological (introduces guidance on how to perform an specific task). These contributions are enumerated in this section.

1.5.1 Theoretical Contributions

The theoretical contributions of this thesis include:

- An state of the Art review of SOA/WS Modeling. [Section 6.4, and (Silva Gallino, 2006)]
- An state of the Art review of Non-Functional Concerns Modeling. [Section 6.1 and (Silva Gallino, 2008)]
- The definition of an analysis and a metric for iMM models. [Section 4.6]
- The definition of a WS-Policy Metamodel. [Section 3.3.4]
- The definition of a WS-SecurityPolicy Metamodel. [Section 4.3.5]
- The definition of an integrated metamodel (iMM) for the particular case of security and access control in SOA/WS systems. [Section 4.3.6]
- An state of the Art review of Model Composition Tool Support. (Silva Gallino et al., 2009a)

1.5.2 Methodological Contributions

With regard of the methodological contributions made by this thesis, those include:

- The definition of a methodology incorporating concepts extracted from MDA, MDSoc, AOM, and DSM for the inclusion of NF properties in the modeling of SOA/WS. [Section 3.2.2]
- The use of high level abstract concepts and ontologies for the discovery an selection of NF models. [Section 4.3.1]
- The combination of functional and non-functional models into a complete system model. [Section 3.2.2]
- The generation of intermediate models (through model transformations) for development and analysis. [Sections 3.2.2 and 4.6]

1.5.3 Practical Contributions

Finally, with respect to practical contributions, this thesis provides three:

- A prototype tool for the model-driven development of SOA/WS with regard of non-functional concerns. [Section 4.7]
- The implementation of the SoaML (OMG, 2009) UML profile [Section 3.3.2 and (Silva Gallino, 2007)].
- The implementation of the metamodels listed in sections 3.3 and 4.3.

1.5.4 Funding Acknowledgements: Participation in Research Projects

This research has been partially funded thanks to the participation in several research projects, most noticeable:

- ITECBAN (Infraestructura Tecnológica y Metodológica de Soporte para un Core Bancario) (CENIT), Spanish Industry Ministry (2006-09).
- MODELWARE (MODELLing solution for softWARE systems) project (IST-511731), under the “Information Society Technologies” Sixth Framework Program (2002-2006).
- RT-MODEL (Real-time platforms for model-driven design of embedded systems) project (TIN2008-06766-C03-01), under the “Plan Nacional de I+D+I” (2008-2011).

1.6 Dissertation Organization

This dissertation is structured as follows. This chapter has provided an introduction to the research foundations and objectives. Chapter 2 presents the background to this research among the different areas through which it spans: service-oriented architectures, model driven methodologies, the modeling of non-functional concerns of systems, model transformations and compositions and current tool support, etc. The proposed methodology for the integration of non-functional properties (NFPs) in the modeling of service-oriented systems is presented in chapter 3, its particular implementation for security and access control as non-functional characteristics in chapter 4, and a well-known use case (WS-I, 2003a) demonstrating its application is provided in chapter 5. Most relevant related work is introduced in chapter 6. Finally, a summary, conclusions, and possible future lines of work are presented in chapter 7. The appendixes include the different inputs (appendix A), some transformation templates (appendix B), and relevant technical results (appendix C) of this work, placed there to facilitate referencing.

Chapter 2

Background

2.1 Model-Driven Approaches

2.1.1 Setting the Cornerstone

One of the main objectives of software engineering is to increase productivity, and in that way shorten the time to market (Lukman and Mernik, 2008). All through software development history, greatest productivity leaps have been tightly tied to a raise in the level of abstraction. Raising the level of abstraction not only has incidence on the time needed for making the product, but also the time implied in maintaining it (Kelly and Tolvanen, 2008). This was the case when third generation languages raised on top of assembler, raising productivity up to 450%, according to some claims (Kelly and Tolvanen, 2008). In the words of (Mellor et al., 2004), the idea behind model-driven approaches is to “..raise the level of abstraction through modeling, and also increase the level of reuse, and in that way improve productivity”.

(Czarnecki, 2005) introduces solution and problem spaces. The problem space is defined as a set of abstractions for expressing application’s domain concepts, and contains the abstract concepts from the domain that one is working on (Hessellund, 2009). Solution space, on the other hand, deals with of implementation-oriented abstractions (Czarnecki, 2005), representing the realization of the problem-domain’s concepts in a concrete technology (Hessellund, 2009).

Many (Mellor et al., 2004; Kleppe et al., 2003; Miller, 2009; Schmidt, 2006) compare model-driven initiatives with the advent of code compilers and the rise of third-generation languages (3GLs). The base of model-driven approaches is abstracting the *problem space*. The abstractions provided by 3GLs, however, focus on implementation technologies: the *solution space* (Schmidt, 2006). Figure 2.1, shows how the level of abstraction has evolved over time. Nowadays, development focus is on code, most likely written in an object-oriented programming language (Kleppe et al., 2003).

Computer-aided software engineering (CASE) emerged in the 1980’s. CASE tools aimed at providing a graphical means of simplifying software development, whilst also generating implementation artifacts. In this way, the effort of coding and debugging was to be reduced (Schmidt, 2006). However, CASE tools failed in providing a leap forward in productivity similar to that of 3GLs, according to (Greenfield and Short, 2003) because “...they were based on proprietary modeling languages, metamodels and ineffective code generators”.

Further improvements have appeared since CASE tools. Application frameworks, for instance, currently provide reusable middleware services, avoiding the need of re-implementing them (Schmidt, 2006). However, some challenges are still to be faced. The ever-increasing, rapidly growing complexity of systems, moving faster than software development technologies can cope with, is one of the most important. The fast evolution of platforms and the appearance of new ones,

Figure 2.1: *Evolution of the Abstraction Level Through the Years.* (Mellor et al., 2004, p. 24)

(Schmidt, 2006) claims, has developers spending a great amount of effort on porting legacy applications. Model-driven reuses the idea of compilers. According to (Mellor et al., 2004), model transformations tools represent the core of the paradigm. They free developers from having to worry about low level elements. Each time the abstraction level raises, the platform on which this new layer depends changes. For model-driven, the new layer is the independence of software platforms (Mellor et al., 2004).

Separation of concerns (Dijkstra, 1982) is a basic idea for current software engineering. A concern can be defined as “some piece of a problem whose isolation as a unique conceptual unit results in a desirable property” (Gray, 2002) or, nicely put in the words of (Nelson et al., 2001) “a slice through the problem domain that addresses a single issue”. Conformant with the ideas of (Kent, 2002), identifying different use cases corresponds to identifying different subject areas. Different subjects areas can be seen as different concerns.

Multidimensional separation of concerns (MDSoc) (Tarr et al., 1999) introduces the concept of “hyperslice”. They describe hyperslices as “intended to encapsulate concerns in dimensions other than the dominant one”. A system is then a collection of hyperslices that are to be composed together to conform the complete system. This type of separation of concerns provides the capacity of capturing several views of a single class, supporting a better understanding of the implementation of each view in isolation (Gray, 2002). These views are composed together by means of a compositor, a concept similar to that of a weaver in aspect-oriented jargon. Model composition and weavers are described in more detail in section 2.1.2.

According to (Liang, 2009), the main technologies and methods supporting and accelerating the development of MDD are:

- Component-Based Software Development (CBSD).
- Design patterns.
- Middleware.
- Application Framework technology.
- Declarative Specification.
- Design by Contract.
- Object-oriented method.

The view in this thesis is that, besides the previously enumerated technologies, the concepts in MDSoc are also called to play an important paper in model-driven approaches.

For (Kelly and Tolvanen, 2008), software development can be divided in different steps:

1. Thinking of a solution in terms of the problem domain.
2. Map this solution into a specification of some language (platform).
3. Finally, implement it into code.

In their view, is quite noteworthy that the first step has traditionally been performed without any tool support, especially when errors in this phase of development transform into the most expensive to solve. Model-Driven approaches come to address this and other different shortcomings of the current methodologies, such as:

- Productivity.
- Portability.
- Maintenance.
- Reusability.
- Interoperability.
- Documentation.

Domain specific languages (DSLs) have been around for almost as long as computing has been done (Fowler, 2010). However, in the last few years (and, according to (Fowler, 2010), probably because of Ruby, Rails, and its many features that make it easy to develop DSLs) the use of DSLs have picked up. To (Fowler, 2010), Rails in particular uses several DSLs which have played a big role in making it so easy to use, in turn encouraging more people to take up these ideas. When it comes to DSLs, he claims, it is the reading that matters more than the writing. By facilitating domain experts to read, and mostly understand, the code that drives a key part of their business, then communication with the developer greatly benefits.

Theres is a clear evolution of software engineering techniques, in the research field primarily, towards the use of model-driven methodologies and domain-specific approaches. This type of methodologies and approaches provide great benefits on areas such as productivity, reusability, and maintainability, and therefore are adopted in this thesis. Nevertheless, using pure MDA or DSM approaches suffer from the lack of properly supporting the inclusion of multiple non-functional aspects of systems. This thesis incorporates MDSoc and AOM elements, the vision of a system being composed of different concerns, each one addressed in a proper view, and adds this to the equation.

2.1.2 Terminology

This section provides definitions for the different terms and elements that compose a model-driven development approach. On occasions, different authors disagree in certain aspects of the definition of these elements, having different views. There, a position is taken, stating which is the view adopted in this dissertation. At the end of the section, a summary is provided in table 2.1, mapping these different elements to standard OMG technologies.

Models

The fundamental concept in model-driven approaches is that of models. To (Mellor et al., 2003), a model is a coherent set of formal elements describing something (for example, a system), built for a determined purpose, addressing a number of subjects, and not needing to be complete. In this way, a model may have multiple views, only some of which are exposed. For (Kleppe et al., 2003), a model is an abstraction of something that exists, and is different from the thing it models (e.g., details are left out or its size is different). Additionally, they assert that a model is “a description of (part

of) a system written in a well-defined language”, and a well-defined language is “a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer”. A model can be described as an abstraction of a system, generated for a certain purpose, often presented a combination of drawings and text (Liang, 2009; OMG, 2003). Those purposes include, but are not limited to (Mellor et al., 2003; Brown, 2004; Brown et al., 2005):

- Communication of ideas between stakeholders and machines.
- Completeness checking.
- Race condition analysis.
- Prediction of system qualities.
- Test case generation.
- Analysis of specific properties.
- Viability in terms of indicators such as cost estimation.
- Standards compliance.
- Transformation into an implementation.

As stated by (Brown, 2004), models allow developers to reason about a system by ignoring extraneous details while focusing on the relevant ones. They remark that all forms of engineering rely on models as essential to understand complex real world systems.

In brief, a model can be considered an abstraction of something (in this case, a system), which highlights the most relevant parts of it, aiding understandability, and subject of analysis and (semi) automated transformation into implementation.

MetaModels

Elegantly put by (Clark et al., 2008), all languages have a concrete syntax, which defines how they are presented, an abstract syntax that describes the concepts of the language, and semantics that describes what the concepts mean. The abstract syntax defines the concepts of a language and their relationships, while the concrete syntax defines the physical appearance of language. For a textual language, this means that it defines how to form sentences. For a graphical language, this means that it defines the graphical appearance of the language concepts, and how they may be combined into a model (den Haan, 2009).

(Brown, 2004) remarks that the ability to analyze, automate, and transform models requires a clear, unambiguous way to describe their semantics. One way in which to describe the syntax and semantics is through metamodels. A metamodel is a model of all these different aspects of a language. Metamodels define the structure, semantics, and constraints for a family of models (Mellor et al., 2004).

In the simplest of views, a metamodel can be pictured as a model of a modeling language (Clark et al., 2008; Mellor et al., 2004): a model corresponds to a particular metamodel. For example, a model that is being described by Unified Modeling Language’s diagrams is captured by the UML metamodel, which describes how UML models can be structured, the elements they can contain, and the properties those elements exhibit (Mellor et al., 2004).

The paper of metamodels in model-driven approaches is a point of discussion. Kleppe, in (Kleppe et al., 2003), acknowledges them, but almost exclusively refers to the UML; (Mellor et al., 2004) notes their growing importance, while they represent the cornerstone in (Kelly and Tolvanen, 2008).

It can be concluded that the perceived usefulness of having multiple metamodels has increased over time, and is directly related to the importance granted by the different authors to the UML. It is almost universally accepted that UML is the “de facto” standard modeling language of choice nowadays. For (Mellor et al., 2004), the existence of this standard, (reasonably) well-defined language reduces the likelihood of misinterpretation by the viewers of models, but in the longer term, domain-specific languages are likely to be an important alternative to the UML. UML is described in more extension in section 2.1.3.

Sometimes metamodels are referred to as Domain-Specific Modeling Languages. According to (Balasubramanian et al., 2006), a DSML is itself a model that defines all the possible models that developers can build using it. We share the view of (Kent, 2002) in that models may have different perspectives, and that those often (although not always) require modeling languages with different properties. This is when the usefulness of multiple domain-specific metamodels comes into play. Even the OMG (the standardization body behind UML) acknowledges the importance of having different metamodels, the result of this being the specification of a standard language for expressing metamodels, the Meta-Object Facility (MOF) (OMG, 2011b).

According to (Karlsch, 2007), the advantages of using a metamodel to describe domain-specific modeling languages include:

- Lower development effort for visualization and editors
- Model verification and checking according to the metamodel
- Extensibility of the metamodel
- Standardized exchange formats
- Models comparison based on metamodel
- Description of the semantics of the language

We agree with (Clark et al., 2008) in that creating a metamodel for a language is not simple. They identify a sequence of steps for the definition of a metamodel. Those steps are:

- defining abstract syntax
- defining well-formedness rules and meta-operations
- defining concrete syntax
- defining semantics
- constructing mappings to other languages

MOF OMG’s Meta Object Facility (MOF) is an standard “meta-metamodel”, a metamodel for defining metamodels. OMG’s metamodeling architecture is sometimes referred to as closed, because the defined metamodel (the MOF) conforms to itself (Karlsch, 2007). In version 2.0 of the standard, the MOF has been divided in pieces: the Complete MOF (CMOF), the Essential MOF (EMOF, holding a subset of the most important elements in CMOF) and the Semantic MOF. In the words of (Karlsch, 2007), the MOF is the foundation for OMG’s Model Driven Architecture (MDA), and the Unified Modeling Language (UML) and all its different diagram types are defined conforming to the MOF.

Ecore ECore can be said to be the Eclipse Modeling Framework’s (EMF) (Steinberg et al., 2009) implementation of EMOF. It is used as the meta-metamodel of all models in EMF. It “can be said to be” because, strictly speaking, ECore is not EMOF, although they are strongly aligned (Karlsch, 2007). In the same way as the MOF, ECore is described in ECore itself.

Abstract and Concrete Syntax Generally, a language is defined by its *abstract* syntax, its *semantics*, and as much *concrete* syntaxes as necessary. The abstract syntax, what is usually considered as the metamodel ‘per se’, defines the elements of a modeling language, and the relations and restrictions, but not the concrete syntax, the physical representation of the data.

While the abstract syntax specifies how a DSML structure should look, a concrete syntax defines the representation of the abstract syntax. Mappings from MOF to middlewares, languages, and exchange formats have been defined, in particular to CORBA, XML Metadata Interchange (XMI), and Java Metadata Interface (JMI). These mappings allow generators to automatically transform a metamodel’s abstract syntax into a concrete representation. A concrete syntax allows a parser to build an abstract syntax tree.

Translating all this into the Eclipse Modeling Framework, the abstract syntax (metamodel) is defined as an instance of the Ecore metamodel. On the other hand, Eclipse’s Graphical Modeling Framework (GMF) is used to develop a graphical concrete syntax. A newer Eclipse project, named Textual Modeling Framework (TMF), is focused on the textual concrete syntax. By default, the concrete syntax of Ecore models is represented in XMI. Figure 2.2 provides a graphical depiction of the relation between the different syntaxes.

Figure 2.2: Relation between abstract and concrete syntaxes. (OMG, 2010a, p. 4)

Expression Languages In occasions, although less frequently in domain-specific metamodels, it is necessary to specify constraints or invariant conditions that must hold for the system being modeled or queries over objects described in a model. Expression languages are used to describe such conditions. Perhaps the most representative may be the Object Constraint Language (OCL) (OMG, 2010c), a formal language used to describe expressions on MOF/UML models. Modelers can use OCL to specify application-specific constraints in their models, or to specify queries on the model, which are completely programming language independent.

Model Transformations

(OMG, 2003) describes model transformations as “the process of converting one model to another model of the same system”. As described by (Brown, 2004), is often necessary to convert between different views of the system at an equivalent level of abstraction, or across different levels of abstraction, usually from a more abstract to less abstract view by adding more detail supplied by the transformation rules, or additional information in the form of mappings.

Mappings between metamodels are the specification of a transformation (OMG, 2003). They represent rules describing how one or more constructs in the source language can be transformed into one or more constructs in the target language (Kleppe et al., 2003). Mapping functions are the way in which model-driven development captures expert knowledge, decoupling the several models so that each can evolve independently (Mellor et al., 2003). These mapping rules are described at the metamodel level, making mappings applicable to all sets of source models that conform to the given metamodel (Mellor et al., 2004; OMG, 2003; Ameller, 2009).

The degree of automation of a transformation is also a point of disagreement. (Kleppe et al., 2003; Schmidt, 2006) see automatic transformations as fundamental for MDA, whilst (Mellor et al., 2004; Brown, 2004) doesn’t consider automation of the transformations of the foremost importance. In this point, we agree with (Kleppe et al., 2003) in that automatic transformations are vital for the usefulness of model-driven approaches, although we coincide with (Brown, 2004) in that not all applications of MDA will be able to benefit from complete end-to-end automated model transformations, nor require it.

(OMG, 2003) also remarks that quality of service (non-functional) requirements may sometimes be incorporated into transformations. They claim that specific transformation choices can be made according to the particular quality requirements. This is of direct application to the subject of this thesis.

There are many ways in which a transformation may be done. For (Brown et al., 2005), in practice there are three common model transformations:

- *Refactoring transformations* reorganize a model according to some well-defined criteria.
- *Model-to-model (M2M) transformations*, convert information from one model or models to another model or set of models, typically where the flow of information is across abstraction boundaries.
- *Model-to-text (M2T) transformations*, which are familiar to any one who has used the code generation capability of a UML modeling tool.

Transforming a model to code is often referred to as “code generation”, and the tools that implement them simply as “generators” (Lukman and Mernik, 2008; Kelly and Tolvanen, 2008; Karlsch, 2007). Generators not only generate code, but they can also produce tests, simulations, or metrics (Kelly and Tolvanen, 2008; Karlsch, 2007). Also known as model to text (M2T) transformations, these are generally performed by scripting or template-based languages (Ameller, 2009).

Model Compositions

Model composition, a special kind of transformation, is a combination of two or more models to generate a different one. Although generically referred to as *model merging*, (Barais et al., 2008) distinguishes between the *merging* of two models of the same language (homogeneous), and the *weaving* of two models of different languages (heterogeneous).

Merging of different models is considered in this thesis as an operation that takes two models as input, a mapping between them, and that produces a new output model that contains all the elements of the input models, but no additional information (Didonet and Valduriez, 2009). A composition, on the contrary, is a much general activity, that may include other actions besides merge. Such actions, or “element composition directives”, include (Reddy et al., 2006):

- Creating new model elements.
- Adding model elements to a namespace.
- Removing model elements from a namespace.
- Changing properties.
- Replacing references to a model element in a namespace.
- Overriding model elements.

There are several model-driven approaches that are based on the composition of models, aspect weaving being the most well-known case, although still just a particular case of it. Nevertheless, the tools that perform model compositions are currently mostly referred to as *model weavers*. (Brunet et al., 2006) discusses model composition in more detail.

Model weaving is critical when systems are developed as a composition of different concern models or viewpoints. Section 3.2.2 proposes solutions to this problem in the context of this thesis.

Viewpoints

Viewpoints (Finkelsetin et al., 1992) recognizes the existence of multiple perspectives in a system under development. In the words of (OMG, 2003), “a viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system”. The authors clarify that the meaning of ‘abstraction’ in this definition is the process of suppressing selected detail to establish a simplified model. It can be considered as a subdivision of the complex specification of the system (Gervais, 2002). There are many definitions, some included in (Gray, 2002, page 88), but most of them describe viewpoints as “partial description”, or associated to a “concern” (or set of concerns). The notion of views/viewpoints is a key part of the IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE, 2000).

When models grow, in size and complexity, into an entity in which it is “unmanageable in its entirety” (Gray, 2002), the division of the model into multiple parts becomes necessary. As previously mentioned in section 2.1.1, a system can be visualized as the composition of multiple concerns. Each concern may be expressed by a model, by means of a particular notation that helps to more clearly express the concepts which are of importance in this particular perspective. These perspectives, or views, present the properties of a system which are considered relevant at any point in time (Brown, 2004).

(Vallecillo, 2010) studies the problem of the combination of domain-specific modeling languages expressing different views. In this thesis, the composition of models that is performed may be categorized as a *viewpoint unification* (Vallecillo, 2010). There, a unified model is created, correspondent to a new metamodel or DSML, and a set of mappings are defined to combine the different viewpoints in it.

Table 2.1: *OMG's Standards related to model-driven technologies.*

Metamodeling Language	Meta Object Facility (MOF) (OMG, 2011b).
Modeling Language	Unified Modeling Language (UML) (OMG, 2010e), Business Process Model and Notation (BPMN) (OMG, 2011a).
UML Profiles	UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms (OMG, 2008d), Service oriented architecture Modeling Language (SoaML) (OMG, 2009).
M2M Transformation	MOF Query/View/Transformation (QVT) (OMG, 2008a).
M2T Transformation	MOF Models to Text Transformation Language (OMG, 2008b).
Graphical Notation Interchange	Diagram Definition (DD) (OMG, 2010a).
Expression Language	Object Constraint Language (OCL) (OMG, 2010c).
Concrete Syntax	MOF 2 XMI Mapping (XMI) (OMG, 2011c).

2.1.3 Model-Driven Initiatives

Different names are used to refer to model-driven initiatives: Model-centric software development, Model-Driven Software Development, Model-Driven Engineering (MDE), Model-Driven Development (MDD), Model-Driven Architecture (MDA), and the list goes on. Many times, those names refer to the same idea, the use of models as the building artifacts in the development of a software system, whilst in some other occasions there is a subtle difference in the different names. This section tries to clarify such commonalities/differences.

First we need to clarify what we mean by “model-driven”. The MDA Guide (OMG, 2003) appropriately defines model-driven as an activity that “provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification”. By means of this definition, the focus will be placed on four different expressions of model-driven initiatives: Model-Driven Engineering, Model-Driven Development, Model-Driven Architecture, and Domain-Specific Modeling (DSM). One view of the relation among these initiatives is the one presented by (Ameller, 2009), and which is visualized in figure 2.3.

Based on the publications introducing these concepts, (Ameller, 2009) describes MDE as being the software engineering area, MDD as the software development process, and MDA as a particular instance of MDD, standardized by the OMG. In this dissertation, that definition is both agreed and disagreed upon. Agreed in that, as defined in the different publications, figure 2.3 shows the actual relation of MDE, MDD, and MDA (DSM is not considered in (Ameller, 2009)). However, it is disagreed in that the cause of such containment relationship (the more generic approaches containing the most specific ones) is due to the fact that MDA is tied to the use of UML and other OMG's standards (this statement is discussed in section 2.1.3). Rather, in this dissertation the reason of such containment relationship is considered to be that MDA specifies a somehow more specific abstract-to-concrete (although quite generic) process or methodology than those of MDE and MDD. The same argument is valid for the case of DSM (as defined in (Kelly and Tolvanen, 2008)). These last two approaches are perceived to be at the same abstraction level, as they both promote a different generic methodology for model-driven software development.

Figure 2.3 above presents the view maintained in this thesis of how these initiatives relate to each other. Nevertheless, there are times in which these acronyms (with the exception of DSM) are

Figure 2.3: A view of the Different Model-Driven Approaches. (inspired on the one in (Ameller, 2009, p. 13))

used interchangeably to refer to a model driven activity. This may have its origins in the fact that MDA and other related terms are registered trademarks from the OMG (OMG, 2010d).

According to (Kent, 2002), a model driven engineering approach must specify the modeling languages, models, translations between models and languages, and the process used to coordinate the construction and evolution of the models. Model-driven engineering is not a new idea, but currently the technology is in place to support it, and therefore we are witnessing an explosion on the amount of research, tools, and usage experiences around it.

(Karlsch, 2007) clearly distinguishes MDE as “an open and integrative approach not tied to a special standard”, being that the reason why many different implementations exist. In this view, MDE’s focus is on reuse, moving models into the center of the development process and out of mere documentation purposes. Models allow a more domain-specific description of problems than the achievable with third generation programming languages. For (Kent, 2002), MDE combines process and analysis with architecture, and is, by definition, wider in scope than MDA.

This vision is shared by (Lukman and Mernik, 2008), describing MDE as a software development approach having the potential to address software engineering’s current challenges. They also consider models as the primary development artifacts, and they are to be used in a formal and precise way. Moreover, they affiliate the view of (Karlsch, 2007) of the importance of domain-specificity. They even go farther, and declare Domain Specific Modeling Languages (DSMLs) and model transformations as the two main components enabling MDE.

The capability of model-driven initiatives to deal with platform complexity, and the rest of the challenges being faced by software engineering, is perceived by (Schmidt, 2006). These challenges could be overcome by combining:

- DSMLs, described by metamodels defining the relationships among concepts in a domain and

precisely specifying the key semantics and constraints associated with these domain concepts.

- Transformations and generator’s engines synthesizing various types of artifacts, such as source code, simulation inputs, XML deployment descriptions, or alternative model representations.

Domain-specific modeling languages’ role in model-driven engineering has grown in importance in the last few years, and will probably keep growing. For (Kent, 2002), the reason is that it is improbable that one limited group of models, transformations and process will be appropriate for all domains, companies, or projects.

For (Schmidt, 2006), instead of general-purpose notations (which are claimed to rarely express application domain concepts and design intent), DSMLs (metamodels) can be tailored to express with great precision the domain’s semantic and syntax. (Waddington and Lardieri, 2006) predicts the use of several views to represent the different concerns that form a system, or fulfill different roles within the development. The approach in this dissertation shares these views. Furthermore, it proposes the use of different, more suitable domain-specific languages for each concern or view.

One important activity in model-driven engineering is model validation (Waddington and Lardieri, 2006; Ameller, 2009; Schmidt, 2006). When domain-specific metamodels are used, these rules are included in the specification of metamodels (for instance, forbidden associations are simply not present in the MM). In this way, incorrect models are just not possible to construct, preventing many errors early in the life cycle (Schmidt, 2006).

Ensuring that models are formally defined and precise is fundamental for leveraging automatic transformations, which is by far the most effective technological means for boosting productivity, according to (Lukman and Mernik, 2008). This automated transformation process is often referred to as “correct-by-construction”, differing from the much more error prone “construct-by-correction” software development (Schmidt, 2006).

Model Driven Architectures

Perhaps the most popular case of an model-driven methodology is the Model Driven Architecture (MDA), defined by the OMG. In the words of (Kent, 2002), the MDA strategy envisages a world where models play a more direct role in software production, being amenable to manipulation and transformation by machine.

To (Brown, 2004), MDA is “a conceptual framework that separates business-oriented decisions from platform decisions to allow greater flexibility when architecting and evolving these systems”, an important approach to enterprise-scale software development that is already having significant impact in the software industry. MDA raises the level at which problems are addressed (Liang, 2009), and is associated with code being (semi-) automatically generated from more abstract models (Brown, 2004).

The basic concepts in MDA include (Brown et al., 2005; Liang, 2009):

1. *Models*. Expressed in a well-defined notation, models are the cornerstone to understanding systems for enterprise-scale solutions.
2. *Metamodels*. Represent a formal description of models. Having an infrastructure for working with metamodels facilitates transformations and integration.
3. *Abstraction*. Any specification of a system only describes the system at a level matching a specific perspective. This is the abstraction for the objective system.
4. *View*. Each view represents particular viewpoint of some concern at a determined level of abstraction.

5. *Refinement.* Model refinement is the gradual change of a model to better match the desired system (Brown et al., 2005). This means the actualization and the process of incorporation further details.
6. *Platform.* MDA concept involves a new type of developmental environment, which is independent of the hardware and software environment. On subsection “Different Models, Dimensions, or Viewpoints in MDA”, on page 23 within this section, a description on how models are divided into the computation independent model (CIM), platform independent model (PIM), and platform specific model (PSM) is provided.

The key idea behind MDA is separating the specification of the operation of a system from the details of the way the system uses the capabilities of its platform (OMG, 2003). According to (Brown et al., 2005), MDA provides the best means to capture commonalty in the problem domain, express transformation rules that convert the problem ideas into a possible solution, and generate a great piece of the solution under the technology of choice.

To (Hessellund, 2009), Model-Driven Architecture proposes that developers design their system on the abstract level and then use automated transformations to generate, in the first round, a platform specific translation, and in the second round actual production code for some platform. It tackles problems caused by system size or complexity by raising the level of abstraction. (Brown et al., 2005) remarks the conceptual framework provided by MDA for using models and applying transformations between them as part of a controlled, efficient software development process.

The primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns (OMG, 2003; Miller, 2009; Asadi and Ramsin, 2008). Portability is achieved by focusing on the development of platform independent models. Maintainability is also improved by the use of modeling (Liang, 2009). MDA is being considered by the software industry today as a means to increase the quality, efficiency, and predictability of large-scale software development (Brown et al., 2005).

According to (OMG, 2003), the promise of Model Driven Architecture is to allow definition of machine-readable applications and data models which allow long-term flexibility of:

- *Implementation*, existing designs can be mapped to any new infrastructure.
- *Integration*, integration bridges can be produced to integrate new infrastructures.
- *Maintenance* having the design in a machine-readable form makes maintenance much simpler.
- *Testing and simulation*, the developed models can be validated against requirements, tested against various infrastructures, or be used to directly simulate the behavior of the system being designed.

Three central principles support MDA (Booch et al., 2004):

1. *Direct representation* refers to the representation of the concepts in the problem domain, rather than the solution domain. The MDA approach allows specifying a system independently of the platform that supports it, and specifying platforms (OMG, 2003).
2. *Automation* as a means for bridging the semantic gap between domain concepts and implementation technology. The next step in MDA after choosing a particular platform for the system, is transforming the system model into one for a particular platform.
3. *Industry standards* as technology boosters, which encourage an ecosystem of vendors producing tools, and foster competition among them (Brown et al., 2005).

The OMG, in (OMG, 2001), states that the MDA approach and the standards supporting it allow the same model (specifying system functionality) to be realized on multiple platforms through auxiliary mapping standards, or through pointing mappings to specific platforms. It also allows different applications to be integrated by explicitly relating their models, enabling integration and interoperability and supporting system evolution as platform technologies come and go. On this, (Kent, 2002) comments: “This identifies the key distinction made in MDA between platform independent and platform specific models (PIM and PSM)”. By abstracting away from platform specific details “it is easier to validate the correctness of the model”, “it is easier to produce implementations on different platforms while conforming to the same essential and precise structure and behavior of the system”, and “integration and interoperability across systems can be defined more clearly in platform-independent terms, then mapped down to platform-specific mechanisms” (OMG, 2001, pp. 7 and 8). For (Clark et al., 2008), this PIM-to-PSM mapping or transformation is “the most widely recognized application of MDA”.

Although some authors (Kleppe et al., 2003) present MDA as having a precisely defined process, the OMG itself describes MDA in (OMG, 2003) as a flexible approach with what could be described more as a set of guidelines than specific steps. This guide describes numerous possible ways in which a model-driven application can be developed. The only clear guideline is the abstract-to-concrete approach of CIM->PIM->PSM transformations.

(Asadi and Ramsin, 2008) claim MDA is not a methodology, but rather a generic approach to software development, a shared view. Moreover, they also state that MDA cannot be useful without software development methodology support. Their concern with respect to the need of having a well defined process for the model-driven development of software systems is shared.

MDA by itself, however, fails to address the non-functional requirements of systems, and needs to be complemented in that area. In this thesis, a development process is defined to integrate non-functional concerns’ models in the model-driven development of Service-Oriented Architectures. The fundamental aspects of MDA are adopted: model-driven characteristics, abstraction, refinement, and the division in CIM, PIM and PSMs. Concepts from MDSoc and AOM are incorporated to address the shortcomings in the NF area, and DSLs are used instead of general-purpose languages. As a proof of the applicability of the approach, special attention is paid to concerns related to the security and access control of SOA systems.

The Unified Modeling Language (UML) and other OMG standards As a general consensus (Brown, 2004; Brown et al., 2005; Clark et al., 2008; Liang, 2009), the current state of the practice employs the Unified Modeling Language, a general-purpose modeling language, as the ‘de facto’ software modeling language, the primary modeling notation. The UML, a consolidation of all the notations in the various object-oriented methodologies of the eighties and nineties, supports the representation of different horizontal concerns, such as behavioral and structural parts of the system, on several levels of abstraction, targeted at modeling software systems at different stages of the development lifecycle (Clark et al., 2008; Hesselund, 2009).

As previously mentioned, domain experts think in terms of the problem space. Programmers think in terms of the solution space. UML, as a general-purpose modeling paradigm targets this problem trying to minimize the gap between problem and solution space (Hesselund, 2009). The means provided by UML to express domain-specific concepts are *profiles*. (France et al., 2006) defines profiles as the way how UML model elements are extended to support usage in a particular modeling context. They are viewed as domain-specific enhancements in (Kelly and Tolvanen, 2008), considering them as a first step toward DSM that provide a limited extension mechanism. Limited as new types can’t be added, nor anything be taken away from the UML language through the use of profiles. This forces any tool (for code generation, analysis, checking, or documentation) to navigate models through possibly unnecessary UML elements.

The advantages of having a common language such as UML include improving communication among participants. Being one of the most well established modeling language for software development, its knowledge is quite generalized. Another benefit of using UML is its wide tool support, including programmatic management of models. The main disadvantage to using UML, according to (Gharavi, 2008), would be that it is not sufficient enough to express enough detail for certain aspects.

Detractors of the UML claim that its great weakness is precisely its strongest feature: its generality (Kelly and Tolvanen, 2008; Hessellund, 2009; Liang, 2009). For (Hessellund, 2009), the UML standard lacks the advanced modularization constructs found in many programming or architecture description languages, and the problem of separation of vertical concerns is handled less adequately. UML major shortcomings are, according to (Clark et al., 2008; France et al., 2006; Kelly and Tolvanen, 2008; Hessellund, 2009):

- Imprecise semantics.
- UML has become an increasingly large language, bulky to be used, and difficult to maintain and test for consistency. The profile mechanism is simple to apply, although limited to constraining existing language constructs, with no possibility to modify or add new language constructs.
- The core models are at (substantially) the same level of abstraction of the supported programming languages.

(France et al., 2006) shares this impression, and states that the UML 2.0 profile mechanism does not provide means for precisely defining semantics associated with extensions. This prevents developers from “...using profiles in their current form to develop domain-specific UML variants that could support the formal model manipulations required in an MDD environment” (France et al., 2006).

There are also claims that UML is simply an abstraction of the solution domain, oppositely to model-driven initiatives, which focus on problem domain models (at least for CIMS and PIMs). (Hessellund, 2009), for instance, concludes that UML’s class diagram only changes the rendering of a concept and not really the level of abstraction, making neither regular UML nor UML with profiles sufficiently restricted to represent the constraints and limitations of a particular domain. (Gray, 2002) deepens on the unsuitability of UML’s profiles for introducing domain-specific knowledge, foreseeing problems when both the domain metamodel and domain model are represented using UML and profiles. He also asserts that the semantics of the domain also tend to be scattered across the domain model diagrams. Finally, he declares that UML models are filled of solution-space concepts that in other domain-specific modeling environments would otherwise be absorbed within the model interpreter, included as part of the mapping to the target platform. (Hessellund, 2009) argues that the only difference between single-language programming and general-purpose modeling is the visual appearance of the program sources. All this would result in violations of domain constraints being detected at a very late stage, an expensive consequence.

The view of (Kelly and Tolvanen, 2008) is shared in that UML has done a great favor to the software industry by emphasizing the need to consider design first. However, they postulate that the UML standard offers very little help in automating development work or increasing productivity. They base this in that UML was originally set up not for automating development but for agreement on modeling concepts, on communication.

On the other side, the OMG acknowledges some of these drawbacks. Therefore MDA, as stated in (OMG, 2003), suggests, but does not mandate, the use of UML as a metamodel for both PIMs and/or PSMs. The employment of UML in MDA is more or less relevant depending on the view of the author. For instance, (Kleppe et al., 2003; Miller, 2009; Hessellund, 2009; Pitkänen and

Mikkonen, 2006) and (Kent, 2002), among others, see the use of UML as central to MDA, while (Mellor et al., 2004) notes its importance, but clearly states that DSMs are a sound alternative.

Conversely, the OMG does mandate the use of MOF as the metamodeling language for MDA (OMG, 2010b; OMG, 2003). The reason given is that this guarantees that the models can be stored in a MOF-compliant repository, parsed and transformed by MOF-compliant tools, and rendered into XMI for transport over a network. In this thesis, a model-driven methodology that makes use of MOF and applies MDA's abstract-to-concrete approach is proposed. The E-MOF implementation of choice is Ecore (Steinberg et al., 2009), arguably the most used MOF implementation these days.

(Clark et al., 2008) accurately asserts that UML should be used where it makes sense to use it, and that other languages should be used where the abstractions provided by UML do not fit. This statement is agreed upon. In this thesis, in correspondence to the new interpretations of MDA, adjusted and suitable MOF-based domain-specific metamodels are proposed for the description of different non-functional concerns. UML is acknowledged as the primary modeling notation for modeling the functional aspects of software systems, and is therefore granted this role. Section 4.3 provides further details on this.

Different Models, Dimensions, or Viewpoints in MDA (Tarr et al., 1999) points out that for any particular software project, one will need to state the dimensions that need to be considered, which includes defining the points of interest along those dimensions (Kent, 2002). (Kleppe et al., 2003) sees the building blocks of the MDA framework as being:

- Consistent and precise high-level models, containing enough information on the system.
- One or more standard, well-defined languages to write high-level models.
- Transformation definitions of how a PIM can be automatically transformed to a specific PSM.
- A formal language in which to write these definitions, as it must be interpreted by the transformation tools.
- Tools that execute the transformation definitions.
- Tools that implement the execution of the transformation of a PSM to code, also known as generators.

As already mentioned, the Model-Driven Architecture specifies three viewpoints, or models, of a system: the Computation Independent Model (CIM), Platform Independent Model (PIM), and the Platform Specific Model (PSM), as described in (OMG, 2003). Complementary to these three views, one may have various Platform Models (PM) on which to map your Platform Independent Models, and then transform them (according to these mappings) to create a Platform Specific Model (PSM). This lexicon of platform-specific and platform-independent models is now widely referenced in the industry (Brown, 2004).

A key aspect of the MDA approach, as described by (Brown et al., 2005), is the concept of transformations being applied to abstract descriptions of some aspect (concern) of a system to add more detail to that description, refine that description to be more concrete, or to convert between representations. MDA defines process guidelines for a model-driven development in which Computation-Independent Models are repeatedly refined through transformations, until the final implementation code is obtained. The general idea of this process is that of transforming CIMs to PIMs, PIMs to PSMs, and PSMs to code. (Brown, 2004)'s perspective is that distinguishing among different kinds of models allows developers to think of software and system development as a series of refinements between different model representations.

(Mellor et al., 2003) describes how conceptual (CIMs, sometimes also referred to as business models (Kleppe et al., 2003; Ameller, 2009), or also domain models (OMG, 2003)) and engineering models (PIMs) address different problem domains. Conceptual models express requirements that can be then addressed in engineering models. The reason of the name Computational Independent Model is that business model does not necessarily say anything about the software systems used within a company (Ameller, 2009). These CIMs contain the requirements of the software system, expressing what a system will do (logically), not how it will do it (physically) (Miller, 2009; Kleppe et al., 2003). (OMG, 2003) describes how the CIM plays an important role in bridging the gap between domain experts and the experts of the design and implementation.

As noticed by (Ameller, 2009; Liang, 2009), literature on CIM to PIM transformations is scarce, and although the scientific community has not exhibit much interest in computational independent models so far, recently the use of process models describing business services (later implemented as SOAs) has given origin to research on the CIM-to-PIM transformation of service-oriented systems. In this CIM-to-PIM transformation, developers evolve the CIM into the platform-independent model, realizing the mapping from the domain-specific concepts of the business analyst into technology-specific concepts of the information technology (IT) architect (Brown, 2004; Miller, 2009).

A Platform Independent Model describes a system without any knowledge of the final implementation platform (Kleppe et al., 2003), void of implementation details, in a more technical manner than the CIM (Miller, 2009). PIMs describe the structure and processes of the system, including considerations of transaction process, information security, data persistence and other technical issues (Liang, 2009). According to (Kleppe et al., 2003), PIM models are an exact representation of the code, thus fulfilling the function of high-level documentation that is needed for any software system. (OMG, 2003) details that a platform independent viewpoint shows the part of the complete specification that does not change from one platform to another, presenting “..a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type”

Platform Specific Models, on the other side, add details that have meaning only within a specific platform (Ameller, 2009), describing a system with full knowledge of the final implementation platform (Kleppe et al., 2003). PSMs are the result of applying specific transformation rules to PIMs (Protopsaltou, 2007), constricting the PIM to an specific platform (Miller, 2009). As correctly stated by (Kleppe et al., 2003; Protopsaltou, 2007), a PSM is a platform-dependent model from which code and deployment description files can be automatically generated.

Most literature (Kent, 2002; Mellor et al., 2004; Kleppe et al., 2003; Brown et al., 2005; Clark et al., 2008) agrees in that PIM and PSM definitions are relative to a context or point of view. This opinion is shared, and there is coincidence with (Clark et al., 2008) in that MDA is too fixed on the notion of platform, and that what constitutes a platform is unclear at best. For (Kent, 2002), the platform specific/platform independent dimension is an example of the more general abstract/concrete dimension. He argues that one can only talk about a model being more abstract/concrete than the other. This is acknowledged by the OMG. (Kleppe et al., 2003) clarifies that the only thing one can say about different models is that one model is more (or less) platform specific than another, and that within an MDA transformation, a more platform-independent model is transformed into a more platform specific model, making the terms PIM and PSM relative terms.

Similarly, there may be more than one CIM, PIM, or PSM of the system. As noticed by (Kent, 2002), there are situations where models are not at different levels of abstraction but still provide alternative perspectives. Acknowledged by (OMG, 2001) is the fact that there may be many PIMs which may take different viewpoints or perspectives on the system, and which may be more or less abstract. (Kent, 2002) adds that in such cases, each model is likely only to define a part of the system, and synchronization needs to be maintained within the different viewpoints. (Kent, 2002) states that this may be achieved by means of some underlying integrated model, a view also shared by (Kelly and Tolvanen, 2008), and employed in this thesis.

PIM-to-PSM transformations, however, are abundantly documented. For (Kleppe et al., 2003), MDA's PIM-to-PSM transformations improves productivity in two ways: first, the PIM developers have less work to do as they avoid designing platform-specific details; second, developers can shift focus from code to PIM, thus paying more attention to solving the business problem at hand. They add that at the PSM and code level, there is much less code to be written, because a large amount of the code is already generated from the PIM, and the platform-specific details are already addressed in the transformation definition.

(Kleppe et al., 2003) also remark that someone still needs to define the exact transformation, and that this is "a difficult and specialized task". However, PIM to PSM transformations need only be defined once and can then be applied for the development of multiple systems. Furthermore, they claim that the payback for the effort to define a transformation is large, although it can only be done by highly skilled people.

In this thesis, an annotated CIM drives the discovery of different PIMs (each one addressing a particular concern of the system), which are then composed into one integrated system PIM. Afterwards, this platform-independent model, containing all functional and non-functional information of the system, is refined, and then transformed into different PSMs (one for each type of artifact to be generated, in the proposed example: code stubs, functional descriptors, service policy descriptors, access control configuration). The different defined metamodels and mappings are presented in chapters 3 and 4.

Domain Specific Modeling

Domain-specific modeling (DSM) is another paradigm that fits under the general umbrella term model-driven engineering, since it operates with models as first-class artifacts (Hessellund, 2009), but in this case based on the use of domain-specific modeling languages. The term domain specific means that the language is explicitly tailored to a target domain or field of application (Karlsch, 2007). As stated by (Ameller, 2009), DSMLs are languages with very specific goals in design and implementation, and are becoming commonplace for specifying systems at a high level of abstraction (Vallecillo, 2010).

When compared to general purpose languages like the UML, DSMLs offer an advantage in expressiveness and ease of use in their domain (Karlsch, 2007). Complex constructs and abstraction of the domain are offered within the language, making it possible to express solutions for domain problems with less effort. DSM is conventionally regarded as a means to raise the level of abstraction in software development, providing an abstraction mechanism to deal with complexity in a given domain (Kelly and Tolvanen, 2008; Liang, 2009; Gray, 2002; Pitkänen and Mikkonen, 2006). These languages reduce the available modeling elements to key, familiar problem-domain concepts (and rules), narrowing down the design space (Kelly and Tolvanen, 2008), to achieve higher levels of productivity. Developers can then concentrate on defining the problem instead of implementation-level details (Pitkänen and Mikkonen, 2006).

While UML's (and other general-purpose languages) constructs abstract the solution space, domain-specific models focus on the problem space. (Lukman and Mernik, 2008) see a big semantic gap between these two spaces, elevating the domain-specific knowledge of an organization to the level of prime intellectual property, and the probable source of its competitive advantage.

As in the previously mentioned model-driven initiatives, model transformations and code generators play an important role in DSM, synthesizing the model into an implementation (Gray, 2002). Different type of analyses, also present in other model-driven approaches, are in this case facilitated due to the closeness to the problem domain. Moreover, some analyses are made unnecessary since, as stated by (Kelly and Tolvanen, 2008), a language can prevent incorrect or poor designs by simply making them impossible to specify, and in this way preventing errors early on (when it is cheaper to correct them). Domain-specific metamodels not only allow for a intrinsic

model correctness, but also help flatten learning curves (Schmidt, 2006).

In the words of (Clark et al., 2008), DSMLs provide “an integrated framework of rich evolvable languages appropriate to their needs”, allowing engineers and domain experts to speak in the languages they understand. However, the view of (Pitkänen and Mikkonen, 2006) in that DSM requires a high initial investment due to the effort needed for designing a DSML and implementing code generators has proven right during the development carried out in this thesis.

Is hard to provide a percentage of the cost of the integration of the multiple DSMLs to form a domain-specific metamodel. It very much depends on numerous factors: the complexity of the domain, the degree of expertise the architect defining the MM has on the domain(s), the experience with model-driven initiatives and languages, etc. In the particular implementation performed for this thesis, learning the domain, the definition and setup of the MMs, generators, integration tools, etc., represented around a 50% of the project’s time frame. Productivity improvements in this case are not representative, as the amount of code in the resulting implementation is not large, and the generators were employed only a few times. For posterior implementations, the framework would already be in place, and the cost of the setup would be amortized with further use. In other cases, on longer-term projects, product families, or similar, productivity gains are claimed to be in the range of 5 to 10 times (Kelly and Tolvanen, 2008).

(Hessellund, 2009; Brown et al., 2005) claim that the development of modern enterprise systems demand the assembly of multiple views. (Hessellund, 2009) argues that multiple DSMLs are needed to better suit each individual view, and that the challenge is “integrating these different views and avoiding the potential cacophony of multiple different languages”. This assembly of multiple views is also anticipated by (Kelly and Tolvanen, 2008) and (Oberortner et al., 2008). (Oberortner et al., 2008) finds that multiple language models reduce the complexity by separation of concerns, providing tailored views for the different stakeholders. The main challenge of splitting, he claims, lies in finding appropriate extension points for merging models. In this thesis, the integration of different specific views is performed to achieve a model of the system with non-functional consciousness (security-awareness, for instance). This is described in more depth in section 4.4.

The key benefits of Domain-Specific Modeling are common to those of other model-driven approaches (Kelly and Tolvanen, 2008): improved productivity, better product quality, hiding complexity, and leveraging expertise. However, claims exist (Kelly and Tolvanen, 2008; Lukman and Mernik, 2008; Hessellund, 2009; Ameller, 2009) stating that productivity and quality increase are greater in DSM because of the above mentioned characteristics. Knowledge reuse is also facilitated by the closeness to the application domain (Lukman and Mernik, 2008).

According to (Kelly and Tolvanen, 2008), applications’ quality improves for various reasons. The two main ones being the inclusion of correctness rules on domain modeling languages, and the use of generators (model-to-code transformations). Those generators use models based on the domain-specific metamodel to produce the required code. Therefore, focusing on a narrow space makes it possible to map a language closer to the problem and make full code generation realistic, which they perceive complicated to achieve with general-purpose modeling languages.

DSMLs can be divided in two categories (Oberortner et al., 2008): DSMLs for domain experts (high-level DSML), and for technical experts (low-level DSML). A suitable separation of concerns can be established by splitting the language model into high and low-level models. Hence, a separation of technical and domain concerns can be established to present only the appropriate ones to each of the different groups of stakeholders (Oberortner et al., 2008). In the particular case of this thesis, high-level domain-specific metamodels were selected, and then mapped into lower level DSMLs, which are closer to the target platform.

Although feasible in some specific situations, the approach in (Kelly and Tolvanen, 2008) is not entirely achievable in all cases. Therefore, an approach amalgamating both domain-specific modeling languages (expressed as metamodels) and MDA’s abstract-to-concrete approach is proposed. Direct code generation from PIM models is possible in some situations, perhaps where platform knowledge

can be incorporated in the transformation code. However, this may result in the loss of an opportunity of reuse, of being able to map your design to multiple platform models (when different platform models are available). Section 3.2.2 describes this in more depth.

According to (Pitkänen and Mikkonen, 2006), the two greater weaknesses of DSM approaches are the high initial investment required for designing a domain-specific language, and the inflexibility with regard to the target platform. They clarify that such inflexibility is not only due to the use of a specific code generator, but also to the tendency of DSM languages to reflect a certain target architecture. In this thesis, the target architecture, SOA/web services, is specific but, at the same time, quite generic, as it counts with implementations in most platforms.

There are a lot of coincidences between this thesis and the view of (Hessellund, 2009). There, the author claims that customized DSMLs change the way of thinking about problems and facilitate more optimal solutions. Moreover, by using multiple languages, a very effective separation of concerns is achieved, where each concern is addressed by an adequate and proper language. Such an approach reduces, he states, the mental gap between domain expert and developer by providing constructs suited to the problem at hand. In this thesis multiple languages are used to address different non-functional concerns. However, as integrated metamodels present a more suitable subject for analyses, the approach differs of that in (Hessellund, 2009), and is closer to the “integration by metamodel” approach mentioned in (Kelly and Tolvanen, 2008, page 254).

2.1.4 Comparing UML and DSML Approaches

UML and DSMLs are two distinct, not mutually-exclusive modeling alternatives. Each one has its strengths and weaknesses. This section tries to provide a comparison between them.

According to (Fenster and Hamilton, 2010), UML can be used for various purposes, from describing system requirements to modeling a set of object-oriented domains and classes, while DSMLs have the advantage of being very specific to their purpose (much more specific than the general-purpose UML). Nowadays, complexity is regarded as one of the major problems of UML (Vallecillo, 2010). One advantage of DSMLs is that the modeling environment can constrain and validate the models created under it, which UML alone can’t (Dalgarno and Fowler, 2008).

The great penetration of UML also allows developers to choose among a wide variety of tool vendors (Dalgarno and Fowler, 2008). On the contrary, and although DSLs have been around for a while, it has only taken off in the past few years, arguably propelled by Microsoft’s DSL tools for Visual Studio (Microsoft, n.d.). In consequence, DSML tooling is less abundant, resulting in the need of a very specific know-how (e.g., specific MMs such as Ecore, related tools, etc.) of the modelers in charge of developing the DSMLs. Table 2.2, based on those in (Fenster and Hamilton, 2010), presents a comparison of the use of UML or DSMLs as modeling languages.

On the overall, UML can be considered a good means to model software implementations, or short-term problem domains. A UML profile may be a good alternative for prototyping domain-specific metamodels which are not yet stable/definite, and in situations (e.g., short-term project) where an investment in the creation and learning curve of a DSML is not worthwhile. In the long run, a DSML is a better solution for both modeling a stable domain, and for achieving a greater percentage of code generation.

Nonetheless, the author agrees with (Dalgarno and Fowler, 2008) in that there are cases in which an hybrid approach, combining UML and DSMLs, draws from the strength of each. In fact, this is the approach taken in this work, in which different DSMLs are used to represent the great majority of the system’s aspects, but UML is reserved a place for the functional modeling of the system.

Table 2.2: *Comparison of UML and DSMLs for modeling applications (based on those in (Dalgarno and Fowler, 2008)).*

	UML	DSML
Cost of initial implementation	Lower: Five standard UML diagrams are included in the box. Profiles must be authored.	Higher: A DSM language (meta-model and notation) must be determined and evolved. A designer (graphical, forms-based, or textual) must be implemented with the toolkit. Interoperability between DSMLs must be implemented.
Learning Curve	Lower: UML diagrams inter-operate in known ways (for example, class diagrams and sequence diagrams).	Higher: Modeler must learn the domain language (theoretically, is easy as already domain-savvy).
Specificity	Lower: All valid UML notations are allowed, even if they don't apply to the domain being modeled. Models have standardized notation, but rely on profiles, stereotypes, and comments to add domain-specific information.	Higher: Language is constrained to the domain that is being modeled. Models have domain-specific notation, DSMLs are custom-built and custom-tailored.
Goals	Design communication and documentation is often the goal. Code-stub generation is commonplace.	Forward-engineering of working software is usually the goal. Platform-specific code generation is commonplace.
Multi-perspective	UML profiles are integrated in one unique model, and appear in all the standard diagrams.	The use of multiple DSMLs support the use of various, very specific perspectives.
ROI	Over time, cost of use is higher.	Over time, cost of use is lower.

2.1.5 State of the Practice in Modeling Transformation Tools

For model-driven methodologies to succeed, sound tool support is necessary. During the process of this thesis, a state of the practice survey on model transformation and composition tools was performed. To this extent, the different tools with support for model transformation and composition of multiple MOF-based metamodels were evaluated for their suitability (in terms of usability, stability, features, and expected evolution) to be used in the application of the defined methodology (presented in section 3.2.2). The reader is referred to (Silva Gallino et al., 2009a) for the results of this survey.

2.2 Modeling Service-Oriented Architectures

2.2.1 Service Oriented Architectures

Many definitions have been provided for Service Oriented Architectures (SOAs). In (OMG, 2009), the standard defines SOA as “a way of describing and understanding organizations, communities and systems to maximize agility, scale and interoperability”. For (OMG, 2008c), SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides uniform means to offer, discover, interact with, and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

A service consists of an implementation that provides business logic and data, a well-specified contract that defines the specifies the functionality, usage, and constraints for a client of the service, and a service interface that physically exposes the functionality (Krafzig et al., 2004; Brown, 2004). According to (Arsanjani, 2004), the three key elements of a SOA are services, flows, and components realizing services. Sharing with (Brown, 2004) the impression that services are related to the notion of components, the difference between them is that services are relatively loosely-coupled, have ‘document-oriented’ interfaces, and may also be late-bound.

(Erl, 2007) remarks how SOA establishes an architectural model that aims to enhance the efficiency, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing. SOA can establish an abstraction of business logic and technology, resulting in a loose coupling between these domains (Erl, 2005).

SOA has grown in importance during the last years, because organizations realize that they can expose their functional capabilities to other organizations and they can build loosely coupled systems, allowing for fast, secure, flexible and automated relationships between enterprises (Larrucea and Alonso, 2008). The key technical concepts of SOA that allow it to cope with the system characteristics just described (Josuttis, 2007) include:

- *Services.* SOA aims to abstract by concentrating on the business aspects of a problem. In essence, a service is an IT representation of some business functionality. Externally, technical details should not be seen.
- *Interoperability.* For SOA, high interoperability is the base from which we start to implement business functionality (services) that is spread over multiple distributed systems.
- *Loose coupling.* Enterprises seek fast times to market, so flexibility is often valued over quality. With all that complexity, a minor problem can stop the whole business. This demands fault tolerance, in addition to previous goals such as flexibility and scalability. The key to fulfilling these goals is loose coupling. Loose coupling is the concept of minimizing dependencies. When dependencies are minimized, modifications have minimized effects, and the systems still runs

when parts of it are broken or down. Minimizing dependencies contributes to fault tolerance and flexibility, and leads to scalability.

The view of (Arsanjani, 2004) that SOA and Web Services (WS) are not synonyms is shared. SOA is more strategic and business-aligned, with Web services being a technical implementation of SOA, perhaps the most widely accepted implementation of SOA. However, many other implementation technologies exist in the market, such as peer to peer or agent technologies (Larrucea and Alonso, 2008; Larrucea and Alonso, 2009). In the context of the enormous quantity of specifications related to the so-called WS-*, (Larrucea and Alonso, 2009) sees Security Policies specification and deployment support as still immature and in need of improvement.

(Josuttis, 2007) summarizes the definition of SOA as:

- An architectural paradigm for dealing with business processes distributed over a large landscape of existing and new heterogeneous systems that are under the control of different owners.
- The key technical concepts of SOA are services, interoperability, and loose coupling.
- The key ingredients of SOA are infrastructure, architecture, and processes (including the meta-process of establishing SOA, governance).
- The key success factors for SOA are understanding, governance, and management support.
- SOA is not an specific technology.
- Web Services are one possible way of realizing the infrastructure aspects of SOA. Using Web Services is often recommended because it seems to be becoming established as the standard technology.

2.2.2 Modeling Service-Oriented Architectures

In the words of (Larrucea and Alonso, 2008), “applying a MDA vision to enterprise architectures as well as systems architectures provides a solution to build model based systems to avoid or reduce the loss of information and to increase the separation of concerns, flexibility and traceability”, adding that MDD and SOA provide capabilities for enabling interoperability among stakeholders. (OMG, 2009) declares that using MDA (in the context of a SOA) helps architectures to outlive the technology of the day, supporting enterprises’ evolution over the long term. This standard adds that MDA helps with separating the concerns of the business or systems architecture from the implementation and technology. For (Kim and Lee, 2008), MDA focuses on levels of abstraction, defining the role of models within a process, while SOA focuses on the stereotypical roles of models based on separation of concerns. Combined, each approach provides a set of benefits, and together they form an holistic approach towards this end (Larrucea and Alonso, 2008).

(Arsanjani, 2004) describes service-oriented modeling as a service-oriented analysis and design (SOAD) process for modeling, analyzing, designing, and producing a SOA that aligns with business analysis, processes, and goals. In this regard, (Zimmermann et al., 2004) claim that SOAD must facilitate end-to-end modeling and have comprehensive tool support. Furthermore, if SOA is supposed to bring flexibility and agility to the business, the same should be expected from its supporting method, spanning from the business to the architecture and application design domains.

The modeling of service-oriented architectures should distinguish between two different activities:

- modeling of high-level, business-to-business workflows, CIMS, etc.
- PIM-level service modeling.

The OMG has recently adopted (OMG, 2009) as its standard for modeling services. This standard supports both the IT and business perspectives of a SOA, but knowingly leaves the non-functional aspects of services out of its scope. As already mentioned, the focus of this thesis is on the description of non-functional properties of services.

To (Brown, 2004), mapping from the domain specific concepts of the business analyst into technology specific concepts of the IT architect represent the CIM-to-PIM mapping. In this work, CIMs, PIMs and PSMs for functional and non-functional aspects of SOAs and web services are provided, and mappings between these different layers are identified.

Using a DSL-based MDD for SOAs, says (Oberortner et al., 2008), enables technical experts and domain experts to work at higher levels of abstraction when compared to using technical interfaces, adding that the multiple concerns of process-driven SOAs can be modeled independently through MDD. They insist in that model-driven DSLs can reduce complexity, and enhance understandability and readability for the individual stakeholders of SOAs, by tailoring different DSMLs for domain and technical experts.

Referring to the particular strategy of the modeling of services, (Bui et al., 2008) notices that the usage of standard-based concepts through modeling constructs significantly improves communication, since it leverages developers' familiarity with existing standards. Using policies as a way to cover both requirements and capabilities in the same model, they claim, is well-received compared to using aspects and ad-hoc non-functional requirements representations from the academic community, as policies also resonate with business analysts (in addition to developers). They argue that many existing MDD approaches pollute high level models with product and platform specific concepts, while standard-based modeling constructs are considered a good solution. In the approach proposed in this thesis, the integrated metamodel adopts a policy-based representation of non-functional properties of the system, and technical characteristics are avoided until low level PSM models.

This thesis is also conscious of the pollution of models, in this case on the pollution of functional models with non-functional characteristics of the services. The methodology proposed in chapter 3 allows for the independent development of functional and extra-functional concerns, fostering designs' reuse.

Implementing a CIM → PIM → PSM Chain for SOA

A plausible approach for addressing the model-driven development of SOAs, addressing both functional and non-functional concerns, would be one that makes use of appropriate DSMLs at each abstraction level, and at each perspective.

For instance, process models are a good means to describe the functional aspects of services compositions at CIM level. UML's activity diagrams, or other standards, such as Business Process Modeling Notation (BPMN) (OMG, 2011a) already address process-modeling. Annotations may be an appropriate mechanism for including non-functional considerations, but a common vocabulary has still to be defined for such goal.

Architects may already have a functional architecture for the system at PIM level. Most probably, it would be described in UML. Being service-oriented, the use of an standard profile, such as (OMG, 2009), provides a great potential for reusability, and is paramount.

At this level of abstraction, technical non-functional aspects should be addressed. Different viewpoints of the system may display the information important for a particular concern, while hiding superfluous details. Currently, there is no metamodel that may address all functional and non-functional concerns of a system, nor it would be practical to define one. At least in a first stage, it makes more sense to limit the focus of the approach to a set of NF characteristics, and define a MM to address those.

There exist, however, a number of MMs which have been defined to describe different individual non-functional concerns. In the particular field of security, (Lodderstedt et al., 2002) for access

control, or (Jürjens, 2002) for security modeling under UML are classical examples. Trying to define new MM addressing the same domains seems unnecessary and impractical, as good solutions already exist. A better approach would be to reuse those solutions to define a MM that considers multiple aspects of a system at the same time. A good amount of work should be placed on selecting the most appropriate languages for each concern, the ones that support greater generality and reusability, and implement them ¹. The greater effort, then, would be placed in identifying correspondences and overlaps, joint points that allow for the integration of those MMs.

At the platform (PSM) level, adopted standards (i.e., WSDL, WS-Policy) already exist, and are heavily employed by the industry. Target programming languages and frameworks, on the other hand, abound, and the more appropriate option may vary from project to project. Generators to such standards may be reusable, while those specific to the framework/language might be specifically developed for each case.

Generation tools, transformation languages, and modeling frameworks, are already available. Tooling is in place: Eclipse EMF (Steinberg et al., 2009) to define DSMLs, ATL (Jouault et al., 2008) or the Query, View, Transformation (QVT) standard (OMG, 2008a) for transformations, Acceleo (OBEO, 2006) or Java Emitter Templates (JET) (Aniszczyk and Marz, 2006) for generation, etc. Nevertheless, the particular transformations and generators must be defined for each pair of source/target (modeling) languages.

In conclusion, a solution in the lines that have been presented appears feasible. DSMLs have been defined for the areas under focus, tools are available, target standards and platforms are also in place. What is still to be done is implement the glue that puts all this together. It may sound simple, but is really not. This dissertation presents a solution that has been defined and implemented towards that end.

2.3 Summary

(Mellor et al., 2004)’s conception of Model-Driven Architecture is very close to how it is understood in this thesis. They describe MDA in terms of the selection of the models and the mapping functions between them and how these must fit together to form the specific process each one applies on his MDA project. The best approach to MDA, they claim, is a mixture of two: finding models at a single level of abstraction, and finding an optimal length for each “hop” in the mapping chain.

The process adopted in this thesis, described in chapter 3, is a combination of MDSoC/AOM concepts, domain-specific metamodels, and MDA. (Mellor et al., 2004)’s idea that a multiple-hop approach simplifies the mapping functions and exposes an intermediate model is shared in this thesis. They add that the choice of intermediate domains may also depend on what is available for reuse (possibly from third parties). Strongly coinciding with this last statement, a focus on reusing existing (useful) metamodels as far as possible is present in this dissertation.

Many (Hessellund, 2009; Kelly and Tolvanen, 2008; Pitkänen and Mikkonen, 2006; Mellor et al., 2004) agree in that an effective approach is to divide up the system into independent subject matters, or problem domains. (Brown et al., 2005) notices how many different kinds of elements can be modeled depending on the context, offering different views that must be reconciled. The intermediate model of (Mellor et al., 2004), sitting between the source model and the target model, can be used to integrate all these views, as suggested in (Kelly and Tolvanen, 2008). Furthermore, such an intermediate model allows mapping functions to be reused. This is precisely the adopted approach in this dissertation.

Besides defining an integrated metamodel, other very important steps in an model-driven approach are composing the different views’ models, specifying mapping functions, constructing

---

¹Definitions are available, but implementations are generally not.

marking models, and transforming the models (Mellor et al., 2004). In this thesis all those steps have been considered, and are explained in section 4.7.

(Kleppe et al., 2003) believes that you can today achieve great gains in productivity, portability, interoperability, and maintenance effort by applying MDA using a good transformation tool. On the other hand, (Ameller, 2009) declares that a very specific domain, and a very restricted group of applications are necessary to be able to generate the full source code from models. Although not the primary focus of this work, code generation has been performed for illustrative purposes. An example demonstrating the proposed approach, is presented in chapter 5.

(Kleppe et al., 2003), differently from (Mellor et al., 2004), places MDA's emphasis on automatic transformations. According to them, for some time already tools have been able to transform a PSM to code, what's new in MDA is that the transformation from PIM to PSM is automated as well. (Mellor et al., 2004), however, argues that the innovation in MDA, and this can be extrapolated to other model-driven approaches, comes with the concept of testing the model for correctness. The analysis of models is also addressed in this thesis, as described in section 4.6.

Chapter 3

Model-Driven Development of a Web Service-Oriented Architecture and Non-Functional Policies

3.1 Introduction

Efforts on achieving model-driven development of web services have been around for a while. Service-oriented architecture's modeling is an area in which, although multiple proposals have been put forward by tools vendors, a consensus has not been achieved just yet. SOA is the common reference for many in-development standards and execution frameworks and, therefore, special attention is being paid to it.

Currently, standardization organizations, such as OMG, have recently adopted standards (i.e., (OMG, 2009)) to offer a solution to this problem. Results are expected in the short term. However, it has not been until very recently (OMG, 2008c) that non-functional aspects of services have been considered for standardization processes. Thus, no solution in this area is expected in the next few years to come.

Although multiple implementation technologies exist to facilitate the development of web services and SOA systems, the lack of a sound methodological base for the development of such applications stresses the need for new modeling methods or techniques that could guarantee the quality of the development of this type of systems. The model-driven methodology defined in this chapter translates into development quality, such as in:

- guarantee an error-free development (due to the use of validated generators),
- guarantee that the resulting implementation adheres to its specification, in requirements satisfaction (both functional and NF), and in architecture.

In their research roadmap for service-oriented computing, (Papazoglou et al., 2008) remark the need for methodologies supporting the specification and design of services and service compositions, associating the design methodology with business process modeling techniques. *QoS-awareness* is also stressed as another grand challenge in the future of service-oriented architectures.

In essence, there is currently no complete solution that addresses non-functional aspects of services as well as the functional ones. This chapter presents a methodological approach which seeks to integrate these non-functional aspects in the development of web services.

Participant Technologies

Service-Oriented Architectures (SOA), counting with great popularity nowadays, and Web Services (WS), the technology generally employed to implement them, go hand in hand in such a way that they are even referred to as WSOA (Web Service Oriented Architecture). They achieve the integration of heterogeneous (e.g., Java, C#, etc.) technologies, providing interoperability, and allowing for the reutilization of pre-existent systems.

At the same time, Model Driven Applications (MDA) (OMG, 2003) arises as a new paradigm that tries to shift models out of a mere documental role, and turn them into a first class development artifact. MDA provides, among other benefits, a greater understanding of the system as a whole and a platform-independent development paradigm, improving the reusability of the design, and simplifying the evolution of the system, thus increasing productivity.

WS-* standards (InnoQ, 2007) provide a strong foundation for the development of services. WS-Policy (W3C, 2006a) and related specifications also offer the possibility of considering extra-functional concerns of those services. However, these specifications' syntax is based on XML, not thought to be read/written by humans. As appreciated in (Satoh et al., 2008), developers at downstream phases (implementation, deployment) do not have enough information to create adequate configurations, and these configurations themselves are too complex for non-experts. An abstraction of such languages is desirable.

Frameworks provide much functionality which would be otherwise necessary to generate for the execution of the different systems, greatly simplifying the development of generators. Newer frameworks' declarative-based configuration, of particular usefulness, support the configuration of non-functional characteristics of the services (such as security or access control, in this case) independently of the functional code. This allows for an autonomous generation of the different artifacts, and the reuse of NF generators independently of the functional ones. This chapter presents a piece of work towards achieving a solution that includes all these previously mentioned concepts.

3.2 In Search Of The Integrated Solution

3.2.1 Requirements and Objectives

From within the different requirements and objectives placed on this research, the ones with a higher influence on the result are enumerated below. Those are, in no particular order:

- Fundamentally, define a development process that allows an independent development of the functional and non-functional concerns, fostering reuse.
- Each concern should be addressed in a convenient manner.
- The functional model should not be polluted with non-functional information, supporting the secrecy of separate designs.
- Where feasible and practical, reuse pre-existent tools/solutions.
- Model-Driven. Operate with models as much as possible.
- Relating to target platforms, make use of declarative configuration and AO functionalities that are becoming available in the different platforms to satisfy extra-functional requirements.

- The resulting tool should generate WSDL descriptors with referenced policies for the services.
- The tool should also generate elements that describe the non-functional configuration of the resulting services.
- Finally, the result should also include code templates for the target platform, to illustrate the marriage of this non-functional approach with the generation of functional artifacts.

3.2.2 Defined Architecture

From within the technologies mentioned in chapter 2, model-driven development and service-oriented architectures are currently very much accepted both in academics and industry (in the latter, SOA enjoys much greater penetration than MDD/MDA). The development approach of defining extra-functional aspects as system-independent constituent blocks is slowly migrating from an exclusively academic subject towards the corporate/industrial evaluation of its usefulness. Research on modeling/design of such independent non-functionality blocks is relatively recent.

The application of model-driven techniques on SOAs is not novel. Published research and experiences are available on the subject. Nonetheless, the communion between the model driven development of non-functional aspects as independent building blocks which are later combined to result in a SOA system is quite innovative. As already mentioned, this is the major research subject of this thesis.

Such a combination of technologies generates really interesting benefits. For instance, MDD provides platform and implementation independence, and facilitates design reuse. In particular, the use of domain-specific modeling languages leverages great productivity increments and facilitates a more appropriate design. On the other hand, service-oriented architectures allow for the creation of loosely-coupled, geographically disperse, on-demand systems. Finally, MDSoc supports greater code/design modularization, leveraging understandability and maintainability of the design.

Proposed Solution

Figure 3.1 shows the overall structure of the designed solution, consisting of a chain of model transformations and compositions (indicated by arrows). Every box in the figure indicates a different type of model, including:

- Business model (CIM Level) indicating non-functional intentions.
- Functional system model (the main input model at PIM 1 Level).
- Non-functional concerns models (PIM 1 Level).
- Integrated model (PIM 2 Level).
- WSDL, WS-Policy, and XML models (PSM Level).

The presented approach proposes the development of each individual concern as an independent model. Moreover, it suggests the use of different DSMLs, each one appropriate to the particular concern in hand. Such models are later composed into a complete model of the system by weaving them together. A set of metamodels for an implementation of the approach, for security and access control as non-functional concerns, is presented in section 4.3.

As illustrated in Figure 3.1, different expert-made NF models, hosted in model repositories, could provide the NF input. Appropriate models are automatically selected from the repositories and offered to the modeler. As a result, the developer of the system need not be savvy in the particular NF concern (e.g, security).

Figure 3.1: *Structure of the proposed solution.*

The objective of the integrated model is achieving an abstraction both of input modeling techniques/metamodels, and output target platforms. It is also designed for holding in one unique model, all information necessary to generate the different output artifacts. This makes PIM2 models the perfect subject if model analyses.

To (Vallecillo, 2010), a proper system specification consists of a set of viewpoints, each one expressed by its proper metamodel, and a set of correspondences between them. There, the author observes that the combination of these viewpoints should lead to a new modeling language, which is the case in this thesis. Moreover, the particular unified metamodel for security and access control defined in this thesis is not a mere combination of the viewpoints' DSML, but a combination of other different (more generic) DSMLs. The definition of this metamodel, iMM, is presented in section 4.4.

3.2.3 Development Process

The sequence of steps to perform in the proposed approach, represented in Figure 3.2, include:

1. A business process model, annotated with abstract NF intents, or, alternatively, a requirements model, is presented as input to the process.
2. The intents at CIM level drive the selection of the appropriate non-functional models. As suggested in (Toma et al., 2008), a semantic approach for describing non-functional properties may basically enable reasoning on these properties, thus supporting the automation of tasks like discovery, selection, composition, etc.
3. A functional PIM model is either derived from the CIM model. or provided by the user as an input functional model at PIM level (PIM 1).

4. By means of composition models, the user indicates which non-functional properties should be associated to which resources in the functional model. Weaving metamodels define the possible associations that functional and NF elements may have. Although these metamodels may be reusable among different NF characteristics, it may be the case (as can be seen in chapter 4) that different NF concerns present different types of associations (and affecting different kinds of functional elements). Ideally, pre-filled composition models should be proposed by the tool.
5. At this stage, the platform-independent integrated model (PIM 2) contains all (functional and extra-functional) information.
6. Next, the PIM2 model is transformed into PSM models. Default values may be used for the generation, unless optional platform information (indicating preferences and/or availability of platform characteristics) is fed to the transformation. For instance, in the particular case of security and AC as NF concerns, a WS-SecurityPolicy model (extended with a '*preference*' attribute) represents the platform model. This is further described in chapter 4.
7. Finally, service and policy descriptors are derived from the PIM 2 iMM model. Resulting WSDL documents hold references to the appropriate policies defined in the different generated WS-Policy documents. Simultaneously, code templates and non-functional configurations are automatically created.

Tool support automating the development process is desirable. The tool should make use of default value options to allow an end-to-end automatic process, while still supporting user feedback for greater flexibility and results' customization. A prototype, partially implementing the development process, has been developed, and is described in 4.7.

The idea behind the approach is that each model in the repositories address one or more non-functional concerns. Experts in the particular NF domain create such models, and also indicate how the different elements map onto the intermediate model. Appropriate metadata is associated to these models to support automatic discovery.

The benefits of this approach is twofold: it frees the developers from the need of non-functional expertise, and allows greater reuse of (functional **and** non-functional) designs. NF models may be reused among different projects. Also, different NF characteristics may be applied to the same functional design to generate different resulting systems.

3.3 NF-Concern Independent Metamodels

Referring to the metamodels employed in this approach, some pre-existent MMs have been adopted, some other where composed together, whilst the rest had to be defined for the occasion. The following sub-sections describe the different MMs which are independent of the non-functional concerns to be addressed, and may be reused along other implementations of the approach. Chapter 4 describes a particular implementation considering security and access control as the NF characteristics under focus. There, the rest of the metamodels, in this case NF-dependant, are presented.

3.3.1 Functional CIM Input Metamodel: BPMN

Business process diagrams have become the mainstream alternative to model the compositions and interactions of services (e.g., in (Wada et al., 2008; Menzel and Meinel, 2009; Satoh et al., 2008)). These diagrams are also the choice adopted in this thesis to model the services at the Computation Independent Model (CIM) level. Such models provide a great means for describing the interaction of the services at a high level of abstraction. They also represent an outstanding instrument to express abstract non-functional intentions, as will be presented in section 4.3.1.

Figure 3.2: *Proposed Development Process.*

For this particular instantiation of the approach, Business Process Management Notation (BPMN), nowadays the mainstream notation with which to model business processes, has been selected. The OMG defines in (OMG, 2011a) the objectives of BPMN as “to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.” Thus, they claim, BPMN creates a standardized bridge for the gap between the business process design and process implementation. Furthermore, they add that it seeks “to ensure that XML languages designed for the execution of business processes... can be visualized with a business-oriented notation.”

BPMN allows for the modeling of collaboration, processes, and choreographies, supporting the expression of these type of interactions between services, and its posterior execution. To this extent, BPMN standard defines both its own execution semantics, and a mapping to WS-BPEL (Web Services Business Process Execution Language) (OASIS, 2007b).

The goal of the approach is to abstractly indicate the NF intents/requirements in a service composition, and then facilitate the implementation of mechanisms fulfilling those requirements. Therefore, the elements subject of securement (in the exemplifying use case) include those that represent the services, its operations, and the data objects exchanged by them. Additionally, to facilitate the expression of security intents which are common to multiple services, the annotation of grouping elements is also supported (the intents are considered applied to every subjects contained in the group). Consequently, the BPMN elements subject of annotations include:

- Message events & flows, Service Tasks, Data Objects.
- Groupings: Group, Pool, Lane.

A description of these, and the remaining BPMN elements may be found in (OMG, 2011a).

3.3.2 Functional PIM Input Metamodel: UML

If a PIM level input is preferred, UML, the modeling standard of choice in most cases nowadays, has been selected as the functional input’s metamodel. An implementation of the SoaML (OMG, 2009) UML’s profile¹ is applied to these input models to guide the transformation. The objective of this profile, described next, is to abstract the transformation of the different ways in which UML is being used (from the experience of the author, different modelers prefer to represent software components with UML component, class, or package model elements).

UML’s SoaML Profile Based on (OMG, 2009), a UML profile is defined to identify the different service-oriented elements of the system. As previously stated, the functional aspects of the system are not the focus of this work. They are acknowledged, but not thoroughly studied. Consequently, the SoaML profile and its mappings to the intermediate metamodel are not studied in full detail. The profile is fully specified, including validation rules, in (Silva Gallino, 2007).

Figure 3.3 presents a conceptual partial view of the SoaML profile. Dark gray elements represent UML metaclasses, while light gray elements represent SoaML concepts. Each light gray metaclass has an equivalent stereotype to represent such concepts in a standard UML model. The profile represents a means to identify service interfaces (and transitively its operations and parameters/messages), service definitions, service ports, etc. The basic idea is that Participants (a generalization of service providers and consumers) offer/require service interfaces through a service interaction point. The mappings from SoaML profile to IMM are presented in section 4.5.4.

¹Strictly speaking, it is an implementation of the state of the standardization process at the time.

Figure 3.3: Conceptual view of the SoaML profile. (OMG, 2009)

3.3.3 WSDL Metamodel

A WSDL metamodel has been used in the transformations and compositions, in order to visualize the future WSDL document as a model and facilitate the specification of policy application points. This metamodel was obtained from the WSDL 1.2 specification schema (using the tools provided to this end by Eclipse’s EMF (Eclipse, 2011b)), and then modified to better fit its use in editors and transformations.

3.3.4 WS-Policy Metamodel

A WS-Policy (W3C, 2006a) metamodel, used both in transformations and compositions, has been developed for the prototype. Its implementation was necessary as the metamodel obtained from the specification schema (as in the case of WSDL’s) was not usable in an editor, and that all other previously existent WS-Policy metamodels (i.e., (Ortiz and Hernández, 2006; Jegadeesan and Balasubramaniam, 2009)), do not allow for the use of both “compact” and “normal” WS-Policy mode, as was the intention. This metamodel, presented in Figure 3.4, contains the usual concepts:

Figure 3.4: WS-Policy Metamodel.

Policy Document, Policy, Policy Alternative, Policy Assertions, etc.

The use of this metamodel in EMF allows for the implementation of Java model repositories, on which new WS-Policy models can be created, either from Java code or from model transformation languages. Transformations rules and later generation of WS-Policy documents operate on this metamodel. This is also the case of models corresponding to particular extensions of this MM, for instance the WS-SecurityPolicy metamodel described in next chapter in section 4.3.5.

3.4 Annotating Models for Measurement and Analysis

The early evaluation of software designs is a frequent goal; this evaluation gives support for solving risks and selecting among alternative solutions. In model-driven development, models are a fundamental part of software design. Architectural models are used not only for the description of functional features, but also include other properties that characterize software components (e.g. components quality, non-functional properties and traceability information).

Nowadays, software measurement has become fundamental for a sound software engineering. Skillful software developers measure characteristics of the software to get some sense of whether the requirements are consistent and complete, whether the design is of high quality, and whether the code is ready to be tested (Fenton and Pfleeger, 1998). Moreover, software measurement is especially useful for aiding in selecting among alternative software solutions. The approach in this thesis considers the execution of analyses and calculation of metrics on software models, for the evaluation and comparison of designs regarding non-functional characteristics.

Analyses of this kind, however, are specific to the metamodel to which the models correspond. In next chapter, a metric and an analysis, specifically defined for the integrated models presented in Figure 3.1, are presented.

3.5 Discussion of the Approach

The presented approach is only one of the multiple alternative solutions to the model-driven development of non-functional aware SOAs. For instance, domain-specific multimodeling (DSMM) (Hessellund, 2009) or the use of one unique metamodel all along would represent two different possibilities. This section discusses the strength and weaknesses of each of these approaches.

3.5.1 Comparison with Domain-Specific Multimodeling

DSMM shares many similarities with the approach presented here. The biggest difference between DSMM and this approach is the used of an integrated metamodel by the latter.

This brings along its proper pros and cons. For instance, the first limitation of using a integrated metamodel, when compared to DSMM, is that the approach is limited to what may be described by this metamodel. DSMM, in contrary, has no limitation in this aspect, although the associations between the different DSMLs have to be defined for each combination of them. This limitation can be softened if the integrated metamodel is defined with enough generality to include multiple characteristics.

On the other hand, an integrated metamodel is the perfect subject for analyses in the early stages of development. Conversely, analyses in DSMM have to be defined for every combination of DSMLs. Moreover, a DSMM approach is subject of the coordination problem (Hessellund, 2009), to which an approached based on an integrated metamodel is not.

3.5.2 Comparison with Unique Metamodel

Representing both functional and non-functional aspects of the system in the same model from the start is another alternative to the proposed approach. Both approaches share the benefit of early analyses, and there is only one language to learn for the development of systems. Furthermore, having all information in the same model facilitates communication between non-functional and functional modelers. This may well be the form of a solution based on UML with profiles.

However, such an approach has two mayor shortcomings. The first one is the use of the same abstraction level for all layers (CIM, PIM, and PSM). Information can be refined, but model elements are basically the same ones in all stages.

The second mayor shortcoming is lack of modularity, hindering reuse. By having both functional and non-functional information in the same model, design is only usable for the particular system in hands. With the solution presented in this dissertation, functional design can be reused in different set-ups, possibly changing completely the non-functional characteristics of the implementation on which it will run. Also, NF designs can also be reused among different systems. Additionally, functional modelers will be exposed to non-functional elements, which they will probably need to understand to grasp how the complete model behaves.

Furthermore, this exposition prevents the achievement of secrecy in the NF design, one of the objectives placed on the approach. Picture, for example, a banking system, where the design and implementation of security aspects would be preferably kept secret of most system designers, and only a few trusted individuals may be selected to be exposed to that design.

In conclusion, the defined approach, although sacrificing a certain flexibility, was selected to both provide an opportunity for analyses, while still supporting reuse. It was designed to abstract non-technical stakeholders from technical details, while still offering a means to express non-functional intentions. Next, in chapter 4, a particular implementation of the approach is presented to clarify how these objectives may be achieved.

3.6 Summary of the Chapter

This chapter has presented a development process for the achievement of an integrated, model-driven solution for the development of service-oriented architectures with non-functional characteristics' awareness.

The proposed development process presents a combination of technologies that generate great benefits:

- Model-driven development for platform and implementation independence, and design reuse.
- Aspect-orientation for code/design modularity, understandability and maintainability of the design.
- An appropriate domain-specific modeling language for each concern, raising productivity and aiding in the achievement of suitable designs.
- Service-oriented architectures as target, supporting the reuse of pre-existent applications and providing interoperability between heterogeneous technologies and systems.

From the different metamodels employed, only those which are independent of the particular non-functional characteristics to be addressed were described in this chapter. The remaining metamodels, which depend on the NF concern, are particular to the different implementations. The proposed solution is not tied to any implementation or target technology, allowing for great flexibility in its evolution regarding the appearance of new standards, technologies, or modeling alternatives.

A particular implementation of the process defined in this chapter for the non-functional characteristics of security and access control is presented next, in chapter 4.

Chapter 4

Security and Access Control as Non-Functional Properties in Service-Oriented Architectures

4.1 Introduction

This chapter presents an implementation of the development process defined in chapter 3 for the definition of SOA systems with security and access control concerns awareness. Security is an aspect of software systems that is continuously growing in importance. A proof of this is that in the evolution of ISO's standard 9126 Software engineering Product quality (ISO/IEC, 2001), the new ISO standard 25010 Systems and software Quality Requirements and Evaluation (SQuaRE) (ISO/IEC, 2011), security is upgraded from a sub-characteristic of functionality, into its own characteristic.

As already described, modeling standards knowingly leave NFPs out of their scope, and it has not been until very recently (OMG, 2008c) that non-functional aspects of services have been considered for standardization processes. Thus, no solution in this area is expected within the next few years.

Although multiple frameworks exist to facilitate the development of web services and SOA systems, the lack of a sound methodological base for the development of such applications stresses the need for new modeling methods or techniques that could guarantee the quality of the development of this type of systems. For instance, as presented in (Juric et al., 2010, Chapter 6, "Securing BPEL Processes"), in IBM's Websphere (IBM, 2011a) latest version (at the time of writing), 16 steps (including more or less technical questions, e.g., Security header layout) are necessary to generate a policy requiring the presence of a UsernameToken in a service request. Furthermore, the average web services user/programmer doesn't know if a UsernameToken is the appropriate kind of identification token to require, or if it represents a suitable security configuration. Service modeling standards, such as (OMG, 2009), make no suggestions on this respect. WS-Policy is defined at a technical level, for a developer role, and the same technicality may not be appropriate for business analysts.

Recalling, the objectives placed in this thesis may be summarized into defining a design a solution that:

- Allows an independent development of the functional and non-functional concerns of service-oriented architectures, fostering reuse.

- Permits that each concern be addressed in a convenient manner.
- Is modular enough to avoid the pollution between functional and non-functional information at the architectural level, and to support secrecy.
- Provides a practical way of measuring and analyzing the system for non-functional characteristics in early stages of the development.

Approaches trying to apply MDA to the development of Web Services already exist (i.e. (Ortiz and Hernández, 2006; Jegadeesan and Balasubramaniam, 2009; Hafner et al., 2006)). However, these approaches do not offer a thorough support for analysis, generation of code, access control descriptors, Web Services Description Language (WSDL) (W3C, 2001), WS-Policy (W3C, 2006a) and WS-SecurityPolicy (OASIS, 2007c).

This thesis proposes the application of the separation of the design into multiple concerns (may them be called views, aspects, or any other alternative name) to integrate the development of non-functional properties of service-oriented systems in a modular, reusable, and maintainable manner. Such a combination of technologies generates really interesting benefits. For instance, model-driven development provides platform and implementation independence, and facilitates design reuse. On the other hand, service-oriented architectures allow for the creation of loosely-coupled, geographically disperse, on-demand systems. Domain-specific metamodels, in turn, provide a much closer representation of the domain, greater understandability, and mayor productivity increments. Finally, MDSoc/AOM concepts supports greater code/design modularization, leveraging understandability and maintainability of the design.

The following sections describe the elements that compose the implementation, and the steps that were necessary to accomplish its development. Section 4.2 presents a high level view of how the different metamodels relate to each other, while section 4.3 presents each one the MMs that compose this particular implementation. Next, section 4.4 deals with the process of defining the particular implementation of the IMM metamodel. The mappings between different metamodels in the different abstraction levels are presented in section 4.5. An analysis and a metric for the IMM metamodel are presented in section 4.6. Then, a description of the implemented prototype is provided in section 4.7. Finally, section 4.8 provides a summary of the chapter.

4.2 Integration, Transformation, and Composition of Meta-models

A great piece of the work performed for this thesis was carried out in the form of definition and implementation of the relations between the different participating metamodels. This section describes, in a broad view, how these metamodels relate to each other. Later, finner grained details are provided on the composition of the IMM metamodel in section 4.3.6, and on the mappings between metamodels that belong to different abstraction layers in section 4.5.

Some order is necessary to introduce how the different MMs relate to each other. A particular way of grouping the participating metamodels may be the just mentioned division by different abstraction layers. Figure 4.1 presents the relations between the metamodels in the CIM layer, and the IMM in the PIM layer. The reader may visualize in the picture the different participant metamodels of the layer: BPMN (annotated with a security vocabulary), SecureUML, QoS framework metamodel, the weaving metamodel, and the IMM integrated metamodel. Each one of these metamodels is further described in a corresponding section below in this chapter. The figure clearly describes how the security vocabulary drives the discovery of the NF models, and the models, corresponding to these different metamodels, are transformed (by means of model-to-model transformations) into the IMM.

Figure 4.1: CIM and PIM Level Metamodels.

Figure 4.2 presents, in turn, the relations between the IMM, the metamodels that compose it, and the metamodels in the PSM layer. Some of these metamodels (WSDL, WS-Policy) were already presented in the previous chapter. The rest are also presented in corresponding sections below in this chapter.

In the picture, the reader may see how three different metamodels (SecureUML, CBDI/SAE's MM policy package, and a proprietary metamodel addressing the functional piece of SOAs) are composed to conform the IMM metamodel. This process is detailed in section 4.3.6. Models corresponding to the IMM are transformed (M2M) into target specific models (WS-Policy and WSDL models, specific to the WS platform). WSDL and WS-Policy documents are later derived from these modes by means of M2T transformations. Code stubs and declarative configuration files for the target programming frameworks are also derived by M2T transformations.

Finally, Figure 4.3 provides the big picture of the relations between all the metamodels. There, also the use of UML annotated with the SoaML profile is included, used in case the modeler decides to provide a PIM level functional input, instead of one derived from the business process model.

4.3 NF Metamodels

In chapter 3, the different metamodels independent of the non-functional concern under consideration have been presented. This section, on the other hand, deals with the presentation of the metamodels that were selected/created to represent the NF characteristics *security* and *access control*.

4.3.1 Modeling Security Intents at the CIM level

As previously mentioned in section 3.3.1, the notation selected to represent services at the CIM level is the Business Process Model and Notation, or BPMN. This section describes how security and access control considerations may be expressed on BPMN to address the different requirements or intents.

(Firesmith, 2004) clearly distinguishes between security requirements, driving security policies to achieve these goals, and the security mechanisms that fulfill these requirements. They propose

Figure 4.2: PIM and PSM Level Metamodels.

Figure 4.3: Relations Between All Metamodels.

the idea that, at requirements level, what should be specified is *what is needed* instead of the mechanisms by which it must be achieved. This thesis follows this guideline.

At the CIM level, abstract security intents (as means for stakeholders to express their security concerns without the need for a detailed technical knowledge) appear as an appropriate strategy. Ontologies have in the last years (noticeably with the advent of the semantic web) risen as useful tools to agree on a common language for the description and discovery of elements (web services, for instance).

The use of an ontology at the CIM level may provide not only a means to define a common language for security intents, which is the task described in this section, but may also represent a tool for the matchmaking¹ of the non-functional concerns models contained in the repositories that may fulfill the expressed requirements. Furthermore, annotating the different elements within the NF models with more specific concepts within the different security characteristics allows, by means of a categorization, the generation of pre-filled composition models.

In the security domain, the NRL ontology (Kim et al., 2005a; Kim et al., 2005b; Kim et al., 2007) represents a very complete option, scoping from high-level security objectives, through security credentials, to technical encryption algorithms, and has been selected to formalize a vocabulary for the security intents. (Asnar et al., 2009) provides a categorization of security mechanisms according to their strength (of particular usefulness for this implementation).

The NRL ontology provides categorization and a matching algorithm, but the preferred syntax for the annotation of the CIM models is defined separately. The reason for the separate definition of such common vocabulary is the objective of isolating the business perspective from the technical vocabulary that is used in the ontology. Several pieces of research² presenting security primitives and means to model security requirements have been considered for the definition of such language.

Table 4.1 presents a list of security primitives available in literature, and the number of publications in the list that make reference to them. These primitives were considered for the definition of the final common language, presented in table 4.2.

In this dissertation, the security categorization in (ISO/IEC, 2011) is employed for the definition of the common security intents vocabulary. The author's criteria and most agreed-on characteristics are also inputs used to determine the common vocabulary. ISO's standard defines *Security* as a top-level characteristic. Sub-characteristics to Security include:

- Confidentiality,
- Integrity,
- Non-Repudiation,
- Accountability,
- Authenticity.

Discussion and Definition of the Security Intents Common Vocabulary

Multiple categorizations are possible. In occasions, the characteristics' names used by the authors are not precisely the ones that appear in the table, but the concept is equivalent. Other times, the concepts address a more technical abstraction level. Here, the focus is on high-level intents, which are then mapped into more detailed mechanisms or particular algorithms.

Not all sub-characteristics identified by all authors are presented in table 4.1, because some are either not referred to by other authors, or they fall out of the scope of the example, or because, sometimes, incompatibilities arise. For instance, Identification, Authentication, and Authorization

¹The process of identifying the elements that fulfill the imposed requirements, and its degree of fulfillment.

²(Dai and Cooper, 2007; Pettersson, 2006; Michael Menzel, 2010; Souza et al., 2009; Toma et al., 2008; Kim et al., 2007; Lin-lin et al., 2008; Asnar et al., 2009; Atluri, 2001; Rodríguez et al., 2007; O'Brien et al., 2005; Firesmith, 2004; Avizienis et al., 2004)

Table 4.1: Security Sub-Characteristics in Literature.

Characteristic	Mentions
Authentication	9
Authorization (Access Control)	7
Encryption	1
Confidentiality	11
Integrity	10
Auditing	5
Non-Repudiation	6
Availability	5
Authenticity	1
Firewalls	1

are regarded in (Firesmith, 2004) as third-level sub-characteristics of Access Control; as well, Anonymity and Confidentiality are considered as third-level sub-characteristics of Privacy.

Availability, for instance, although considered closely related to *dependability*, is not regarded in this dissertation as a sub-characteristic of security, but as an independent, first-level attribute, a view that is shared (Asnar et al., 2006; Lin-lin et al., 2008). (ISO/IEC, 2011) considers *Availability* as a sub-characteristic of *Reliability* (when required for use). Hence, *Availability* is left outside the security intents dialect.

Other characteristics that are not included are those related to risks, threats, hazards, and attacks, as the modeling of attackability or threats to the system is not within the focus of the example. In the same way, trust and federations are also considered out of the scope.

Regarding the discussion of the different security characteristics that repeatedly appear in literature, and that address the requirements that fall into the scope of the particular example, it has to be emphasized that at this abstraction level the selected strategy is to model **security intents** rather than **security capabilities or mechanisms**. In that way, the necessary security expertise of the modeler is kept at a minimum. Consequently, a distinction has to be performed between concepts that represent the goal to be achieved, and those that represent a mechanisms to achieve that goal.

For instance, it can be argued that, at times, *authentication* may be a goal to achieve. However, at the abstraction level being addressed, *authentication* by itself does not represent a goal, but a mechanism to achieve, among others, goals such as *authorization*, *non-repudiation*, or *auditing*. Therefore, *authentication* is left out of the security vocabulary. By the same argument, *encryption* is, as well, a mechanism to achieve, for example, *confidentiality*, and is also left out.

Auditing can be regarded as a mechanism for *non-repudiation*. Nevertheless, *auditing* can be a goal by itself, if the desire is to maintain a register of events, but no necessarily related to the identification of the author of the action or the prevention of the denial of having taken such action. Hence, *auditing* and *non-repudiation* are both maintained in the vocabulary.

Accountability is, in (ISO/IEC, 2011), defined as “the degree to which the actions of an entity can be traced uniquely to the entity”. This resembles very much *non-repudiation*, which is defined as “the degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later”. In this dissertation, *non-repudiation* is interpreted as the trace of an undeniable action towards the participant that produced the action. Therefore, both *accountability* and *non-repudiation* are included in the semantics of the *non-repudiation* intent.

It has already been described how *availability* is not considered as a sub-characteristic of security. The concept of *authenticity*, as described in (O’Brien et al., 2005), relates to the trustworthiness in

Table 4.2: Security Intents Common Vocabulary.

Intent	Attributes	NRL Ontology Mapping
Authorization	Pair{Attribute, RequiredValue}	Authorization
Confidentiality	Level (Strong, Mild, Soft)	Confidentiality
Integrity	Level (Strong, Mild, Soft)	MessageIntegrity
Auditing	No Attributes	UserAuthentication, MessageAuthentication
Non-Repudiation	No Attributes	UserAuthentication, MessageAuthentication, MessageIntegrity, ReplayPrevention

that the identified user is responsible for the action. Hence, it spans over other concepts such as *authentication*, *integrity* and *non-repudiation*. A strong *integrity*, together with *non-repudiation*, would achieve a strong *authenticity*. *Authentication*, as already described, has been considered a required mechanism for the achievement of *non-repudiation*.

Firewalls, as presented in (Pettersson, 2006), is included in table 4.1 to illustrate on the different abstraction levels and technicality in which non-functional characteristics are addressed. *Firewalls*, no doubt, are a technical mechanism useful to achieve many security considerations, but represents a much lower abstraction level than the one addressed here (at the CIM level).

The resulting security intents vocabulary, presented in table 4.2, represents only one particular classification of security primitives; many alternative taxonomies are possible. It is, nevertheless, appropriate for the description of the different security concepts appearing in the use case presented in chapter 5.

Attributes are included in the vocabulary to allow a further refinement of the security intents. For instance, a certain degree of confidentiality may be specified, either strong, mild, or soft. In order to map these attributes to implementations, a categorization of the strength of the available algorithms is necessary. In this dissertation, the categorization available in (Asnar et al., 2009) is adopted.

A different attribute is defined for the case of *authorization*. Inspired in attribute-based access control (ABAC) (Priebe et al., 2004; Yuan and Tong, 2005), a 2-tuple (pair) attribute-value describes the required credential to be presented for the authorization to take place. In chapter 5, where role-based access control is used, the pair would be (attribute='Role', RequiredValue='RequiredRole').

Table 4.2 also presents the mappings of the different terms in the common vocabulary to the security objectives defined in the NRL ontology.

4.3.2 Access Control Input Metamodel: SecureUML

The selected metamodel for access control inputs is SecureUML (Lodderstedt et al., 2002), an RBAC (one of the currently most used access control techniques) metamodel. The IMM access control part, described below in section 4.3.6, incorporates a more generic metamodel (presented in (Mouelhi, Baudry and Fleurey, 2008) and (Mouelhi et al., 2010)), supporting not only RBAC but other access control techniques.

As described by (Lodderstedt et al., 2002), SecureUML introduces the metamodel types “User”, “Role”, “Resource”, “Action”, “AuthorizationConstraint”, and “Permission” as well as relations between these types. Figure 4.4 portrays SecureUML.

Class *Resource*. A *Resource* represents an element subject of being secured.

Figure 4.4: SecureUML Metamodel.

Class *Permission*. A relation class, connecting a role to a *Resource*, via an *Action* (e.g., read, write, or execute) that can be performed on that *Resource*. The semantics of *Permission* is defined by the *Action* elements used to classify the permission.

Class *Action*. Every *Action* represents a class of security-relevant operations on a particular type of protected resource. A security relevant action “execute” on a method/operation, or CRUD actions on attributes are plausible examples.

Class *AuthorizationConstraint*. An *AuthorizationConstraint* is a part of the access control policy of an application. It expresses a precondition imposed on every call to an operation on a particular resource, which usually depends on the dynamic state of the resource, the current call, or the environment. The current time, day of the week, or being the owner of the resource represent possible constraints. Such a constraint is attached indirectly, via *Permission*, to a particular model element representing a protected resource.

For example, imagine the modeler wants to create an access control rule allowing the “admin” role to execute the “helloWorld()” operation, but only from Mondays to Fridays. Under SecureUML, such rule would require the following instances of the different metaclasses:

- an instance of “Role” named *admin*,
- an instance of “Resource” representing the “helloWorld()” operation,
- an *execute* instance of “Action”, associated to the operation,
- *workingdays*, an instance of “AuthorizationConstraint”, representing Mondays to Fridays,
- an instance of “Permission”, relating the *execute* action on the *helloWorld()* operation with the *admin* role and the *workingdays* constraint.

4.3.4 Weaving Metamodel

Weaving metamodels guide the composition mechanisms of the different non-functional concerns models with the functional model. In this case, weaving access control models and weaving security policy models imply different weaving associations. Therefore, two different weaving associations metaclasses, *AddPolicyReference* and *Combine Resource*, are defined, as presented in Figure 4.6.

The semantics of the associations are implemented as part of the composition rules. In the case of policy references, one new “PolicyReference” object is created on the model, with its URL value derived from the ID property of the policy, and the location property of the document containing it.

For access control compositions, a *Resource* is associated with some particular actions (i.e. read, write, execute) described in the access control model. Correspondent actions are then created for the resource, and all associated permissions are introduced in the model (permissions connect roles with actions on resources, according to some attached constraints).

Currently, *resources* that are subject of access control rules include operations, entities, interfaces, etc. Permissions on interfaces propagate to all its operations and all its realizations, in the same way that all permissions on entities propagate to all operations accessing them. Conflict resolution between propagated permissions is one of the possible analysis to be performed on the unified IMM model, as presented in section 4.6.2.

An example may clarify the use of weaving associations. Picture, for instance, an operation named “helloWorld()”. Imagine that the modeler wants to ensure that authorization is enforced for this operation, and selects a *UsernameToken* to fulfill such intent. An *AddPolicyReference* association links the operation to the security token in the weaving model.

For the *UsernameToken* security policy, a *PolicyDocument* is created in the IMM model, containing a *Policy*, and within it, a unique *PolicyAlternative* containing a *UsernameToken PolicyAssertion*. Inside the model, a *PolicyReference* is associated to the “helloWorld()” operation. This *PolicyReference* points to the *UsernameToken* policy recently defined.

The modeler also requires a *Clerk* role to be able to execute this operation, and may not execute it on weekends, for instance. In a second weaving model, a *Combine Resource* association links the “helloWorld()” operation with a *Permission* element in the SecureUML model. This permission relates the “Clerk” role, the “Execute” action, and the “Weekdays” constraint. At the time of composing the models, a new action (“helloWorldExecuteAction”) is created under the “helloWorld” operation. This new action references all associated permissions, in this case grouping together the “Clerk” role and the “Weekdays” constraint.

The use of these composition mechanisms is described again in the example in chapter 5. For the purposes of this implementation, non-functional properties are assumed to be orthogonal, and no composition is defined between them. The base weaving metamodel, the definition of weaving metamodel extensions, and the process of model compositions is described in (Didonet and Valduriez, 2009).

The description of viewpoints relations by the specification of individual relations does not scale well if performed by hand by a modeler. Tooling providing (semi)automatic support for the generation of the above mentioned pre-filled composition models is of the foremost importance, almost imperative. The implementation of the CIM2PIM step on top of the existing prototype is planned for the near future. The generation of these tentative composition models should be addressed by this implementation.

4.3.5 WS-SecurityPolicy Metamodel

A WS-SecurityPolicy metamodel has been defined for this prototype as an extension to the WS-Policy metamodel presented in the previous chapter. This allows its use inside the policy editor, facilitating the expression of standard security policy assertions, and allowing the incorporation

Figure 4.6: Weaving Metamodel.

Figure 4.7: WS-SecurityPolicy Metamodel (Conceptual Partial View).

of rules (defined by the standard and implemented in the prototype) for the validation of such assertions, preventing the definition of invalid policies.

It also provides a means, thanks to the inclusion of a “*Preference*” attribute, to indicate the available and/or preferred security mechanisms (encryption algorithms, authentication tokens, etc.) in the platform. This use of the WS-SecurityPolicy represents, in MDA terminology (OMG, 2003), a *Platform Model*.

For instance, if a mapping to WS-SecurityPolicy implies that a authentication token is to be included in the policy, the transformation rule may query the platform model for available authentication tokens. Imagine that the platform model includes X509, Kerberos, and SAML tokens, each type assigned a *preference* value of 1, 10, and 20 respectively.

If the transformation rule specifies, for any reason, one of this particular tokens specifically, SAML for instance, it may be used, as it is available in the platform model, and therefore assumed to be supported by the target infrastructure. If, on the contrary, it only demands the use of authentication tokens in general, by considering the preference value, X509 tokens would be the ones selected, as are the preferred choice.

The WS-SecurityPolicy metamodel, a conceptual view of which is presented in Figure 4.7, is, to the best knowledge of the author, the first implementation of a WS-SecurityPolicy metamodel available.

4.3.6 iMM Metamodel

The intermediate metamodel developed for this approach is one of the fundamental parts of the framework. It was designed to be as general as possible, in order to maximize support of access control techniques and policy standards. It is composed of three differentiated parts, depicted in 4.8 and further described in table 4.3:

- Functional part (marked “System Design” in Figure 4.13).
- Access Control part (marked “Access Control” in the figure).
- Service Policies part (marked “Policy”).

The functional part of the metamodel existed previously, and some tools, mappings, and transformations were already available around it (nevertheless, such elements were not employed in

Figure 4.8: Composition of the iMM Metamodel.

Table 4.3: Parts of the IMM metamodel.

Part	Origin	Characteristic
Functional	(CDTI, 2006)	CBSD approach, used to describe service components, entities and interfaces, among other metaclasses. Capable of modeling workflows and events.
Authorization (Access Control)	(Mouelhi, Fleurey, Baudry and Le Traon, 2008)	Supports DAC, MAC, RBAC, and OrBAC techniques, among others, and mutation between those techniques.
Policy	(Dodd et al., 2007)	Describes policies associated to services.

this dissertation). It follows a component-based approach, and is also designed to model workflows and events. However, being the focus of this work on non-functional properties, these and other capabilities that the MM presents for modeling functional concerns are currently not in use.

As described in the previous section, Mouelhi’s generic metamodel used for the access control part (included in Figure 4.13) allows for the use of different access control techniques as inputs (DAC (Lampson, 1974)], MAC (Bell and LaPadula, 1976), RBAC (Sandhu et al., 2002), OrBAC (Kalam et al., 2003), among others), and mutation between those techniques (Mouelhi, Fleurey, Baudry and Le Traon, 2008). This possibility of mutating one access control technique into another is of great usefulness when targeting different AC platforms and infrastructures.

Finally, the “Policy” package of CBDI-SAE Meta Model for SOA Version 2.0 (Dodd et al., 2007) was selected for the service policies part due to its grade of maturity and flexibility. The complete CBDI-SAE Meta Model was not adopted as it does not include access control support, and it spans over different abstraction levels (the IMM only addresses a technical PIM level).

The three selected metamodels, where then studied to identify correspondences, equivalent concepts/metaclasses that provided points of union. This not only represent identifying redundant concepts, but also generalizations/specializations of concepts, relationships (i. e., which metaclasses in the functional model represent the *resource* concept included in the access control metamodel, or which represent a policy subject), etc.

The process of the composition of the IMM metamodel is presented next in section 4.4. Section 4.4.4 describes how these different concepts were identified, and the final result can be found in section 4.4.5.

4.4 Definition of the IMM Metamodel for Security and Access Control

This section describes the process of definition and composition of the IMM metamodel. In this thesis, several view models are used to specify a non-functional aware system. In cases like this one, when the view models are expressed in different Domain Specific Modeling Languages, a problem arises because a heterogeneous composition, a model unification is required.

A heterogeneous composition is the integration of two models expressed in two different languages. To (Yie et al., 2009), this means that is necessary to define the compositional semantic for every pair of concepts in both languages. This appreciation is not shared. As is shown in this section, this is not strictly the case in this thesis, where compositional semantics are defined only for overlapping elements, or for elements between which an association has been identified.

(Vallecillo, 2010) defines model unification as:

Given a set of models $M_1, M_2, \dots, M_n$, and a set of correspondences between them $c_{ij} = C(M_i, M_j) \subseteq \mathbb{P}(M_i) \times \mathbb{P}(M_j)$, a unification is a new model M_G and a set of functions $t_i : M_G \rightarrow M_i$ (projections) that respect the set of correspondences, i.e., $C(t_i(M_G), t_j(M_G)) \subseteq C(M_i, M_j)$.

In this dissertation, the composition was addressed by selecting a set of generic metamodels, where each could describe a number of the addressed concerns. Next, a process of identification of correspondences or overlaps (matching) between these generic metamodels takes place, to later compose (weave, rather than merge) the different metamodels into the final expression of the iMM.

The resulting metamodel is able to represent SOA systems that consider a number of, in this case, security and access control demands. Afterwards, mappings (transformations rules) are defined between the high-level views metamodels selected to express the different concerns and the iMM.

It is important to observe that model elements never appear in more than one of the different available viewpoint models. In other words, functional elements are presented on the functional concern model only, access control elements in the AC model only, and so on. The weaving models are the ones in charge of expressing the correspondences/links between the elements. This approach, similar to the one employed in AOM, avoids the coordination problem described in (Hessellund, 2009).

The mechanism used to define the integrated metamodel is not the only available approach: organization in different detail levels of the same metamodel, referencing foreign elements (present in another model, corresponding to other MM), etc. The case of the WS-SecurityPolicy metamodel is another example, where it was designed as an extension, a specialization of the WS-Policy MM, or the case of the weaving metamodel in section 4.3.4, defined as an extension of a generic weaving MM (Didonet and Valduries, 2009).

The next sub-sections describe the different metamodels that compose the iMM, the matching phase, and the resulting iMM metamodel. The use of the iMM in the model-driven development of a SOA system is presented in chapter 5.

4.4.1 Functional Metamodel

The functional part of the metamodel specializes in the construction of SOA applications, defining the concepts present in such applications. It is divided in five packages, as presented in Figure 4.9, each package grouping a particular type of concepts.

The Component package contains concepts related to the general structure of the model (i.e., components and interfaces composing the application). The implementation of a component will be integrated by elements defined in other packages. The Entity package fundamentally describes the entity concept, concepts related to data persistence, and other related concepts, such as entity associations, queries, or identity restrictions. Interaction package deals with graphical user interfaces' concepts. Process package contains concepts used for the specification of process flows, most concepts based on the BPMN standard, in order to support process modeling tools. Finally, the Services package represents the concepts used in the definition of the implementations of the interfaces provided by the different components. The fundamental elements include Service and Remote Service.

The intention behind the use of Component-Based Software Development (CBSD) is to provide a “map” of the system. This concepts indicate among which components the different functionality of the system is distributed, how such components depend from each other, and their external dependencies. Fundamental concepts in a component model are interfaces, the proper components, and its implementations.

Concepts in use in the context of the functional architecture include System, Components, Ports, Interfaces, Operations, etc.

Figure 4.9: Functional Metamodel, Packages View.

Each component may have different associated implementations. A usual case is that of component versioning, where a component may have several active versions of itself within a system.

Interfaces represent contracts between different components. Services, as shown in Figure 4.10, are the materialization of these interfaces. A service implements all the operations defined by an interface (public operations), although it may also define other auxiliary (private) operations. Services will use, for the logic of their operations, other elements such as entities or remote services.

Besides the elements described above, the metamodel also counts with different concepts that allow for the modeling of:

- Graphical User Interface.
- Persistence.
- Processes.

Those elements fall out of the scope of this thesis and are, thus, not described in this document.

4.4.2 Generic Access Control Metamodel

The generic access control metamodel, defined in (Mouelhi, Baudry and Fleurey, 2008; Mouelhi et al., 2010) and shown in Figure 4.11, is divided into two levels of instantiation:

- *Policy formalisms* are defined with the classes *PolicyType*, *ElementType* and *RuleType*. A *PolicyType* defines a set of element types (*ElementType*) and a set of rule types (*RuleType*). Each rule type has a set of parameters that are typed by element types.
- *Policies* may then be defined using the classes *Policy*, *Rule* and *Parameter*. A *Policy* must have a type (an instance of *PolicyType*) and defines rules and parameters. The type of a policy constrains the types of parameters and rules it can contain. Each parameter has a type which must belong to the element types defined for the policy type. If the hierarchy property of the parameter type is true, then the parameter can contain children of the same type as itself. Policy rules can be defined by instantiating the *Rule* class. Each rule has a type that belongs to the policy type and a set of parameters whose types must match the types of the parameters of the type of the rule.

Figure 4.10: Functional Metamodel, Services View.

With reference to SecureUML (RBAC) examples of *RuleTypes* are the associations between *Subject* and *Role*, or between *Resource* and *Action*. Each of these metaclasses (*Resource*, *Action*, etc.) represent the *ElementTypes* typing the different parameters of the rules. The representation of SecureUML elements under this metamodel will become clearer in section 4.5.2.

These two parts of the metamodel have to be instantiated sequentially: first define a formalism, then define a policy according to this formalism. As already described, mappings from MAC and DAC (Mouelhi, Fleurey, Baudry and Le Traon, 2008) and RBAC and OrBAC (Mouelhi, Baudry and Fleurey, 2008; Mouelhi et al., 2010) have been defined for this metamodel. Most importantly, and one of the main reasons for its selection, this access control metamodel supports mutation (Mouelhi, Baudry and Fleurey, 2008; Mouelhi, Fleurey, Baudry and Le Traon, 2008) between access control models, allowing to express the access control rules in the A.C. mechanisms available in the target platform.

4.4.3 Policy Package of the CBDI-SAE Metamodel for SOA

CBDI Service Architecture & Engineering (CBDI-SAE) is a comprehensive, defined approach for service oriented architecture (SOA) including taxonomy, classification and policies together with repeatable service engineering processes that guide the delivery of the agile enterprise, implemented in a knowledge-base with integrity between the architecture concepts, processes, tasks, techniques and deliverables (Dodd et al., 2007). One of the objectives of this metamodel is to supply a MM that is helpful to software vendors building tooling for SOA.

The CBDI-SAE meta model considers a policy to be a strategy or directive defined independently from how it is carried out; it is some kind of rule to be followed, though the model envisages some rules are mandatory while others are guidelines, as explained by the 'strictness' attribute of the

Figure 4.11: Generic Access Control Metamodel (Mouelhi, Baudry and Fleurey, 2008; Mouelhi et al., 2010).

Policy meta type. As defined in (Dodd et al., 2007), it is intended that Business and IT policies be represented by this model, as well as SOA/Service policies. In our case, only the last type of policies are applicable. Therefore, not all metaclasses defined in this package are necessarily present in the iMM metamodel. This section describes the package exactly as defined in (Dodd et al., 2007), and presented in Figure 4.12. When the use in iMM differs from that in (Dodd et al., 2007), it is explicitly highlighted.

For a simple *Policy*, there only needs to be one *PolicyAlternative* and one *PolicyAssertion*. All the assertions within one alternative are AND-ed together. The applicability attribute of *PolicyAlternative* may define a strict rule, or helpful guidance, on when to choose that alternative. Which one it is depends on how it is written.

A *Policy* should always govern something. So the *Policy* is associated with an instance of *PolicySubjectType* which represents the type of thing being governed by the assertions. For example, within the whole CBDI-SAE metamodel, this could be a type already defined in the meta model such as Automation Unit or Service Level Agreement, or it might be a type that is not explicitly represented by a meta type, such as Supplier or Business Analyst. In the case of the iMM, and as described in 4.4.4, specific metaclasses have been identified as policy subject types.

A *Policy* must always belong to a policy category, which is identified as *PolicyType*. The model allows a policy type hierarchy to be defined. A *PolicyType* helps identify the proper generator/transformation which should be applied to a *Policy*. UML’s QoS profile (OMG, 2008d) provides a categorization that may prove useful in this regard.

A *Policy* is not necessarily “universal”, so the model requires:

- Every *Policy* to be associated with the “Organization Unit” to which it applies. This will often be the top-level Organization Unit (the entire enterprise), but some might be restricted to a division of the company, or to partner organizations. This metaclass seemed unnecessary for the purpose of the implementation and is, therefore, excluded.
- It can also be restricted to certain instances of the subject type. This restriction is defined by means of the Policy Context concept.

The rest of the metaclasses defined in the CBDI-SAE MM seemed unnecessary within the same level of abstraction (they may be useful to identify technical policy assertions implementing high level business/service policies), and are not included in iMM.

Figure 4.12: CBDI-SAE Metamodel for SOA's Policy Package (Dodd et al., 2007).

Referring back to the “helloWorld()” operation example, in which authorization is enforced by a *UsernameToken*, under this metamodel the *PolicyType* value would be “Security”, the *Policy* would only contain a unique *PolicyAlternative* and within it, the *UsernameToken PolicyAssertion*. The *PolicyEffect*, serving only documentary purposes, would have the value “Authorization”. Finally, the *PolicySubjectType* would include operation, and a *PolicyReference* would be attached to its “helloWorld()” instance.

4.4.4 Identification of Correspondences

The next step after selecting the different metamodels that would compose the iMM is to identify overlapping concepts among them. Being the focus of the metamodels mostly different and independent domains, overlapping concepts between the functional metamodel and the other two are limited to those of the model elements that may be affected by service policies, access control rules, or both.

Conversely, overlapping concepts between the access control and the policy metamodels reside in more abstract elements regarding the notion of policies and its expressions. In other words, concepts that are not dependant of the specific domain for which the metamodel was defined.

Policy and Functional Parts

As previously described, overlapping concepts between these two metamodels are limited to those of the model elements that may be affected by service policies. In this way, elements subject of being affected by a service policy are to be defined as *PolicySubjectType*, the metaclass present in the policy package in Figure 4.12 which corresponds to this concept.

In turn, it is necessary to identify the concepts in the functional model which are subtle to a service policy. It was decided to follow the directives specified in the WS-PolicyAttachment specification (W3C, 2006b) as a guide for this task. Although a technical specification for a particular target platform, it is conceptually close to the service-oriented domain, and is appropriate for this duty. Therefore, and following these directives, the identified metaclasses are:

- Interface
- Parameter
- Service
- Operation
- Port

Accordingly, all mentioned metaclasses are defined as specializations of *PolicySubjectType*.

Access Control and Functional Parts

The access control metamodel, as presented in Figure 4.11, does not incorporate a metaclass to represent the concept of the system element affected by an access control rule. Elements subject to be the target of access control rules are to be identified as *Resources*, and are affected by the *Policy* concept in the AC metamodel.

Multiple resources could be subjected to AC rules. A distinction can be made between external (accessible from the exterior) and internal (not directly accessible from the exterior) resources. However, as the functional modeling is not of primary importance in this thesis, only access rules to two internal elements will be modeled: to *Entity* and *EntityAttribute*. The use of these two elements is considered sufficient to illustrate the usefulness of the approach.

Additionally, operations are elements exposed to the exterior of the system, and which are also subjects to access control. Groupings (such as interfaces) and particular implementations of operations may be subjected to AC rules to facilitate the modeling activity. On the other hand, and as the only means to access a inner element are through exterior elements, by transitive AC

rules applied to such inner elements equally apply to parameters or operations accessing them. As a result, the complete list of access control resources include:

- Interface
- Port
- EntityAttribute
- Operation
- Service
- Parameter
- Entity

Policy and Access Control Parts

It was previously commented that overlapping concepts between these two metamodels may reside in abstract elements, related to the notion of policies and the expressions of policies, which were not dependant of the specific domain for which the metamodel was defined.

Precisely, two common metaclasses appeared in the metamodels in figures 4.11 and 4.12:

- Policy
- PolicyType

For the concepts to be equivalent, one further restriction has to be imposed to the AC metamodel: all rules in a particular *Policy* are to be applied to the same resource (and not individually assigned). Thereupon, the *Policy* concept becomes equivalent to the grouping of assertions that apply to a particular policy subject.

PolicyType presents a concept that is close in both metamodels. (Dodd et al., 2007) includes this metaclass but does not provide a categorization to use it, leaving it to the criterion of the modeler. Comparatively, (Mouelhi, Baudry and Fleurey, 2008) describes how a *PolicyType* imposes the available rules and parameters types for a *Policy*. This is, again, a categorization element, an equivalent concept.

As a consequence of this identified overlapping, a new element appears in the range of the AC metamodel: *PolicySubjectType* serves the equivalent purpose of that of *Resource*. Moreover: *PolicySubjectType* represents a sub-collection of *Resource*. Accordingly, both concepts were merged and unified under the *PolicySubjectType* metaclass. This is depicted in Figure 4.13. An OCL rule may be created to prevent service policies from being applied to inner elements, identifying them by means of their *PolicyType*.

4.4.5 Resulting iMM Metamodel

This concludes the identification of correspondences, finally defining the iMM metamodel. The resulting iMM metamodel is presented in Figure 4.13. The particular implementation of the process presented in chapter 3, employing the metamodels described above, is presented in Figure 4.14.

The black arrows in Figure 4.14 between the PIM level and the platform are conceptual and represent the development/mapping of the NF into the target platform. The remaining arrows represent either transformations or compositions between models, and the corresponding mappings are presented next in section 4.5.

Figure 4.13: iMM Metamodel (conceptual partial view).

Figure 4.14: Implementation of the Approach in Figure 3.1 for Security and Access Control.

4.5 Metamodels' Mappings

The different compositions and transformations in the development process are guided by rules that take into account mappings between source and target metamodels. This section describes the mappings between the metamodels in the different abstraction levels. The example in chapter 5 makes use of these mappings, and may clarify some aspects of their use.

First, the CIM-to-PIM mapping is described, presenting a BPMN to SoaML mapping based on the work of several authors. Next, PIM refinements are introduced: SecureUML, QoS, and SoaML mappings towards iMM. Finally, the PIM 2-to-PSM mappings (defined for the generation of results) are presented, detailing the derivation of WSDL and WS-Policy models from iMM.

4.5.1 CIM to PIM

As already described, at CIM level, an annotated business process model describes the different services and its non-functional intents. On the non-functional domain, for security and access control concerns the NRL ontology (Kim et al., 2005a) has been proposed to offer a categorization of the concepts, and its matchmaking algorithm as the mechanism to drive the automatic discovery of fulfilling NF models.

It was previously mentioned that the functional aspect is not the focus of the thesis. Nevertheless, the functional (business) behavior is the center of attention in most model driven software developments, and because of this, the topic is taken into account. Therefore, a business process to functional PIM model mapping is presented here, derived from research referred to below. Such mapping is the subject of several pieces of research, and is properly addressed in related publications.

On the functional side, the MINERVA (Model drIveN & sERvice oRiented framework for the

Table 4.4: CIM to PIM Mappings (Delgado et al., 2010; Lemrabet et al., 2010; Elvesæter et al., 2011).

Process Model Concept	SoaML Concept
Pool/Lane /Role	Participant
Activity (w/incoming flow)	ServicePoint
Activity (w/outgoing flow)	RequestPoint
Message/Data flows/Data Objects	Message types/Entities

continuous business process improvement & related tools) framework (Delgado et al., 2010) defines a BPM to SoaML mapping by means of an ontology and QVT transformations. Similarly, (Lemrabet et al., 2010) also maps BPM to SoaML using ATL rules.

Within the MDSE (Model-Driven Service Engineering) methodology, defined in (Elvesæter et al., 2011), again BPMN and SoaML are used to describe services at CIM and PIM levels. Although separate business diagrams are suggested in MDSE for carrying out a refinement process that link BPMN and SoaML concepts, making use of non-standard extensions to BPMN, mappings from BPMN to SoaML are defined.

The transformation rules in the previous three publications are equivalent, and are presented in table 4.4. A SoaML model would represent an already supported functional input. However, this intermediary steps is not necessary, and a direct transformation to a iMM model is possible by unifying the mappings in tables 4.4 and 4.6.

Alternatively, UML's activity diagrams could have been used. The CIM-to-PIM-to-PSM procedure would be practically identical. BPMN to UML's activity diagrams mappings, and vice-versa (with some limitations) are defined in (Macek and Richta, 2009).

4.5.2 SecureUML to iMM

SecureUML is an access control metamodel which has been selected to represent one of the possible inputs into the development process. RBAC models in SecureUML are mapped into the access control part of the iMM metamodel, presented in Figure 4.13. The mappings between SecureUML and the AC part of iMM, presented in Figure 4.15, are based on the modeling of RBAC rules as defined in (Mouelhi, Baudry and Fleurey, 2008).

As already described in section 4.4.2, the access control part in iMM is itself composed of two parts, describing policy formalisms and the actual policies respectively.

Policy-formalisms define the particular associations (or rules) and the types available in an access control model. Its mappings deal with *RuleType* and *ElementType* elements. In the case of RBAC, rule types include all associations:

- Permission,
- Subject-Role,
- Action-Resource,
- Group-Group,
- User-Group,
- Role-Role,
- AtomicAction-CompositeAction,
- CompositeAction-CompositeAction.

*ElementType*s include most metaclasses:

- Role,
- Group,
- User,
- AuthorizationConstraint,
- AtomicAction,
- CompositeAction,
- Resource.

Figure 4.15: SecureUML to iMM Mapping.

Rules and parameters have the correspondent *RuleType* or *ElementType* associated. *PolicyType* may be defined as “SecureUML” or “RBAC”.

Policies, on the other hand, represent the actual instantiations of the rules defined in the formalism. As a general rule, the different instances of the metaclasses in SecureUML (*Roles*, *Subjects* (Groups or Users), *Resources*, *Constraints*, and *Composite* or *Atomic Actions*) are mapped into instances of *Parameter* of the iMM AC part, and associations in SecureUML are mapped into instances of *Rule* in iMM (composed by the appropriate “parameters”). The only exception to this rule is in the case of SecureUML’s *Permission* metaclass. *Permission* plays the role of an association class, linking resources with roles, constraints, and actions. Accordingly, the actual mapping of a *Permission* is a rule composed of *resources*, *roles*, *constraints*, and *actions* parameters.

Recalling the example in the SecureUML section, under iMM’s AC part, the different elements would be represented as:

- the whole permission association would be mapped into an instance of *Rule* of a “Permission” *RuleType*, containing all the following parameters.
- *admin* would be an instance of *Parameter*, with “Role” as its *ElementType*.
- Following the same approach, *helloWorld* would be an instance of a “Resource” typed *Parameter*.

Table 4.5: QoS to iMM Mapping.

QoS Metaclass	iMM Metaclass
QoSCategory	PolicyType
QoSContext	PolicyAlternative
QoSValue	PolicyAssertion

Table 4.6: UML's SoaML Profile to iMM Mapping.

Stereotype	iMM Metaclass
ServiceSpecification	Component (definition of a component)
ServiceProvider	Service and ComponentImplementation
ServiceConsumer	RemoteService
ServiceInteractionPoint	Port
ServiceInterface	Interface (provided by a Service)
MessageType	Message

- The *helloWorldExecute* action is the correspondent “Action” parameter.
- Finally, *workingdays* would be the “AuthorizationConstraint” parameter of the rule.

4.5.3 QoS Framework Metamodel to iMM

Mapping the QoS Framework metamodel into iMM is a bit more complicated, as the mapping depends not only on the the metaclass, but on its particular instance. The QoS framework defines:

- A taxonomy defining the different QoS Characteristics and Dimensions within a QoS Category.
- The actual values (instances) of these characteristics applied to the subjects.

This is due to the generic nature of the QoS framework. The taxonomy is necessary to be able to model the NFPs within the framework. In contrast, the iMM model should only contain the actual instances/values of the non-functional characteristics that are applied to the different subjects.

In the QoS metamodel, the *QoSCharacteristic* and *QoSDimension* metaclasses are used to define the vocabulary. On the other hand, *QoSCategory*, *QoSContext*, and *QoSValue* are the elements to be transformed into iMM model elements. Table 4.5 presents the mappings between these two metamodels.

The example in chapter 5 illustrates on the use of these mappings in the case of security policies.

4.5.4 UML with SoaML Profile to iMM

In case a functional PIM input is provided, a UML model with applied SoaML profile (OMG, 2009) is currently supported. The different concepts present in the SoaML profile address the same domain that the functional part of the iMM metamodel. Therefore, the mapping is relatively straight, as presented in table 4.6.

4.5.5 iMM to XML Schema

Based on the many mappings definitions from UML to XML Schema (e.g., (Huemer and Liegl, 2007; Routledge et al., 2002; IBM, 2005; Gorelik, 2008)), a iMM to XML schema mapping was

Table 4.7: iMM to WSDL Mapping.

iMM Metaclass	WSDL Metaclass
Service	Service
Interface (provided by a Service)	Porttype
Operation (offered by a Service Interface)	Operation
Parameter	Message, Part, Element
Port	Port
PolicyAssertion (applied to a subject)	PolicyAttachment

defined. This is such a common mapping that is not described here. Automatic generation of XML schemas from Ecore models is also available in the tooling (Steinberg et al., 2009).

4.5.6 iMM to WSDL

An iMM to WSDL mapping was defined. Being the target domain so close, or even the WSDL domain contained within the service-oriented domain, both iMM and WSDL metamodels share most common metaclasses. Therefore, the mapping is quite direct, as presented in table 4.7. The implementation of the iMM-to-WSDL transformation considers the different values of the “Style” and “Use” variables (rpc-literal, document-literal, document-literal-wrapped), as different values of these variables result in different WSDL elements being created. Listing B.1 shows how a template of ATL’s transformation rules for the iMM to WSDL mapping looks like.

The policy attachments, following the definitions in (W3C, 2006b), are appended to the different elements and reference a particular policy in a document. Such policies and documents are generated as described in section 4.5.7.

4.5.7 iMM to WS-Policy

The mapping between iMM and the WS-Policy metamodel is also quite direct. The reason is that the CBDI-SAE Policy package very much resembles the WS-Policy standard elements. Although the CBDI-SAE Policy package is used along different abstraction levels, in the technical aspect, is very close to WS-Policy metamodel. Table 4.8 describes the mappings between these metamodels.

However, these mappings must consider the actual policy assertion as defined in the corresponding target standard, driven by *QoSCategory*’s value (in the case of *Security*, WS-SecurityPolicy (OASIS, 2007d) assertions). These standards define the selection of the transformation rules that are to be applied to the instances. The transformation rule itself holds the know-how of the particular standard assertion to generate.

Table 4.8: iMM to WS-Policy Mapping.

iMM Metaclass	WS-Policy Metaclass
Service	PolicyDocument
Policy	Policy
PolicyAlternative	PolicyAlternative
PolicyAssertion	PolicyAssertion

4.6 Analyses in the iMM Metamodel

Analyses and metrics are fundamental tools to quantify the trustworthiness of the design. As already mentioned, errors detected in this stage are much less costly to correct. Multiple analyses and metrics can be defined or adapted to be applied on the iMM implementation for security-related properties. Two analyses, defined during the participation on different research projects, are briefly presented here to illustrate on the possibilities. These examples describe how the information available in the model can be interpreted to evaluate the quality of the design.

4.6.1 Requirements Addressing Metric

This metric indicates the extent to which the non-functional intents expressed in the annotated input model have been considered. For its calculation, means to indicate the mapping between the CIM level intents and the PIM level non-functional properties are necessary. A traceability model/document may represent an alternative, although in the approach presented in chapter 3 and above in this chapter, this information is the result of the NRL ontology matching applied between the BPMN model and the NF models in the repositories. In the presented approach, non-functional PIM models are not fully generated from CIM requirements, but alternatively harvested from model repositories. The tool which performs the matches, and proposes the different candidate NF models to the user, should record the particular selection of the developer. If tentative composition models are automatically generated based on the CIM requirements, these weaving models represent a suitable subject for this metric. Otherwise, the composition models created by the user must be employed.

Later, a mapping from PIM level characteristics to PSM/implementation would also be necessary if the scope to be measured is desired to be extended up to this point. Under the particular transformations tools in use, different traceability mechanisms (i.e., (Jouault, 2005; Yie and Wagelaar, 2009; Allilaire, 2009)) can be applied.

The metric itself is quite simple. On the CIM level model, the different annotations (intents) have to be enumerated. By means of traceability models, CIM level elements are associated to PIM level elements. Individually, each intent must have at least one PIM level counterpart. If this is not the case, requirements have not been addressed in a 100%. The process is equivalent in the PIM-to-PSM situation.

4.6.2 Access Control Conflict Analysis

The composition of different realizations of the non-functional intents may introduce conflicts in the resulting architecture. The analysis presented below provides an idea of how these conflicts may be identified in the iMM model.

At architectural (PIM) level, due to the different resources that may be affected by an access control rule, and to the propagation of these permissions, conflicts may rise. For example, if a determined role (assuming the use of RBAC) is required to access objects from a particular class, and a less privileged role is required to execute an operation affecting such objects, the conflict is evident.

Moreover, conflicts may even rise in cases where required roles are compatible, but permission constraints are not. Alternatives exist: in case one constraint is more general than the other, the more restrictive constraint may be maintained while discarding the other, and a warning be presented to the user; or if exclusive, all constraints are kept, and the sum of the restrictions apply. The definition of rules compatibility for the different languages (OCL, Java, etc.) is out of the scope of this work.

As defined in section 4.4.4, the complete list of access control resources, representing inner and external elements, include:

- Interface
- Operation
- Parameter
- Port
- Service
- Entity
- EntityAttribute

Hence, the propagation of the rules on an element inwards may cause that conflicts rise between the elements as indicated in table 4.9. Consequently, the analysis must navigate from a resource with access control rules, through its aggregated resources, and up to the affected elements ³, and determine if a conflict arises from incompatible roles or constraints.

Table 4.9: Access Control Propagation Conflicts

	Serv	Port	Interf	Op	Param	Entity	Ent.Att.
Service		X	X	X	X	X	X
Port			X	X	X	X	X
Interface				X	X	X	X
Operation					X	X	X
Parameter						X	X
Entity							X
Ent.Att.							

4.7 Implementation of a Prototype

As a proof of concept, a prototypical implementation has been developed. In line with the requisite of not reinventing the wheel, an extensive use of available tools and metamodels has been performed. Implementation has taken place over the Eclipse Modeling platform, trying to make the most of the functionalities provided by it.

Although the proposed approach is not tied to a specific implementation technology, after an evaluation of the available tools (Silva Gallino et al., 2009a), it was decided to base the implementation on the use of the ATL (Jouault et al., 2008) transformation engine and its companion tools (AMW (Didonet del Fabro et al., 2005) for model composition, and AM3 (Bézivin et al., 2005) for the extraction of models to/from XML). ACCELEO (OBEO, 2006), a model-to-text (M2T) tool, is employed for code templates and access control descriptors generation.

The implementation of the metamodels in Eclipse (some definitions have been reused, but no implementations were available at the time), the definition of the transformation and composition rules, and the implementation of the different assistants (wizards, cheat sheets, etc.) implied a considerable amount of work.

The adopted strategy was to divide the implementation in different Eclipse plugins, in order to allow for easier maintainability and evolution. In this way, the different artifacts to be generated (WSDL, WS-Policy, code templates), and the different metamodels to be composed (access control, policies) are to be delivered in different plugins. Figure 4.16 shows the proposed division, with each plugin marked by 'PG' and a number. The plugin in charge of the composition of the intermediate model, marked as PG1 in Figure 4.16, is independent of latter generation plugins. The inclusion of

³It is assumed that the functional model appropriately indicates the elements (entities, entities attributes, etc.) affected by operations and parameters, which is the case in iMM.

Figure 4.16: Structure of the proposed solution.

new metamodels for composition, or generation plugins (possibly using different technologies) is made possible by means of Eclipse’s extension point mechanism.

The generation of the different output artifacts has also been made independent between themselves, to allow a separate evolution of each tool according, for instance, to new standards.

The user interface of this prototype offers a cheat sheet and a menu for the modeler to access the different functions. The cheat sheet enumerates the steps described in section 3.2.2, providing mechanisms to launch these different steps on the selected models. The different steps would either launch wizards to select/import models, or launch a transformation/composition. The user can follow these steps all along as presented in the cheat sheet or, alternatively, take the different steps independently as (s)he sees fit by selecting them in the menu.

This prototype was used through the application of the development process to the example in chapter 5. The resulting artifacts (WSDL and WS-Policy documents, target platform’s XML configuration files, access control configuration) were automatically generated, and correctly validated, providing a proof of the usefulness of the approach.

The CIM2PIM step presented in this chapter has been designed, but still not implemented as part of the prototype. The integration in the prototype of ontologies and matchmaking for the automatic discovery of appropriate non-functional models is among the future lines of work. Reusable assets (Reusable Modeling Tools Assets (de Miguel et al., 2011)), IBM’s RAM repository (Rational Asset Manager, implementing the Reusable Asset Specification (OMG, 2005a)) for Eclipse (IBM, 2011b), or the “emftriple” (Hillairet, 2010) project (an effort towards the development of a triple-store repository for EMF models with support for semantic annotations) are capable of hosting these models and associated metadata, and could represent possible alternatives to be integrated in the prototype.

4.8 Summary of the Chapter

An implementation of a framework applying the process defined in chapter 3 for the particular case of security and access control as non-functional characteristics of service-oriented architectures has been described in this chapter. All the different metamodels used for the implementation have been presented. Among them, the intermediate metamodel stands out, capable of containing all necessary information for generating the resulting artifacts. The WS-Policy (and WS-SecurityPolicy) metamodel supports the definition and validation of assertions within an editor, and are equally supported as input or output metamodels. Moreover, the WS-SecurityPolicy metamodel can be also employed as a platform model to further refine the PIM2PSM transformation.

An analysis and a metric for iMM models were proposed. The iMM analysis/metric illustrate the simpleness of performing calculations on NF characteristics of integrated metamodels, such as iMM. Non-functional aware models such as these are the perfect subject of those types of analyses.

The proposed solution, of great flexibility, is an improvement compared to editor-based approaches (such as the one employed in state of the practice tools, as described in (Juric et al., 2010)) tightly coupled to the technologies they have been developed for. The implemented prototype, on the other hand, provides a set of functionalities not always found together in one individual tool.

The application of the defined approach, with the help of the implemented tool, in the development of the well-known Web Services Interoperability Organization's (WS-I) Supply Chain Management (SCM) use case (WS-I, 2003b) is presented next in chapter 5.

Chapter 5

The WS-I Supply Chain Management Use Case

5.1 Introduction

To illustrate on the use of the proposed methodology and tools, the Web Services Interoperability Organization's (WS-I) Supply Chain Management (SCM) use case (WS-I, 2003b) has been selected. The WS-I (Web Services Interoperability Organization, n.d.) is an open industry organization chartered to establish best practices for web services interoperability for selected groups of web services standards, across platforms, operating systems and programming languages. WS-I comprises a diverse community of web services leaders from a wide range of companies and standards development organizations. Members of the WS-I include IBM, Microsoft, SAP, Nortel, Intel, Oracle, Hitachi, and other leading companies.

WS-I's deliverables provide resources for Web services developers to create interoperable web services and verify that their results are compliant with WS-I guidelines. Key WS-I deliverables include profiles, sample applications and testing tools. WS-I committees and working groups create profiles and supporting testing tools based on best practices for selected sets of web services standards. The profiles and testing tools are available for its use by the web services community to aid in developing and deploying interoperable web services.

Sample applications demonstrate web services applications that are compliant with WS-I guidelines. These implementations are developed using multiple platforms, languages and programming tools, demonstrating interoperability in action, and providing readily usable resources for the web services developer. Sample applications serve as working examples for developers looking to follow the WS-I guidelines in their programming environment of choice.

WS-I's SCM use case represents a perfect example to demonstrate the usefulness of the approach proposed in this thesis. The existence of multiple implementations and testing tools provide great means to validate the implementation resulting from the use of the methodology. Testing tools are used to monitor the messages and analyze the resulting log to identify any known interoperability issues, to determine whether the messages exchanged with a web service conform to WS-I guidelines. It is for this reason that this particular use case has been selected.

The sample application described in this section is not intended to exhibit all of the characteristics of a real world SCM design and implementation, but rather to demonstrate how WS-I Basic Profile (BP) (Web Services Interoperability Organization, 2006) conformant web services might be designed, implemented and deployed. One of the goals of the sample application is to explore as many of the features found in the BP as possible. To this end, the application employs a variety of schema

Figure 5.1: Supply Chain Management Use Case (WS-I, 2006)

naming conventions, SOAP message formats, SOAP message styles, and WSDL design practices that are all Basic Profile conformant. A complete description of the system architecture can be found in (WS-I, 2003b), as a set of use cases in (WS-I, 2003a), and in (WS-I, 2006) for security requirements. As the WS-I profiles deal only with interoperability issues, not implementation-related matters, no access-control requirements were originally specified for this use case. For this reason, some access control criteria has been specified, to achieve a full-blown implementation.

The WS-I states that, in order to inject some realism into the sample application, a single top down design was not attempted. Rather each main system (Retailer and Manufacturer) has been designed and architected separately and brought together via web services. This should reflect the real world of connecting autonomous organizations together without the luxury of a global architecture and design.

The SCM application (WS-I, 2003b; WS-I, 2003a) presents a typical business-to-consumer (B2C) model: a *Retailer* offering consumer electronic goods to consumers, as shown in Figure 5.1. To fulfill orders the *Retailer* has to manage stock levels in the three *Warehouses*. When an item in stock falls below a certain threshold, the *Warehouse* must restock the item from the relevant *Manufacturer*'s inventory (a typical business-to-business (B2B) model). In order to fulfill a *Retailer*'s request, a *Manufacturer* may have to execute a production run to build the finished goods (not presented as a WS). Each use case includes a logging call to a monitoring system in order to monitor the activities of the services from a single monitoring service. The primary goal of the application is to demonstrate all of the scenarios in the WS-I Basic Profile.

(WS-I, 2006) provides a common architecture and design document demonstrating the interoperability of the Basic Security Profile. The main objective of the security requirements is to demonstrate the wire-level interoperability of messages between applications, developed on platforms from multiple vendors that each conform to the Basic Security Profile version 1.1 (Web Services Interoperability Organization, 2010). As stated by (WS-I, 2006), these key architectural objectives focus on how Web services adhering to the WS-I Basic Security Profile Version 1.1 might be modeled, rather than demonstrating Web services security best practices. The same spirit has been used for the defined access control requirements. These are not supposed to represent a logical or 'best practice' access control strategy, but merely to provide a means to illustrate its integration through the methodology.

Although the complete use case has been implemented, this chapter mostly describes the "Warehouse" service, considered complete enough to illustrate the use of the methodology and tool. The sequence of steps that are going to be performed in this example is that of Figure 4.16, recalled here in Figure 5.2 to simplify visualization. In each section a similar image will be presented to the reader, to aid in following the use of the approach within the example.

Figure 5.2: Structure of the proposed solution.

5.2 The SCM Use Case, Functional Specification

The *Retailer* presents a web service for some third party system (Web Client Application in Figure 5.1) to access it. *Retailer*'s WS provides a façade onto the *Retailer System*, providing operations to access the catalog of products and to place orders. Within the *Retailer System*, three instances of the *Warehouse* WS exist, one for each of the warehouses (A, B and C) as defined in (WS-I, 2003a). These warehouses will, in turn, call out to the three *Manufacturing* services. To facilitate this interaction each warehouse has to provide a callback interface. The *Callback* and *Retailer* WS are the only external entry points into the *Retailer* system. Figure 5.1 presents the deployment view of the whole system (Web Client Application, *Retailer*, and *Manufacturer* systems).

The retailer must manage stock in three warehouses. If Warehouse A can not fulfill an order, the retailer checks Warehouse B; if Warehouse B also cannot, the retailer checks Warehouse C. When any warehouse's inventory of a particular product falls below a defined threshold, the warehouse orders more units from the correspondent manufacturer. There are three different manufacturers: manufacturers A, B and C, all providing different products. Any Warehouse may place an order with any manufacturer.

As already mentioned, the *Retailer System* will contain the following web services:

- Retailer.
- Warehouse.
- Warehouse Callback.

Retailer and Warehouse services belong to the same system, but communicate through WS request/response mechanism. Only the security requirements are a bit more lax than with external services. In addition, the *Retailer System* depends on two external services, defined elsewhere:

- Logging.
- Manufacturer.

5.2.1 Warehouse Service

This section describes the functional specification of the *Warehouse* service. The complete specification of this and the rest of the services in the Retailer System can be found in (WS-I, 2003b). The Warehouse service provides the following operations/message types, following the synchronous request/reply scenario:

Operation *ShipGoods* Description

A call to this service results in the warehouse checking each line item in the order. If for each line item the required quantity is in stock, the warehouse will ship that line item (and hence reduce the stock level). It will record which ones it has shipped and which ones it does not have enough stock for and hence cannot ship; the response will contain this list. A *ConfigurationFault* fault is returned if there is a problem in the configuration header. The reduction of the stock level below the minimum level will trigger the warehouse to call on the *Supply* service of the relevant manufacturer. Table 5.1 details this operation.

Scenario The scenario used will be Synchronous Request/Response using SOAP over HTTP.

Message Style The rpc/literal message style will be used.

5.2.2 Warehouse Callback Service

The Warehouse Callback service provides the following operations/message types:

Operation *SubmitSN* Description

A call to this service indicates that the manufacturer has finished processing an order. If the manufacturer's processing has been successful, the manufacturer will submit the shipping notice using the *SubmitSN* operation. If the shipping notice can be correlated to an order placed with a manufacturer, a positive acknowledgement is sent in the reply; otherwise a "Callbackfault" is returned.

Scenario The scenario used will be the reply/callback portion of the Basic Callback Scenario using SOAP over HTTP.

Message Style The doc/literal message style will be used.

Operation *ErrorPO* Description

A call to this service indicates that the manufacturer has finished processing an order but there has been an error in doing so. If the Manufacturer's processing of the order has not been successful, the Manufacturer will provide a reason using the ErrorPO operation. For any PO (Purchase Order), a Manufacturer may only invoke one of submitSN or ErrorPO.

Scenario The scenario used will be the reply/callback portion of the Basic Callback Scenario using SOAP over HTTP.

Message Style The doc/literal message style will be used.

Table 5.1: *Warehouse & WarehouseCallback Services*

Operation	Msg. Type	Message	Parameters	Type
Warehouse:ShipGoods	Input	ShipGoodsRequest	ItemList Customer Configuration Header	ItemList CustomerReferenceType Configuration
	Output	ShipGoodsResponse	Response	ItemShippingStatusList
	Fault	ConfigurationFault		Client
WhCbck:SubmitSN	Input	SNSubmit	shippingNotice ConfigurationHeader CallbackHeader	doc Configuration CallbackHeader
	Output	ackSN	Response	boolean
	Fault	ConfigurationFault		Client
	Fault	CallbackFault		Server
WhCbck>ErrorPO	Input	ProcessPOFault	ProcessPOFault	SubmitPOFault
	Output	AckPO	Response	boolean
	Fault	ConfigurationFault		Client
	Fault	CallbackFault		Server

Figure 5.3: Retailer System Services Interaction (WS-I, 2006)

Table 5.2: *Security Requirements (WS-I, 2006)*

Sender → Receiver	Operation/Message	Message Integrity	Authent.	Conf.	Algorithm
Retailer → Warehouse n	ShipGoods/ ShipGoodsRequest	R X.509: Body, ConfigHeader, Timestamp	Cert Auth	None	Key: RSA 1.5, Digest: SHA1
Warehouse n → Retailer	ShipGoods/ ShipGoodsResponse	Wn X.509: Body, Timestamp	Cert Auth	None	Key: RSA 1.5, Digest: SHA1
Warehouse n → Manufacturer	submitPO/ POSubmit	Wn X.509: Body, Configheader, Starthead,er, Timestamp	Cert Auth	Mn X.509: Body, Starthead,er, Signature.	Key: RSA 1.5, Data Encrypt.: AES256 Digest: SHA1
Manufacturer → Warehouse n	submitPO/ ackPO	Wn X.509: Body, Timestamp	Cert Auth	None	Key: RSA 1.5, Data Encrypt.: AES256 Digest: SHA1

5.3 Services Security Specification

Each system within the sample application possesses assets with unique security requirements. These assets include data, code, and configuration information. Only the assets exposed either as messages or as points of entry to the application, such as the Web client, were considered in this sample application. In (WS-I, 2006), each operation's security requirements are specified for message integrity, authentication and confidentiality. Figure 5.3 illustrates the sequence of web services calls in the sample application.

Table 5.2 provides a summary of the port level requirements for message integrity, authentication, and confidentiality used for the *Warehouse* service. The table is self-explanatory. A more complete description can be found in (WS-I, 2006).

5.4 Access Control Specification

The WS-I SCM use case knowingly leaves access control considerations out of the scope. Therefore, some access control requirements are presented here for illustration purposes. The requirements presented in this section do not pretend to be thorough nor appropriate. The only objective is to exemplify how such requirements could be modeled, and subsequently transformed to finally obtain the proper configuration artifacts.

Only access control rules on service interfaces are considered, sufficient to depict the different stages and steps which compose the development process. Equivalent access control rules can be expressed on entities (domain objects representations) or its attributes, for example. The stages such rules step through in the development process are identical to the ones presented here. Table 5.3 presents the defined rules.

Table 5.3: *Access Control Requirements*

Service	Operation	Required Role	Constraints
Retailer	getCatalog	WebClient	None
	submitOrder	WebClient	None
Warehouse	ShipGoods	Retailer	Working Days
Warehouse Callback	SubmitSN	Manufacturer	Working Days, Office Hours
	ErrorPO	Manufacturer	Working Days, Office Hours

5.5 Model Driven Development of WS-I's Supply Chain Management Use Case with Non-Functional Requirements

5.5.1 Business Process Model

A business process model represents the first input of the approach, as can be visualized in Figure 5.4. Figure 5.5 presents a BPMN diagram picturing an annotated (simplified conceptual version) of the SCM use case process. It shows the messages interchanged between the *Retailer*, *Warehouse*, *Manufacturer*, and *WarehouseCallback* services. The figure presents descriptive names for the activities. If the derivation of a functional PIM model is intended, the activities should be named

Figure 5.4: Annotated BPMN Model Input.

after the services/operations. Alternatively, a traceability model could link activities to PIM-level elements. The functional mapping process has been described in 4.5, and is not included here for the sake of brevity.

The notes in Figure 5.5 represent the non-functional intents placed on these interactions, following the common vocabulary defined in section 4.3.1. The focus will be placed on two intents in particular to illustrate the mapping procedure: *Warehouse:ShipGoods*'s authorization and non-repudiation intents, and *WarehouseCallback:SNSubmit*'s integrity and confidentiality intents.

First let's concentrate on the authorization intent. From there you can derive three elements; from the intent itself (*Authorization*), you can restrict to the *SecurityPolicy* class in the NRL Ontology (Kim et al., 2005a). Furthermore, the value of the "Attribute" property (in this case, *Role*), drives the search in the repository for a model categorized as "RBAC". The SecureUML model represents an exact match.

The third element to be derived is the value of the "RequiredValue" attribute from the *Authorization* intent, the role itself. The particular role can be matched to one present in the model, or introduced in the model instance (customizing the model to this particular system). Once the role has been matched, a weaving association (associating the specific *role* to the service interface) can be derived from the data-flow in the process model.

The *Non-Repudiation*, *Confidentiality*, and *Integrity* intents remain to be mapped. In this case, they (each one independently) drive a search for models in the repository that match the *MessageIntegrity*, *Confidentiality*, and *MessageAuthentication* classes in the ontology. The QoS model matches these *SecurityObjectives* from the ontology.

As in the case of the *Authorization* intent, the value of the attribute provides an input for the derivation of weaving associations. *QoSValues* for "Integrity" and "Confidentiality" may be associated to the *WarehouseCallback* service interface, according to the process model. Equivalently, a "Non-Repudiation" *QoSValue* can be associated to the *Warehouse* service.

Figure 5.5: Retailer's Collaboration Business Process Model

Integrity and confidentiality intents' *Level* attribute provides a mechanism to describe the objective in more granularity. Its value, "Mild" in the integrity case, delimits the possible values to be associated to the *Integrity Algorithm* and *SignatureEncryption Algorithm* QoSDimensions. The "Strong" value for the confidentiality level also reduces the possible matched values. Following the categorization of algorithms' strength in (Asnar et al., 2009, pp. 42-43, 45), the algorithms RSA 1.5, SHA1 and AES256 match the intents¹. As presented below, these algorithms later (in the PIM 1-to-PIM 2 mapping) map to WS-SecurityPolicy's *AlgorithmSuite* assertion *Basic256* value.

This example provides a proof that the technical requirements can be automatically derived from the high-level CIM model up to the degree of the algorithms. It may be argued that the CIM-to-PIM mapping is not the appropriate level on which to perform the derivation of algorithms. Ideally, it would be carried out in the PIM-to-PSM transformation, and also take advantage of the platform information (platform model) that is fed to it. In this dissertation, it was placed here to illustrate on the potential of the automatic derivation of technical details from high-level models.

5.5.2 PIM Input Models

Section 5.5.1 has already described how different elements could be derived, from the annotated process model, for the population of the QoS and SecureUML models. In this section, the use of these metamodels for the description of security and access control characteristics is presented, providing an insight for the understanding of the rest of the security artifacts' generation process.

Although a tentative structure of the services (components, interfaces, etc.) can be derived from the CIM model (represented as "T1" transformation in Figure 5.6), the internal implementation of the system still has to be described (e.g., different services which are implemented by the same component, etc.). A UML model of the Warehouse service, annotated with the SoaML profile (the selected input for this stage) is also presented in this section. Figure 5.6 shows where exactly in the sequence of steps of the development process these models are positioned.

Figure 5.6: SoaML, QoS, and SecureUML Inputs.

¹Those algorithms were the ones required in (WS-I, 2006).

In an already established development environment, the NF concerns models would most likely be available in repositories. Different expert-made NF models, suitable for different concerns in different domains would represent a pool of options from which to select the most appropriate one. Such models could be used “as is”, or modified to better address the needs of the particular development in hands. This section describes the creation of these models. In this case, the models have a certain specificity for the SCM use case which is not the ideal approach. Generic models, supporting greater reuse, would provide a better alternative for long-term (i.e., used more than once) frameworks. Elements derived, matched, and/or created by the modeler would complement these models, customizing them for the particular system.

Retailer System’s UML Model

Functional models of the different systems will most likely be developed from scratch (or based in a previous system model) for each new project. The ‘de facto’ standard for functional systems’ modeling is UML, so it was decided to adopt it to model the Retailer system. As described in section 4.4.1, the SoaML profile is used to guide its posterior transformation to IMM. Being the functional development not the focus of this dissertation, it is not described here in much detail. Figure A.1 in the appendix presents a partial view of the annotated UML model for the Warehouse and WarehouseCallback services. Table 5.4 presents the different annotated model elements in the implementation. UML class diagrams describing the internal implementation of the rest of the services are available in (WS-I, 2003b).

Modeling of the Security Requirements with the QoS Metamodel

The first step towards modeling the security requirements under the QoS metamodel is defining a *QoS Catalog* for the target (one or more) *QoS Category*. A catalog defines the different characteristics (may be regarded as metaclasses in a metamodel) and dimensions (the variables defining a particular characteristic) that may be present in a model of such category. In this particular case, only one category will be modeled: Security. Within this category, three characteristics are defined:

- Integrity
- Confidentiality
- Authentication

Table 5.4: Warehouse Service’s SoaML Model Elements

UML Model Element	SoaML Stereotype
WarehouseShipmentsPortType	ServiceInterface
WarehouseCallbackPortType	ServiceInterface
WarehouseService	ServiceProvider
WarehouseCallbackService	ServiceSpecification
WarehouseCbck	ServiceProvider
ProcessPOFault	MessageType
AckPO	MessageType
CallbackFault	MessageType
SubmitSN	MessageType
ShipGoodsRequest	MessageType
ShipGoodsResponse	MessageType
ConfigurationFault	MessageType
ackSN	MessageType

Table 5.5: *QoS Catalog for the QoS Category “Security”*

Charac.	Dimension	Unit Type	ShipGoodsRequest
Integ.	Integrity Algorithm	Enumeration Literal	SHA1
	TimestampRequired	Boolean	true
	EncryptSignature	Boolean	false
	SignatureEncryption Algorithm	CryptoAlgorithm	Not Applicable (NA)
Conf.	CryptoAlgorithm	String	NA
Auth.	TokenType	Enumeration Literal	X509
	TokenKey Algorithm	Enumeration Literal	RSA15
	SignToken	Boolean	true
	EncryptToken	Boolean	false

Table 5.5 presents the different *QoSDimensions* defined for these three characteristics, and the values assigned to them in the case of the “ShipGoodsRequest” message, which can be matched to the requirements presented in Table 5.2.

Notice that QoS Dimensions can be defined in reference to other dimensions. This is the case of *SignatureEncryptionAlgorithm*, which makes reference to a *CryptoAlgorithm* dimension to define its value.

Representing the Access Control Requirements Under SecureUML

Figure 5.7 presents the implementation under SecureUML of the requirements presented in Table 5.3. The different roles, resources, permissions, actions, and constraints are declared in the model. The particular users (principals), however, are not included in the model, and are not associated to the roles. Although users could have been included, the general idea is to keep the different concern models generic and reusable (this case, however, is not a fair example, as the access control model is very supply chain management specific, as was developed for example). Nevertheless, the reader should keep in mind that the idea is to have model repositories containing generic concern-models, which could be browsed, selected by suitability, and possibly customized for the particular application. Users could be later included either directly in the implementation, or as property files inputs to the transformations.

Particular attention is paid to the *WorkingDays* constraint, detailed in the lower part of Figure 5.7, under the properties view. A partial view of the constraint expression can be visualized in the figure. Constraints are boolean expressions which evaluation grants/denies a particular principal access to a determined action on a resource, all this according to the description of an specific permission. The constraint expression must be translated to the target implementation language in the final generation of the access control descriptors.

In this occasion, the target language (Java) was selected to describe the constraint expression, so no translation would be necessary. OCL would perhaps represent a more appropriate alternative for a platform-independent constraint language in a production environment. OCL parsers for Java (an other languages) are readily available. In the case of this example, it was preferred to keep it simple, and avoid the translator. Therefore, the constraint expression is directly incorporated in the target access decision voters, described below in section 5.5.5.

Figure 5.7: SecureUML Model

5.5.3 Matching Non-Functional Models

The idea behind the matching of the non-functional models is to have metadata associated to such models in the repository. In this case, the different models would have the different security objectives they address associated to them, as shown in Table 5.6.

The security objectives used for the metadata description are the ones defined in (Kim et al., 2007). As already mentioned, the different intents are mapped to these security objectives as described in Table 4.2 in chapter 4. Based on these mappings, the SecureUML and the QoS models in the repository would match access-control and the remaining requirements respectively.

The particular elements within the model could also be semantically annotated in order to illustrate which objective they are realizing, and by means of which mechanism (the ontology provides classes for credentials and security algorithms, for instance, fulfilling the different security objectives). This inner annotation of the model elements is skipped, for the sake of brevity.

The automatic matching of QoS and SecureUML models, annotated with metadata as described here, belong to the sequence of steps in the development process as shown in Figure 5.6. In the figure, QoS models are represented by the “Generic Security Model” box, while SecureUML models are identified by the “Generic Access Control Model” box.

Table 5.6: Annotated Non-Functional Models

Characteristic	Metamodel	Annotations
Security	QoS Metamodel	Confidentiality MessageIntegrity MessageAuthentication UserAuthentication ReplayPrevention
Access Control	SecureUML	Authorization

Figure 5.8: PIM Inputs to iMM (PIM 2) Stage.

5.5.4 Transformation Into the iMM Model

The transformation into the iMM model is divided in two steps. First, weaving models are created/derived, linking functional and non-functional elements to be composed. Then, and following the mappings already defined in section 4.5, the different functional and non-functional input models are composed.

This second step is the actual transformation of the models. The different input models are complemented with the composition models to drive this transformation. The implemented prototype splits this transformation into multiple sequential transformations, first transforming the input functional model, and then each subsequent one incorporating a different non functional concern into the iMM model.

Figure 5.8 depicts this second step in the development process. It is represented by a transformation from SoaML to iMM (marked “T2” in Figure 5.8), and the two compositions: SecureUML with functional iMM (marked “C1”) and QoS with functional iMM (marked “C2”). The weaving models are omitted in the figure.

Composition of Retailer System’s UML Model with Non-Functional Models

In this particular example, having two different non-functional aspects of the system under consideration, two composition models are necessary. Included in appendix A, Figure A.2 and Figure A.3, present the three-panel view provided by the AMW tool to create such composition models.

The first model relates the model elements of the *Retailer* system with the security characteristics modeled in the QoS model. In Table 5.7, the different model elements are associated to its security requirements following the specification described in section 5.3. The functional elements are associated to one *QoSValue* per *QosCharacteristic* (i.e., Integrity, Confidentiality, and Authentication) as appropriate. In turn, each of these *QoSValues* is composed of child *QoSValues* which are the ones in charge of specifying a value to each one of the attributes that uniquely define the particular characteristic. Figure A.2 in the appendix brings out, for instance, the association of

Table 5.7: Warehouse Service and QoS Models Composition

Model Element	QoS Characteristic	QoS Value
WarehouseService	Integrity	Integrity = SHA1
		TimestampRequired = True
		EncryptSignature = False
	Confidentiality	None
	Authentication	TokenType = X509
		TokenKey Algorithm = RSA15
		SignToken = True
		EncryptToken = False

ShipGoodsResponse with the security characteristics of signing, with the corresponding Warehouse X.509 certificate, the whole message and the timestamp element.

The second composition model relates the access control requirements with the different functional elements, according to the requirements in Table 5.3. In this case, Figure A.3 brings out the execution permission of the *shipGoods* operation for the *Retailer* role (through the *ExecuteWarehouse* Action), and its associated constraints. At this time, the modeler may introduce, if desired, the different system users in the SecureUML, and map them to the present roles. In this way, the users may be included in the later derivation of implementation artifacts. A view of the resulting iMM model after all transformations is included in Figure A.5 in appendix A.

Transformation to iMM

According to the mappings defined in section 4.5.4, transformations from UML+SoaML to iMM, QoS + Weaving model to iMM, and SecureUML + Weaving model to iMM were defined.

Retailer's SoaML to iMM Transformation SoaML to iMM represents the first step in the transformation chain. The subsequent transformations (composition transformations) use the resulting iMM model and the weaving models to incorporate the different non-functional elements. The traceability between the UML and the iMM model elements is implicit in this subsequent transformations thanks to the knowledge of the mappings between UML and iMM. Other forms of traceability (Jouault, 2005; Yie and Wagelaar, 2009) could also be used.

QoS to iMM Transformation The different QoS characteristics are transformed into high level policies in the iMM model, and associated to the corresponding functional elements. As will be described in 5.5.5, once the target platform has been determined, these policies are subject of a new refinement transformation to map them into a platform specific form (in this case WS-SecurityPolicy) prior to the generation of target artifacts.

SecureUML to iMM Transformation The SecureUML input model is transformed into the generic access control part of the iMM model by following the directives defined in section 4.5. The corresponding composition model drives the creation of associations between these access control rules and the functional elements.

5.5.5 Generation of Resulting Artifacts

The Java programming language has been selected as the target platform for the generation of the resulting Retailer system. To be more precise, two frameworks have been selected to support the

Figure 5.9: Outputs of the Development Process.

execution of the system:

- Spring framework (SpringSource, 2010a) and Spring WS (SpringSource, 2010b), a framework leveraging enterprise-level functionalities for Java POJOs (Plain Old Java Objects).
- Apache CXF (Apache, 2010) framework, additionally supporting RPC style Web Services (as demanded by the SCM use case specification).

The use of frameworks is of great importance in model-driven development. Frameworks provide much functionality which would be, otherwise, necessary to generate for the execution of the different systems. This greatly simplifies the development of generators.

Declarative-configuration frameworks in particular (making use of schema-based XML documents to configure characteristics of the system), support the configuration of non-functional characteristics of the services (such as security or access control, in this case) independently of the functional code. This is of great usefulness, allowing for an independent generation of the different artifacts, and the reuse of functional Java generators for different target frameworks.

The current stage in the development process is presented in Figure 5.9.

Functional Code Generation

The implemented prototype generates code stubs for the Java classes to be employed in the resulting system. Acceleo (OBEO, 2006), a model-to-text open source tool is used to generate these classes. Not being the focus of this dissertation, the description of the generation of the functional code is excluded. The reader may refer to (OBEO, 2010), which was used as a base for the functional code generation for this target platform. *SecurityConnector*, a Java class generated for adapting Spring and CXF security engines, is included in Listing C.1.

WSDL, WS-Policy, and XSD Generation

The first step towards the WSDL descriptors (and the rest of these artifacts) is to split the iMM model into separate models of each individual service. This is performed, in the case of the prototype, by means of Eclipse's EMF API.

Each service model contains all elements related to the corresponding service which are necessary for the generation of the different documents:

- services, ports, interfaces, operations and parameters for WSDL descriptors,
- associated policies, alternatives, and assertions for WS-Policy documents,
- and the different used types to be included in XSD schemas.

Figure 5.10 presents the resulting WSDL document for the *Warehouse* service. The reader may notice the reference to the corresponding policies within the description of the service elements.

As already mentioned, the high level policies originally derived from the QoS model need to be further refined into valid equivalent policies in the target domain. In this particular case, the policies need to represent valid WS-Policy (more precisely, WS-SecurityPolicy) assertions. Therefore, a new transformation is defined for these policies which takes into account the QoS Catalog definitions². Figure A.4 in appendix A presents an example of the refinement of such policies.

Afterwards, the now WS-Policy policies are separated in individual models (corresponding to the WS-Policy MM) per policy document, in a similar way to that of the different service models.

The generation of both WSDL and WS-Policy documents is not directly performed from the WSDL and WS-Policy models by means of a M2T tool as in the case of the code stubs. In contrast, these models are first transformed into a XML model (according to a metamodel available in the AM3 tool suite) for which an extractor is available. This represents an alternative mechanism on how the generation of artifacts can take place in model-driven development, in which a model is sequentially transformed into different models, the final model being syntactically expressed (concrete syntax) in the desired target format. The resulting WS-Policy document containing the policies applied to the Warehouse service is presented in Figure 5.11 and included in the appendix as Listing C.5.

Finally, XSD schemas are also generated for the system, making use of the functionalities provided by Eclipse's EMF and XSD APIs. The generation of XSD schemas from Ecore models is anything but new, and therefore is not included here.

---

²The different catalogs that may drive such transformations may be provided as libraries, supporting the use of reusable transformations.

```

<?xml version = '1.0' encoding = 'UTF-8' ?>
<wsdl:definitions name = 'WarehouseServicePortImp_System'
    targetNamespace = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse.wsdl'
    xmlns:ximp1 = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Configuration.xsd' ...>

  <wsdl:types>
    <xsd:schema targetNamespace = 'http://iBuilder.generation.by.upm/types' xmlns:xsd1 = 'http://warehouseserviceportimp_system/types'>
      <xsd:import schemaLocation = './Configuration.xsd'
        namespace = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Configuration.xsd' />
      ...
      <xsd:element name = 'ConfigurationHeader' type = 'ximp1:ConfigurationType' />
      <xsd:element name = 'ConfigurationFault' type = 'ximp1:ConfigurationFaultType' />
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name = 'ShipGoodsRequest'>
    <wsdl:part name = 'ConfigurationHeader' element = 'types:ConfigurationHeader' />
    <wsdl:part name = 'Customer' type = 'ximp3:CustomerReferenceType' />
    <wsdl:part name = 'ItemList' type = 'ximp3:ItemList' />
  </wsdl:message>

  ...

  <wsdl:message name = 'ConfigurationFault'>
    <wsdl:part name = 'ConfigurationFault' element = 'types:ConfigurationFault' />
  </wsdl:message>

  <wsdl:portType name = 'WarehouseServiceInterface_PortType'>
    <wsp:PolicyReference URI = 'WS_Context.xml#warehouseService_Binding_Binding_Policy' wsdl:required = 'true' />
    <wsdl:operation name = 'ShipGoods'>
      <wsdl:input message = 'tns:ShipGoodsRequest' name = 'ShipGoodsRequest' />
      <wsdl:output message = 'tns:ShipGoodsResponse' name = 'ShipGoodsResponse' />
      <wsdl:fault name = 'ConfigurationFault' message = 'tns:ConfigurationFault' />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name = 'WarehouseServiceInterface_Binding' type = 'tns:WarehouseServiceInterface_PortType'>
    <soap:binding style = 'rpc' transport = 'http://schemas.xmlsoap.org/soap/http' />
    <wsdl:operation name = 'ShipGoods'>
      <soap:operation soapAction = '' style = 'rpc' />
      <wsdl:input>
        <soap:body use = 'literal' parts = 'Customer ItemList'
          namespace = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse.wsdl' />
        <soap:header message = 'tns:ShipGoodsRequest' part = 'ConfigurationHeader' use = 'literal' wsdl:required = 'true'>
          <soap:headerfault message = 'tns:ConfigurationFault' part = 'ConfigurationFault' use = 'literal' />
        </soap:header>
      </wsdl:input>
      <wsdl:output>
        <soap:body use = 'literal' namespace = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse.wsdl' />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name = 'warehouseServiceService'>
    <wsdl:port name = 'warehouseService' binding = 'tns:WarehouseServiceInterface_Binding'>
      <soap:address location = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse.wsdl/warehouseService' />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

Figure 5.10: Resulting Warehouse Service's WSDL document

```

<?xml version = '1.0' encoding = 'UTF-8' ?>
<wsp:Policy xmlns:sopa = 'http://types.sopware.org/qos/SOPAssertions/1.1'
            xmlns:sopwsp = 'http://types.sopware.org/qos/WS-PolicyExtensions/1.1'
            xmlns:wsu = 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'
            xmlns:sp = 'http://schemas.xmlsoap.org/ws/2002/12/secext'
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:wsp = 'http://www.w3.org/ns/ws-policy'>

  <wsp:Policy name = 'warehouseService_Binding_Binding_Policy' wsu:Id = 'warehouseService_Binding_Binding_Policy'>
    <wsp:All>
      <sp:AsymmetricBinding wsp:Ignorable = 'false' wsp:Optional = 'false'>
        <sp:IncludeTimestamp wsp:Ignorable = 'false' wsp:Optional = 'false'>
        <sp:InitiatorToken wsp:Ignorable = 'false' wsp:Optional = 'false'>
          <wsp:Policy name = 'InitiatorTokenNestedPolicy' wsu:Id = 'InitiatorTokenNestedPolicy'>
            <wsp:All>
              <sp:X509Token wsp:Ignorable = 'false' wsp:Optional = 'false'
                sp:IncludeToken = 'http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient'>
                <wsp:Policy name = 'InitiatorTokenX509TokenNestedPolicy' wsu:Id = 'InitiatorTokenX509TokenNestedPolicy'>
                  <wsp:All>
                    <sp:WssX509V3Token10 wsp:Ignorable = 'false' wsp:Optional = 'false'>
                    </wsp:All>
                  </wsp:Policy>
                </sp:X509Token>
              </wsp:All>
            </wsp:Policy>
          </sp:InitiatorToken>
          <sp:RecipientToken wsp:Ignorable = 'false' wsp:Optional = 'false'>
            <wsp:Policy name = 'RecipientTokenNestedPolicy' wsu:Id = 'RecipientTokenNestedPolicy'>
              <wsp:All>
                <sp:X509Token wsp:Ignorable = 'false' wsp:Optional = 'false'
                  sp:IncludeToken = 'http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToInitiator'>
                  <wsp:Policy name = 'RecipientTokenX509TokenNestedPolicy' wsu:Id = 'RecipientTokenX509TokenNestedPolicy'>
                    <wsp:All>
                      <sp:WssX509V3Token10 wsp:Ignorable = 'false' wsp:Optional = 'false'>
                      </wsp:All>
                    </wsp:Policy>
                  </sp:X509Token>
                </wsp:All>
              </wsp:Policy>
            </sp:RecipientToken>
            <sp:AlgorithmSuite wsp:Ignorable = 'false' wsp:Optional = 'false'>
              <wsp:Policy name = 'AlgorithmSuiteNestedPolicy' wsu:Id = 'AlgorithmSuiteNestedPolicy'>
                <wsp:All>
                  <sp:Basic128Rsa15 wsp:Ignorable = 'false' wsp:Optional = 'false'>
                  </wsp:All>
                </wsp:Policy>
              </sp:AlgorithmSuite>
            </sp:AsymmetricBinding>
            <sp:SignedParts wsp:Ignorable = 'false' wsp:Optional = 'false'>
              <sp:Header wsp:Ignorable = 'false' wsp:Optional = 'false' Name = 'ConfigurationHeader'
                Namespace = 'http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse'>
              <sp:Body wsp:Ignorable = 'false' wsp:Optional = 'false'>
              </sp:SignedParts>
            </wsp:All>
          </wsp:Policy>

```

Figure 5.11: Warehouse Service's WS-Policy document

Non-Functional-Configuration Generation

WS-Security Configuration Although the possibility existed to use the generated WS-Policy documents to configure the framework (CXF support this feature), it was decided to generate the frameworks' proprietary configuration files to make the example more illustrative. Listing B.2 presents a ACCELEO template for the generation of Spring configuration files.

Figure 5.12 shows an excerpt of the application configuration file generated for the Warehouse service. In it, the "WarehouseA" service is defined as a `jaxws:endpoint` and configured correspondingly (lines 9-20). The rest of the warehouses' (B and C) configuration is equivalent, and not included for the sake of brevity. The actual bean implementing the service ("warehouseAService") is passed as a reference to the endpoint (line 11).

The implementor bean is itself configured with the different properties it needs (lines 23-27). In the current state of the prototype, this is the only piece of the configuration code that is not automatically generated, as it depends on the actual business code implementation of the service, which is not known at design time. The use of the Inversion of Control (IoC) pattern, available on modern frameworks, is a possible solution to this issue.

Security interceptors are also defined for the service endpoint. Such interceptors can be defined in a separate configuration file, simplifying the modularization of the generators, but are in this case defined in the same file to facilitate visualization. Only the output interceptor (*WarehouseEndTimestampSign_Response*) is presented in lines 33-48 of Figure 5.12. The input interceptor's configuration is, again, very similar, and therefore excluded.

Encircled in this configuration, two entries stand out:

- The different securing actions to be performed (key="action" and, according to the requirements, value="Timestamp Signature", line 38).
- The actual elements to be signed (key="signatureParts" and, again following the requirements, value="..Timestamp; ..Body; ..Header", lines 43 and 44).

Security interceptors configuration represent, therefore, the actual piece of the configuration file that depends of the security policy part of the model. It can be completely generated from the model, with no need for human intervention. The complete configuration files for the *Warehouse* service are included in appendix C.

Access Control Configuration The access control configuration has been defined in a separate file, presented in Figure 5.13. All the information necessary to generate the access control configuration file is available in the IMM model, supporting a complete generation without the need for human interaction.

In this case, being a proof-of-concept implementation, an "in-memory" user details service ³ (id="userDetailsServ", lines 38-44 in the figure) has been defined. Such user-details service relates the different user names (principals) with the corresponding passwords, and the authorities (roles) that are to be granted to this user. This information, as already described, is readily available in the access control part of the IMM model.

For a production-quality implementation, data base, Java Authentication and Authorization Service (JAAS), or Lightweight Directory Access Protocol (LDAP) alternatives are available. For the aforementioned services to be correctly used within the system, extra information may be needed. Properties files/libraries containing such extra information may be used to drive the generation of the access control configuration. The configuration of such services is, however, not the focus of this dissertation, and are consequently not included.

³The names and passwords are defined in the configuration file itself. It is a practical mechanism for prototyping.

A *PreAuthenticatedAuthenticationProvider* (id="preAuthProvider", highlighted in lines 29-35 in Figure 5.13) is also defined in this configuration file. This bean supports the use of WS-Security-based authentication (the user of the service is determined from the digital signature included in the message). To integrate the web services' security interceptors, described in the previous section, with the access control infrastructure, a "SecurityConnector" had to be included in the generation.

SecurityConnector's purpose is to create an "AuthenticationToken" (containing the "principal", the user name basically), and pass it to the authentication manager defined in the access control configuration (the X509 certificate was previously validated). It basically connects CXF's JAX-WS authentication with Spring's access control infrastructure. The authentication manager, leveraging the user-details service, is then in charge of granting the principal the appropriate roles. The "SecurityConnector" class is available in Listing C.1 in appendix C.

The previous detailed elements take part in assigning the appropriate roles to the principal, once authenticated. The *global-method-security* element (lines 4-10), on the other hand, is the one in charge of indicating the assigned roles the principal is required to have to be able to execute a particular method. Basically, what the "Permission" class indicates in SecureUML. There are multiple alternative ways in which to perform such indication, including, for instance, the use of Java annotations. In this case it was preferred to make use of the AspectJ (Kiczales et al., 2001) pointcut language, as indicated by the "protect-pointcut" element in lines 5-9.

Figure 5.13 presents a pointcut expression requiring the retailer (access="ROLE_RETAILER") role for the execution of the "shipGoods(..)" method of any warehouse service's implementation class (WarehouseAService, WarehouseBService, or WarehouseCService). The expression and required role are marked in the figure (lines 8 and 9).

Finally, the *AccessDecisionManager* element provides a means for the implementation of constraints filtering the access to the different methods. It grants or denies access to a secured resource based on different voters, in this particular case requiring an unanimous affirmative decision on the voters for granting access, as highlighted in line 18 of Figure 5.13.

For the generation of the retailer system, it was decided that constraints should be represented as voters. These constraints were initially defined in the SecureUML model, and transformed into the iMM model. The necessary information is, therefore, once again present in the iMM model. *WeekDaysConstraintVoter*'s (marked in line 22 of the figure) implementation is included in Listing C.2.

5.6 Summary of the Chapter

This chapter has presented the well-known WS-I's Supply Chain Management use case. Focusing on its "Warehouse" service, all functional, security, and access control specifications have been presented. The different steps carried out for this use case's implementation (following the development process presented in chapter 3) are described in this chapter. The generated implementation results have been validated using the correspondent tools provided by the WS-I, reassuring the trustworthiness of the approach and prototype.

This example, although limited, provides a proof of the feasibility and usefulness of the proposed approach. Regretfully, the possibilities the approach provides for the independent evolution of the different concerns is not simple to transmit in a limited number of pages.

This chapter then closes a circle, completing the objectives placed in chapter 1 with regard to the modeling, measurement and analysis of non-functional characteristics of software systems. The remaining chapters of this document provide a context to this dissertation by discussing this and similar work. First, chapter 6 presents most representative related work in the different areas this work spans on. Finally, chapter 7 provides the summary and discussion of the work being presented in this dissertation.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:jaxws="http://cxf.apache.org/jaxws"
4       ...>
5
6     <!-- SERVICES -->
7     <!-- WarehouseA -->
8     <jaxws:endpoint id="warehouseA" address="/warehouseA" implementorClass="org.ws_i.sampleapplications....WarehouseAService">
9       <jaxws:implementor>
10         <ref bean="warehouseAService"/>
11       </jaxws:implementor>
12       <jaxws:outInterceptors>
13         <ref bean="WarehouseEndTimestampSign_Response"/>
14       </jaxws:outInterceptors>
15       <jaxws:inInterceptors>
16         ...
17       </jaxws:inInterceptors>
18     </jaxws:endpoint>
19     ...
20     <bean id="warehouseAService" class="org.ws_i.sampleapplications....WarehouseAService">
21       <property name="logStub" ref="loggingClient"/>
22       <property name="manufacturerAService" ref="manufacturerAClient"/>
23     </bean>
24     ...
25     <!--
26     WSS4JOutInterceptor for timestamping and signing the SOAP response.
27     -->
28     <bean
29       id="WarehouseEndTimestampSign_Response"
30       class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor">
31       <constructor-arg>
32         <map>
33           <entry key="action" value="Timestamp Signature"/>
34           <entry key="user" value="warehouse"/>
35           <entry key="signaturePropFile" value="warehouseKeystore.properties"/>
36           <entry key="passwordCallbackClass" value="org.ws_i.sampleapplications....WarehousePasswordCallback"/>
37           <entry key="signatureParts" value="{Element}{http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd}Timestamp:
38             {Element}{http://schemas.xmlsoap.org/soap/envelope/}Body"/>
39         </map>
40       </constructor-arg>
41     </bean>
42     ...
43 </beans>

```

Figure 5.12: Warehouse Service's Security Configuration

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans:beans xmlns="http://www.springframework.org/schema/security" ...>
3   ...
4   <global-method-security access-decision-manager-ref="myAccessDecisionManagerBean">
5     <protect-pointcut
6       expression="execution(* org.ws_i.sampleapplications.supplychainmanagement._2002_08.
7                                     warehouse.Warehouse*Service.shipGoods(..))"
8     />
9     access="ROLE RETAILER" />
10  </global-method-security>
11  <!-- Define custom Voter -->
12  <beans:bean id="WeekDaysConstraintVoter"
13    class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.util.
14                                     WeekDaysConstraintVoter"/>
15
16  <!-- Define AccessDecisionManager as UnanimousBased -->
17  <beans:bean id="myAccessDecisionManagerBean"
18    class="org.springframework.security.vote.UnanimousBased">
19    <beans:property name="decisionVoters">
20      <beans:list>
21        <beans:ref bean="WeekDaysConstraintVoter" />
22        <beans:bean class="org.springframework.security.vote.RoleVoter" />
23        <beans:bean class="org.springframework.security.vote.AuthenticatedVoter" />
24      </beans:list>
25    </beans:property>
26  </beans:bean>
27
28  <beans:bean id="preAuthProvider"
29    class="org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvider">
30    <beans:property name="preAuthenticatedUserDetailsService" ref="userDetailsWrapper"/>
31    <beans:property name="order" value="1"/>
32    <custom-authentication-provider/>
33  </beans:bean>
34
35  ...
36
37  <user-service id="userDetailsServ">
38    <user authorities="ROLE_WEBCLIENT" name="webclient" password="webclientpass"/>
39    <user authorities="ROLE_RETAILER" name="retailer" password="retailerpass"/>
40    <user authorities="ROLE_WAREHOUSE" name="warehouse" password="warehousepass"/>
41    <user authorities="ROLE_MANUFACTURER" name="manufacturer" password="manufacturerpass"/>
42  </user-service>
43
44  </beans:beans>
45

```

Figure 5.13: Warehouse Service's Authorization Configuration

Chapter 6

Related Work

The work presented in this thesis focuses on the model driven development of non-functional characteristics of component-based and service-oriented systems. The approach, as presented in the previous chapters, proposes the use of domain-specific modeling languages to independently describe different non-functional concerns and then generate, by a combination of compositions and transformations, a complete model of the system on which metrics' calculations and analyses can be performed. The proposed approach follows the abstract-to-concrete methodology proposed by MDA, and UML profiles represent the selected mechanisms to incorporate functional information. An example, generating executable artifacts and standard service descriptors, is also provided.

The thesis spans, therefore, over multiple areas of research (i.e. the modeling of non-functional properties, measuring and analysis of models, modeling of service-oriented systems, etc.). Accordingly, this chapter presents the most representative related work in the different subjects that fall within the scope of this dissertation.

This chapter is structured correspondingly. Section 6.1 introduces some interesting pieces of research on the subject of the modeling of non-functional properties, further distinguishing between UML-based and other type of approaches. Some related model-driven methodologies and development processes for service-oriented systems are presented in section 6.2. Work on the subject of analyses, metrics, and model measuring is reported in section 6.3. Finally, noteworthy results in the specific area of model-driven service-oriented systems is described in 6.4.

6.1 Modeling Non-Functional Properties

This sections presents representative related work on the subject of the modeling of non-functional properties. What represents a non-functional aspect of software systems is sometimes arguable. The International Organization for Standardization (ISO) presented an standard (ISO/IEC, 2001), and very recently a revision (ISO/IEC, 2011) on software quality attributes. Such standards provide a categorization (a quality model) that can be mapped to non-functional characteristics and sub-characteristics of software. This provides a good base of what can be considered non-functional characteristics. Noteworthy, in the particular case of security, the revision has promoted it to characteristic, rather than a sub-characteristic of functionality (original classification), and includes confidentiality, integrity, non-repudiation, accountability and authenticity as sub-characteristics.

On the specific area of software security, ISO provides, in (ISO, 1989), a description of the different services and associated mechanisms, organized per-level of the ISA security architecture reference model. There, the security services include identification and authentication, auditing, authorization (logical access control), confidentiality, integrity, and non-denial/non-repudiation.

Different research pieces addressing the non-functional characteristics described in these quality models are presented below.

6.1.1 UML Non-Functional-Aware Models

Approaches based on UML Profiles

The OMG has standardized a number of UML profiles. Among them, closely related to the subject of extra-functional characteristics are “The UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms” (QoS) (OMG, 2008d) and the “Modeling and Analysis of Real-Time and Embedded Systems” (MARTE) (OMG, 2007). These two profiles, and the superseded “UML Profile for Schedulability, Performance and Time” (SPT) (OMG, 2005b), appear numerous times, as will be presented in this chapter, in research using UML to model non-functional properties.

UMLSec (Jürjens, 2002; Jürjens, 2005) is an extension to UML for modeling security characteristics such as confidentiality and access control. In UMLSec, security requirements are specified using multiple stereotyped UML diagrams, to generate Abstract State Machines models and later evaluate them against associated constraints. The focus on UMLSec is to model the particular security mechanisms that are applied to a system, and hence address security at a very low, technical level, making it difficult to understand the reason for the introduction of such mechanisms (Asnar et al., 2006). In this dissertation, non-functional requirements/intents (security being the selected example) are first modeled at a high abstraction level, to be them mapped into less abstract fulfilling models (with varied degree of platform-dependence) until the final resulting artifacts are generated. Consequently, different description mechanisms are used at different abstraction levels. Moreover, domain specific alternatives have been preferred over general purpose ones, due to their closeness to the problem domain.

The ALARCOS group has participated in several lines of work (Blanco et al., 2009a; Rodríguez et al., 2010; Delgado et al., 2011) which are related in different aspects to the one presented in this thesis. Focused on data warehouses, (Blanco et al., 2009a; Blanco et al., 2009b) define a UML profile to include security requirements at business level. Greatly resembling the conception of the IMM metamodel, this line of ALARCOS’s work makes use of a *multidimensional* MM at PIM level, composed of three different metamodels: a security configuration, an RBAC, and a cube MM.

Also by ALARCOS, (Rodríguez et al., 2010) presents a model-driven approach for the consideration of security in BPMN models, which are latter transformed into UML activity diagrams, and from there into class and use case diagrams. This piece of research shares with the one presented in this dissertation the definition of an abstract vocabulary for security intents at the CIM level, and the use of a abstract-to-concrete chain of transformations. Differently from the approach in this thesis, instead of using text notes to express security requirements, they define a BPMN metamodel extension (Rodríguez et al., 2007). Although addressing a different target domain, and using a general purpose language such as UML, these last two lines of work share many commonalities with the approach described in this dissertation. These approaches focus on the development of security-aware systems from scratch, while the approach presented in this thesis, although capable of this, leans towards adding the possibility of reusing previously existent non-functional (not limited to security) architectural designs.

The approach used by *Majzik et al* is very similar to the one presented in this thesis. Their work in (Majzik et al., 2003) is devoted to dependability modeling and a model based evaluation approach based on UML models. It addresses the early phases of system design to capture dependability attributes like reliability and availability, thus providing guidelines for selecting among different architectural and design solutions.

Access control is a non-functional characteristic that is normally addressed in literature indepen-

dently from other NFPs. (Reznik et al., 2007) proposes the model-driven generation of access control policies. For this purpose, they define a UML profile (eUMLSec) and extend the CORBA Component Model standard to incorporate security concepts. They present the application of Role-Based Access Control (RBAC) techniques, although they claim the approach also supports Mandatory Access Control. The target platform is limited to SecureMiddleware (a CCM implementation).

Aspect-Oriented Approaches

The model-driven development of non-functional requirements in embedded real-time systems is addressed in (Wehrmeister et al., 2007) and (Freitas et al., 2007). This work presents DERAf (Distributed Embedded Real-Time Aspects Framework), an AO framework to handle NFRs at earlier design stages. DERAf, although extensible, is currently limited to timing, precision, performance, distribution, and embedded behavior aspects. Initially, it adapts the FRIDA (From Requirements to Design using Aspects) (de Castro Bertagnolli and Lisbôa, 2003) methodology, naming it RT-FRIDA, and at design stage makes use of UML extended with the standard SPT profile. This approach, as most aspect-oriented approaches, makes use of the same general purpose modeling language (generally UML) to represent all functional and non-functional concerns. Additionally, it only addresses requirements and design stages (no mapping towards implementation is provided), and is currently limited to performance and timing characteristics.

(Pavlich-mariscal et al., 2007) takes an aspect-oriented approach to augment UML with security diagrams to support role-based, discretionary, and mandatory access control techniques. The approach decomposes MAC, DAC, and RBAC into *security features*. Such features are then composed into an augmented UML metamodel, from which code and configuration artifacts can be generated. The work seems promising. However, some aspects of the framework are still to be addressed. For instance, no validation of the combination of security features is performed (RBAC, DAC and MAC elements could be mixed in the augmented model, and is not clear how such elements could be individualized). Furthermore, no mappings to a target platform or access control policy languages are provided.

An aspect-oriented approach to access control is presented in (Hovsepyan et al., 2007), using UML and the concepts of “access interfaces” and “view connectors”. In (Hovsepyan et al., 2009), they shift towards exploring the use of DSMLs in this AO approach, and use the access control concerns as the illustrative example. They weave these concerns into a UML base model by means of a “concern interface”, which is dependent of the concern to be composed. The approach in (Hovsepyan et al., 2009) is similar to the one presented in this thesis in that it proposes the use of domain-specific modeling languages to describe individual concerns. Both methodologies allow the independent evolution of the concerns models. However, (Hovsepyan et al., 2009) directly generate code taking all (composition, base, and aspect) models as input. This approach severely handicaps the possibility of model analysis and validation of the non-functional concerns. It resembles domain-specific multi-modeling, which implies that the approach is exposed to the coordination problem (Hessellund, 2009). Our method follows the recommendations of (Kelly and Tolvanen, 2008) in that all concern models should be integrated into a unified one prior to code generation.

6.1.2 Domain Specific Languages

Domain-Specific Modeling (Kelly and Tolvanen, 2008) is an approach that makes use of domain-specific metamodels for the description of each aspect of interest in the software system. As adopted in this thesis’ approach, DSM integrates the different metamodels in one larger monolithic metamodel used to describe the whole system. In DSM, the different software systems are modeled at one particular abstraction level (defined by the selected/assembled metamodel), and all software should be derived from it. Therefore, DSM, as presented in (Kelly and Tolvanen, 2008), does not

follow an abstract-to-concrete series of transformations, as proposed by MDA, for instance. This thesis advocates for the use of such a methodology for the development of non-functional concerns as separate aspects with proper domain-specific modeling languages.

Lockheed Martin proposes Model-Centric Software Development (MCSD), its own form of model-driven engineering (Waddington and Lardieri, 2006). MCSD is an integrated approach in which models are central to all phases of the development process, although subtly different from other software modeling efforts which concentrate largely on generating implementation artifacts from models. Instead, MCSD is based on the following concepts (Waddington and Lardieri, 2006):

- Avoiding a one-language-does-all approach, using DSMLs to represent "aspects of interest" such as atomicity of data access.
- Automated generation of partial implementation artifacts. Instead of being restricted to program skeletons, partial implementations also can include fine-grained concrete functionality and specifications for software simulators and emulators.
- Integration of legacy assets through reverse engineering.
- Model verification and checking, through static analysis.

All this is achieved by integrating selected technologies in metamodeling, model checking and verification, code generation, and reverse engineering. Apparently, the supporting environment is built in a rather ad hoc fashion (Montangero and Semini, 2008).

Model-Integrated Computing (MIC) (Jackson, 2007; Sztipanovits and Karsai, 1997; Lédeczi et al., 2001) presents a system-level multi-aspects modeling approach. The focus is on the rapid and cost-effective composition of domain-specific design environments. In MIC, domain-specific metamodels specify the modeling language syntax. The semantics are introduced by the different transformations. Metamodels, originally described by UML class diagrams and supported by OCL constraints, are currently based on DSMLs. State machines are used to model the behavior of the system.

Microsoft presents in (Greenfield and Short, 2003) its own domain-specific model-driven approach, "Software Factories", exclusively targeting Microsoft Windows as its objective platform. The approach presented here is more general in its scope.

Other high-level approaches addressing the expression of non-functional requirements, such as the NFR (Non-Functional Requirements) framework (Chung et al., 2000) (appearing repeatedly in literature, in the field of non-functional characteristics research) represent a plausible alternative to business processes. In the framework, NFRs are represented as (soft)goals, and the notion of *satisficing* (Simon, 1996) (reflecting partial contributions towards or against the achievement of the goal) is incorporated. The NFR framework can even be placed into a higher level or previous step, and be mapped into the CIM workflow. Mapping rules need to be defined accordingly. This falls within the scope of future work.

Access control is also addressed by proposals which are not based on UML. (Lodderstedt et al., 2002; Basin et al., 2006), for example, take a domain-specific approach on access-control by defining SecureUML, an RBAC metamodel, although they also define a UML profile to create SecureUML models with UML. Mappings to access control infrastructures are defined for SecureUML and later, in (Basin et al., 2007; Basin et al., 2009), rules for the validation of SecureUML policies are proposed. (Clavel et al., 2008) presents the result of the application of their approach in an industrial experience.

(Braga, 2009) leverages on this previous work and defines a mapping to an abstract aspect-oriented access-control metamodel (Aspects for Access Control, AAC) and applies the validation rules to it. Similarly to this thesis' proposal, (Braga, 2009) merges SecureUML and the AAC metamodel, and uses this as an intermediate metamodel for code generation. The approach in this

dissertation supports not only RBAC, but other access-control techniques models by incorporating a generic access control metamodel (Mouelhi et al., 2010; Mouelhi, Baudry and Fleurey, 2008). SecureUML models are one of the possible inputs. Moreover, the target platforms available to this approach are not limited to aspect-oriented frameworks, although such frameworks are supported. The intermediate metamodel is independent of the platform, and the mappings should hold the necessary knowledge to create a particular implementation.

Approaches based on Aspect-Oriented Ideas

Aspect-oriented architecture description languages (ADLs), such as (Cuesta et al., 2002; Pérez et al., 2003; Cooper et al., 2003; Pessemier and Seinturier, 2004; Mencl and Bures, 2005; Pinto et al., 2005; Chavez et al., 2006; Palma et al., 2006; Barais et al., 2006; Pinto and Fuentes, 2007; Navasa et al., 2009), bring together the benefits of ADLs and aspect orientation. An attempt to unify best merits of multiple AO-ADLs into an integrated aspect-oriented modeling for software architecture is described in (Krechetov et al., 2006). The result, however, is not capable of modeling all aspect concepts. Aspect oriented approaches use, in most cases, the same language (usually a general purpose language) to describe all aspects (functional and non-functional). Alternatively, the approach presented in this dissertation proposes the use of individual modeling languages adequate for each specific concern, supporting a higher abstraction level, and therefore greater productivity.

In (Lohmann and Ebert, 2003), the Hyperspace approach (Ossher and Tarr, 2000) is generalized by using multiple metamodels to describe different concerns, in a multi-dimensional separation style. Concerns models are later combined in a integrated metamodel, in which new relationships connect unit types from different languages. Some ideas in (Lohmann and Ebert, 2003), although the realization differs, are shared in this thesis.

Close to the view presented in this dissertation is that of Domain-Specific Multi-Modeling (DSMM) (Hessellund, 2009; Lohmann and Hessellund, 2009). In the words of (Hessellund, 2009), DSMM is a development paradigm that relies on multiple domain-specific languages to express the different concerns of an enterprise application. Using multiple languages is a way of separating concerns during development. Just like in the approach proposed in this thesis, the different concerns of a system are conceptually separated and made explicit as independent domain-specific languages. Both approaches share the increase in productivity and quality by raising the overall level of abstraction (Hessellund, 2009). However, in domain-specific multimodeling, the different concern models are never woven together. This introduces a new problem: that of coordinating multiple different languages in a single system, which the author calls “the coordination problem”. The author uses a coordination model to enforce consistency across multiple languages, including how elements from different languages can refer to each other. In this last aspect, coordination models fulfill a role similar to that of the weaving models in the approach in this work. In contrast, as models are integrated in a unique model in the proposed approach, the coordination problem simply doesn’t occur. In DSMM, in the final system, the concerns must be re-integrated in a successful manner, a subject of not minor importance, as different domain-specific (meta)models are composed differently, as presented in section 3.2.2. This problem is not addressed in (Hessellund, 2009).

Programmable Logic Approaches

In (Jackson et al., 2009) structured logic programming, instead of metamodeling, is used as a formal approach to specifying non-functional requirements as constraint-systems over the space of models. The claim is that many NFPs can be regarded as constraint-systems. Similarly to the work presented in this memoir, this constraint-systems’ approach supports the independent development of NFPs, which can be later composed together. Logic programming is used to model platforms, and the constraints apply to such models. However, no graphical means are provided to create

these models. Although probably useful for systems with strict certification needs, such as safety critical software, logic programming as means to model systems add little to the abstraction level of code. Therefore, no productivity gains may be obtainable from such shift in the development habits. The approach in this thesis makes use of domain specific metamodels as the modeling language for each individual concern, benefiting from the productivity increase provided by DSM, as stated in (Kelly and Tolvanen, 2008). In addition, is also not clear how non-measurable NFPs (i.e., security as opposed to throughput, performance, worst execution time, etc.) could be expressed in this logic programming. The expression of high level authorization languages as constraints systems is identified as future work (Jackson et al., 2009), but it has not been presented so far.

Surveys and discussions on different approaches for modeling and analysis of the architecture of secure software systems (not limited neither to access control aspect of security, nor to UML-based approaches) is presented in (Dai and Cooper, 2007), and limited to aspect-oriented approaches in (Dehlinger and Subramanian, 2006).

6.1.3 Author's Publication on the Subject

The different publications in which the author has participated, and that address, in a greater or lesser manner, the subject of the modeling of non-functional properties in different type of systems (e.g., safety-aware, service-oriented, etc.) include: (Briones et al., 2008b; Briones et al., 2008a; de Miguel, Massonet, Silva Gallino and Briones, 2008; Briones et al., 2009a; Briones et al., 2009b; Silva Gallino et al., 2011; Silva Gallino, Juan Pedro and de Miguel, Miguel and Briones, Javier F. and Alonso, Alejandro, 2011; Silva Gallino et al., 2010; Briones, Alonso, de Miguel and Silva Gallino, 2010; Silva Gallino et al., 2009b; Briones, de Miguel, Alonso and Silva Gallino, 2006).

6.2 Methodologies and Development Processes

Agile Modeling (AM) (Ambler and Jeffries, 2002; Erickson et al., 2005) is a practice-based methodology for effective modeling and documentation of software-based systems. Meant to be tailored into other agile methodologies such as XP or RUP (Ambler, 2009), it significantly differs from model-driven methodologies, in which models play an important role in the generation of the final executable artifacts. In this dissertation, a model-driven development process is proposed to address non-functional characteristics of services.

In (Anaby-Tavor et al., 2008), a metamodel and technology independent model-driven methodology is proposed for the functional aspects of services' engineering. The development process proposed in this thesis is, similarly, metamodel independent. Every concern should be modeled with an appropriate metamodel, and then unified into a intermediate model. IMM is an particular implementation of such a unified metamodel, allowing the definition of services' models with awareness of non-functional concerns. The approach in (Anaby-Tavor et al., 2008), on the contrary, leaves non-functional characteristics out of the scope.

Methodologies that make use of UML

Grønmo and Jaeger (Grønmo and Jaeger, 2005) present a methodology for building QoS-optimized web service compositions. Their work addresses the QoS-based selection of services for the composition by means of UML and extending the standard QoS profile (OMG, 2008d). This work, in contrast, focuses on the development of services, including their required non-functional characteristics.

SOA-MDK (Barn et al., 2008) is a model-driven methodology for component-based SOA systems, making use of UML and BPEL. It proposes the use of reference models, acknowledging the existence

of multiple viewpoints, focusing, however, on the functional aspects of the system. The idea of the existence of multiple viewpoints is shared, although in this dissertation the different views represent a different (functional or non-functional) concern of the system.

Again with participation of the ALARCOS group, (Delgado et al., 2011) extends the Business Process Service Oriented Methodology (BPSOM) to include service generation in SoaML. The work by Delgado is discussed in further detail in section 6.4.

(Cortellessa et al., 2006; Cortellessa et al., 2007a) take performance as a non-functional concern as its subject. The authors propose horizontal transformations of UML models annotated with the SPT profile, to Queuing Networks models at the three MDA's abstraction levels (CIM, PIM, PSM). They name the respective methodologies PRIMA (Performance Incremental Validation) and COBRA (COmponent Based Reliability Assessment). In the queuing models' CIM-PIM-PSM chain, vertical transformations do not work as a traditional MDA's refinement process, but only provide a input contribution to the next level's horizontal transformation. During the work performed towards completing this dissertation, although not directly performed by the author, model-to-model horizontal transformations were used to generate Fault Tree Analysis (FTA) and Failure Mode Effects and Criticality Analysis (FMECA) models out from annotated design models (de Miguel, Briones, Silva Gallino and Alonso, 2008). Differently from (Cortellessa et al., 2007a), the need to use a more abstract analysis model as input did not arise.

Methodologies with Multiple Perspectives

Viewpoints (Finkelsetin et al., 1992) is an open framework supporting any methodology, including the one proposed in this memoir. Just as it was proposed here, Viewpoints recognizes the existence of multiple perspectives in a system under development. In the authors view, different development-team members use different representation styles chosen to be particularly appropriate to their area of work (Finkelsetin et al., 1992). Such *representation styles* map, in the development process proposed in this thesis, to the different domain-specific metamodels that describe each (functional or non-functional) concern. In fact, the Viewpoints approach, together with Multi-Dimensional Separation of Concerns, Model Driven Architecture, and Domain Specific Modeling, provide the basis on which the the development process proposed in this dissertation is founded.

A model-driven methodology for early aspects is presented in (Sánchez et al., 2010). There, the authors leverage previous work to define a process that departs from requirements, modeled with the UML profile for Aspectual Scenario Modeling (ASM) which is based on (Clarke and Baniassad, 2005), and following the Aspect-Oriented Requirements Analysis (AORA) approach (Brito and Moreira, 2003), towards architectural descriptions (expressed in UML's profile for CAM (Pinto et al., 2005)). The approach bases on applying recurrent aspectual patterns on the (semi-automatic) transformations from requirements to architectural descriptions. The approach in this thesis and that of (Sánchez et al., 2010) share various coincidences. Nonetheless, as in most aspect-oriented proposals, in (Sánchez et al., 2010) all aspects are described with the same general purpose language. Reviews of the state of the art of MDD in the context of non-functional aspects are available in (Bakker et al., 2005; Chitchyan et al., 2005; Zhu and Liu, 2009; Ameller et al., 2010; Bombonatti and Melnikoff, 2009).

Formal Methodologies

The π -Method (Oquendo, 2006) proposes the use of π -ADL (a formal architecture description language, based on the π -calculus) and related languages in a formal model-driven methodology for software engineering. Having π -ADL a textual notation, the authors propose a UML profile to graphically model the system. π -ADL and the π -Method have been applied in different domains (including service-oriented architectures (López-Sanz et al., 2008; Oquendo, 2008)), and present the

benefit of enabling rigorous analysis of the system properties. On the other hand, being focused on formalizing the architecture elements (solution domain), it is distant from the problem domain, lacking the productivity gains that may be achieved with domain-specific modeling. All in all, π -ADL and related languages look promising, and may represent an interesting formal target as a platform-independent language in cases where strict architecture analyses are needed. Mappings need to be defined to it from higher-level domain specific metamodels in order to take advantage of both DSM and formality. This may represent an interesting future line of work.

Methodologies Focused on Security

Tropos (Castro et al., 2002), an agent-oriented methodology that uses an extended i^* Modeling Framework's (Eric Yu, 2011) notation, covers all software development phases (from requirements, through design, and up to implementation). Secure Tropos (Mouratidis et al., 2003; Mouratidis and Giorgini, 2007) complements the Tropos methodology by including security concepts and a set of consistency and validation rules. Secure Tropos extends not only the Tropos language, but its development process as well. At the detailed design phase, the components identified in early stages of the methodology are described by means of the Agent Unified Modeling Language (AAML) (Odell et al., 2000). A review of (Secure) Tropos, and other methodologies capable of considering security and dependability concerns is presented in (Asnar et al., 2006).

Process for Web Services Security (PWSec) is a methodology, presented in (Gutiérrez et al., 2006), to enable the integration of security specific stages in the development of web services. PWSec defines three stages: one for security requirements, a second for security architecture, and a third for security technologies (WS standards). The work presented in this memoir addresses part of the architecture stage, and the whole of the security technologies stage, as defined in (Gutiérrez et al., 2006). Non-functional requirements gathering, and its mapping onto the CIM/PIM model is a probable future line of work.

A methodology for an end to end SOA security configuration is proposed by IBM's Tokyo Research Laboratory in (Satoh et al., 2008). The approach is model-driven, and makes use of templates to express identified security patterns. Initially, security intents are added as abstract keywords that represent security requirements, first at business level, and then at a service model level. Security primitives are expressed as UML stereotypes in (Nakamura et al., 2005; Satoh et al., 2006) to be later transformed into WS-SecurityPolicy artifacts (Satoh and Yamaguchi, 2007). Alternatively, patterns may be indicated in place of intents. This mapping from business-level requirements to technical intents is a manual task, performed by a software architect. Security intents are complemented with a Security Infrastructure Model (SIM) to later generate a concrete configuration. By the same research group, access control is addressed as a pattern in (Satoh et al., 2008) in a similar way to access control lists. However, no intents are available to express the "authorization" requirement. Validation of the security policies is addressed in (Nakamura et al., 2007; Ono et al., 2007), and with respect to requirements in (Satoh and Uramoto, 2010). The approach is at this time closely-related to SCA (Service Component Architecture) (OASIS, 2007a) (security patterns are expressed by means of SCA's Service Component Definition Language, SCDL).

The approach in (Satoh et al., 2008) shares some characteristics with the approach presented in this dissertation. First, both processes follow MDA's abstract to concrete model refinement approach. Second, their use of abstract security intents in the first development stages is shared. Later, the SIM, although closer to topology modeling, fulfills an equivalent functionality to that of the WS-SecurityPolicy platform model used in section 4.3.5. The expression of high level security intentions, and the use of a means (e.g. model, library, transformation sub-module) for indicating the available security implementation possibilities of the platform are shared approaches. Moreover,

the use of security patterns is compatible with the approach being presented here ¹, and is not discarded in advance. On the other hand, some differences are noticeable. Both approaches differ in that the one in (Sato et al., 2008) is limited to security, and that it does not offer the same flexibility in the selection of modeling languages. Above all, the most significant difference is in what is, perhaps, the greatest strength in this proposal: the use of separate concern models to address each non-functional characteristic, a feature that aids reuse and modularity of design and implementation.

An alternative methodology for security-aware services is Sec-MoSC (Souza et al., 2009) expressing security requirements at different levels of abstraction: incorporating security requirements in business process models, and down through the services' composition, up to the BPEL execution and runtime monitoring. Sec-MoSC (Security for Model-oriented Service Composition) presents a very interesting approach. It is, however, addressed to the composition of services. The approach in this memoir focuses more on the development of the services, including their non-functional characteristics. Additionally, Sec-MoSC methodology is limited to security concerns.

6.2.1 Author's Publication on the Subject

Published material that deal, to some extent, with the subject of methodologies or development processes, and which the author of this dissertation has either authored or co-authored, include: (Silva Gallino et al., 2005; Briones, de Miguel, Silva Gallino and Alonso, 2006; de Miguel et al., 2006; de Miguel, Briones, Silva Gallino and Alonso, 2008; de Miguel, Massonet, Silva Gallino and Briones, 2008; Silva Gallino et al., 2009b; Silva Gallino et al., 2010; Silva Gallino et al., 2011; Silva Gallino, Juan Pedro and de Miguel, Miguel and Briones, Javier F. and Alonso, Alejandro, 2011).

6.3 Analyses, Metrics, and Model Measuring

6.3.1 Model Metrics

On the field of model measuring, (Monperrus et al., 2008; Monperrus et al., 2009) present one generic metric for metamodel level, which can be used to specify a wide range of model metrics. They interestingly describe the state of the art of model measuring as part of their research. This metric (sigma), based on the set theory and first order logic, can be used to express other domain metrics. Lines of Code (LoC), requirements, and engineering metrics are provided as examples and expressed in terms of the sigma metric. The sigma metric could be useful for the uniform processing of metrics, but needs specific metrics for the domain to be defined in terms of sigma. It doesn't provide any useful measures by itself.

Size estimation of non-functional requirements (NFR) is the subject addressed in (Kassab et al., 2008). The authors present the NFR Size Measurement (NFSM) method, proposing the use of the ISO 19761 (ISO/IEC, 2003) standard to calculate the size of non-functional requirements, expressed by means of the NFR framework (Chung et al., 2000).

(Bettin, 2003) introduces Atomic Model Elements (AME) to measure modeling effort as an equivalent of Lines of Code (LoC) in models. The AME metric is applied to a sample application, but no real world studies are performed, and no validation of the metric is provided.

As described later in section 6.4, Gilmore, in (Gilmore et al., 2005), applies process calculus to the analysis of security and performance of a choreography of web services.

Känsälä addresses the assessment of risks that may affect the normal execution of a system development process in (Känsälä, 1997). There, the author defines "RiskMethod" and presents

---

¹Patterns, best practices, and other forms of "know-how" are part of the model transformation rules

“RiskTool”, providing metrics for the estimation of how risks (such as the volatility of the requirements, the logical complexity of the software, etc.) may affect the final development cost of the project. This assessment may also be performed on the type of projects addressed in this dissertation.

A preliminary version of security and dependability metrics for the SERENITY project is presented in (Asnar et al., 2009). There, an ontology providing security primitives is presented. Metrics are defined, such as cost of measure, complexity of measure, value of goal, integrity and confidentiality protection metrics, etc., some of these metrics are quite abstract, while others are specified with more precision. Those metrics were then applied to the risk assessment of workflow’s execution, and also to the Air Traffic Management domain. The work in (Asnar et al., 2009) is applicable to the models used in this thesis. An extended version of this work is intended, and may be interesting to see.

(Krautsevich et al., 2010) presents an initial effort towards the definition of a formal approach to security metrics, and the characterization of what a ‘*more secure*’ relation means. The approach focuses on the attackability of a existent system, as deduced from the set of presented metrics: number of attacks, minimal cost of attacks, maximal probability of attack, attack surface metric, etc. In this thesis, the focus is on metrics that may be applied at early stages of the system development (e.g., architectural design) to help predict/estimate the degree of fulfillment of the non-functional characteristics of the resulting system. Such metrics would also help the developers select among different design alternatives regarding the non-functional aspects.

Recently (Di Marco et al., 2010), a metrics’ framework for evolving and dynamic dependability and QoS metrics was defined, within the context of the CONNECT (Emergent Connectors for Eternal Software Intensive Networked Systems) European project. This article presents ongoing work towards defining a new metric classification framework characterized by orthogonal dimensions, to perform online and offline analysis in CONNECT architecture. Elaborating on traditional, well-understood dependability metrics, a conceptual model has been developed as an structured framework, which refines generic metrics into CONNECT-dependent and context-dependent metrics. The elements and relations in this conceptual model are being formalized into a metamodel. In (Di Giandomenico et al., 2010), verification and validation, stochastic model checking, and state-based stochastic analyses are performed for dependability. Although specific to CONNECT, the research presents an interesting approach, and is worth of attention to its future progress.

6.3.2 Model-Driven Analyses

Design and analysis are considered as two orthogonal spaces in (Jonkers et al., 2005), complementing MDA with horizontal model-to-model transformations from what they call the “design domain” into the “analysis domain”. This approach resembles one applied in some work developed in our group, presented in (de Miguel, Briones, Silva Gallino and Alonso, 2008). In (Jonkers et al., 2005) in particular, UML and the standard QoS and SPT profiles are used to guide the transformations, and make use of the performance analysis presented in (Balsamo and Simeoni, 2001b; Balsamo and Simeoni, 2001a). An important distinction the authors of this work make is the difference between quantifiable aspects and others which qualitative nature cannot be quantified, a remark which is shared.

(Henriksson et al., 2005) defines Quality-Driven Development (QDD), based on OMG’s MDA, arguing that software development should not only rely on code verification, but on design verification also. QDD extends MDA in that it defines separate MDA stacks for Model-Driven Verification (MDV). This appears as just a new name for horizontal transformations from design to analysis, and then refine the analysis with more platform-specific data. As an example, UML stereotypes are used to drive the design-to-analysis transformation. Analysis models are then used as inputs in a model checker, and a worst case execution time analysis (WCETA). The applied technique

is to transform all diagram's information into state charts, and later transform those into timed automata, which are the subject of the analysis.

(Bhargavan et al., 2005; Bhargavan et al., 2008) addresses validation of web services security policies. In (Bhargavan et al., 2005) a formal model is derived from WS-Security semantics in the shape of a dialect of pi calculus, which is then analyzed for correctness. The results were very low level, and hard to manage even for experts. Consequently, a new, more abstract, policy advisor is proposed in (Bhargavan et al., 2008). The validation tools presented in (Bhargavan et al., 2005; Bhargavan et al., 2008) may be very useful to achieve a safe configuration of security policies. Such tools could be used in concert with our framework to validate the modeled security configurations. (Nakamura et al., 2007; Ono et al., 2007; Satoh and Uramoto, 2010), also addressing validation of the security policies, are described in section 6.4.

Validation of the deployment of access control security policies is addressed in (Preda et al., 2010), by means of an OrBAC (Organization-Based Access Control) model and a tracing path algorithm. The B-Method is used as the theorem prover. A known infrastructure deployment is needed to perform the validation. The focus of (Preda et al., 2010) is different from the one of this work. It only supports OrBAC access control, and validates the deployment, not the policy. It resides, therefore, at a later stage in the development lifecycle. Both approaches could be used together (not without some adaptation) if desired.

Liu presents in (Liu and Traore, 2007; Liu, 2008) an interesting work on security analysis, proposing the User System Interaction Effect (USIE) model to derive and analyze security concerns from service-oriented software architectures. In particular, Liu's work focuses in the concept of *attackability*, which represents the extent to which software systems or services can be the target of successful attacks. It also defines and makes use of security metrics for the analysis of software security, arguing that metrics present a more practical alternative to be applied in the industry when compared to that of formal analysis methods. This view is partly agreed with, but both metrics and formal analysis are considered to be complementary. Which of this should be the most appropriate choice depends on the particular aspect under evaluation. (Liu, 2008) also presents a good survey of related work on security analysis of software systems designs.

Reliability (Rodrigues et al., 2004; Rodrigues et al., 2005; Wassermann and Emmerich, 2006) and performance (Skene and Emmerich, 2003) are the non-functional aspects addressed by the work in the University College London. In (Skene and Emmerich, 2003), UML SPT's performance sub-profile and OCL are used to derive queuing networks models for performance analysis. The analysis to be performed on these queuing network models is not presented in the publication. In (Rodrigues et al., 2004), both SPT and QoS profiles are extended to model reliability at PIM level, and the Enterprise Java Beans (EJB) profile is extended and targeted at PSM level. Again, analysis are not presented. (Rodrigues et al., 2005) deals with reliability prediction in MDD, based on scenario specification. The approach taken is to extend UML's QoS and SPT profiles to support the necessary dynamic aspects of reliability. This annotated UML model is then transformed into a Labeled Transition System (LTS), used for automated reliability prediction and identification of implicated scenarios. The particular case of reliability in atomic web services orchestrated in Business Process Execution Language (BPEL) (or similar) workflows, is addressed in (Wassermann and Emmerich, 2006). This work proposes the guidelines of the authors' future research in this particular scenario. No published results from this line of research could be found at the time of writing.

These two NFPs (reliability and performance) are also the target of research in the University of L'Aquila. Similarly to (Rodrigues et al., 2004), in (Cortellessa et al., 2007b) the authors define platform independent and platform specific models, but also computational independent models in the non-functional domain. To this purpose they define the Non-Functional Model Driven Architecture (NFMDA) framework. In it, UML models are annotated with profiles to guide the horizontal transformations towards the non-functional domain. The particular profiles that should

be applied in each case are not specified, but it is assumed that standard UML profiles are used. In (Berardinelli et al., 2009), MARTE (Modeling and Analysis of Real Time and Embedded Systems) (OMG, 2007) and DAM (Dependability Analysis and Modeling) (Bernardi et al., 2009) profiles are the ones used to address reliability, availability, and performance.

Other research on performance analysis approaches address the subject by deriving queuing network models (Alsaadi, 2004; Gu and Petriu, 2002), labeled (stochastic) Petri Nets (Bernardi et al., 2002; López-Grao et al., 2004), and Stochastic Process Algebra (SPA) or PEPA (Performance Evaluation Process Algebra) nets (Canevet et al., 2004; Pooley, 1999). A survey on the subject can be found in (Balsamo and Simeoni, 2001a).

BPEL workflows are the subject of multiple pieces of related research. Among them, (Zarras et al., 2004) uses BPEL for the dependability analysis of web services. In that work, BPEL workflows are primarily transformed into UML models complemented by a proprietary profile. This annotated UML model is then automatically transformed into block diagrams, and then into Markov models, which are the final subjects of the analysis. The approach in (Zarras et al., 2004) is presented with a simple example, and no real use cases or tools are available.

(Basin et al., 2009) describes and exemplifies the use of OCL for the analysis of SecureUML (Lodderstedt et al., 2002; Basin et al., 2006) models. This analysis is implemented over the SecureMova (SecureMova, n.d.) tool. Although a viable approach, the use of OCL syntax is cumbersome for non-experts. Graphical means for deriving such OCL rules would be desirable, but are not provided. Moreover, some queries on the model can not be expressed with OCL formulas and, according to the authors, require theorem proving. Nevertheless, the application of OCL analysis is possible on the metamodels presented in this thesis. The particular formulas, being metamodel-specific, would have to be customized for the participant metamodels.

Other alternative analysis methods include the evaluation of contracts between components. The assembly and coupling of components developed by third parties create risks because of the incompatibility of the security and reliability characteristics of components. Some experimental methods are used for the specification of security attributes and for the evaluation of contracts (Khan and Han, 2002).

6.3.3 Author's Publication on the Subject

On the particular field of model metrics and model-driven analysis, several pieces of research, with different implication of the author, have resulted in published papers. These publications include: (Silva Gallino et al., 2005; Briones, de Miguel, Silva Gallino and Alonso, 2006; de Miguel et al., 2006; Briones et al., 2007; de Miguel, Briones, Silva Gallino and Alonso, 2008; Briones, Alonso, de Miguel and Silva Gallino, 2010; Briones, de Miguel, Alonso and Silva Gallino, 2010; Briones, de Miguel, Silva Gallino and Alonso, 2010).

6.4 Modeling Service-Oriented and Web Services Systems

Some approaches applying MDA to the development of service-oriented systems and their NFPs already exist. However, in most occasions these approaches do not offer thorough support for access control descriptors, code generation, and, in the case of web services, WSDL and WS-Policy. This section describes some of these approaches.

6.4.1 Proposals Addressing the Functional Aspects of SOA Systems

Many pieces of research have already address the model-driven development of service-oriented architectures exclusively focusing on the functional aspect of the system (Cao et al., 2004; Bézin

et al., 2004; Grønmo et al., 2004; Patrascioiu, 2004; Jiang et al., 2005; Lopes et al., 2005; Yu et al., 2007). Some (Bézivin et al., 2004; Patrascioiu, 2004; Jiang et al., 2005) make use of the EDOC (Enterprise Distributed Object Computing) (OMG, 2004) UML profile to express Web Services' concepts. The use of the EDOC profile facilitates the generation of more complete service models (Bézivin et al., 2004) by presenting concepts closer to the problem domain. Precisely to improve this closeness to the problem domain, in this dissertation the SoaML profile and domain-specific metamodels are used for expressing the different functional and non-functional concerns.

The Kybele group proposes the Service-Oriented Development Method (SOD-M) (Valeria de Castro and Esperanza Marcos and Juan M. Vara, 2011), a methodology for the derivation of PIM/PSM models from CIM-level business models of service compositions. In SOD-M, business processes are represented by means of BPMN and value models (Gordijn and Akkermans, 2003), and specifically designed DSMLs are used at PIM and PSM levels. The most remarkable element of SOD-M may be its focus on the modeling and generation of the behavioral aspect of information systems. SOD-M is integrated into the MIDAS framework (Marcos et al., 2006).

The work by Delgado, in collaboration with the ALARCOS group, focuses on the model-driven development of the functional aspect of service-oriented architectures. To that end, the authors extend a methodology (Delgado et al., 2011), define an ontology (Delgado et al., 2010), and developed the MINERVA framework (Delgado et al., 2009). Defining a MD process that transforms BPMN business processes into SoaML models, and making use of ontologies at the CIM level, the approach by Delgado is seen as complementary to the one presented in this thesis. It would be interesting to see in the future how both approaches could be combined to achieve a full-blown model-driven development of service-oriented systems with non-functional awareness.

6.4.2 Non-Functional Approaches Based on UML Profiles

Among the proposals that deal with the different aspects supported by WS-Policy, the approach by Ortiz and Hernández (Ortiz and Hernández, 2006) is one the most representatives. Ortiz makes use of model-driven development and aspect-oriented modeling for including extra-functional concerns in web services parting from a platform independent model. Ortiz's solution makes use of SCA modeling and a UML profile for modeling aspects. This work, in change, proposes the use of DSMLs, more adapted to each of the non-functional aspects in hand, to later combine these DSMLs into one integrated model. Ortiz's work also does not describe any means to generate the implementation of the security aspects previously mentioned. In (Ortiz et al., 2008), the authors compare the use of aspect-oriented techniques and model-driven approaches to traditional development of web services' development. A set of metrics are applied to evaluate separation of concerns, complexity, coupling, cohesion, size, and some performance measures are taken. Some of those metrics could be adapted to be applied in the development process presented in this memoir, and results could later be compared between the two approaches. This belongs to the future lines of work.

Gilmore, in (Gilmore et al., 2005), applies process calculus to the analysis of security and performance of a choreography of web services. Standard UML and profiles are used to model the services, and process algebra is used for the analysis. More recently, in (Gilmore et al., 2010), they define a new UML profile for SOA compositions (UML4SOA), extended for contract-based non-functional properties (UML4SOA-NFP), and the MARTE profile for quantitative characteristics of NFPs such as performance. Authorization, although regarded as one of the security concepts, is not addressed.

(Kaviani et al., 2008) present the General Policy Modeling Language (GPML), a generic metamodel that extracts common policy concepts from several policy languages, and grounds them to a deontic logic foundation. An UML profile is provided to express such policy concepts in UML. Transformations to KAoS, Rei, and Ponder2 policy languages are provided. Contrarily, our work focuses on the WS-Policy umbrella of standards.

Reliable messaging (RM) is the non-functional property addressed in (Gönczy et al., 2006), and a performability analysis of these reliability-aware models is performed in (Gönczy et al., 2008). (Gönczy et al., 2006) proposes a model-driven approach towards the generation of configuration artifacts for middleware frameworks with support of RM standards. An OWL description of the model is also generated, to enable model queries for analysis and validation. The system is modeled in UML, and reliable-messaging concepts are added through stereotypes (derived from a RM metamodel defined by the authors). The mappings to the standards are not defined, although they are direct, as the concepts presented in the metamodel are the same ones described in the standards (there is no increment in the abstraction level). The approach in (Gönczy et al., 2006) is similar to the one presented here, although limited to reliable messaging. New metamodels and transformations need to be defined to integrate other non-functional concerns.

(Oladimeji et al., 2007) presents a UML extension to model security requirements in the form of authorization and obligation security policies. It makes use of stereotype properties in UML ports (in component diagrams) to describe security mechanisms (encryption, etc.), and an adaptation of the Ponder policy specification language to model security policies in inner elements. Once again, no mapping to an implementation platform is provided.

(Wada et al., 2006; Wada et al., 2007; Wada et al., 2009) uses UP-SNFR, a UML profile, to support the specification of NF requirements (NFRs) inside UML models for SOA systems. UP-SNFR models security at a low, technical level, describing properties such as security tokens, algorithms, timeout, priority, synchrony, etc. Feature modeling (Czarnecki et al., 2002) represents the basic mechanism applied to represent NFPs. In (Wada et al., 2008), the approach is extended to BPMN business processes models. A weaver takes a feature configuration model and the BPMN model as inputs, and weaves both together. The result is finally transformed into a UP-SNFR model. The abstraction level applied to the non-functional properties is the same in both business process and UML models. The technical level adopted forces at least a middling degree of knowledge on the non-functional concern (a certain knowledge on security mechanisms, for instance). The intention behind the work in this dissertation is to relieve the designer/developer from the need of expertise in the particular non-functional field. Stakeholders, such as business analysts, express abstract, non-technical intentions at a high-level model of the system, which are later automatically refined into more technically detailed descriptions with a certain platform specificity.

The previously presented proposals, as all approaches that make use of UML, leverage on the use of a standardized way of expressing software artifacts. This provides some advantages, e.g. with respect to tooling compatibility. However, and as already described, they suffer from the drawbacks of general-purpose languages when compared to domain-specific strategies. In this thesis, each non-functional concern is addressed by a suitable modeling language.

6.4.3 Other Approaches Addressing Non-Functional Properties

Following the same line of work of (Ortiz and Hernández, 2006) is the proposal in (Jegadeesan and Balasubramaniam, 2009). Although more recent than Ortiz's work, their approach is very similar. Jegadeesan makes use of model driven development, and, differently from (Ortiz and Hernández, 2006), a domain-specific Policy metamodel. Aspects are divided in three levels: service, business, and domain aspects, defining a vocabulary for the domain constraints. These different aspects are later expressed in the shape of WSDL 2.0 and WS-Policy documents. A service policy metamodel is defined for the later. WS-Policy is selected due to being a widely accepted standard, and WS-PolicyConstraints is the choice to specify domain-independent generic constraints using XACML-based functions. Our approach, on the contrary, leaned towards the use of the much more accepted WS-SecurityPolicy (OASIS, 2007d) standard. Even though Jegadeesan and Ortiz present use cases in their publications that prove their concepts, no tool implementing their approaches has been provided so far.

Quality models, equivalent to those in (ISO/IEC, 2001) but in this case specific to the field service-oriented architectures, can be found in (Pettersson, 2006; O'Brien et al., 2005; O'Brien et al., 2007). Both models share some characteristics (e.g., modifiability, reliability, security, etc.), but differ in others (e.g., return of investment (ROI) and development cost in (Pettersson, 2006), or Testability and Auditability in (O'Brien et al., 2005)). On security in particular, (Pettersson, 2006) identifies four sub-characteristics: authentication, authorization and access control, firewalls, and encryption. (O'Brien et al., 2005), however, further classifies security into confidentiality, authenticity, integrity, and availability. In this dissertation, availability is considered closely related to dependability, although not regarded as a sub-characteristic of security, but as an independent, first-level attribute.

(Tsai et al., 2005; Tsai et al., 2007) introduce PSML-S, a modeling language for Service-Oriented computing, and (Tsai et al., 2008) puts forward Pi4SOA for verification and control of service collaboration. PSML-S is heavily focused on supporting simulation of service-oriented architectures, and not so much on the modeling of non-functional properties. The PSML-S language is divided in packages, one of which (constraints package) deals with non-functional concepts, providing means for policy specification. The use made of NFPs in this approach is to constraint simulation characteristics such as timing, performance, and throughput, and for policy enforcement in Pi4SOA. They define PSML-P (a new policy language described in (Tsai et al., 2008)) to be used by the framework. PSML-P policies are textually applied to models. A framework is also defined to enforce these policies. The approach in this thesis differs in two ways. First, policies are graphically expressed leveraging a policy metamodel. It can be argued that graphically expressing policies aids in the visualization, rapid understanding and the readability of models. Secondly, standard policy languages are used. No new policy languages are defined. The selected policy metamodel is generic, and, although only mappings to the WS-Policy umbrella of standards have been defined in this work, independent of the policy language used as target platform.

(OASIS SCA Policy TC, 2010) specifies a Policy framework for Service Component Architecture (SCA). The proposal deals with Quality of Service (QoS) goals expressed as "Intents" (a sort of quality primitives) which cover three areas: Security, Reliability, and Transaction. Such intents are, at a lower level, satisfied by policy assertions. (OASIS SCA Policy TC, 2010) has WS-Policy in mind, but is independent of it and moves at a higher level of abstraction. Therefore, the QoS intents defined in this specification represent the CIM level primitives, representing an alternative to those used in chapter 4. Substituting the QoS intents makes both approaches compatible. Adopting SCA in IMM in place of the current proprietary functional metamodel, in search of a more standard solution, makes sense, and is a possible future line of work.

Security

Concurrent to the work presented here, a security metamodel is presented in (Menzel and Meinel, 2009; Menzel et al., 2009) to model security considerations in business processes, map them to service-oriented architectures, and generate configurations in WS-SecurityPolicy. The focus is on confidentiality, integrity, identification, and authentication using patterns. Security goals (primitives) are expressed in these models, and policy assertions satisfying these goals are specified. Mappings are defined for WS-Policy and WS-SecurityPolicy. In (Michael Menzel, 2010; Menzel and Meinel, 2010), the author presents SecureSOA, a security design language for the definition of security intentions in SOA. SecureSOA leverages SecureUML schema, defined to integrate the security modeling language with the functional design language. The concrete syntax of SecureSOA is defined using UML profiles. The modeled security intentions are then mapped to the metamodel defined in (Menzel and Meinel, 2009).

Previous and simultaneous work by the same group (Wolter and Schaad, 2007; Wolter et al., 2008; Wolter et al., 2009) focuses on modeling authorization characteristics of business processes. In this thesis, domain specific independent metamodels are used to describe the individual concerns

facilitates the understanding, analysis, and generation of the different aspects of the service individually. The work in (Menzel and Meinel, 2009) uses an unique metamodel to express all characteristics of the services hindering, therefore, the autonomous evolution of the different concerns. Additionally, the work in this dissertation is not limited to security concerns, being flexible enough to integrate other type of policies. Nonetheless, the work in (Menzel and Meinel, 2009; Menzel et al., 2009) is sound, and worth of consideration. Many useful ideas can be derived from it. In future work, the representation of high level abstractions of other non-functional characteristics different from security, and the automatic matching of correspondent PIM models in repositories is to be addressed.

Besides the previously described ones, many pieces of research have addressed the specification of security requirements in business process models (Neubauer and Heurix, 2008a; Neubauer and Heurix, 2008b; Rodríguez et al., 2007; Lehmann and Matthes, 2005; Johnston, 2004). Some of them were considered for defining the security intents vocabulary described in chapter 4 (e.g., (Satoh et al., 2008; Souza et al., 2009)). A survey of the available (up to 2008) model alternatives for dependability (i.e., security and reliability), although limited to those based on UML, is presented in (Jaeger and Werner, 2009). (Jakoubi et al., 2009) presents a survey of nine different approaches addressing security on business process models.

Service composition's security is addressed in a different way in (Charfi and Mezini, 2005; Charfi et al., 2006). There, a framework designed around AO4BPEL (an aspect-oriented extension to the BPEL execution language) is used to secure the composition. Security-as-a-service and deployment descriptors form the base of the approach. The deployment descriptor specifies which process activities have which security requirements. The individual processes, however, remain insecure if accessed from outside the BPEL engine.

Compared to the approaches in the two previous paragraphs, the work presented here is not limited neither to the specification of requirements/intents, nor to security as a non-functional characteristic. Security, nonetheless, is used as the exemplifying NFP. Additionally, the development of the non-functional functionality is also considered in this work, proposing a model-driven process based on an specific DSML appropriate for the non-functional concern.

Access Control

Emig defines in (Emig et al., 2007) an access-control metamodel for web service-oriented architecture (WSOA) derived from RBAC and ABAC (Attribute-Based Access Control). In (Priebe et al., 2004; Yuan and Tong, 2005), and in (Emig et al., 2008) a markup language (Web Services Access Control Markup Language, WSACML) is used to express this policies in a platform independent manner. Contrasting with the approach in section 3.2.2, instead of supporting the multiple already available access-control models, their focus is on defining a new one. They also define a platform independent XML syntax to express such policies, and later define mappings between WSACML and platform-specific syntaxes. The focus in this work is clearly different, as it doesn't try to define a new access-control technique. This effort seeks to provide means for the developers to be able to select the most appropriate access-control model from different techniques to be used in their systems' designs. Later, the platform specific models/code can be derived from the intermediate model by following the defined transformation rules. In our work, Mouelhi's access-control metamodel plays the role of WSACML, and SecureUML (RBAC), DAC, MAC, and OrBAC models can be used as input. A mapping from ABAC to Mouelhi's metamodel has yet to be defined, although it appears possible, as ABAC can be considered as a generalization of RBAC.

Arguing that traditional RBAC approaches are not appropriate for SOAs, (Sun et al., 2010) defines SOAC (Service Oriented Authorization Control) an access control approach specifically addressed to service-oriented architectures. SOAC is an extension to RBAC that is focused on the authorization issues of composite web services, addressed to the coordination of the constraints

of the different WS. In this thesis, access control (authorization) is addressed as a non-functional characteristic related to security. The intention is to support the largest number of techniques as possible, not to define a new technique. Hence, a generic AC metamodel, supporting multiple AC techniques, is adopted as part of the intermediate metamodel. Nevertheless, the IMM metamodel is only one of the many possible alternatives of an integrated metamodel that may be defined for the presented development process.

Semantic Approaches

Ontologies and taxonomies are also used to describe non-functional properties. In the case of (Paoli et al., 2008), the Policy Centered Metamodel (PCM), a metamodel for the description of non-functional properties of web services, is proposed. This metamodel is formalized by an ontology, following a BNF grammar syntax, and logical groundings on both OWL-DL (Ontology Web Language-Description Logics) and WSM (Web Service Modeling Language). Differently from this work, (Paoli et al., 2008) focuses on the description of web services' NFPs to aid in service selection. No mapping to the expression of NFPs in policy languages is provided.

Zschaler et al take a semantic approach towards the modeling of non-functional properties of component-based software systems. They consider services to be implemented by components; this view is applied in this thesis. They define QML/CS (Quality-Modeling Language for Component-based Systems), an specification language for modeling NFPs of components (Zschaler, 2010). Previously, in (Röttger and Zschaler, 2006), a methodology for the refinement of measurement-based non-functional requirements by means of model transformations was presented. OCL-annotated UML diagrams are used in QML/CS, supported by a formalism based on extended temporal logic of actions (Lamport, 2002). Their work uses the concepts of *measurement* for representing NF dimensions of a system. It can be argued that, although appropriate for some NFPs such as throughput, performance, etc., measurements are not practical for other aspects like security characteristics, for instance. Policy-based approaches, as the one presented here, better suit this situations. Nonetheless, both approaches can be complementary, and measurement-based and policy-based techniques can be used together to address different NFPs.

A taxonomy for the description of non-functional requirements of service oriented development is presented in (Galster and Bucherer, 2008). The different NFRs are divided into process, external, and service requirements. (Aier et al., 2007) proposes an extension to the OWL-S ontology for service lifecycle management and QoS (based on OMG's QoS UML profile (OMG, 2008d)). An equivalent approach is taken in (Dobson et al., 2007) for non-functional requirements under the OWL-S ontology to support negotiation and compliance monitoring of Service Level Agreements (SLAs).

The Common Non-Functional Aspects (NFA) Ontology is introduced in (Kabilan et al., 2007), to provide an integrated perspective of non-functional aspects, aiming at conceptualizing the domain of interest (to enhance semantic enterprise interoperability). The authors define various specialized sub-ontologies: trust, business contract risk, threat and misuse, and privacy ontologies. Kagal proposed authorization and privacy extensions first for DAML-S (Denker et al., 2003) and then for OWL-S ontology (Kagal et al., 2004), expressing related rules as Rei policies (Kagal, 2002). Again, the focus of these proposals is on description and matchmaking of services.

Non-functional properties of web services are addressed in (Toma et al., 2008) by extending WSMO/WSML (Web Services Modeling Ontology/Language). This work also provides a survey of ontological non-functional properties approaches. A comparison of OWL and WSMO is provided in (Lara and Roman, 2004).

Security is, again, the NF concern addressed in (Carminati et al., 2006). In this work, the authors propose a method for expressing security constraints, and a brokered architecture, to build web services compositions according to some security requirements. After agreeing on an

OWL extension to define a common language for security characteristics, SAML (OASIS Web Services Security (WSS) TC, 2006) attribute statements are used to indicate security capabilities. Security constraints are expressed as boolean formulas in disjunctive normal form. In (Carminati et al., 2006), the matchmaking algorithm is expressed, and the required architecture is presented. Two services in the architecture are noteworthy: the Secure Capability Authority, certifying the different web services' security capabilities, and the Secure Web Services Broker (SWS-Broker), composed by the Security Matchmaker, in charge of pruning the candidate services according to the security constraints.

The approach in (Carminati et al., 2006) shares some correspondences with the one presented in this thesis. First, ontologies are proposed in both approaches to agree on a terminology for non-functional requirements. In this thesis, the ontology is proposed for the matchmaking of the models in the repositories. Later, both express security capabilities, (Carminati et al., 2006) in the shape of SAML assertions, here as WS-SecurityPolicy with preference attributes. Finally, the mechanisms in both approaches can be extended to incorporate other NFPs outside from security.

Nevertheless, some differences arise. The approach in (Carminati et al., 2006) is mostly focused on service compositions and closely tied to a particular architecture. The one in this dissertation, however, primarily addresses the modeling and development of non-functional aware atomic services. Compositions can be later created from them. Moreover, this thesis assumes the existence of no particular architecture. Deployment of the services, and the use of patterns such as enterprise service buses (ESBs) or security-as-a-service (SAAS) are left for future lines of work.

The NRL ontology, a very thorough security extension to the OWL ontology, is defined in (Kim et al., 2005a; Kim et al., 2005b; Kim et al., 2007) for the description of resources. Designed to facilitate automatic discovery and invocation, the ontology is accompanied by a matchmaking algorithm that takes into account not just the concepts, but also the properties of the concept. The ontology is very complete, scoping from high-level security objectives, through security credentials, to technical encryption algorithms.

As defined in chapters 3 and 4, and exemplified in chapter 5, the use of ontologies is proposed for the CIM level to specify non-functional intents in business models, and to leverage its matching mechanisms for identifying models in the repositories that may satisfy such requirements.

6.4.4 Author's Publication on the Subject

On the subject of model-driven development of service-oriented systems, the publications in which the author of this thesis has participated include: (Silva Gallino et al., 2009b; Silva Gallino et al., 2010; Silva Gallino et al., 2011; Silva Gallino, Juan Pedro and de Miguel, Miguel and Briones, Javier F. and Alonso, Alejandro, 2011; de Miguel, Massonet, Silva Gallino and Briones, 2008).

6.5 Summary of the Chapter

This chapter has presented the most relevant related work in the different areas that fall within the scope of this thesis: NF properties' modeling, non-functional aware methodologies and development processes, analysis of models, and fundamentally, modeling of service-oriented systems.

The different contributions provided by this thesis in these various areas have been enumerated. From them, it may be concluded that the approach brings something new to the table of the modeling of NF-aware SOAs. The benefits that may be obtained from the use of this approach, which is not exclusive and may even be made compatible with others, makes it attractive to be adopted in different industries.

Next chapter, chapter 7, concludes this memoir by elaborating on the different conclusions and findings that may be derived from the work carried out in this thesis.

Chapter 7

Summary & Conclusions

7.1 Summary and Major Contributions

This dissertation has presented work towards the achievement of an integrated, model-driven solution for the development of service-oriented architectures with non-functional awareness. A particular instantiation of the approach for security policies and access control support was described, and a well-known use case has been included.

Standardization organizations have already proposed standards to for service-oriented modeling. However, these standards do not address non-functional characteristics of the systems. OMG's Service oriented architecture Modeling Language (SoaML) standard (OMG, 2009), for instance, explicitly focuses on the functional characteristics of services, and leaves the NF concerns out of the scope. The modeling of non-functional concerns of services is a subject still under study, which standardization bodies consider still not mature/stable enough.

In this work, different abstraction levels were addressed: from very abstract non-functional intents, through a platform-independent expression of policies, towards a final implementation of technical aspects of software. High to medium abstraction level mappings (business requirements, or CIM to PIM), and PIM to PSM mappings were defined for this purpose.

This memoir presents a model driven approach to service-oriented systems development. The approach, as presented in the previous chapters, spans over multiple areas of research. It proposes the use of domain-specific modeling languages to independently describe different non-functional concerns and then generate, by a combination of compositions and transformations, a complete model of the system on which to perform analyses and metric calculations to evaluate the suitability of architectural designs related to extra-functional properties. The defined approach follows the abstract-to-concrete methodology proposed by MDA, and supports its application on inputs either at the CIM or PIM levels.

At the CIM level, a process model annotated with non-functional intentions may be used to drive the selection of the appropriate artifacts for the PIM level. Ontologies have in the last years risen, noticeably with the advent of the semantic web, as useful tools to agree on a common vocabulary for the description and discovery of elements (web services, for instance).

The use of an ontology for categorization at the CIM level provides not only means to define a common language for security intents, but also represents a tool for the matchmaking of the non-functional concerns models contained in the repositories that may fulfill the expressed requirements. Other technical results, such as taxonomies classifying algorithm strengths, allow the possibility to add a selection mechanism that maintains the abstraction level of the intents, while still supporting an automated mapping to implementation mechanisms. By the same token, UML profiles represent

the selected mechanisms to incorporate additional information on functional models that may be used as inputs at the PIM level. An example of the application of the approach, generating executable artifacts and standard service descriptors, is also provided.

Chapter 4 has presented a particular implementation of the defined methodology, for the particular field of security and access control. All the different metamodels used for the implementation have been presented. Among them, two stand out: the intermediate metamodel, capable of containing all necessary information for generating the resulting artifacts, and the WS-Policy (and WS-SecurityPolicy) metamodel, allowing the definition and validation of assertions, and the expression of security capabilities.

This implementation is applied in chapter 5 to the well known WS-I's Supply Chain Management (SCM) use case. The SCM use case was originally developed to demonstrate interoperability in action. Sample applications serve as working examples for developers looking to follow the WS-I guidelines in their programming environment of choice. WS-I's SCM use case represents a perfect example to demonstrate the usefulness of the approach defined in this thesis.

The results are promising regarding the objective of achieving an integrated, model-driven solution. The implemented prototype provides a set of functionalities not always found together in one unique tool. The proposed solution is not tied to any implementation or target technology, allowing for great flexibility in its evolution regarding the appearance of new standards or technologies. This is an improvement compared to editor-based approaches, tightly coupled to the technologies they have been developed for.

The dissertation presents, in section 4.6, an approach for the evaluation of software architectures, illustrating on how analyses can be performed on an integrated metamodel for NF characteristics present in the model. An example on the use of this technique for the evaluation of safety-critical software with metrics for the evaluation of safety/cost trade-off is provided in (de Miguel, Briones, Silva Gallino and Alonso, 2008). In brief, the measurement techniques presented in this memoir may be found useful for software development and architectural evaluation in general.

7.1.1 Contributions

As first described in this dissertation's introduction, this thesis provides a variety of contributions to the field of the model-driven development of the non-functional concerns of service-oriented architectures. These contributions are recalled in this section.

Theoretical Contributions

The theoretical contributions of this thesis include:

- An state of the Art review of SOA/WS Modeling. [Section 6.4, and (Silva Gallino, 2006)]
- An state of the Art review of Non-Functional Concerns Modeling. [Section 6.1 and (Silva Gallino, 2008)]
- The definition of an analysis and a metric for iMM models. [Section 4.6]
- The definition of a WS-Policy Metamodel. [Section 3.3.4]
- The definition of a WS-SecurityPolicy Metamodel. [Section 4.3.5]
- The definition of an integrated metamodel (iMM) for the particular case of security and access control in SOA/WS systems. [Section 4.3.6]
- An state of the Art review of Model Composition Tool Support. (Silva Gallino et al., 2009a)

Methodological Contributions

With regard of the methodological contributions made by this thesis, those include:

- The definition of a methodology incorporating concepts extracted from MDA, MDSoc, AOM, and DSM for the inclusion of NF properties in the modeling of SOA/WS. [Section 3.2.2]
- The use of high level abstract concepts and ontologies for the discovery an selection of NF models. [Section 4.3.1]
- The combination of functional and non-functional models into a complete system model. [Section 3.2.2]
- The generation of intermediate models (through model transformations) for development and analysis. [Sections 3.2.2 and 4.6]

Practical Contributions

Finally, with respect to practical contributions, this thesis provides three:

- A prototype tool for the model-driven development of SOA/WS with regard of non-functional concerns. [Section 4.7]
- The implementation of the SoaML (OMG, 2009) UML profile [Section 3.3.2 and (Silva Gallino, 2007)].
- The implementation of the metamodels listed in sections 3.3 and 4.3.

7.2 Discussion

This dissertation has addressed several elements in order to achieve a solution that would satisfy the different objectives placed on this work. The proposed solution is considered to adequately fulfill the demands posed in section 1.2.

In the inception of this thesis, it was identified that there exists a lack of a design solution that:

- Allows an independent development of the functional and non-functional concerns of service-oriented architectures. A mechanism that allows for a separation of NF (e.g., security) modules for secrecy or privacy reasons. A mechanism that would allow reuse of expertise in the shape of models.
- Permits that each concern be addressed in a convenient manner, fostering understandability.
- Is modular enough so that the functional model is not polluted with non-functional information, leveraging reusability.
- Provides a practical way of measuring and analyzing the system for non-functional characteristics in early stages of the development, supporting validation of design.

This section enumerates the different goals and discusses them individually.

Independent development of concerns in SOAs The development process presented in chapter 3 promotes the individual management of the different concerns. This becomes evident in the example in chapter 5, where the mappings of different concern models are first defined, and the complete system is posteriorly obtained by deriving values for each element from the IMM model.

Appropriate notation for individual concerns The use of multiple domain-specific models, each one independently addressing one (or more) non-functional concern, favors that each concern be addressed in a convenient manner. These languages, as mentioned in the introduction, reduce the available modeling elements to key, familiar problem-domain concepts (and rules), narrowing down the design space, to achieve higher levels of productivity (Kelly and Tolvanen, 2008).

Modularization and independence of data The application of AOM (Aspect-Oriented Modeling) and MDSoc (Multi-Dimensional Separation of Concerns) ideas increase the modularity of the approach. Additionally, the solution permits that each concern design, functional or not, is not polluted with extraneous information. The management of the individual concern's artifacts (e.g., update, corrections, mapping to novel target platforms, definition of modeling rules, etc.) is performed independently. Such separation of designs supports the objective of secrecy/privacy of design placed on the thesis.

Measurement and analysis adequacy Integrated metamodels, such as iMM, provide a perfect subject of analysis and calculations of metrics. In the particular case of the selected example, an access control conflicts analysis, and a requirements addressing metrics were defined. Such analysis and metrics take advantage of the information already available in the model.

General discussion of the approach Thanks to the application of model-driven techniques, the separation into multiple independent concerns, and the use of domain-specific vocabulary, the proposed approach provides, among other benefits, a greater understanding of the system as a whole and a platform-independent development. This improves the reusability of designs, and simplifies the evolution of the system, thus increasing productivity. Moreover, (semi) automatic generation of artifacts speeds prototyping and, later, the generation of repetitive production code, relieving developers from tedious, error-prone tasks.

As an additional merit, not part of the original objectives, the proposed solution presents great flexibility in its evolution regarding the appearance of new standards or technologies, as it is not tied to any implementation or target platform, nor closed in its support for NF characteristics. New concern models can be defined and incorporated to the model any time.

The use of hand made composition (weaving) models would not scale well for its application on large system models. Therefore, the approach proposes the automatic generation of pre-filled composition models. The current state of the prototype tool still does not incorporate this element, but it has already been designed, and is planned for future versions of the prototype.

One shortcoming that can be argued to the proposed approach is that it is limited, in each implementation, to what can be expressed with the intermediate metamodel. Although true, the intermediate metamodel may be defined in a generic manner, addressing this shortcoming.

In the particular implementation, focused on security and access control, the iMM is based on generic parts (CBSD, generic access control metamodel, service policies), which makes its spectrum broad enough for the objectives of security and access control that were placed on it.

Moreover, the WS-Policy framework itself has been defined in a generic fashion, allowing for many standards to be formulated based on it. Any new standard that is defined under its umbrella will automatically be supported by this solution, and assertions defined in them can readily be applied with this approach. Thanks to this, all policy standards already available or in process of being defined (e.g., WS-Addressing (Box et al., 2004), WS-RM Policy (Davis et al., 2005), WS-AtomicTransaction (OASIS, 2006)) are capable of being represented by the iMM. This permits the use of the approach (the particular implementation) for developing not only security-aware services, but also including timing constraints, secure exchange, transactions, etc. The use of generic

standard solutions, such as WS-Policy, supports the design of solutions with a extensive application spectrum.

7.3 Future Lines of Work

With respect to future work, in the short term it will be focused on finishing the implementation of the prototype, mostly regarding the automatic selection of appropriate PIM-level models addressing the non-functional concerns expressed in the requirements.

Some work has been done to integrate Eclipse's CDO (Connected Data Objects) (Eclipse, 2011a) repositories within the prototype. An extension of CDO to support semantic capabilities would be necessary. The "emftriple" (Guillaume Hillairet, n.d.) project is an effort towards the development of a repository for EMF models with support for semantic annotations, and could represent a possible alternative to be integrate in the prototype. Reusable assets (Reusable Modeling Tools Assets (de Miguel et al., 2011)), IBM's Reusable Assets Specification (RAS) repositories also provide a great alternative for hosting models annotated with categorization metadata.

Other high-level approaches addressing the expression of non-functional requirements, such as the NFR (Non-Functional Requirements) framework (Chung et al., 2000) (appearing repeatedly in literature in the field of non-functional characteristics research) represent an plausible alternative to business processes. The NFR framework can even be placed into a higher level or previous step, and be mapped into the CIM workflow. Mapping rules need to be defined accordingly.

The automatic generation of baseline weaving models, providing a tentative composition of the functional and non-functional concerns is also one of the objectives in the future. If the functional model at the PIM level is derived from the process model at the CIM level, or even if it is provided to the tool, a matching between the actors (services) in the CIM model, and the services defined at the PIM model can be found. Once a mapping is found, the intents in the CIM model indicate where (to which service interfaces) the particular NFPs must be applied. This composes the weaving model. Such automatic generation of a tentative composition model is considered feasible.

In a longer term, it is planned to evaluate the expression of other policy aspects under standardization process in the implemented IMM. As previously mentioned, the approach is flexible enough to support any WS-Policy-based standard, but the implementation is not able to use the information intrinsic to the assertion for its benefit (shaping of code generation, configuration of target platform, generation/use of an extra-functional aspect, e.g.).

Finally, one attractive and different line of work derived from the experience acquired in this project, and inspired in Domain Specific Multi-Modeling (Boronat et al., 2009; Hessellund, 2009), would be shifting from an approach where the generation is based on an integrated metamodel, towards one in which a it is based on weaving associations. In such an approach any metamodel could be used to generate artifacts based on a set of model-weaving relations. This approach, although more flexible in the sense that is not limited by what the intermediate metamodel supports, suffers from other drawbacks, such as the the coordination problem (Hessellund, 2009), or its inadequacy to be the subject of analysis and measurements.

As a final epilogue, the belief of the author is that the work presented within this thesis, although leaving room for further improvement, provides developers with sound, productive, and useful means for the design of service-oriented software systems with non-functional awareness, that may wishfully result in the improvement of the quality of the implementations.

Appendices

Appendix A

Input and Composition Models for WS-I's SCM Use Case

Figure A.1: Warehouse Service's SoaML Model

Figure A.2: Composition of Retailer's QoS and UML Models

Figure A.3: Composition of Retailer's SecureUML and UML Models

Figure A.5: Resulting Retailer's iMM Model

Appendix B

Transformation Templates

B.1 iMM2WSDL.AtlanZoo.doc.lit.wrapped.atl

134

Listing B.1: *iMM2WSDL.AtlanZoo.doc.lit.wrapped.atl*

```
1  --@atlcompiler atl2006
2  module iBuilderMM2WSDL_AtlanZoo; -- Module Template
3  create OUT : WSDL_AtlanZoo from IN : iBuilderMM;
4
5  -----
6  -----
7  helper def: targetNS : String = 'http://iBuilder.generation.by.upm';
8  helper def: schemaFolder : String = '../schemas/';
9  helper def: XSDElements : Set(WSDL_AtlanZoo!XSDElement) = Set{};
10
11 helper def: mapTypes(type : iBuilderMM!Type) : String =
12     if( type.ocIsTypeOf(iBuilderMM!ComplexType)) then type.name
13     else if( type.package = 'XSDDataTypes' ) then 'xsd:'+type.name
14     else if( type.name = 'String' ) then 'string'
15     else if( type.name = 'Integer' ) then 'int'
16     else if( type.name = 'Boolean' ) then 'boolean' else type.name
17 endif endif endif endif endif;
18
19 helper def: findPart(setPo : Set( iBuilderMM!Parameter), elem : WSDL_AtlanZoo!XSDElement, messName : String) :
20     WSDL_AtlanZoo!Part =
21         let elemName : String = if messName.endsWith('Request') then 'Body' else messName
22         endif in
23         let complex : Boolean = if setPo->size() > 1 then true else false endif in
24         if( WSDL_AtlanZoo!Part->allInstancesFrom('OUT')
25             ->exists(a | a.name = elemName and a.element = elem))
26         then WSDL_AtlanZoo!Part->allInstancesFrom('OUT')
27         ->select(a | a.name = elemName and a.element = elem)->first()
28         else thisModule.getPart(setPo, elem, messName) endif;
29
30 helper def: findPolicyApplication( setPo : Set( iBuilderMM!Parameter) ) : Set( iBuilderMM!Policy) =
31     if( setPo->exists(e | (e.appliedPolicy->size() > 0) ) )
32     then setPo->select(e | (e.appliedPolicy->size() > 0) )
33     else Set {} endif;
34
35 helper def: findElement(setPo : Set(iBuilderMM!Parameter), elemName : String, recursive : Boolean) :
36     WSDL_SoloDesc!XSDElement =
37         let complex : Boolean = if setPo->size() > 1 then true else false endif in
38         if( WSDL_AtlanZoo!XSDElement->allInstancesFrom('OUT')->exists(a | a.name =
```

```

36         elemName )
37     then let elemento : WSDL_AtlanZoo!XSDElement = WSDL_AtlanZoo!XSDElement->
        allInstancesFrom( 'OUT' )
38                                     ->select( a | a.name = elemName )->
                                     first() in
39     if( elemento.subElements = setPo->reject( a | a.name = elemName )->
        collect( e |
40         if (not e.oclIsUndefined())
41         then thisModule.findElement( Set{e}, e.name, false )
42         else e endif )->flatten()->asSequence()->excluding(
        OclUndefined)
43     )
44     then elemento else thisModule.findElement(setPo, elemName ,
        recursive) endif
45 else thisModule.findElement(setPo, elemName , recursive) endif;
46
47 ----- ENTRYPOINT RULE -----
48
49 ----- called ENTRYPOINT rule firstThingsFirst -----
50 entrypoint rule firstThingsFirst()
51 {
52     do{
53         let c : String = 'Running iBuilder2WSDL_AtlanZoo Transformation' in c.println();
54         iBuilderMM!System.allInstancesFrom( 'IN' )->first();
55         thisModule.XSDElements <- thisModule.XSDElements->including(
56             iBuilderMM!Operation.allInstancesFrom( 'IN' )->collect( e |
57                 if e.parameters->flatten().asSet()->select( e | e.direction.toString() = 'in' )->
                    flatten()->asSet().notEmpty()
58                 then thisModule.findElement( e.parameters->flatten().asSet()->select( e | e.
                    direction.toString() = 'in' )
59                                     ->flatten()->asSet(), e.name
60                                     , true)
61                 else OclUndefined endif )->flatten()->asSet()->excluding(OclUndefined);
62         thisModule.XSDElements <- thisModule.XSDElements->including(
63             iBuilderMM!Operation.allInstancesFrom( 'IN' )->collect( e |
64                 if e.parameters->flatten().asSet()->select( e | e.direction.toString() = 'out' or e
65                     .direction.toString() = 'return' )
66                                     ->flatten()->asSet().
                    notEmpty()
67                 then thisModule.findElement( e.parameters->flatten().asSet()->select( e |
                    e.direction.toString() = 'out' or e.direction.toString() = 'return' )
                    ->flatten()->asSet(), e.name+'Response', true)

```

```

67         else OclUndefined endif))->flatten()->asSet()->excluding(OclUndefined);
68     thisModule.XSDElements <- thisModule.XSDElements->including( thisModule.getElement(Set{}, 'Empty'
69         , false)
70         )->flatten()->asSet()->excluding(OclUndefined);
71     thisModule.XSDElements;
72 }
73 }
74
75 -----
76 -----
77 -----
78 rule System
79 {
80     from e : iBuilderMM!System
81     to out : WSDL_AtlanZoo!System
82     (
83         name <- e.name,
84         typesDeclaration <- thisModule.newTypesDeclaration(),
85         services <- e.ports,
86         targetNamespace <- e.baseNS --thisModule.targetNS
87     )
88 }
89
90 -----
91 abstract rule PolicySubjectType
92 abstract rule PolicySubjectType {
93     from e : iBuilderMM!PolicySubjectType
94     to out : WSDL_AtlanZoo!PolicySubject
95     (
96         policyReference <- if( (e.appliedPolicy->size() > 0) )
97         then e.appliedPolicy->collect(a | thisModule.newPolicyReference(a) )
98         else Sequence{} endif
99     )
100 }
101
102 -----
103 rule Service
104 rule Service extends PolicySubjectType
105 {
106     from e : iBuilderMM!Port
107     to out : WSDL_AtlanZoo!Service
108     (
109         name <- e.name,
110         system <- iBuilderMM!System.allInstancesFrom('IN')->first(),
111         ports <- thisModule.newPort(e)
112     )
113 }

```



```

108 }
109
110 rule PortType
111 rule PortType extends PolicySubjectType
112 {
113     from e : iBuilderMM!Interface
114     to   out : WSDL_AtlanZoo!PortType
115         ( name <- ( e.name+'_PortType' ),
116           system <- iBuilderMM!System.allInstancesFrom( 'IN' )->first(),
117           operations <- e.operations
118         )
119 }
120
121 rule Operation
122 rule Operation extends PolicySubjectType
123 {
124     from e : iBuilderMM!Operation
125     using { setPa : Set(iBuilderMM!Parameter) = e.parameters->flatten().asSet(); }
126     to out : WSDL_AtlanZoo!Operation
127     (
128         name <- e.name,
129         type <- e.interface,
130         input <- thisModule.getInputMessage( setPa->select(a | a.direction.toString() = 'in' )
131         ->flatten()->asSet(), e.name+'Request', out ),
132         output <- thisModule.getOutputMessage( setPa->select(a | a.direction.toString() = 'out'
133         or a.direction.toString() = 'return' )->flatten()->asSet(), e.name+'Response', out )
134     )
135 }
136
137 Abstract Rule Type
138 abstract rule Type
139 {
140     from e : iBuilderMM!Type
141     to   out : WSDL_AtlanZoo!Type
142         ( name <- thisModule.mapTypes(e) )
143 }
144
145 Rule SimpleType
146 rule SimpleType extends Type
147 {
148     from e : iBuilderMM!PrimitiveType
149     to   out : WSDL_AtlanZoo!SimpleType
150         ( baseType <- if(not e.restriction.oclIsUndefined())

```

```

150             then e.restriction.baseType else OclUndefined endif,
151             restriction <- if(not e.restriction.ocIsUndefined())
152             then e.restriction.restrictions->collect(a | thisModule.getRestrictions(a))
153                     else Set{} endif
154         )
155     }
156
157     _____ unique lazy Rule getRestrictions _____
158     unique lazy rule getRestrictions
159     {
160         from e : iBuilderMM!BasicMapEntry
161         to out : WSDL_AtlanZoo!BasicMapEntry
162         (
163             name <- e.name,
164             value <- e.value
165         )
166     }
167
168     _____ Rule ComplexType _____
169     rule ComplexType extends Type
170     {
171         from e : iBuilderMM!ComplexType (e.ocIsTypeOf(iBuilderMM!ComplexType))
172         to out : WSDL_AtlanZoo!ComplexType
173         (
174             property <- e.properties->collect(i | i.type)—thisModule.getElement(Set{i}, i.name, false)) )
175     }
176
177     _____ CALLED RULES _____
178
179     —called rule newPolicyReference
180     rule newPolicyReference(dest : iBuilderMM!Policy )
181     {
182         to out : WSDL_AtlanZoo!PolicyReference
183         (
184             URI <- dest.document.location+'#'+dest.wsuId )
185         do {out;}
186     }
187
188     _____ called rule newPort _____
189     rule newPort(port : iBuilderMM!Port)
190     {
191         to out : WSDL_AtlanZoo!Port
192         (
193             name <- port.name,
194             binding <- port.exportedInterfaces ->collect(e | thisModule.newBinding(e, port.name))->

```

```

191         flatten()->asSet()
192     do { if ( port.appliedPolicy->size() > 0 )
193         { out.policyReference <- port.appliedPolicy->collect(a | thisModule.newPolicyReference(a) );
194         }
195     }
196
197     ----- called rule newBinding() -----
198     rule newBinding(interf : iBuilderMM!Interface , portName : String)
199     {
200         to out : WSDL_AtlanZoo!Binding
201         ( name <- ( portName+'_Binding' ) ,
202           type <- interf ,
203           system <- iBuilderMM!System.allInstancesFrom( 'IN' )->first() ,
204           bindingProtocol <- thisModule.newSoapBinding() ,
205           operations <- interf.operations ->collect(e | thisModule.newBindingOperations(e) )->flatten()
206           ->asSet()
207         )
208         do { out; }
209     }
210
211     ----- called rule newSoapBinding -----
212     rule newSoapBinding()
213     {
214         to out : WSDL_AtlanZoo!BindingProtocol
215         ( name <- 'SOAP' ,
216           protocol <- 'http://schemas.xmlsoap.org/wsdl/soap/' ,
217           style <- 'document' ,
218           use <- 'literal' ,
219           transport <- 'http://schemas.xmlsoap.org/soap/http' ,
220           wrapped <- true
221         )
222         do { out; }
223     }
224
225     ----- called rule newTypesDeclaration -----
226     rule newTypesDeclaration()
227     {
228         to out : WSDL_AtlanZoo!TypesDeclaration
229

```

```

230         ( elements <- thisModule.XSDElements->flatten()->asSet() ,
231           types <- iBuilderMM!Type.allInstancesFrom('IN') ,
232           schema <- thisModule.newXSDSchema() ,
233           import <- thisModule.newImport(out.schema.declaredNamespace)
234         )
235     do { out.import <- out.import->including(iBuilderMM!Type.allInstancesFrom('IN')
236       ->select(c | not c.package.oclIsUndefined())->select(b | b.package.
        startsWith('http:'))
        ->collect(a | thisModule.newImport(a.package)))->flatten();
237       out;}
238 }
239
240 ----- called rule newXSDSchema -----
241 rule newXSDSchema()
242 {
243     to out : WSDL_AtlanZoo!XSDSchema
244     (
245       targetNamespace <- thisModule.targetNS+'/types' ,
246       prefixDeclaration <- 'xmlns:xsd1' ,
247       declaredNamespace <- 'http://'+WSDL_AtlanZoo!System.allInstancesFrom('OUT')
248         ->first().name.toLowerCase()+'/types'
249     )
250     do { out; }
251 }
252
253 ----- unique lazy rule newImport -----
254 unique lazy rule newImport
255 {
256     from dns : String
257     to out : WSDL_AtlanZoo!Import
258     (
259       namespace <- dns ,
260       location <- if(dns.endsWith('.xsd')) then thisModule.schemaFolder+dns else thisModule.
        schemaFolder+dns+'.xsd' endif
261     )
262     do { out; }
263 }
264
265 ----- called rule newBindingOperations() -----
266 rule newBindingOperations(op : iBuilderMM!Operation)
267 {
268     to out : WSDL_AtlanZoo!BindingOperation
269

```

```

270         (   name <- op.name,
271           binding <- op
272         )
273     do { out; }
274 }
275
276 ----- called rule getInputMessage -----
277 rule getInputMessage(setPo : Set(iBuilderMM!Parameter),
278                     messName : String, op : WSDL_AtlanZoo!Operation )
279 {
280     using { elemName : String = if setPo->isEmpty() then 'Empty' else if messName.endsWith('Request')
281                               then messName.regexReplaceAll('Request', '') else messName endif endif; }
282     to out : WSDL_AtlanZoo!Input
283     (
284         name <- messName,
285         element <- thisModule.XSDElements->select(e | e.name = elemName )->first(),
286         parts <- thisModule.findPart(setPo, out.element, messName ),
287         operation <- op
288     )
289     do { elemName;
290         if (thisModule.findPolicyApplication(setPo)->size() > 0)
291         { out.policyReference <- thisModule.findPolicyApplication(setPo)
292           ->collect(e | thisModule.newPolicyReference(e));
293         }
294         out;}
295 }
296 ----- called rule getOutputMessage -----
297 rule getOutputMessage(setPo : Set(iBuilderMM!Parameter),
298                     messName : String, op : WSDL_AtlanZoo!Operation )
299 {
300     using { elemName : String = if setPo->isEmpty() then 'Empty' else messName endif; }
301     to out : WSDL_AtlanZoo!Output
302     (
303         name <- messName,
304         element <- thisModule.XSDElements->select(e | e.name = messName)->first(),
305         parts <- thisModule.findPart(setPo, out.element, messName ),
306         operation <- op
307     )
308     do { if (thisModule.findPolicyApplication(setPo)->size() > 0)
309         { out.policyReference <- thisModule.findPolicyApplication(setPo)
310           ->collect(e | thisModule.newPolicyReference(e));
311         }
312         out;}

```

```

312 }
313
314 ----- called rule getElement -----
315 rule getElement(setPo : Set(iBuilderMM!Parameter), elemName : String, recursive : Boolean)
316 {
317     using { complex : Boolean = if setPo->size() > 1 then true else false endif; }
318     to out : WSDL_AtlanZoo!XSDElement
319         ( name <- elemName )
320         do { if ( recursive )
321             { out.subElements <- setPo->collect( e |
322                 if (not e.ocIsUndefined())
323                 then thisModule.findElement( Set{e}, e.name, false )
324                 else e endif )->flatten()->asSequence()->excluding(OclUndefined);
325             }
326             if ( (not setPo->isEmpty()) and (not recursive) )
327             {
328                 if ( not setPo->asSequence()->first().type.ocIsUndefined() )
329                 { out.type <- thisModule.mapTypes(setPo->asSequence()->first().type); }
330             }
331         out;}
332 }
333
334 ----- called rule getPart -----
335 rule getPart(setPo :Set( iBuilderMM!Parameter), elem : WSDL_AtlanZoo!XSDElement, messName : String)
336 {
337     using { elemName : String = if messName.endsWith('Request') then 'Body' else messName endif;
338             complex : Boolean = if setPo->size() > 1 then true else false endif; }
339     to out : WSDL_AtlanZoo!Part
340         ( name <- elemName,
341           element <- if not setPo->isEmpty()
342                       then elem
343                       else thisModule.XSDElements->select(a| a.name = 'Empty')->first() endif,
344           type <- if not setPo->isEmpty()
345                   then setPo->asSequence()->first().type
346                   else OclUndefined endif
347         )
348     do { out; }
349 }
350

```

B.2 appContext.mt

Listing B.2: *appContext.mt*

```

1  <% metamodel http://ibuilder.indra.es/iMM import propertiesScripts import services.IMMServices %>
2  <%script type="iMM.System" name="appContext" file="<%name%>Project/src/main/resources/appContext.xml"%>
3  <?xml version="1.0" encoding="UTF-8"?>
4  <beans xmlns="http://www.springframework.org/schema/beans"
5  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/
   context"
6  xmlns:jee="http://www.springframework.org/schema/jee" xmlns:jaxws="http://cxf.apache.org/jaxws"
7  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/jee
8  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
9  http://www.springframework.org/schema/jee/spring-jee-2.5.xsd http://www.springframework.org/schema/context
10 http://www.springframework.org/schema/context/spring-context-2.5.xsd
11 http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd"
12     default-dependency-check="none" default-lazy-init="false">
13 <%if ( service::hasRPCEndpoints() || script::rpcClients){%>
14     <import resource="classpath:META-INF/cxf/cxf.xml" />
15     <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />
16     <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />
17 <%}%>
18 <%if (!service::hasRPCEndpoints()){%> <%-- SHOULD CHECK PER WSDL IN A FOR --%>
19     <!-- WSDL DECLARATIONS -->
20     <%script::wsdlDeclarations%>
21     <!-- ENDPOINT MAPPINGS -->
22     <%script::EndpointMapping%>
23 <%}%>
24     <!-- SERVICES -->
25     <%script::serviceEndpoints%>
26 <%if (script::hasClients){%>
27     <!-- CLIENTS -->
28     <%script::Clients%>
29 <%}%>
30     <!-- (UN)MARSHALLER -->
31     <bean id="marshaller" class="org.springframework.oxm.jaxb.Jaxb2Marshaller">
32         <property name="contextPaths">
33             <list>
34 <%for (implementations){%>
35                 <value><%package%></value>
36 <%}%>
37             </list>

```

```

38         </property>
39     </bean>
40 </beans>
41
42 <%script type="iMM.System" name="serviceEndpoints" post="trim()" %>
43 <%for (implementations){%>
44     <%for (elements.filter("Service")){%>
45         <%for (implementedInterface){%>
46 <!-- <%name%> -->
47         <%if (current(1).style.equalsIgnoreCase("RPC")){%> <!-- CURRENT = THE SERVICE -->
48 <jaxws:endpoint id="<%name%>Endpoint" address="<%current(2).url%" wsdlLocation="/WEB-INF/<%name%>Impl.wsdl"> <%
    -- CURRENT = THE SERVICE -->
49     <jaxws:implementor>
50         <ref bean="<%current(3).name%>Bean" /> <!-- CURRENT = THE IMPLEMENTATION -->
51     </jaxws:implementor>
52 </jaxws:endpoint>
53     <%}else{%>
54 <bean id="<%name%>Endpoint" class="<%current(3).package%>.<%name%>Endpoint"> <!-- CURRENT = THE IMPLEMENTATION --
    %>
55     <property name="<%current(2).name%>Service" ref="<%current(3).name%>Bean" /> <!-- CURRENT = THE SERVICE -->
56     <property name="marshaller" ref="marshaller" />
57     <property name="unmarshaller" ref="marshaller" />
58 </bean>
59     <%}%>
60     <%}%>
61     <%}%>
62
63 <bean id="<%name%>Bean" class="<%package%>.<%name%>">
64     <%for (elements.filter("Service")){%>
65         <%for (remoteServices){%>
66     <property name="<%name%>Stub" ref="<%name%>Client" />
67         <%}%>
68     <%}%>
69 </bean>
70 <%}%>
71 <%script type="iMM.System" name="wsdlDeclarations" post="trim()" %>
72 <%for (implementations){%>
73     <%for (elements.filter("Service")){%>
74         <%for (implementedInterface){%>
75 <bean id="<%name%>WSDL" class="org.springframework.ws.wsdl.wsdl11.SimpleWsdl11Definition">
76     <constructor-arg value="/WEB-INF/<%name%>Impl.wsdl" />
77 </bean>

```



```

78         <%}%>
79     <%}%>
80 <%}%>
81 <%script type="iMM.System" name="Clients" post="trim()" %>
82 <%for (implementations){%>
83     <%for (elements.filter("RemoteService")){%>
84 <!-- <%name%> Client -->
85     <%if (style == "RPC"){%>
86 <jaxws:client id="<%name%>Client"
87     serviceClass="<%current(2).package%>.<%name%>PortType" <%-- CURRENT = THE IMPLEMENTATION --%>
88     address="<%url%>" />
89     <%}else{%>
90 <bean id="<%name%>Client" class="<%current(2).package%>.<%name%>Client"> <%-- CURRENT = THE IMPLEMENTATION --%>
91     <property name="defaultUri" value="<%url%>" /> <%-- CURRENT = THE REMOTESERVICE --%>
92     <property name="marshaller" ref="marshaller" />
93     <property name="unmarshaller" ref="marshaller" />
94 </bean>
95     <%}%>
96 <%}%>
97 <%}%>
98 <%script type="iMM.System" name="EndpointMapping" post="trim()" %>
99 <bean class="org.springframework.ws.server.endpoint.mapping.UriEndpointMapping">
100     <property name="mappings">
101         <props>
102 <%for (implementations){%>
103     <%for (elements.filter("Service")){%>
104         <%for (implementedInterface){%>
105             <prop key="<%current(1).url%>"><%name%>Endpoint</prop> <%-- CURRENT = THE SERVICE --%>
106         <%}%>
107     <%}%>
108 <%}%>
109     </props>
110     </property>
111 <%if (interceptors){%>
112     <property name="interceptors">
113         <list>
114             <ref bean="manufacturerInterceptor" />
115         </list>
116     </property>
117 <%}%>
118 </bean>

```


Appendix C

Generated Files for WS-I's SCM Use Case

C.1 SecurityConnector.java

Listing C.1: *SecurityConnector.java*

```

1  package org.ws_i.sampleapplications.supplychainmanagement._2002_08.util;
2
3  import java.rmi.RemoteException;
4  import java.security.cert.X509Certificate;
5  import java.util.HashSet;
6  import java.util.Iterator;
7  import java.util.List;
8  import java.util.Set;
9
10 import org.apache.cxf.binding.soap.SoapMessage;
11 import org.apache.cxf.interceptor.Fault;
12 import org.apache.cxf.phase.Phase;
13 import org.apache.cxf.phase.PhaseInterceptor;
14 import org.apache.cxf.ws.security.wss4j.AbstractWSS4JInterceptor;
15 import org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor;
16 import org.apache.cxf.message.Message;
17 import org.apache.ws.security.WSSecurityEngineResult;
18 import org.apache.ws.security.handler.WSHandlerConstants;
19 import org.apache.ws.security.handler.WSHandlerResult;
20 import org.springframework.beans.factory.annotation.Autowired;
21 import org.springframework.security.Authentication;
22 import org.springframework.security.AuthenticationManager;
23 import org.springframework.security.context.SecurityContextHolder;
24 import org.springframework.security.providers.AuthenticationProvider;
25 import org.springframework.security.providers.preauth.
26     PreAuthenticatedAuthenticationToken;
27 import org.springframework.security.providers.x509.X509AuthenticationToken;
28 import org.ws_i.sampleapplications.supplychainmanagement._2002_08.
29     loggingfacility.LogEventRequestType;
30 import org.ws_i.sampleapplications.supplychainmanagement._2002_08.
31     loggingfacility.LoggingFacilityLogPortType;
32
33
34 public class SecurityConnector extends AbstractWSS4JInterceptor
35 {
36
37     private AuthenticationManager authenticationManager;
38     private org.springframework.security.providers.ProviderManager providerManager;

```

```

39
40     public AuthenticationManager getAuthenticationManager() {
41         return authenticationManager;
42     }
43
44     public void setAuthenticationManager(AuthenticationManager authenticationManager) {
45         this.authenticationManager = authenticationManager;
46     }
47
48     private LoggingFacilityLogPortType logStub = null;
49
50     public LoggingFacilityLogPortType getLogStub() {
51         return logStub;
52     }
53
54     public void setLogStub(LoggingFacilityLogPortType logStub) {
55         this.logStub = logStub;
56     }
57
58     public SecurityConnector()
59     {
60         super();
61         setPhase(Phase.PRE_PROTOCOL);
62         getAfter().add(WSS4JInInterceptor.class.getName());
63     }
64
65     public void handleMessage(SoapMessage message) throws Fault
66     {
67         System.out.println("Enters SECURITYCONNECTOR");
68         List<Object> results =
69             (List<Object>)message.get(WSHandlerConstants.RECV_RESULTS);
70         if (results == null) {
71             return;
72         }
73         for (Iterator iter = results.iterator(); iter.hasNext();) {
74             WSHandlerResult hr = (WSHandlerResult) iter.next();
75             if (hr == null || hr.getResults() == null) {
76                 return;
77             }
78
79             for (Iterator it = hr.getResults().iterator(); it.hasNext();)
80             {

```

```

81         WSSecurityEngineResult er = (WSSecurityEngineResult) it.next();
82
83         if (er != null && er.getCertificate() instanceof X509Certificate)
84         {
85             String principal = er.getPrincipal().getName();
86             if(principal.startsWith("CN="))
87                 principal = principal.substring(principal.lastIndexOf("=")+1);
88             System.out.println("Principal: "+principal);
89
90             Authentication authentication =
91                 new PreAuthenticatedAuthenticationToken(principal, er.getCertificate());
92
93             System.out.println("Starts authentication in SECURITYCONNECTOR");
94             authentication = authenticationManager.authenticate(authentication);
95             System.out.println("Principal: " + authentication.getPrincipal().toString()
96                 + " Authorities: " + authentication.getAuthorities());
97             SecurityContextHolder.getContext().setAuthentication(authentication);
98         }
99     }
100 }
101 }
102 }

```

C.2 WeekDaysConstraintVoter.java

Listing C.2: *WeekDaysConstraintVoter.java*

```

1  package org.ws_i.sampleapplications.supplychainmanagement._2002_08.util;
2
3  import org.springframework.security.Authentication;
4  import org.springframework.security.ConfigAttribute;
5  import org.springframework.security.ConfigAttributeDefinition;
6  import java.util.Arrays;
7  import java.util.Calendar;
8  import java.util.HashSet;
9  import java.util.Set;
10
11 public class WeekDaysConstraintVoter
12     implements org.springframework.security.vote.AccessDecisionVoter
13 {
14     private static Set<Integer> weekDays = new HashSet<Integer> (
15         Arrays.asList (Calendar.MONDAY, Calendar.TUESDAY,
16             Calendar.WEDNESDAY, Calendar.THURSDAY,
17             Calendar.FRIDAY));
18
19     public boolean supports(ConfigAttribute attribute)
20     { return false; }
21
22     @SuppressWarnings("unchecked")
23     public boolean supports(Class clazz)
24     { return true; }
25
26     public int vote(Authentication authentication, Object object, ConfigAttributeDefinition config)
27     {
28         int dayOfWeek = java.util.Calendar.getInstance().get(Calendar.DAY_OF_WEEK);
29         if ( dayOfWeek == java.util.Calendar.MONDAY ||
30             dayOfWeek == java.util.Calendar.TUESDAY ||
31             dayOfWeek == java.util.Calendar.WEDNESDAY ||
32             dayOfWeek == java.util.Calendar.THURSDAY ||
33             dayOfWeek == java.util.Calendar.FRIDAY )
34             return ACCESS_GRANTED;
35         else return ACCESS_DENIED;
36     }
37 }

```

C.3 SCM Use Case Application Context (Security)

152

Listing C.3: *Application Context (Security)*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:jee="http://www.springframework.org/schema/jee"
6       xmlns:jaxws="http://cxf.apache.org/jaxws"
7       xmlns:aop="http://www.springframework.org/schema/aop"
8       xsi:schemaLocation=
9         "http://www.springframework.org/schema/beans
10        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
11        http://www.springframework.org/schema/jee
12        http://www.springframework.org/schema/jee/spring-jee-2.5.xsd
13        http://www.springframework.org/schema/context
14        http://www.springframework.org/schema/context/spring-context-2.5.xsd
15        http://www.springframework.org/schema/aop
16        http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
17        http://cxf.apache.org/jaxws
18        http://cxf.apache.org/schemas/jaxws.xsd"
19       default-dependency-check="none" default-lazy-init="false">
20
21     <!-- SERVICES -->
22     <!-- Load the needed resources that are present in the cxf* jars -->
23     <import resource="classpath:META-INF/cxf/cxf.xml" />
24     <import resource="classpath:META-INF/cxf/cxf-extension-*.xml" />
25     <!-- <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" /> -->
26     <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />
27
28     <context:annotation-config />
29
30     <!-- WarehouseA -->
31     <jaxws:endpoint id="warehouseA" address="/warehouseA"
32       implementorClass="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
33         warehouse.WarehouseAService">
34       <jaxws:implementor>
35         <ref bean="warehouseAService" />
36       </jaxws:implementor>
37       <jaxws:outInterceptors>
38       <ref bean="WarehouseEndTimestampSign_Response" />
```



```

39     </jaxws:outInterceptors>
40     <jaxws:inInterceptors>
41         <ref bean="WarehouseEndTimestampSign_Request" />
42         <ref bean="SecurityConnector" />
43     </jaxws:inInterceptors>
44     </jaxws:endpoint>
45
46     <bean id="SecurityConnector"
47         class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.util.SecurityConnector">
48         <property name="logStub" ref="loggingClient" />
49         <property name="authenticationManager" ref="authenticationManager" />
50     </bean>
51
52     <bean id="warehouseAService"
53         class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
54             warehouse.WarehouseAService">
55         <property name="logStub" ref="loggingClient" />
56         <property name="manufacturerAService" ref="manufacturerAClient" />
57         <property name="manufacturerBService" ref="manufacturerBClient" />
58         <property name="manufacturerCService" ref="manufacturerCClient" />
59     </bean>
60
61     <!-- WarehouseB -->
62     <jaxws:endpoint id="warehouseB" address="/warehouseB"
63         implementorClass="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
64             warehouse.WarehouseBService">
65         <jaxws:implementor>
66             <ref bean="warehouseBService" />
67         </jaxws:implementor>
68         <jaxws:outInterceptors>
69             <ref bean="WarehouseEndTimestampSign_Response" />
70     </jaxws:outInterceptors>
71     <jaxws:inInterceptors>
72         <ref bean="WarehouseEndTimestampSign_Request" />
73         <ref bean="SecurityConnector" />
74     </jaxws:inInterceptors>
75     </jaxws:endpoint>
76
77     <bean id="warehouseBService"
78         class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.warehouse.WarehouseBService">
79         <property name="logStub" ref="loggingClient" />
80         <property name="manufacturerAService" ref="manufacturerAClient" />

```

```

81         <property name="manufacturerBService" ref="manufacturerBClient" />
82         <property name="manufacturerCService" ref="manufacturerCClient" />
83     </bean>
84
85     <!-- WarehouseC -->
86     <jaxws:endpoint id="warehouseC" address="/warehouseC"
87         implementorClass="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
88             warehouse.WarehouseCService">
89         <jaxws:implementor>
90             <ref bean="warehouseCService" />
91         </jaxws:implementor>
92         <jaxws:outInterceptors>
93             <ref bean="WarehouseEndTimestampSign_Response" />
94         </jaxws:outInterceptors>
95         <jaxws:inInterceptors>
96             <ref bean="WarehouseEndTimestampSign_Request" />
97             <ref bean="SecurityConnector" />
98         </jaxws:inInterceptors>
99     </jaxws:endpoint>
100
101     <bean id="warehouseCService"
102         class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.warehouse.WarehouseCService">
103         <property name="logStub" ref="loggingClient" />
104         <property name="manufacturerAService" ref="manufacturerAClient" />
105         <property name="manufacturerBService" ref="manufacturerBClient" />
106         <property name="manufacturerCService" ref="manufacturerCClient" />
107     </bean>
108
109     <!-- WarehouseCallback -->
110     <jaxws:endpoint id="warehouseCallback" address="/warehouseCallback">
111         <jaxws:implementor>
112             <ref bean="warehouseCallbackService" />
113         </jaxws:implementor>
114         <jaxws:outInterceptors>
115             <ref bean="WarehouseEndTimestampSign_Response" />
116         </jaxws:outInterceptors>
117         <jaxws:inInterceptors>
118             <ref bean="WarehouseEndTimestampSign_Request" />
119         </jaxws:inInterceptors>
120     </jaxws:endpoint>
121
122     <bean id="warehouseCallbackService"

```

```

123         class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
124             warehouse.callback.WarehouseCallbackService">
125         <property name="logStub" ref="loggingClient" />
126     </bean>
127
128     <!-- CLIENTS -->
129     <!-- Manufacturer Clients -->
130     <jaxws:client id="manufacturerAClient"
131         serviceClass="org.ws_i.sampleapplications.supplychainmanagement._2002_10.
132             manufacturer.ManufacturerPortType"
133         address="http://localhost:8888/ManufacturerService/ManufacturerA">
134     <jaxws:inInterceptors>
135         <bean class="org.apache.cxf.interceptor.LoggingInInterceptor" />
136         <ref bean="ManufacturerClientTimestampSign_Response" />
137     </jaxws:inInterceptors>
138     <jaxws:outInterceptors>
139         <ref bean="ManufacturerClientTimestampSign_Request" />
140     </jaxws:outInterceptors>
141 </jaxws:client>
142
143     <jaxws:client id="manufacturerBClient"
144         serviceClass="org.ws_i.sampleapplications.supplychainmanagement._2002_10.
145             manufacturer.ManufacturerPortType"
146         address="http://localhost:8888/ManufacturerService/ManufacturerB">
147     <jaxws:inInterceptors>
148         <bean class="org.apache.cxf.interceptor.LoggingInInterceptor" />
149         <ref bean="ManufacturerClientTimestampSign_Response" />
150     </jaxws:inInterceptors>
151     <jaxws:outInterceptors>
152         <ref bean="ManufacturerClientTimestampSign_Request" />
153     </jaxws:outInterceptors>
154 </jaxws:client>
155
156     <jaxws:client id="manufacturerCClient"
157         serviceClass="org.ws_i.sampleapplications.supplychainmanagement._2002_10.
158             manufacturer.ManufacturerPortType"
159         address="http://localhost:8888/ManufacturerService/ManufacturerC">
160     <jaxws:inInterceptors>
161         <bean class="org.apache.cxf.interceptor.LoggingInInterceptor" />
162         <ref bean="ManufacturerClientTimestampSign_Response" />
163     </jaxws:inInterceptors>
164     <jaxws:outInterceptors>

```

```

165         <ref bean="ManufacturerClientTimestampSign_Request" />
166     </jaxws:outInterceptors>
167 </jaxws:client>
168
169     <!-- Logging Client -->
170     <bean id="loggingClient"
171         class="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
172             loggingfacility.LoggingClient">
173         <property name="defaultUri" value="http://localhost:8888/logging/services/logging" />
174         <property name="marshaller" ref="loggingMarshaller" />
175         <property name="unmarshaller" ref="loggingMarshaller" />
176     </bean>
177
178     <bean id="loggingMarshaller" class="org.springframework.oxm.jaxb.Jaxb2Marshaller">
179         <property name="contextPath"
180             value="org.ws_i.sampleapplications.supplychainmanagement._2002_08.loggingfacility" />
181     </bean>
182
183     <!-- SECURITY INTERCEPTORS -->
184     <!-- MANUFACTURER CLIENT SECURITY INTERCEPTORS -->
185     <!--
186         This bean is an Out interceptor which will add a timestamp,
187         and then sign the timestamp, body, and ConfigurationHeader.
188     -->
189     <bean
190         class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor"
191         id="ManufacturerClientTimestampSign_Request">
192     <constructor-arg>
193         <map>
194             <entry key="action" value="Timestamp Signature" />
195             <entry key="user" value="warehouse" />
196             <entry key="signaturePropFile" value="warehouseKeystore.properties" />
197             <entry key="passwordCallbackClass"
198                 value="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
199                     warehouse.WarehousePasswordCallback" />
200             <entry key="signatureParts"
201                 value=
202                 "{Element}{ http://docs.oasis-open.org/wss/2004/01/
203                     oasis-200401-wss-wssecurity-utility-1.0.xsd}Timestamp;
204                 {Element}{ http://schemas.xmlsoap.org/soap/envelope/}Body;
205                 {Element}{ http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/
206                     Configuration.xsd}Configuration;" />

```

```

207         {Element}{http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-10/
208             Manufacturer/CallBack}StartHeader"/>
209     </map>
210 </constructor-arg>
211 </bean>
212
213 <!--
214     This bean is an In interceptor which will validate a signed
215     response, and timestamp it.
216 -->
217 <bean
218     class="org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor"
219     id="ManufacturerClientTimestampSign_Response">
220     <constructor-arg>
221         <map>
222             <entry key="action" value="Timestamp Signature"/>
223             <entry key="signaturePropFile" value="warehouseKeystore.properties"/>
224             <entry key="passwordCallbackClass"
225                 value="org.ws-i.sampleapplications.supplychainmanagement._2002_08.
226                                     warehouse.WarehousePasswordCallback"/>
227         </map>
228     </constructor-arg>
229 </bean>
230
231 <!-- WH ENDPOINT SECURITY INTERCEPTORS -->
232 <!--
233     WSS4JInInterceptor for validating the signature of the
234     SOAP request.
235 -->
236 <bean
237     id="WarehouseEndTimestampSign_Request"
238     class="org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor"
239     >
240     <constructor-arg>
241         <map>
242             <entry key="action" value="Timestamp Signature"/>
243             <entry key="signaturePropFile" value="warehouseKeystore.properties"/>
244             <entry key="passwordCallbackClass"
245                 value="org.ws-i.sampleapplications.supplychainmanagement._2002_08.
246                                     warehouse.WarehousePasswordCallback"/>
247         </map>
248     </constructor-arg>

```

```

249     </bean>
250
251     <!--
252         WSS4JOutInterceptor for timestamping and signing the SOAP response.
253     -->
254     <bean
255         id="WarehouseEndTimestampSign_Response"
256         class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor"
257         >
258         <constructor-arg>
259             <map>
260                 <entry key="action" value="Timestamp Signature"/>
261                 <entry key="user" value="warehouse"/>
262                 <entry key="signaturePropFile" value="warehouseKeystore.properties"/>
263                 <entry key="passwordCallbackClass"
264                     value="org.ws_i.sampleapplications.supplychainmanagement._2002_08.
265                                     warehouse.WarehousePasswordCallback"/>
266                 <entry key="signatureParts"
267                     value=
268                         "{Element}{http://docs.oasis-open.org/wss/2004/01/
269                                     oasis-200401-wss-wssecurity-utility-1.0.xsd}Timestamp;
270                                     {Element}{http://schemas.xmlsoap.org/soap/envelope/}Body"/>
271             </map>
272         </constructor-arg>
273     </bean>
274
275 </beans>

```

C.4 SCM Use Case Access Control Context

Listing C.4: *Access Control Context*

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans:beans xmlns="http://www.springframework.org/schema/security"
3      xmlns:beans="http://www.springframework.org/schema/beans"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xmlns:aop="http://www.springframework.org/schema/aop"
6      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/security
7          http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
8          http://www.springframework.org/schema/security/spring-security-2.0.4.xsd
9          http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">
10
11      <http auto-config="true"></http>
12      <aop:aspectj-autoproxy proxy-target-class="true"/>
13      <global-method-security>
14          <protect-pointcut expression="execution(* org.ws.i.sampleapplications.supplychainmanagement._2002_08.
15              warehouse.Warehouse*Service.shipGoods(..))" access="ROLE_RETAILER"/>
16      </global-method-security>
17
18      <authentication-manager alias="authenticationManager"/>
19      <user-service id="userDetailsServ">
20          <user authorities="ROLE_WEBCLIENT" name="webclient" password="webclientpass"/>
21          <user authorities="ROLE_RETAILER" name="retailer" password="retailerpass"/>
22          <user authorities="ROLE_WAREHOUSE" name="warehouse" password="warehousepass"/>
23          <user authorities="ROLE_MANUFACTURER" name="manufacturer" password="manufacturerpass"/>
24      </user-service>
25
26      <beans:bean id="preAuthProvider"
27          class="org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvider">
28          <beans:property name="preAuthenticatedUserDetailsService" ref="userDetailsWrapper"/>
29          <beans:property name="order" value="1"/>
30          <custom-authentication-provider/>
31      </beans:bean>
32      <beans:bean id="userDetailsWrapper"
33          class="org.springframework.security.userdetails.UserDetailsByNameServiceWrapper">
34          <beans:property name="userDetailsService" ref="userDetailsServ"/>
35      </beans:bean>
36  </beans:beans>

```

C.5 Warehouse Service Generated Policy Document

160

Listing C.5: *Warehouse Policy*

```
1 <?xml version = '1.0' encoding = 'ISO-8859-1' ?>
2 <wsp:Policy
3   xmlns:sopa='http://types.sopware.org/qos/SOPAssertions/1.1'
4   xmlns:sopwsp='http://types.sopware.org/qos/WS-PolicyExtensions/1.1'
5   xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'
6   xmlns:sp='http://schemas.xmlsoap.org/ws/2002/12/secext'
7   xmlns:xs='http://www.w3.org/2001/XMLSchema'
8   xmlns:wsp='http://www.w3.org/ns/ws-policy'
9 >
10
11
12 <wsp:Policy name = 'warehouseService_Binding_Binding_Policy'
13   wsu:Id = 'warehouseService_Binding_Binding_Policy'>
14   <wsp:All>
15     <sp:AsymmetricBinding wsp:Ignorable = 'false' wsp:Optional = 'false'>
16       <sp:IncludeTimestamp wsp:Ignorable = 'false' wsp:Optional = 'false' />
17       <sp:InitiatorToken wsp:Ignorable = 'false' wsp:Optional = 'false'>
18         <wsp:Policy name = 'InitiatorTokenNestedPolicy' wsu:Id = 'InitiatorTokenNestedPolicy'>
19           <wsp:All>
20             <sp:X509Token wsp:Ignorable = 'false' wsp:Optional = 'false'
21               sp:IncludeToken=
22                 'http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
23                   IncludeToken/AlwaysToRecipient'>
24               <wsp:Policy name='InitiatorTokenX509TokenNestedPolicy'
25                 wsu:Id='InitiatorTokenX509TokenNestedPolicy'>
26                 <wsp:All>
27                   <sp:WssX509V3Token10 wsp:Ignorable = 'false' wsp:Optional = 'false' />
28                 </wsp:All>
29               </wsp:Policy>
30             </sp:X509Token>
31           </wsp:All>
32         </wsp:Policy>
33       </sp:InitiatorToken>
34       <sp:RecipientToken wsp:Ignorable = 'false' wsp:Optional = 'false'>
35         <wsp:Policy name='RecipientTokenNestedPolicy' wsu:Id='RecipientTokenNestedPolicy'>
36           <wsp:All>
37             <sp:X509Token wsp:Ignorable = 'false' wsp:Optional = 'false'
38               sp:IncludeToken=
```



```

39         'http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
40             IncludeToken/AlwaysToInitiator '>
41     <wsp:Policy name='RecipientTokenX509TokenNestedPolicy '
42         wsu:Id = 'RecipientTokenX509TokenNestedPolicy '>
43         <wsp:All>
44             <sp:WssX509V3Token10 wsp:Ignorable = 'false' wsp:Optional = 'false' />
45         </wsp:All>
46     </wsp:Policy>
47 </sp:X509Token>
48 </wsp:All>
49 </wsp:Policy>
50 </sp:RecipientToken>
51 <sp:AlgorithmSuite wsp:Ignorable = 'false' wsp:Optional = 'false '>
52     <wsp:Policy name = 'AlgorithmSuiteNestedPolicy' wsu:Id = 'AlgorithmSuiteNestedPolicy '>
53         <wsp:All>
54             <sp:Basic128Rsa15 wsp:Ignorable = 'false' wsp:Optional = 'false' />
55         </wsp:All>
56     </wsp:Policy>
57 </sp:AlgorithmSuite>
58 </sp:AsymmetricBinding>
59 <sp:SignedParts wsp:Ignorable = 'false' wsp:Optional = 'false '>
60     <sp:Header wsp:Ignorable='false' wsp:Optional='false' Name='ConfigurationHeader '
61         Namespace='http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Warehouse' />
62     <sp:Body wsp:Ignorable = 'false' wsp:Optional = 'false' />
63 </sp:SignedParts>
64 </wsp:All>
65 </wsp:Policy>
66
67 </wsp:Policy>

```


Bibliography

- Aier, S., Offermann, P., Schönherr, M. and Schröpfer, C. (2007). Implementing Non-functional Service Descriptions in SOAs, *in* D. Draheim and G. Weber (eds), *Trends in Enterprise Application Architecture*, Vol. 4473 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 40–53.
- Allilaire, F. (2009). Towards Traceability support in ATL with Obeo Traceability, *Model Transformation with ATL*.
- Alsaadi, A. (2004). A performance analysis approach based on the UML class diagram, *SIGSOFT Softw. Eng. Notes* **29**(1): 254–260.
- Ambler, S. W. (2009). An Introduction to Agile Modeling.
URL:<http://www.agilemodeling.com/essays/introductionToAM.htm>
- Ambler, S. W. and Jeffries, R. (2002). *Agile modeling: effective practices for extreme programming and the unified process*, John Wiley & Sons, Inc., New York, NY, USA.
- Ameller, D. (2009). *Considering Non-Functional Requirements in Model-Driven Engineering*, Master, Universitat Politècnica de Catalunya.
- Ameller, D., Franch, X. and Cabot, J. (2010). Dealing with Non-Functional Requirements in Model-Driven Development, *Requirements Engineering, IEEE International Conference on* **0**: 189–198.
- Anaby-Tavor, A., Amid, D., Sela, A., Fisher, A., Zhang, K. and Jun, O. T. (2008). Towards a Model Driven Service Engineering Process, *Services, IEEE Congress on* **0**: 503–510.
- Aniszczyk, C. and Marz, N. (2006). Create more – better – code in Eclipse with JET.
URL:<http://www.ibm.com/developerworks/java/library/os-ecl-jet/index.html?ca=drs>
- Apache (2010). Apache CXF.
URL:<http://cxf.apache.org/>
- Arsanjani, A. (2004). Service-oriented modeling and architecture, *IBM developer works*.
URL:<http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/>
- Asadi, M. and Ramsin, R. (2008). *MDA-Based Methodologies: An Analytical Survey*, Vol. 5095/2010 of *Lecture Notes in Computer Science*, Springer, Berlin / Heidelberg, chapter 15, pp. 419–431.
- Asnar, Y., Bonato, R., Bryl, V., Holtmanns, S., Klobucar, T., Massacci, F., Pazzaglia, J.-C., Porekar, J., Riccucci, C., Thomas, R. and Zannone, N. (2006). SERENITY (System Engineering for Security & Dependability) project Deliverable A1. D1. 1 – State of the Art

Review.

URL:http://www.serenity-project.org/IMG/pdf/A1.D1.1_state_of_the_art_review_v1.16_Final-2.pdf

Asnar, Y., Felici, M., Kokolakis, S., Li, K., Saidane, A. and Yautsiukhin, A. (2009). Serenity Project Deliverable A1.D5.1 - Preliminary version of S&D Metrics.

URL:http://www.serenity-project.org/IMG/pdf/A1.D5.1_preliminary_version_of_s_d_metrics_final_version.pdf

Atluri, V. (2001). Security for workflow systems, *Information Security Technical Report* **6**(2): 59–68.

Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Trans. Dependable Secur. Comput.* **1**(1): 11–33.

Bakker, J., Tekinerdogan, B. and Aksit, M. (2005). Characterization of Early Aspect Approaches, *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design Workshop*, Chicago, Illinois, USA, p. 7.

Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J. and Neema, E. (2006). Developing Applications Using Model-driven Design Environments, *IEEE Computer* **39**(2): 33–40.

Balsamo, S. and Simeoni, M. (2001a). Deriving Performance Models from Software Architecture Specifications, *Proceedings of the 15th European Simulation Multiconference (ESM2001)*, SCS - Society for Computer Simulation, pp. 6–9.

Balsamo, S. and Simeoni, M. (2001b). On transforming UML models into performance models, *ETAPS01: Workshop on Transformations in UML (WTUML)*.

Barais, O., Klein, J., Baudry, B., Jackson, A. and Clarke, S. (2008). Composing multi-view aspect models, *Commercial-off-the-Shelf (COTS)-Based Software Systems, International Conference on* **0**: 43–52.

Barais, O., Meur, A.-F. L., Duchien, L. and Lawall, J. (2006). Safe Integration of New Concerns in a Software Architecture, *Engineering of Computer-Based Systems, IEEE International Conference on the* **0**: 52–64.

Barn, B., Dexter, H., Oussena, S. and Sparks, D. (2008). SOA-MDK: Towards a Method Development Kit for Service Oriented System Development, in G. Magyar, G. Knapp, W. Wojtkowski, W. G. Wojtkowski and J. Zupančič (eds), *Advances in Information Systems Development*, Springer US, pp. 191–201.

Basin, D. A., Clavel, M., Doser, J. and Egea, M. (2007). A metamodel-based approach for analyzing security-design models, in G. Engels, B. Opdyke, D. C. Schmidt and F. Weil (eds), *MoDELS*, Vol. 4735 of *Lecture Notes in Computer Science*, Springer, pp. 420–435.

Basin, D., Clavel, M., Doser, J. and Egea, M. (2009). Automated analysis of security-design models, *Information and software technology* **51**(5): 815–831.

Basin, D., Doser, J. and Lodderstedt, T. (2006). Model driven security: From UML models to access control infrastructures, *ACM Transactions on Software Engineering and Methodology (TOSEM)* **15**(1): 39 – 91.

Bell, D. E. and LaPadula, L. J. (1976). Secure computer system: Unified exposition and Multics interpretation.

URL:<http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA023588>

- Berardinelli, L., Bernardi, S., Cortellessa, V. and Merseguer, J. (2009). UML profiles for non-functional properties at work: analyzing reliability, availability and performance, *Proc. of 2nd International Workshop on Non-functional System Properties in Domain Specific Modeling Languages (NFPinDSML2009)*, Denver, Colorado, USA.
- Bernardi, S., Donatelli, S. and Merseguer, J. (2002). From UML sequence diagrams and statecharts to analysable petri net models, *Proceedings of the 3rd international workshop on Software and performance*, WOSP '02, ACM, New York, NY, USA, pp. 35–45.
- Bernardi, S., Merseguer, J. and Petriu, D. (2009). A dependability profile within MARTE, *Software and Systems Modeling* pp. 1–24.
- Bettin, J. (2003). Model-Driven Architecture Implementation & Metrics.
URL:<http://www.softmetaware.com/publicwhitepapers/mda-implementationandmetrics.pdf>
- Bézivin, J., Hammoudi, S., Lopes, D. and Jouault, F. (2004). Applying MDA approach for web service platform, *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International* pp. 58 – 70.
- Bézivin, J., Jouault, F. and Touzet, D. (2005). An Introduction to the ATLAS Model Management Architecture, *Technical report*, Laboratoire D Informatique de Nantes -Atlantique.
URL:<http://www.sciences.univ-nantes.fr/lina/atl/www/papers/RR-LINA2005-01.pdf>
- Bhargavan, K., Fournet, C. and Gordon, A. D. (2008). Verifying policy-based web services security, *ACM Trans. Program. Lang. Syst.* **30**(6): 1–59.
- Bhargavan, K., Fournet, C., Gordon, A. D. and O'Shea, G. (2005). An advisor for web services security policies, *SWS '05: Proceedings of the 2005 workshop on Secure web services*, ACM, New York, NY, USA, pp. 1–9.
- Blanco, C., de Guzmán, I. G. R., Fernández-Medina, E., Trujillo, J. and Piattini, M. (2009a). Applying an MDA-Based Approach to Consider Security Rules in the Development of Secure DWs, *ARES*, IEEE Computer Society, pp. 528–533.
- Blanco, C., de Guzmán, I. G. R., Fernández-Medina, E., Trujillo, J. and Piattini, M. (2009b). Including Security Rules Support in an MDA Approach for Secure DWs, *ARES*, IEEE Computer Society, pp. 516–521.
- Bombonatti, D. L. G. and Melnikoff, S. S. S. (2009). Survey on early aspects approaches: non-functional crosscutting concerns integration in software sytems, *Proceedings of the 4th WSEAS international conference on Computer engineering and applications*, CEA'10, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 137–142.
- Booch, G., Brown, A. W., Iyengar, S., Rumbaugh, J. and Selic, B. (2004). An MDA Manifesto, *Business Process Trends/MDA Journal* .
- Boronat, A., Knapp, A., Meseguer, J. and Wirsing, M. (2009). *What Is a Multi-modeling Language?* , Vol. 5486 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 71–87.

- Box, D., Christensen, E., Curbera, F., Ferguson, D., Frey, J., Hadley, M., Kaler, C., Langworthy, D., Leymann, F., Lovering, B., Lucco, S., Millet, S., Mukhi, N., Nottingham, M., Orchard, D., Shewchuk, J., Sindambiwe, E., Storey, T., Weerawarana, S. and Winkler, S. (2004). Web services addressing (WS-Addressing).
URL:<http://www.immagic.com/eLibrary/ARCHIVES/SUPRSDDED/W3C/W040810B.pdf>
- Braga, C. (2009). From access control policies to an aspect-based infrastructure: A metamodel-based approach, *Models in Software Engineering*.
- Briones, J. F., Alonso, A., de Miguel, M. A. and Silva Gallino, J. P. (2010). Evaluating non-functional properties globally, *Proceedings of the 2010 3rd Simposio de Sistemas de Tiempo Real in the Congreso Español de Informática (CEDI)*, Valencia, Spain.
- Briones, J. F., de Miguel, M. A., Alonso, A. and Silva Gallino, J. P. (2006). Quality and Resource Management in Components Framework, in S. Robert and R. Sanz (eds), *Proceedings of Towards Off-the-Shelf Embedded Real-Time Software (OSERTS)*, Robert, Sylvain Sanz, Ricardo, Turin, Italy.
- Briones, J. F., de Miguel, M. A., Alonso, A. and Silva Gallino, J. P. (2008a). Modeling Quality of Service Adaptability, in M. van Sinderen, J. P. Andrade Almeida, L. Ferreira Pires and M. Steen (eds), *Enterprise Distributed Object Computing Workshops, International Conference on*, IEEE Computer Society, Munich, Germany, pp. 50–27.
- Briones, J. F., de Miguel, M. A., Alonso, A. and Silva Gallino, J. P. (2008b). Software Modeling of Quality-Adaptable Systems, in A. Llemosí and J. Proenza (eds), *Proceedings of 11th Jornadas de Tiempo Real*, Llemosí, Albert Proenza, Julián, Palma de Mallorca, Spain, pp. 21–26.
- Briones, J. F., de Miguel, M. A., Alonso, A. and Silva Gallino, J. P. (2009a). Quality of Service Composition and Adaptability of Software Architectures, *ISORC*.
- Briones, J. F., de Miguel, M. A., Alonso, A. and Silva Gallino, J. P. (2009b). Quality of Service Composition and Adaptability of Software Architectures, *XII Jornadas de Tiempo Real*, Leganés, Madrid, Spain.
- Briones, J. F., de Miguel, M. A., Alonso, A. and Silva Gallino, J. P. (2010). Analysis of Quality Dependencies in the Composition of Software Architectures, in M. I. Capel Tuñón and J. A. H. and (eds), *XIII Jornadas de Tiempo Real*, Capel Tuñón, Manuel I. Holgado Terriza, Juan Antonio, Granada, Spain.
- Briones, J. F., de Miguel, M. A., Silva Gallino, J. P. and Alonso, A. (2006). Integration of safety analysis and software development methods, *IET Conference Publications* **2006**(CP515): 275–284.
- Briones, J. F., de Miguel, M. A., Silva Gallino, J. P. and Alonso, A. (2007). *Application of Safety Analyses in Model Driven Development*, Vol. 4761 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 93–104.
- Briones, J. F., de Miguel, M. A., Silva Gallino, J. P. and Alonso, A. (2010). On the Requirements for Quality Composability Modeling and Analysis, *Proceedings of the 2010 1st IEEE International Workshop on Model-Based Engineering for Real-Time Embedded Systems Design (MoBE-RTES)*, IEEE Computer Society, Los Alamitos, CA, USA, Carmona, Spain, pp. 123–129.
- Brito, I. S. and Moreira, A. M. D. (2003). Advanced Separation of Concerns for Requirements Engineering, in E. Pimentel, N. R. Brisaboa and J. Gómez (eds), *JISBD*, pp. 47–56.

- Brown, A. W. (2004). Model driven architecture: Principles and practice, *Software and Systems Modeling SoSyM* **3**(4): 314–327.
- Brown, A. W., Conallen, J. and Tropeano, D. (2005). *Introduction: Models, Modeling, and Model-Driven Architecture (MDA)*, Springer-Verlag, Berlin/Heidelberg, chapter 1, pp. 1–16.
- Brunet, G., Chechik, M., Easterbrook, S., Nejati, S., Niu, N. and Sabetzadeh, M. (2006). A manifesto for model merging, *GaMMA '06: Proceedings of the 2006 international workshop on Global integrated model management*, ACM, Shanghai, China, pp. 5–12.
- Bui, N. B., Zhu, L., Liu, Y. J., Tasic, V. and Jeffery, R. (2008). Automating Web Service Development Using a Unified Model, *Enterprise Distributed Object Computing Conference Workshops, 2008 12th*, IEEE Computer Society, Washington, DC, USA, pp. 301–308.
- Canevet, C., Gilmore, S., Hillston, J., Kloul, L. and Stevens, P. (2004). Analysing UML 2.0 activity diagrams in the software performance engineering process, *SIGSOFT Softw. Eng. Notes* **29**(1): 74–78.
- Cao, F., Bryant, B. R., Zhao, W., Burt, C. C., Raje, R. R., Olson, A. M. and Auguston, M. (2004). A Meta-modeling approach to Web Services, *IEEE International Conference on Web Services (ICWS'04)*, Vol. 0, IEEE Computer Society, pp. 796–799.
- Carminati, B., Ferrari, E. and Hung, P. C. K. (2006). Security Conscious Web Service Composition, *Web Services, IEEE International Conference on* **0**: 489–496.
- Castro, J., Kolp, M. and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the Tropos project, *Information Systems* **27**(6): 365–389.
- CDTI (2006). ITECBAN.
URL:<http://www.daedalus.es/i-d-i/proyectos-nacionales/itecban/>
- Charfi, A. and Mezini, M. (2005). Using Aspects for Security Engineering of Web Service Compositions, *Web Services, IEEE International Conference on* **0**: 59–66.
- Charfi, A., Schmeling, B., Heizenreder, A. and Mezini, M. (2006). Reliable, Secure, and Transacted Web Service Compositions with AO4BPEL, *Web Services, 2006. ECOWS '06. 4th European Conference on*, pp. 23–34.
- Chavez, C., Garcia, A., Lucena, C. and Kulesza, U. (2006). Aspectual Connectors: Supporting the Seamless Integration of Aspects and ADLs, *Proceedings of Brazilian Symposium on Software Engineering SBES* pp. 17–32.
- Chitchyan, R., Rashid, A., Sawyer, P., Garcia, A., Pinto, M., Bakker, J., Tekinerdogan, B., Clarke, S. and Jackson, A. (2005). Survey of aspect-oriented analysis and design approaches.
- Chung, L., Nixon, B. A., Young, E. and Mylopoulos, J. (2000). *Non-functional requirements in software engineering*, Kluwer Academic Publishing, Norwell, Massachusetts, USA.
- Clark, T., Evans, A., Sammut, P. and Willans, J. (2008). *Applied Metamodelling, A Foundation for Language Driven Development*, Vol. 2005, second edn, Ceteva, London.
- Clarke, S. and Baniassad, E. (2005). *Aspect-Oriented Analysis and Design: The Theme Approach*, Addison-Wesley Professional.

- Clavel, M., Silva, V., Braga, C. and Egea, M. (2008). Model-Driven Security in Practice: An Industrial Experience, *ECMDA-FA '08: Proceedings of the 4th European conference on Model Driven Architecture*, Springer-Verlag, Berlin, Heidelberg, pp. 326–337.
- Clements, P. and Northrop, L. (2001). *Software Product Lines: Practices and Patterns*, SEI Series in Software Engineering, 1st edn, Addison-Wesley Professional.
- Cooper, K., Dai, L., Deng, Y. and Dong, J. (2003). Towards an Aspect-oriented Architectural Framework, *Proceedings of the Second International Workshop on Aspect-Oriented Requirements Engineering and Architecture Design (Early Aspects)*.
- Cortellessa, V., Di Marco, A. and Inverardi, P. (2007a). *Integrating Performance and Reliability Analysis in a Non-Functional MDA Framework*, Vol. 4422 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 57–71.
- Cortellessa, V., Di Marco, A. and Inverardi, P. (2007b). Non-Functional Modeling and Validation in Model-Driven Architecture., *WICSA*, IEEE Computer Society, p. 25.
- Cortellessa, V., Marco, A. D. and Inverardi, P. (2006). Software performance model-driven architecture., *SAC'06*, pp. 1218–1223.
- Cuesta, C. E., de la Fuente, P., Barrio-Solórzano, M. and Beato, M. E. (2002). Introducing Reflection in Architecture Description Languages, *Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance*, WICSA 3, Kluwer, B.V., Deventer, The Netherlands, The Netherlands, pp. 143–156.
- Czarnecki, K. (2005). Overview of Generative Software Development, *Lecture Notes in Computer Science* **3566/2005**(Unconventional Programming Paradigms): 326–341.
- Czarnecki, K., Østerbye, K. and Völter, M. (2002). Generative Programming, in J. Hernández and A. Moreira (eds), *Object-Oriented Technology ECOOP 2002 Workshop Reader*, Vol. 2548 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 15–29.
- Dai, L. and Cooper, K. (2007). A Survey of Modeling and Analysis Approaches for Architecting Secure Software Systems, *I. J. Network Security* **5**(2): 187–198.
- Dalgarno, M. and Fowler, M. (2008). UML vs. Domain-Specific Languages, *Methods Tools* **16**(2): August, 14, 2008–2; 9.
URL:www.methodsandtools.com
- Davis, D., Karmarkar, A., Pilz, G. and Yalçinalp, Ü. (2005). Web services reliable messaging policy assertion (WS-RM Policy), *Specification*, Feb.
- de Castro Bertagnolli, S. and Lisbôa, M. L. B. (2003). The FRIDA model, *Proceedings of the Workshop on Analysis Aspect-Oriented Software*, Darmstadt, Germany.
- de Miguel, M. A., Briones, J. F., Silva Gallino, J. P. and Alonso, A. (2006). Model based integration of safety analysis and development, *Proceedings of 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, IEEE Computer Society, Washington, DC, USA, Gyeongju, Korea, pp. 323–326.
- de Miguel, M. A., Briones, J. F., Silva Gallino, J. P. and Alonso, A. (2008). Integration of safety analysis in model-driven software development, *IET Software* **2**(3): 260–280.

- de Miguel, M. A., Massonet, P., Silva Gallino, J. P. and Briones, J. F. (2008). Model Based Development of Quality-Aware Software Services, *ISORC*.
- de Miguel, M. A., Salazar, E., Silva Gallino, J. P. and Briones, J. F. (2011). *Reusable Modelling Tools Assets: Deployment of MDA Artefacts*, IGI Global, Hershey, PA, pp. 0–0.
- Dehlinger, J. and Subramanian, N. V. (2006). Architecting Secure Software Systems Using an Aspect-Oriented Approach: A Survey of Current Research.
- Delgado, A., Ruiz, F., de Guzmán, I. G. R. and Piattini, M. (2009). MINERVA: Model drIveN and sService oRiented Framework for the Continuous Business Process improvEment and relAted Tools, in A. Dan, F. Gittler and F. Toumani (eds), *ICSOC/ServiceWave Workshops*, Vol. 6275 of *Lecture Notes in Computer Science*, pp. 456–466.
- Delgado, A., Ruiz, F., de Guzmán, I. G. R. and Piattini, M. (2010). Towards an ontology for service oriented modeling supporting business processes, in P. Loucopoulos and J.-L. Cavarero (eds), *RCIS*, IEEE, pp. 415–424.
- Delgado, A., Ruiz, F., de Guzmán, I. G. R. and Piattini, M. (2011). Business Process Service Oriented Methodology (BPSOM) with Service Generation in SoaML, in H. Mouratidis and C. Rolland (eds), *CAiSE*, Vol. 6741 of *Lecture Notes in Computer Science*, Springer, pp. 672–680.
- den Haan, J. (2009). MDE - Model Driven Engineering - reference guide.
URL:<http://www.theenterpriseearchitect.eu/archive/2009/01/15/mde---model-driven-engineering----reference-guide#abstract-syntax>
- Denker, G., Kagal, L., Finin, T., Paolucci, M. and Sycara, K. (2003). Security for DAML Web Services: Annotation and Matchmaking, in D. Fensel, K. Sycara and J. Mylopoulos (eds), *The Semantic Web - ISWC 2003*, Vol. 2870 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 335–350.
- Di Giandomenico, F., Kwiatkowska, M., Martinucci, M., Masci, P. and Qu, H. (2010). {D}ependability {A}nalysis and {V}erification for {C}onnected {S}ystems, *{ISOLA} 2010, 4th {I}nternational {S}ymposium {O}n {L}everaging {A}pplications of {F}ormal {M}ethods, {V}erification and {V}alidation*, {H}eraklion {G}rèce.
- Di Marco, A., Bertolino, A., Di Giandomenico, F., Masci, P. and Sabetta, A. (2010). Metrics for QoS analysis in dynamic, evolving and heterogeneous connected systems, *Proceedings of the Eighth International Workshop on Dynamic Analysis*, WODA '10, ACM, New York, NY, USA, pp. 32–37.
- Didonet del Fabro, M., Bézivin, J. and Jouault, F. (2005). AMW: a generic model weaver, *Proceedings of the Using metamodels to support MDD Workshop, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005)*.
- Didonet, M. and Valduriez, P. (2009). Towards the efficient development of model transformations using model weaving and matching transformations, *Software and System Modeling* 8(3): 305–324.
- Dijkstra, E. W. (1982). *On the Role of Scientific Thought*, Springer-Verlag, pp. 60–66.
- Dobson, G., Hall, S. and Kotonya, G. (2007). A Domain-Independent Ontology for Non-Functional Requirements, *e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on*, pp. 563–566.

- Dodd, J., Allen, P., Butler, J., Olding, S., Veryard, R. and Wilkes, L. (2007). CBDI-SAE Meta Model for SOA Version 2, *Technical report*, Everware-CBDI.
URL:http://www.cbdiforum.com/public/meta_model_v2.php
- Eclipse (2011a). Connected Data Objects.
URL:<http://wiki.eclipse.org/CD0>
- Eclipse (2011b). Eclipse Modeling Framework Project (EMF).
URL:<http://www.eclipse.org/modeling/emf/>
- Elrad, T., Aldawud, O. and Bader, A. (2002). Aspect-Oriented Modeling: Bridging the Gap between Implementation and Design, in D. Batory, C. Consel and W. Taha (eds), *Generative Programming and Component Engineering*, Vol. 2487 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 189–201.
- Elvesæter, B., Carrez, C., Mohagheghi, P., rgen Berre, A.-J., Johnsen, S. G. and Solberg, A. (2011). *Model-driven Service Engineering with SoaML*, Springer Vienna, chapter 2, pp. 25–54.
- Emig, C., Brandt, F., Abeck, S., Biermann, J. and Klarl, H. (2007). An Access Control Metamodel for Web Service-Oriented Architecture, *International Conference on Software Engineering Advances (ICSEA 2007)*, IEEE, Cap Esterel, France, pp. 57–57.
- Emig, C., Kreuzer, S., Abeck, S., Biermann, J. and Klarl, H. (2008). Model-driven development of access control policies for web services, *Proceedings of the 9th IASTED International Conference Software Engineering and Applications (SEA 2008)*, Orlando, Florida, USA.
- Eric Yu (2011). *Modeling Strategic Relationships for Process Reengineering*, 1 edn, Massachussets Institute of Technology, Boston, chapter 2, pp. 11 – 152.
- Erickson, J., Lyytinen, K. and Siau, K. (2005). Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research, *Journal of Database Management* **16**(4): 88–100.
- Erl, T. (2005). *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*, Prentice Hall, Upper Saddle River, NJ, USA.
- Erl, T. (2007). *SOA Principles of Service Design*, Prentice Hall.
- Fenster, L. and Hamilton, B. (2010). UML or DSL: Which Bear Is Best?, *The Architecture Journal* (23): 32 – 37.
URL:<http://msdn.microsoft.com/en-us/architecture/ff476944>
- Fenton, N. E. and Pfleeger, S. L. (1998). *Software Metrics: A Rigorous and Practical Approach*, 2nd edn, PWS Publishing Co, Boston, MA, USA.
- Finkelsetin, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M. (1992). Viewpoints: A framework for integrating multiple perspectives in system development, *International Journal of Software Engineering and Knowledge Engineering* **2**.
- Firesmith, D. (2004). Specifying reusable security requirements, *Journal of Object Technology* **3**(1): 61–75.
- Fowler, M. (2010). *Domain-Specific Languages*, first edn, Addison-Wesley Professional; 1 edition.
URL:<http://www.amazon.com/Domain-Specific-Languages-Addison-Wesley-Signature-Fowler/dp/0321712943>

- France, R., Ghosh, S., Dinh-Trong, T. and Solberg, A. (2006). Model-Driven Development Using UML 2.0: Promises and Pitfalls, *Computer* **39**(2): 59–66.
- Freitas, E., Wehrmeister, M., Pereira, C., Wagner, F., Silva, E. and Carvalho, F. (2007). Using Aspect-Oriented Concepts in the Requirements Analysis of Distributed Real-Time Embedded Systems, in A. Rettberg, M. Zanella, R. Dömer, A. Gerstlauer and F. Rammig (eds), *Embedded System Design: Topics, Techniques and Trends*, Vol. 231 of *IFIP International Federation for Information Processing*, Springer Boston, pp. 221–230.
- Galster, M. and Bucherer, E. (2008). A Taxonomy for Identifying and Specifying Non-Functional Requirements in Service-Oriented Development, *Services, IEEE Congress on* **0**: 345–352.
- Gervais, M.-P. (2002). Towards an MDA-Oriented Methodology, *COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, IEEE Computer Society, Washington, DC, USA, pp. 265—270.
- Gharavi, V. (2008). *Applying Model-Driven Development to Reduce Programming Efforts for Small Application Development*, Master thesis, Delft University of Technology, The Netherlands.
- Gilmore, S., Gönczy, L., Koch, N., Mayer, P., Tribastone, M. and Varró, D. (2010). Non-functional properties in the model-driven development of service-oriented systems, *Software and Systems* . . . .
- Gilmore, S., Haenel, V., Kloul, L. and Maidl, M. (2005). Choreographing security and performance analysis for web services, *Formal Techniques for Computer* . . . .
- Gönczy, L., Ávéd, J. and Varró, D. (2006). Model-based deployment of web services to standards-compliant middleware, in P. Isaías, M. Baptista Nunes and I. J. Martinez (eds), *Proc. of the Iadis International Conference on WWW/Internet 2006(ICWI2006)*, Iadis Press, Murcia, Spain.
- Gönczy, L., Déri, Z. and Varró, D. (2008). Model Driven Performability Analysis of Service Configurations with Reliable Messaging, *Proc. of Model Driven Web Engineering Workshop(MDWE) 2008*.
- Gordijn, J. and Akkermans, J. M. (2003). Value-based requirements engineering: exploring innovative e-commerce ideas, *Requir. Eng.* **8**(2): 114–134.
- Gorelik, D. (2008). Transformation to SOA: Part 3. UML to SOA.
URL:http://www.ibm.com/developerworks/rational/library/08/0115_gorelik/
- Gray, J. G. (2002). *Aspect-Oriented Domain-Specific Modeling: A Generative Approach Using A Metaweaver Framework*, Phd thesis, Vanderbilt University.
- Greenfield, J. and Short, K. (2003). Software factories: assembling applications with patterns, models, frameworks and tools, in R. Crocker and G. L. S. Jr. (eds), *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA*, ACM, pp. 16–27.
- Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-marc Loingtier and John Irwin (1997). Aspect-oriented programming, *ECOOP'97 - Object-Oriented Programming 1997* **1241/1997**: 220 – 242.

- Grønmo, R. and Jaeger, M. (2005). Model-Driven Methodology for Building QoS-Optimised Web Service Compositions, in L. Kutvonen and N. Alonistioti (eds), *Distributed Applications and Interoperable Systems*, Vol. 3543 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 1119–1120.
- Grønmo, R., Skogan, D., Solheim, I. and Oldevik, J. (2004). Model-Driven Web Service Development, *e-Technology, e-Commerce, and e-Services, IEEE International Conference on* **0**: 42—45.
- Gu, G. P. and Petriu, D. C. (2002). XSLT transformation from UML models to LQN performance models, *Proceedings of the 3rd international workshop on Software and performance*, WOSP '02, ACM, New York, NY, USA, pp. 227–234.
- Guillaume Hillairet (n.d.). emftriple.
URL:<http://code.google.com/a/eclipselabs.org/p/emftriple/>
- Gutiérrez, C., Fernández-Medina, E. and Piattini, M. (2006). Towards a process for web services security, *Journal of Research and Practice in Information Technology* **38**(1): 57–68.
- Hafner, M., Breu, R., Agreiter, B. and Nowak, A. (2006). SECTET: An extensible framework for the realization of secure inter-organizational workflows, *Internet Research* .
- Henriksson, A., Aßman, U. and Hunt, J. (2005). Improving Software Quality in Safety-Critical Applications by Model-Driven Verification, *Electronic Notes in Theoretical Computer Science* .
- Hessellund, A. (2009). *Domain-Specific Multimodeling*, PhD thesis, IT University of Copenhagen.
- Hillairet, G. (2010). emftriple.
URL:<http://code.google.com/a/eclipselabs.org/p/emftriple/>
- Hovsepian, A., Baelen, S., Berbers, Y. and Joosen, W. (2009). *Specifying and Composing Concerns Expressed in Domain-Specific Modeling Languages* , Vol. 33 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 116–135.
- Hovsepian, A., Van Baelen, S., Yskout, K., Berbers, Y. and Joosen, W. (2007). Composing application models and security models: on the value of aspect-oriented technologies, *Proc. of the Eleventh international workshop on aspect-oriented modeling (AOM@MODELS 2007)* pp. 1—10.
- Huemer, C. and Liegl, P. (2007). A UML Profile for Core Components and their Transformation to XSD, *2007 IEEE 23rd International Conference on Data Engineering Workshop*, IEEE, Istanbul, pp. 298–306.
- IBM (2005). Generating XSD schemas from UML models.
URL:<http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.xtools.transformations.doc/topics/txsdover.html>
- IBM (2007). New to SOA and web services.
URL:<http://www.ibm.com/developerworks/webservices/newto/>
- IBM (2011a). IBM Software - WebSphere.
URL:<http://www-01.ibm.com/software/websphere/#>
- IBM (2011b). Rational Asset Manager.
URL:<http://www-01.ibm.com/software/rational/products/ram/>

- IEEE (2000). IEEE Recommended practice for architectural description of software-intensive systems.
- InnoQ (2007). InnoQ: Web Services Standards Overview Poster.
URL:<http://www.infoq.com/news/2007/03/innoq-ws-standards-poster>
- ISO (1989). Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture.
- ISO/IEC (2001). ISO/IEC 9126: Information technology-Software product evaluation-Quality characteristics and the guidelines for their use.
- ISO/IEC (2003). ISO/IEC 19761:2003 Software engineering – COSMIC-FFP – A functional size measurement method.
- ISO/IEC (2011). *ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, ISO, Geneva, Switzerland.
- Jackson, E. K. (2007). The Model Integrated Computing Approach to Software Architecture, in A. Prinz (ed.), *Proceedings of the ASM'07 The 14th International ASM Workshop*, GRIMSTAD, NORWAY.
- Jackson, E. K., Seifert, D., Dahlweid, M., Santen, T., Bjørner, N. and Schulte, W. (2009). Specifying and Composing Non-functional Requirements in Model-Based Development, *Software Composition* **5634**: 72–89.
- Jaeger, M. C. and Werner, M. (2009). *Web Services Dependability, Managing Web Service Quality*, IGI Global, US; Hershey, PA, pp. 151–167.
- Jakoubi, S., Tjoa, S., Goluch, G. and Quirchmayr, G. (2009). A Survey of Scientific Approaches Considering the Integration of Security and Risk Aspects into Business Process Management, in A. M. Tjoa and R. Wagner (eds), *DEXA Workshops*, IEEE Computer Society, pp. 127–132.
- Jegadeesan, H. and Balasubramaniam, S. (2009). A Model-driven Approach to Service Policies, *Journal of Object Technology* **8**(2): 163–186.
- Jiang, Q., Zhao, J., Li, X. and Zheng, G. (2005). A MDA model transformation tool for EDOC ER models, *Journal of Nanjing University* **41**: 512 – 520.
- Johnston, S. (2004). Modeling security concerns in service-oriented architectures.
URL:<http://www.ibm.com/developerworks/rational/library/4860.html>
- Jonkers, H., Iacob, M.-E., Lankhorst, M. M. and Strating, P. (2005). Integration and Analysis of Functional and Non-Functional Aspects in Model-Driven E-Service Development, *EDOC '05: Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference*, IEEE Computer Society, Washington, DC, USA, pp. 229–238.
- Josuttis, N. M. (2007). *SOA in Practice: The Art of Distributed System Design (Theory in Practice)*, O'Reilly Media.
- Jouault, F. (2005). Loosely Coupled Traceability for ATL, *Proceedings of the European Conference on Model Driven Architecture (ECMDA) workshop on traceability*, Nuremberg, Germany.
- Jouault, F., Allilaire, F., Bézivin, J. and Kurtev, I. (2008). ATL: A model transformation tool, *Science of Computer Programming* **72**(1-2): 31–39.

- Juric, M. B., Chandrasekaran, S., Frece, A., Hertis, M. and Srdic, G. (2010). *WS-BPEL 2.0 for SOA Composite Applications with IBM WebSphere 7*, Packt Publishing Ltd.
- Jürjens, J. (2002). UMLsec: Extending UML for Secure Systems Development, *UML '2002* The Unified Modeling Language **2460**: 412–425.
- Jürjens, J. (2005). *Secure systems development with UML*, Springer-Verlag, Berlin Heidelberg.
- Kabilan, V., Johannesson, P., Ruohomaa, S., Moen, P., Herrmann, A., Åhlfeldt, R.-M. and Weigand, H. (2007). Introducing the Common Non-Functional Ontology, in R. J. Gonçalves, J. P. Müller, K. Mertins and M. Zelm (eds), *Enterprise Interoperability II*, Springer London, pp. 633–645. URL:http://dx.doi.org/10.1007/978-1-84628-858-6_71
- Kagal, L. (2002). Rei: A policy language for the me-centric project. URL:<http://www.hpl.hp.com/techreports/2002/HPL-2002-270.pdf>
- Kagal, L., Finin, T., Paolucci, M., Srinivasan, N., Sycara, K. and Denker, G. (2004). Authorization and privacy for semantic Web services, *Intelligent Systems*, *IEEE* **19**(4): 50–56.
- Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Mieke, A., Saurel, C. and Trouessin, G. (2003). Organization based access control, *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, IEEE Computer Society, pp. 120 – 131.
- Känsälä, K. (1997). Integrating Risk Assessment with Cost Estimation, *IEEE Software* **14**: 61–67.
- Karlsch, M. (2007). *A model-driven framework for domain specific languages*, Master thesis, Hasso-Plattner-Institute of Software Systems Engineering.
- Kassab, M., Ormandjieva, O., Daneva, M. and Abran, A. (2008). Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP, in J. Cuadrado-Gallego, R. Braungarten, R. Dumke and A. Abran (eds), *Software Process and Product Measurement*, Vol. 4895 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 168–182.
- Kaviani, N., Gasevic, D., Milanovic, M., Hatala, M. and Mohabbati, B. (2008). Model-Driven Engineering of a General Policy Modeling Language, *Policies for Distributed Systems and Networks, IEEE International Workshop on* **0**: 101–104.
- Kelly, S. and Tolvanen, J.-P. (2008). *Domain-specific modeling: enabling full code generation*, Wiley-IEEE, Hoboken, New Jersey, USA.
- Kent, S. (2002). Model Driven Engineering, *IFM '02: Proceedings of the Third International Conference on Integrated Formal Methods*, Vol. 2335 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin / Heidelberg, pp. 286–298.
- Khan, K. M. and Han, J. (2002). Composing security-aware software, *Software*, *IEEE* **19**(1): 34–41.
- Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and Griswold, W. (2001). An Overview of AspectJ, in J. rgen Knudsen (ed.), *ECOOP 2001 - Object-Oriented Programming*, Vol. 2072 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 327–354.
- Kim, A., Luo, J. and Kang, M. (2005a). Security Ontology for Annotating Resources. URL:<http://www.nrl.navy.mil/chacs/pubs/05-1226-0470.pdf>

- Kim, A., Luo, J. and Kang, M. (2005b). Security Ontology for Annotating Resources, in R. Meersman and Z. Tari (eds), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, Vol. 3761 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 1483–1499.
- Kim, A., Luo, J. and Kang, M. (2007). Security Ontology to Facilitate Web Service Description and Discovery, *Journal on Data Semantics IX*, Vol. 4601 of *Lecture Notes in Computer Science*, Springer Berlin, pp. 167–195.
- Kim, H.-K. and Lee, R. Y. (2008). MS2Web: Applying MDA and SOA to Web Services, *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* **149/2008**: 163–180.
- Kleppe, A. G., Warmer, J. B. and Bast, W. (2003). *MDA explained: the model driven architecture : practice and promise*, Addison-Wesley, Boston, Massachusetts, USA.
- Krafzig, D., Banke, K. and Slama, D. (2004). *Enterprise SOA: Service-Oriented Architecture Best Practices*, Prentice Hall.
- Krautsevich, L., Martinelli, F. and Yautsiukhin, A. (2010). Formal approach to security metrics. What does “more secure” mean for you?, *2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, Xi’an, China.
- Krechetov, I., Tekinerdogan, B., Garcia, A., Chavez, C. and Kulesza, U. (2006). Towards an Integrated Aspect-Oriented Modeling Approach for, *Software Architecture Design. 8th Workshop on Aspect-Oriented Modelling (AOM.06)*, AOSD.06.
- Lamport, L. (2002). *Specifying systems: The TLA+ language and tools for hardware and software engineers*, Addison-Wesley, Reading.
- Lampson, B. W. (1974). Protection, *ACM SIGOPS Operating Systems Review* **8**(1): 18–24.
- Lara, R. and Roman, D. (2004). *A conceptual comparison of WSMO and OWL-S*, Vol. 3250 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 254–269.
- Larrucea, X. and Alonso, R. (2008). ISOAS: Through an independent SOA Security specification, *Composition-Based Software Systems, 2008. ICCBSS 2008. Seventh International Conference on*, IEEE Computer Society, Madrid, Spain, pp. 92–100.
- Larrucea, X. and Alonso, R. (2009). Modelling and Deploying Security Policies, *WEBIST 2009 - Proceedings of the Fifth International Conference on Web Information Systems and Technologies*, INSTICC Press, Lisboa, Portugal, pp. 411–414.
- Lédeczi, Á., Bakay, Á., Maróti, M., Völgyesi, P., Nordstrom, G., Sprinkle, J. and Karsai, G. (2001). Composing Domain-Specific Design Environments, *Computer* **34**(11): 44–51.
- Lehmann, K. and Matthes, F. (2005). Meta Model Based Integration of Role-Based and Discretionary Access Control Using Path Expressions, *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, IEEE Computer Society, Washington, DC, USA, pp. 443–446.
- Lemrabet, Y., Touzi, J., Clin, D., Bigand, M. and Jean-Pierre-Bourey (2010). Mapping of BPMN models into UML models using SoaML profile, *Proceedings of the 8th ENIM IFAC International Conference of Modeling and Simulation*, Lavoisier, Hammamet, Tunisie, pp. 1668–1673.

- Liang, Z. (2009). *A Meta-Modelling language definition for specific domain.*, Phdthesis, De Montfort University.
- Lin-lin, Z., Shi, Y., You-cong, N., Jing, W., Kai, Z. and Peng, Y. (2008). Towards Multi-Dimensional Separating of NFRs in Software Architecture, *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*, IEEE Computer Society, Washington, DC, USA, pp. 104–107.
- Liu, Y. (2008). *Quantitative security analysis for service-oriented software architectures*, Phd thesis, University of Victoria.
- Liu, Y. and Traore, I. (2007). Systematic Security Analysis for Service-Oriented Software Architectures, *IEEE International Conference on e-Business Engineering*.
- Lochmann, H. and Hesselund, A. (2009). An integrated view on modeling with multiple domain-specific languages, *Proceedings of the IASTED International Conference Software Engineering (SE 2009)*, ACTA Press, pp. 1–10.
- Lodderstedt, T., Basin, D. and Doser, J. (2002). SecureUML: A UML-based modeling language for model-driven security, *UML 2002 - The Unified Modeling Language* **2460/2002**: 426–441.
- Lohmann, D. and Ebert, J. (2003). A generalization of hyperspace approach using meta-models, *The 2003 AOSD Early Aspects Workshop (AOSD-EAWS'03)*.
- Lopes, D., Hammoudi, S., Bézivin, J. and Jouault, F. (2005). Generating transformation definition from mapping specification: Application to web service platform, *Advanced Information Systems Engineering* **3520/2005**: 309–325.
- López-Grao, J. P., Merseguer, J. and Campos, J. (2004). From UML activity diagrams to Stochastic Petri nets: application to software performance engineering, *SIGSOFT Softw. Eng. Notes* **29**(1): 25–36.
- López-Sanz, M., Qayyum, Z., Cuesta, C., Marcos, E. and Oquendo, F. (2008). Representing Service-Oriented Architectural Models Using π -ADL, in R. Morrison, D. Balasubramaniam and K. Falkner (eds), *Software Architecture*, Vol. 5292 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 273–280.
- Lukman, T. and Mernik, M. (2008). Model-Driven Engineering and its introduction with metamodeling tools, in P. B. (ur.) V: GASPERIN Matej (ur.) (ed.), *9th International PhD Workshop on Systems and Control: Young Generation Viewpoint*, Izola, Slovenia.
- Macek, O. and Richta, K. (2009). The BPM to UML activity diagram transformation using XSLT, in K. Richta, J. Pokorný and V. Snásel (eds), *Proceedings of the DATESO 2009 Annual International Workshop on DATABASES, TEXTS, SPECIFICATIONS AND OBJECTS, SPINDLERUV MLYN, Czech Republic, April 15-17, 2009*, Vol. 471 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 119–129.
- Majzik, I., Pataricza, A. and Bondavalli, A. (2003). Stochastic Dependability Analysis of System Architecture Based on UML Models, in R. de Lemos, C. Gacek and A. Romanovsky (eds), *Architecting Dependable Systems*, Vol. 2677 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 219–244.
- Marcos, E., Acuña, C. J. and Cuesta, C. E. (2006). Integrating software architecture into a mda framework, in V. Gruhn and F. Oquendo (eds), *EWSA*, Vol. 4344 of *Lecture Notes in Computer Science*, Springer, pp. 127–143.

- Mellor, S. J., Clark, A. and Futagami, T. (2003). Model-Driven Development, *Ieee Software* **20**(5): 14–18.
- Mellor, S. J., Scott, K. and Weise, D. (2004). *MDA distilled: principles of model-driven architecture*, Addison-Wesley.
- Mencl, V. and Bures, T. (2005). Microcomponent-Based Component Controllers: A Foundation for Component Aspects, *Asia-Pacific Software Engineering Conference* **0**: 729–737.
- Menzel, M. and Meinel, C. (2009). A Security Meta-model for Service-Oriented Architectures, *2009 IEEE International Conference on Services Computing*, IEEE, Bangalore, India, pp. 251–259.
- Menzel, M. and Meinel, C. (2010). SecureSOA, *Services Computing, IEEE International Conference on* **0**: 146–153.
- Menzel, M., Thomas, I. and Meinel, C. (2009). Security requirements specification in service-oriented business process management, *2009 International Conference on Availability, Reliability and Security*, Vol. 0, IEEE Computer Society, Fukuoka, Japan, pp. 41—48.
- Michael Menzel (2010). Modelling Security in Service-oriented Architectures, in Christoph Meinel, Hasso Plattner, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Neumann and Holger Giese (eds), *Proceedings of the 4th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering*, Universitätsverlag Potsdam, Postdam, Germany, p. 248.
- Microsoft (n.d.). Microsoft Domain-Specific Language (DSL) Tools.
URL:<http://www.microsoft.com/download/en/details.aspx?id=2379>
- Miller, C. (2009). An analysis of model-driven development.
URL:<http://www.scribd.com/doc/27134123/An-Analysis-of-Model-Driven-Development>
- Monperrus, M., Jezequel, J.-M., Champeau, J. and Hoeltzener, B. (2008). Measuring Models, in J. Rech and C. Bunse (eds), *Model-Driven Software Development: Integrating Quality Assurance*, IDEA Group.
- Monperrus, M., Jézéquel, J.-M., Champeau, J. and Hoeltzener, B. (2009). *Measuring models*, IGI Global, chapter VII, pp. 147–169.
- Montangero, C. and Semini, L. (2008). Barbed Model-Driven Software Development: A Case Study, *Electronic Notes in Theoretical Computer Science* **207**: 171–186.
- Mouelhi, T., Baudry, B. and Fleurey, F. (2008). A generic metamodel for security policies mutation, *SecTest 08 : 1st International ICST workshop on Security Testing*, RSM - D{é}pt. R{é}seaux, S{é}curit{é} et Multim{é}dia (Institut T{é}l{é}com-T{é}l{é}com Bretagne), IRISA - Institut de recherche en informatique et syst{è}mes al{é}atoires (INRIA), SINTEF - The Foundation for Scientific and Industrial Research (SINTEF).
- Mouelhi, T., Fleurey, F., Baudry, B. and Le Traon, Y. (2008). Mutating DAC And MAC Security Policies: A Generic Metamodel Based Approach, *Modeling Security Workshop In Association with MODELS '08, 28th September, Toulouse, France*, Toulouse, France.
- Mouelhi, T., Fleurey, F., Baudry, B. and Le Traon, Y. (2010). A model-based framework for security policy specification, deployment and testing, *Model Driven Engineering Languages and Systems* **5301/2010**: 537–552.

- Mouratidis, H. and Giorgini, P. (2007). Secure Tropos: A Security-Oriented Extension of the Tropos methodology, *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)* **17**(2): 285–309.
- Mouratidis, H., Giorgini, P. and Manson, G. (2003). Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems, in J. Eder and M. Missikoff (eds), *Advanced Information Systems Engineering*, Vol. 2681 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, p. 1031.
- Nakamura, Y., Satoh, F. and Chung, H.-V. (2007). Syntactic Validation of Web Services Security Policies, in B. J. Krämer, K.-J. Lin and P. Narasimhan (eds), *Service-Oriented Computing - ICSOC 2007, Fifth International Conference*, Vol. 4749 of *Lecture Notes in Computer Science*, Springer, Vienna, Austria, pp. 319–329.
- Nakamura, Y., Tatsubori, M., Imamura, T. and Ono, K. (2005). Model-Driven Security Based on a Web Services Security Architecture, *SCC '05: Proceedings of the 2005 IEEE International Conference on Services Computing*, IEEE Computer Society, Washington, DC, USA, pp. 7–15.
- Navasa, A., Pérez-Toledano, M. A. and Murillo, J. M. (2009). An ADL dealing with aspects at software architecture stage, *Information and Software Technology* **51**(2): 306–324.
- Nelson, T., Cowan, D. D. and Alencar, P. S. C. (2001). Supporting Formal Verification of Crosscutting Concerns, *Lecture Notes In Computer Science; Vol. 2192*.
- Neubauer, T. and Heurix, J. (2008a). Defining Secure Business Processes with Respect to Multiple Objectives, *Availability, Reliability and Security, International Conference on* **0**: 187–194.
- Neubauer, T. and Heurix, J. (2008b). Objective Types for the Valuation of Secure Business Processes, *Computer and Information Science, ACIS International Conference on* **0**: 231–236.
- OASIS (2006). Web Services Atomic Transaction (WS-AtomicTransaction) 1.1.
- OASIS (2007a). Service Component Architecture (SCA) | OASIS Open CSA.
URL:<http://www.oasis-open.org/sca>
- OASIS (2007b). Web Services Business Process Execution Language Version 2.0.
URL:<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- OASIS (2007c). WS-SecurityPolicy 1.2.
- OASIS (2007d). WS-SecurityPolicy 1.2.
- OASIS SCA Policy TC (2010). SCA Policy Framework Specification Version 1.1 CD03.
- OASIS Web Services Security (WSS) TC (2006). Web Services Security: SAML Token Profile 1.1 OASIS Standard, 1 February.
- OBEO (2006). Acceleo.
URL:<http://www.eclipse.org/acceleo/>
- OBEO (2010). Acceleo Spring Module.
URL:<http://www.acceleo.org/pages/uml2-to-jee-java-struts-hibernate-generator/en>
- Oberortner, E., Zdun, U. and Dustdar, S. (2008). Domain-specific languages for service-oriented architectures: An explorative study, *Towards a Service-Based Internet* **5377/2008**: 159–170.

- O'Brien, L., Bass, L. and Merson, P. (2005). Quality Attributes and Service-Oriented Architectures, *Technical report*, Defense Technical Information Center OAI-PMH Repository [<http://stinet.dtic.mil/oai/oai>] (United States).
- O'Brien, L., Merson, P. and Bass, L. (2007). Quality Attributes for Service-Oriented Architectures, *International Workshop on Systems Development in SOA Environments (SDSOA'07: ICSE Workshops 2007)* pp. 3–9.
- Odell, J., Parunak, H. V. D. and Bauer, B. (2000). Extending uml for agents, *Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, pp. 3–17.
- Oladimeji, E. A., Supakkul, S. and Chung, L. (2007). A model-driven approach to architecting secure software, *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, Knowledge Systems Institute Graduate School, Boston, USA, pp. 535–551.
- OMG (2001). Model Driven Architecture - A Technical Perspective .
URL:<http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01>
- OMG (2003). MDA Guide Version 1.0.1.
URL:<http://www.omg.org/cgi-bin/doc?omg/03-06-01>
- OMG (2004). UML Profile for Enterprise Distributed Object Computing.
- OMG (2005a). Reusable Asset Specification.
- OMG (2005b). UML Profile for Schedulability, Performance, and Time Specification.
- OMG (2007). Specification. A UML Profile for MARTE.
- OMG (2008a). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification.
URL:<http://scholar.google.com/scholar?q=Query+View+Transformation#0>
- OMG (2008b). MOF Model to Text Transformation Language, v1.0.
- OMG (2008c). Request for Information. Handling Non-Functional Properties in Service Oriented Architecture.
- OMG (2008d). UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms Version 1.1.
- OMG (2009). Service oriented architecture Modeling Language (SoaML)-Specification for the UML Profile and Metamodel for Services (UPMS).
- OMG (2010a). Diagram Definition Version 1.0 - FTF Beta 1.
- OMG (2010b). MDA Specifications.
URL:<http://www.omg.org/mda/specs.htm>
- OMG (2010c). Object Constraint Language, Version 2.3.
- OMG (2010d). OMG Trademarks.
URL:http://www.omg.org/legal/tm_list.htm
- OMG (2010e). Unified Modeling Language (UML) Specification.
URL:<http://www.omg.org/spec/UML/2.3/>

- OMG (2011a). Business Process Model and Notation (BPMN).
- OMG (2011b). OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1.
- OMG (2011c). OMG MOF 2 XMI Mapping Specification, Version 2.4.1.
URL:<http://www.omg.org/spec/XMI/>
- Ono, K., Nakamura, Y., Satoh, F. and Tateishi, T. (2007). Verifying the Consistency of Security Policies by Abstracting into Security Types, *2007 IEEE International Conference on Web Services (ICWS 2007)*, IEEE Computer Society, Salt Lake City, Utah, USA, pp. 497–504.
- Oquendo, F. (2006). pi-Method: a model-driven formal method for architecture-centric software engineering, *ACM SIGSOFT Software Engineering Notes* **31**(3): 1–13.
- Oquendo, F. (2008). Formal Approach for the Development of Business Processes in Terms of Service-Oriented Architectures Using Pi-ADL, *Service-Oriented System Engineering, IEEE International Workshop on* **0**: 154–159.
- Ortiz, G., Bordbar, B. and Hernández, J. (2008). Evaluating the use of AOP and MDA in web service development, *Internet and Web Applications and Services, 2008. ICIW '08. Third International Conference on* pp. 78 – 83.
- Ortiz, G. and Hernández, J. (2006). Service-oriented model-driven development: Filling the extra-functional property gap, *Service-Oriented Computing-ICSOC 2006* **4294/2006**: 471–476.
- Ossher, H. and Tarr, P. (2000). Multi-Dimensional Separation of Concerns and The Hyperspace Approach, *Proceedings of the Symposium on Software Architectures and Component Technology The State of the Art in Software Development*, Kluwer, pp. 293–323.
- Palma, K., Eterovic, Y. and Murillo, J. M. (2006). Extending the Rapid ADL to Specify Aspect-Oriented Software Architectures, *15th International Conference on Software Engineering and Data Engineering (SEDE-2006)*, ISCA, Los Angeles, California, USA, pp. 170–167.
- Paoli, F. D., Palmonari, M., Comerio, M. and Maurino, A. (2008). A Meta-model for Non-functional Property Descriptions of Web Services, *ICWS '08: Proceedings of the 2008 IEEE International Conference on Web Services*, IEEE Computer Society, Washington, DC, USA, pp. 393–400.
- Papazoglou, M. P., Traverso, P., Dustdar, S. and Leymann, F. (2008). Service-oriented computing: a research roadmap, *Int. J. Cooperative Inf. Syst.* **17**(2): 223–255.
- Patrascoiu, O. (2004). Mapping EDOC to Web services using YATL, *Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04)* **0**: 286–297.
- Pavlich-mariscal, J., Michel, L. and Demurjian, S. (2007). Enhancing UML to Model Custom Security Aspects [Position Paper], *Proc. of 11th International Workshop on Aspect-Oriented Modeling, co-located with MODELS 2007*, Nashville, TN, USA.
- Pérez, J., Ramos, I., Jaén, J., Letelier, P. and Navarro, E. (2003). PRISMA: Towards Quality, Aspect Oriented and Dynamic Software Architectures, *Quality Software, International Conference on* **0**: 59.
- Pessemier, N. and Seinturier, L. (2004). Components, ADL and AOP: Towards a Common Approach, *In Workshop ECOOP Reflection, AOP and Meta-Data for Software Evolution (RAM-SE04)*, pp. 61—69.

- Pettersson, A. (2006). *Service-Oriented Architecture (SOA) quality attributes – A research model*, Master thesis, University of Lund.
- Pinto, M. and Fuentes, L. (2007). AO-ADL: An ADL for Describing Aspect-Oriented Architectures, in A. Moreira and J. Grundy (eds), *Early Aspects: Current Challenges and Future Directions*, Vol. 4765 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 94–114.
- Pinto, M., Fuentes, L. and Troya, J. M. (2005). A Dynamic Component and Aspect-Oriented Platform, *The Computer Journal* **48**(4): 401–420.
- Pitkänen, R. and Mikkonen, T. (2006). Lightweight Domain-Specific Modeling and Model-Driven Development, *OOPSLA 6th Workshop on Domain Specific Modeling*, Portland, Oregon, USA, pp. 159–168.
- Pooley, R. (1999). Using UML to Derive Stochastic Process Algebra Models, *UKPEW '99, Proceedings of the Fifteenth UK Performance Engineering Workshop*, The University of Bristol, pp. 23–33.
- Preda, S., Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. and Toutain, L. (2010). Model-Driven Security Policy Deployment: Property Oriented Approach, *Lecture Notes in Computer Science - Engineering Secure Software and Systems* **5965/2010**: 123–139.
- Priebe, T., Fernandez, E., Mehlaui, J. and Pernul, G. (2004). A Pattern System for Access Control, in C. Farkas and P. Samarati (eds), *Research Directions in Data and Applications Security XVIII*, Vol. 144 of *IFIP International Federation for Information Processing*, Springer Boston, pp. 235–249.
- Protopsaltou, A. (2007). *Rapid prototyping of web applications combining Ruby on Rails and Reverse Engineering*, Master thesis, IT UNIVERSITY OF GÖTEBORG.
- Rashid, A., Sawyer, P., Moreira, A. and Araújo, J. (2002). Early Aspects: A Model for Aspect-Oriented Requirements Engineering, *Requirements Engineering, IEEE International Conference on* **0**: 199.
- Reddy, Y. R., Ghosh, S., France, R. B., Straw, G., Bieman, J. M., McEachen, N., Song, E. and Georg, G. (2006). Directives for Composing Aspect-Oriented Design Class Models, *Trans on Aspect Oriented Development* **3880**(1): 75–105.
- Reznik, J., Ritter, T., Schreiner, R. and Lang, U. (2007). Model driven development of security aspects, *Electronic Notes in Theoretical Computer Science (2007)* **163**(2): 65–79.
- Rodrigues, G., Roberts, G. and Emmerich, W. (2004). *Reliability support for the model driven architecture*, Vol. 3069 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 394–412.
- Rodrigues, G., Rosenblum, D. and Uchitel, S. (2005). Reliability Prediction in Model-Driven Development, in L. Briand and C. Williams (eds), *Model Driven Engineering Languages and Systems*, Vol. 3713 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 339–354.
- Rodríguez, A., de Guzmán, I. G. R., Fernández-Medina, E. and Piattini, M. (2010). Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach, *Information & Software Technology* **52**(9): 945–971.

- Rodríguez, A., Fernández-Medina, E. and Piattini, M. (2007). A BPMN Extension for the Modeling of Security Requirements in Business Processes, *IEICE - Trans. Inf. Syst.* **E90-D**(4): 745–752.
- Röttger, S. and Zschaler, S. (2006). Tool support for refinement of non-functional specifications, *Software and Systems Modeling journal (SoSyM)* .
- Routledge, N., Bird, L. and Goodchild, A. (2002). UML and XML schema, *Australian Computer Science Communications* **24**(2): 157–166.
- Sánchez, P., Moreira, A., Fuentes, L., Araújo, J. and Magno, J. (2010). Model-driven development for early aspects, *Information and Software Technology* **52**(3): 249–273.
- Sandhu, R., Coyne, E., Feinstein, H. and Youman, C. (2002). Role-based access control models, *IEEE Computer* **29**(2): 38 – 47.
- Satoh, F., Nakamura, Y., Mukhi, N., Tatsubori, M. and Ono, K. (2008). Methodology and Tools for End-to-End SOA Security Configurations, *2008 IEEE Congress on Services, SERVICES I*, IEEE Computer Society, Honolulu, Hawaii, USA, pp. 307–314.
- Satoh, F., Nakamura, Y. and Ono, K. (2006). Adding Authentication to Model Driven Security, *Web Services, 2006. ICWS '06. International Conference on*, pp. 585–594.
- Satoh, F. and Uramoto, N. (2010). Validating Security Policy Conformance with WS-Security Requirements, in I. Echizen, N. Kunihiro and R. Sasaki (eds), *Advances in Information and Computer Security*, Vol. 6434 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 133–148.
- Satoh, F. and Yamaguchi, Y. (2007). Generic Security Policy Transformation Framework for WS-Security, *2007 IEEE International Conference on Web Services (ICWS 2007)*, IEEE Computer Society, Salt Lake City, Utah, USA, pp. 513–520.
- Schmidt, D. C. (2006). Guest Editor's Introduction: Model-Driven Engineering, *IEEE Computer* **39**(2): 25–31.
- SecureMova (n.d.). SecureMova.
URL:http://maude.sip.ucm.es/mova/index.php?option=com_content&task=view&id=39&Itemid=29
- Silva Gallino, J. P. (2006). 12.02.2 Definición del dominio de trabajo, estado del arte, análisis de situación y perspectivas de evolución de MDA: Estado del arte del modelado de arquitecturas de servicios.
- Silva Gallino, J. P. (2007). 12.02.4.01 T23 Lenguaje PSM para plataformas SOA.
- Silva Gallino, J. P. (2008). 12.10.01.01 TXX Estado del arte del Modelado de Aspectos (AOM).
- Silva Gallino, J. P., de Miguel, M. A., Briones, J. F. and Alonso, A. (2005). Safety Metrics for the Analysis of Software Architectures, in H. Giese, I. H. Krüger and K. M. L. Cooper (eds), *Workshop on Visual Modeling for Software Intensive Systems*, Proceedings of 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05, Dallas, USA.
- Silva Gallino, J. P., de Miguel, M. A., Briones, J. F. and Alonso, A. (2009a). Composing Models with Six Different Tools: a Comparative Study, in F. Jouault (ed.), *Proceedings of the 1st International Workshop on Model Transformation with ATL*, Jouault, Frédéric, Nantes, France, pp. 103–118.

- Silva Gallino, J. P., de Miguel, M. A., Briones, J. F. and Alonso, A. (2009b). Desarrollo Guiado por Modelos de una Arquitectura Orientada a Servicios Web y Políticas de Seguridad, *in* C. J. Acuña, V. de Castro, A. Ruiz-Cortés and D. Pascual (eds), *V Jornadas Científico-Técnicas en Servicios Web y SOA, JSWEB 2009*, Acuña, César J. de Castro, Valeria Ruiz-Cortés, Antonio Pascual, David, Madrid, Spain, pp. 251–257.
- Silva Gallino, J. P., de Miguel, M. A., Briones, J. F. and Alonso, A. (2010). Model-Driven Development of a Web Service-Oriented Architecture and Security Policies, *2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, IEEE Computer Society, Los Alamitos, CA, USA, Carmona, Spain, pp. 92–96.
- Silva Gallino, J. P., de Miguel, M. A., Briones, J. F. and Alonso, A. (2011). Domain-Specific Multi-Modeling of Security Concerns in Service-Oriented Architectures, *LNCS - 8th International Workshop on Web Services and Formal Methods, WS-FM'11*.
- Silva Gallino, Juan Pedro and de Miguel, Miguel and Briones, Javier F. and Alonso, Alejandro (2011). Multi Domain-Specific Modeling of the Security Concerns of Service-Oriented Architectures, *Services Computing, IEEE International Conference on* **0**: 761–762.
- Simon, H. A. (1996). *The Sciences of the Artificial - 3rd Edition*, The MIT Press.
- Skene, J. and Emmerich, W. (2003). A model-driven approach to non-functional analysis of software architectures, *Automated Software Engineering, International Conference on* **0**: 236.
- Souza, A., Silva, B., Lins, F., Damasceno, J., Rosa, N., Maciel, P., Medeiros, R., Stephenson, B., Motahari-Nezhad, H., Li, J. and Northfleet, C. (2009). Incorporating Security Requirements into Service Composition: From Modelling to Execution, *in* L. Baresi, C.-H. Chi and J. Suzuki (eds), *Service-Oriented Computing*, Vol. 5900 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 373–388.
- SpringSource (2010a). Spring Framework.
URL:<http://www.springsource.org/>
- SpringSource (2010b). Spring Web Services.
URL:<http://static.springsource.org/spring-ws/sites/1.5/>
- Steinberg, D., Budinsky, F., Paternostro, M. and Merks, E. (2009). *EMF Eclipse Modelling Framework*, second edn, Pearson Education, Addison-Wesley, Boston, USA.
- Sun, H., Zhao, W. and Yang, J. (2010). SOAC: A Conceptual Model for Managing Service-Oriented Authorization, *Services Computing, IEEE International Conference on* **0**: 546–553.
- Sztipanovits, J. and Karsai, G. (1997). Model-integrated computing, *Computer* **30**(4): 110–111.
- Tarr, P., Ossher, H., Harrison, W. and Sutton Jr., S. M. (1999). N degrees of separation: multi-dimensional separation of concerns, *International Conference on Software Engineering* pp. 107 – 119.
- Toma, I., Foxvog, D., Paoli, F. D., Comerio, M., Palmonari, M. and Maurino, A. (2008). Non-functional properties in web services.
- Tsai, W.-T., Cao, Z., Wei, X., Paul, R., Huang, Q. and Sun, X. (2007). Modeling and simulation in service-oriented software development, *Simulation* **83**(1): 7–32.

- Tsai, W.-T., Paul, R., Wei, X. and Cao, Z. (2005). PSML-S: A process specification and modeling language for service oriented computing, *The 9th IASTED International Conference on Software Engineering and Applications (SEA)*, Phoenix, AZ, USA.
- Tsai, W.-T., Zhou, X. and Wei, X. (2008). A policy enforcement framework for verification and control of service collaboration, *Information Systems and E-Business . . . .*
- Valeria de Castro and Esperanza Marcos and Juan M. Vara (2011). Applying CIM-to-PIM model transformations for the service-oriented development of information systems, *Information & Software Technology* **53**(1): 87–105.
- Vallecillo, A. (2010). On the Combination of Domain Specific Modeling Languages, *Modelling Foundations and Applications* pp. 305–320.
URL:<http://www.springerlink.com/index/Q750875H113QH2X1.pdf>
- W3C (2001). Web services description language (wsdl) 1.1, W3C Note.
- W3C (2006a). Web Services Policy 1.2 - Framework (WS-Policy).
- W3C (2006b). Web services policy 1.2-attachment (WS-policyattachment).
- Wada, H., Suzuki, J. and Oba, K. (2006). Modeling Non-Functional Aspects in Service Oriented Architecture, *Services Computing, IEEE International Conference on* **0**: 222–229.
- Wada, H., Suzuki, J. and Oba, K. (2007). A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture, *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pp. 187–195.
- Wada, H., Suzuki, J. and Oba, K. (2008). Early Aspects for Non-Functional Properties in Service Oriented Business Processes, *Services, IEEE Congress on* **0**: 231–238.
- Wada, H., Suzuki, J. and Oba, K. (2009). *A Model-Driven Development Framework for Non-Functional Aspects in Service Oriented Architecture*, Software Applications, IGI Global, US; Hershey, PA, pp. 942–974.
- Waddington, D. and Lardieri, P. (2006). Model-centric software development, *IEEE Computer* **39**(2): 28–29.
- Wassermann, B. and Emmerich, W. (2006). Reliable scientific service compositions, *Service-Oriented Computing ICSOC 2006* .
- Web Services Interoperability Organization (2006). WS-I Basic Profile - Version 1.1.
URL:<http://www.ws-i.org/Profiles/BasicProfile-1.1.html>
- Web Services Interoperability Organization (2010). WS-I Basic Security Profile Version 1.1.
URL:<http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html>
- Web Services Interoperability Organization (n.d.). <http://www.ws-i.org>.
URL:<http://www.ws-i.org/>
- Wehrmeister, M. A., Freitas, E. P., Pereira, C. E. and Wagner, F. R. (2007). An Aspect-Oriented Approach for Dealing with Non-Functional Requirements in a Model-Driven Development of Distributed Embedded Real-Time Systems, *Object-Oriented Real-Time Distributed Computing, IEEE International Symposium on* **0**: 428–432.

- Wolter, C., Menzel, M. and Meinel, C. (2008). Modelling security goals in business processes, *Proceedings of the GI Modellierung 2008*.
- Wolter, C., Menzel, M., Schaad, A., Miseldine, P. and Meinel, C. (2009). Model-driven business process security requirement specification, *J. Syst. Archit.* **55**(4): 211–223.
- Wolter, C. and Schaad, A. (2007). Modeling of task-based authorization constraints in BPMN, *BPM'07: Proceedings of the 5th international conference on Business process management*, Springer-Verlag, Berlin, Heidelberg, pp. 64–79.
- WS-I (2003a). Sample Architecture Usage Scenarios.
- WS-I (2003b). Supply Chain Management Sample Architecture.
URL:<http://www.ws-i.org/SampleApplications/SupplyChainManagement/2003-12/SCMArchitecture1.01.pdf>
- WS-I (2006). Sample Applications Security Architecture Document.
URL:<http://www.ws-i.org/SampleApplications/SupplyChainManagement/2006-04/SCMSecurityArchitectureWGD5.00.doc>
- Yie, A., Casallas, R., Deridder, D. and Wagelaar, D. (2009). A practical approach to multi-modeling views composition, *ECEASST* **21**.
- Yie, A. and Wagelaar, D. (2009). Advanced Traceability for ATL, in F. Jouault (ed.), *Model Transformation with ATL (MtATL2009)*, CEUR-WS, Nantes, France.
- Yu, X., Zhang, Y., Zhang, T., Wang, L., Hu, J., Zhao, J. and Li, X. (2007). A model-driven development framework for enterprise Web services, *Information Systems Frontiers* **9**(4): 391–409.
- Yuan, E. and Tong, J. (2005). Attributed Based Access Control (ABAC) for Web Services, *Proceedings of the IEEE International Conference on Web Services, ICWS '05*, IEEE Computer Society, Washington, DC, USA, pp. 561–569.
URL:<http://dx.doi.org/10.1109/ICWS.2005.25>
- Zarras, A., Vassiliadis, P. and Issarny, V. (2004). *Model-driven dependability analysis of webservices*, Vol. 3291/2004 of *Lecture Notes in Computer Sciences*, Springer-Verlag Berlin Heidelberg, pp. 1608–1625.
- Zhu, L. and Liu, Y. (2009). Model Driven Development with non-functional aspects, *International Conference on Software Engineering*.
- Zimmermann, O., Krogdahl, P. and Gee, C. (2004). Elements of Service-Oriented Analysis and Design.
URL:<http://www.ibm.com/developerworks/webservices/library/ws-soad1/>
- Zschaler, S. (2010). Formal specification of non-functional properties of component-based software systems, *Software and Systems Modeling*.