

IMT Institute for Advanced Studies, Lucca

Lucca, Italy

**Business-Oriented Model Driven Development
of Service Oriented Architectures**

PhD Program in Computer Science and Engineering

XXI Cycle

By

Luca Abeti

2009

Program Coordinator:
Prof. Ugo Montanari,
Computer Science Department, University of Pisa

Supervisor:
Prof. Paolo Ciancarini,
Computer Science Department, University of Bologna
ciancarini@cs.unibo.it

Tutor:
Prof. Paolo Ciancarini,
Computer Science Department, University of Bologna

The dissertation of Luca Abeti has been reviewed by:

Prof. Daniela Damian,
Department of Computer Science, University of Victoria

Prof. John Mylopoulos,
Dep. of Information and Communication Technology, University of Trento

IMT Institute for Advanced Studies, Lucca

2009

to Myself

Contents

List of Figures	x
List of Tables	xii
Acknowledgements	xiii
Vita and Publications	xv
Abstract	xviii
1 Introduction	1
1.1 Motivations	3
1.1.1 Global Software Engineering of SOA's	3
1.1.2 The Need for a Business-Oriented Approach to Software Services Development	5
1.2 Contributions of the Thesis	10
1.2.1 General Objectives for the Thesis	11
1.2.2 Summary of Contributions and Limitations	12
1.3 Previous Work of the Author Related to the Thesis	17
1.4 Services in e-Government and Civil Protection	23
1.4.1 Services for Critical Infrastructure Protection and Emergency Management	24
1.4.2 The Marche Region SISSI Scenario	28
1.4.3 An Inquiry on the Causes of the SISSI Failure	30
1.5 Outline of the Thesis	32

2	State of Art of Business-Oriented Model Driven Development	34
2.1	Business Modeling and Reengineering	35
2.1.1	A Historical Perspective of Business Modeling . . .	36
2.1.2	The Discipline of Business Modeling in Computer Science	39
2.1.3	Business Process Modeling is not Process Modeling	45
2.1.4	Notations for Business Process Modeling	51
2.1.5	The ATHENA Model-Driven Interoperability Frame- work	53
2.2	Global Requirements Engineering	59
2.2.1	Requirements Engineering	60
2.2.2	Requirements in Distributed Projects	64
2.2.3	The SOP-wiki and Softwiki Projects	73
2.2.4	The Goal Oriented Approach to Requirements En- gineering	76
2.3	Modeling Enterprise-centric Computing	93
2.3.1	MDE and MDA in Enterprise Computing	94
2.3.2	MDA Tools	101
2.3.3	How Eclipse supports MDA	104
3	My proposal: the Enterprise-Service-Implementation (ESI) De- sign Method	115
3.1	A Business-oriented Approach to Software Modeling . . .	117
3.1.1	The Enterprise Modeling phase	120
3.1.2	The Service Modeling phase	128
3.1.3	Platform Specific Implementations	129
4	My Technological Framework	132
4.1	The Framework Transformations	132
4.1.1	The Goals 2 Services Transformation	132
4.1.2	The Services 2 Web Services & Portlets Transforma- tions	136
4.2	Plug-in Architectures as Patterns for SOA Interaction . . .	138
4.2.1	SOA Infrastructures	139
4.2.2	Plug-ins as a Reference Architecture for ESB's . . .	143

5	My Tools Supporting the ESI Method	147
5.1	Semantic Wikis for Requirements Engineering	147
5.1.1	The WikiReq tool	148
5.1.2	WikiReq argumentation feature	153
5.1.3	WikiReq to Eclipse export	154
5.1.4	WikiReq: an Example Scenario	155
5.2	The SMOTE Tool	157
5.2.1	Goals2Service as a CIM2PIM MDA Transformation	157
5.2.2	Services2WebServices&Portlets as PIM2PSM MDA Transformations	163
6	The Civil Protection Case Study	167
6.1	The Case Study	167
6.1.1	The IDEA Enterprise Modeling Phase	168
6.1.2	The IDEA Service Modeling Phase	173
6.1.3	The IDEA Platform Specific Implementation Phase	177
6.2	Critical Evaluation and Results of the Experiment	180
6.2.1	Lessons Learned and Threats to Validity	184
7	Conclusions and Future Works	186
	Bibliography	193
	URL's	210

List of Figures

1	The ISA framework matrix	41
2	Roles and Artifacts of business modeling in RUP	43
3	Modeling levels in BPM.	47
4	Theory family tree for BPM standards as envisioned in (86)	50
5	BPMN core: flow objects	52
6	BPMN core: connecting objects, swimlines and artifacts . .	54
7	Research areas of the ATHENA Interoperability Project . .	55
8	The ATHENA Realization Framework	59
9	Global Requirements Engineering main stakeholders as described in (57)	65
10	The SOP-wiki Architecture	74
11	Pros and Cons of wikis used in requirements engineering .	76
12	The Tropos metamodel of the actor concept and the dependency relation specified by means of UML.	86
13	The paint "The Treachery of Images", René Magritte, 1929.	96
14	The OMG 4-layer Metamodeling Architecture.	99
15	Model to model transformation.	100
16	Our three views for Business Modeling.	119
17	The three main phases of our ESI method.	121
18	The Si*, BMPN and UseCase graphical syntaxes exploited by the ESI method.	131
19	Our BPR framework architecture	133

20	WikiReq tool: Main Page	149
21	WikiReq tool: the Actor semantic form	150
22	WikiReq tool: the Actor viewpoint page	152
23	The actor model exported from WikiReq by the RDFtoEMF script and loaded in the TAOM4E plug-in	156
24	The SMOTE Architecture	158
25	SMOTE tool: the Si* metamodel used by SMOTE	159
26	SMOTE tool: the BPMN metamodel used by SMOTE	160
27	SMOTE tool: a screenshot of the SMOTE ATL configuration	161
28	SMOTE tool: the metamodel for Service PIM's	164
29	The WikiReq Actor Viewpoint page for the IDEA project	171
30	The <i>To speed up administrative processes</i> goal page in WikiReq	172
31	A subset of the IDEA stakeholders goals	173
32	A sceenshot of the IDEA Service Model in SMOTE	178
33	A screenshot of the IDEA palmtop software	179
34	The results of the questionnaire about the WikiReq system.	182

List of Tables

1	A glossary for business processes related terms	111
2	TOP 5 threats faced by distributed projects identified by Smite (147)	112
3	A comparison among wiki platforms and CMS platforms features in GSE	112
4	The Tropos main concepts	113
5	An overview on tools that currently support MDA	114

Acknowledgements

First of all, I wish to express my gratitude to my tutor and mentor, Prof. Paolo Ciancarini, for his wise advices and his bear with me. I thank the Coordinator of the CSE program, Prof. Ugo Montanari, for the chance to study at IMT Lucca. I would like to gratefully acknowledge my evaluation committee, Prof. Paola Inverardi and Prof. Davide Rossi for their suggestions. I am forever indebted to Dr. Rocco Moretti for the hard work we carry out in these years. I am also grateful to Prof. Maurizio Lenzerini for the involvement in the TOCAI.it FIRB project; the support given by the CINI consortium; my two external referees Prof. John Mylopoulos and Prof. Daniela Damian; Prof. Fausto Marincioni and the IMT PhD Office, in particular, Silvia Lucchesi and Silvia Doberti.

Desidero esprimere la mia gratitudine in primo luogo al mio tutor e mentore, il Prof. Paolo Ciancarini, per i saggi insegnamenti e la pazienza dimostratami in questo articolato e lungo percorso formativo, per aver sempre creduto in me e nelle mie capacità, introducendomi con rigore nell'avvincente e complesso mondo della ricerca scientifica. Ringrazio il coordinatore del dottorato CSE IMT, il Prof. Ugo Montanari, per l'opportunità di crescita concessami qui a Lucca e per l'alto profilo con il quale ha improntato questa ed altre iniziative a cui ho avuto modo di prender parte. Un cordiale ringraziamento ai membri della commissione di valutazione, la Prof.ssa Paola Inverardi ed il Prof. Davide Rossi per gli utili suggerimenti a guida del mio percorso di ricerca. Una particolare gratitudine al Dr. Rocco Moretti per le infinite ore di lavoro insieme, senza la cui pazienza ed il supporto difficilmente avrei raggiunto questo risultato. Desidero inoltre ringraziare il Prof.

Maurizio Lenzerini per il coinvolgimento nel progetto TO-CAI; il Prof. Fausto Marincioni; il consorzio CINI; i due referee esterni della tesi Prof. John Mylopoulos e Prof.ssa Daniela Damian ed il PhD Office dell'IMT, in particolare Silvia Lucchesi e Silvia Doberti.

Sono grato alla mia famiglia, a mio padre e mia madre che mi hanno insegnato il valore della conoscenza e che sono stati sempre, insieme a Sara, coinvolti come parte integrante in questo oneroso lavoro. Il ringraziamento più grande va senz'altro alla mia futura moglie, Elisabetta, che con pazienza ed amore ha patito insieme a me le rinunce e la fatica di uno sforzo così impegnativo, dimostrandomi in tal modo, in maniera chiara, ciò che di più bello al mondo un uomo può desiderare.

Ringrazio il resto della famiglia per la stima e l'affetto: Marino, Matteo, Antonia, Giovanni, Valentina, Davide, Matteo, Claudia, Mimmo, gli zii Giancarlo, Tiziana, nonne Ersilia, Emilia, Ilda ed Elisabetta.

Un sentito grazie al Dott. Roberto Oreficini Rosi ed al Dott. Maurizio Ferretti per aver dato spazio alla mia crescita formativa e per aver così dimostrato di credere nella mia persona ancora prima che nelle mie capacità professionali. Grazie anche ai cari amici e colleghi del Centro Funzionale per il supporto logistico ed umano in mille modi offertomi, in particolare a Paola e Gianluca per la pazienza sulle frequenti necessità dettate da questa attività di ricerca.

Grazie ai miei amici, indubbiamente i più trascurati per via di questo dottorato, con i quali mi ripropongo di recuperare negli anni a venire.

Vita

- April 26, 1980** Born, Ascoli Piceno, Italy
- March 2003** Master Degree in Computer Science
University of Bologna, Italy
- From 2004** Information Systems Officer
Civil Protection and Security Department
Marche Region, Italy
- July 2005** Laurea Specialistica Degree in Computer Science
Final mark: 110/110 cum laude
University of Bologna, Italy
- February 2006** PhD Admission for the XXI Cycle
IMT Institute for Advanced Studies of Lucca, Italy
- From 2006** Collaborations with the Computer Science Dep.
Involved in the TOCAL.it project
University of Bologna and CINI, Italy
- July 2007** Summer school "Advances in Software Engineering"
International School for Computer Science Researchers
University of Catania, Italy
- May 2009** Chair in a special session of the "Collaboration and Social Networking in Emergency" Track (ISCRAM 2009)
ISCRAM Community, Int.

Papers Related to the Thesis

1. L. Abeti, P. Ciancarini, and R. Moretti. Business-Oriented Model Driven Development of Services in Civil Protection. To appear in *International Journal of Emergency Management (IJEM)*, Jean-Luc Wybo, Luca Abeti, Paolo Ciancarini Ed., volume, 1, Inderscience, UK, Apr 2010.
2. L. Abeti, P. Ciancarini, and R. Moretti. Wiki-based Requirements Management for Business Process Reengineering. In A. Aguiar, U. Dekel, and P. Merson, editors, *ICSE Companion Wikis4SEWorkshop Proceedings of the International Conference on Software Engineering (ICSE) 2009, volume CFP0935G, May 16-24, 2009 Vancouver, British Columbia, Canada*, IEEE Computer Society.
3. L. Abeti, P. Ciancarini, and R. Moretti. Wiki-based Requirements Management for Business Process Reengineering. In *Proceedings of The International Symposium on Wikis, September 8 - 10, Porto, Portugal*, Porto, Portugal, Sep 2008.
4. L. Abeti, P. Ciancarini, and R. Moretti. Requirement-Driven Development for Emergency Management Systems. In *Proceedings of International Workshop on Mobile HCI for Emergencies in conjunction with CHI*, April 5 - 10, Florence, Italy, Florence, Italy, Apr 2008. ACM SIGCHI.
5. L. Abeti, P. Ciancarini, and R. Moretti. Business process modeling for organizational knowledge management. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 301–311, Berlin / Heidelberg, 2008. Springer-Verlag.
6. L. Abeti, P. Ciancarini, and R. Moretti. Model driven development of ontology-based grid services. In *WETICE*, pages 229–234, Los Alamitos, CA, USA, 2007. IEEE Computer Society Press.
7. L. Abeti, P. Ciancarini, and R. Moretti. Service oriented software engineering for modeling agents and services in Grid systems. *Multiagent and Grid Systems*, 2(2):135–148, Apr 2006.

Other Publications

1. L. Abeti and F. Marincioni. Geoinformatics and Communication Technology for Climate Change Hazards; Musing Beyond the Technical Issues. To appear in P.K. Joshi, A. Joshi and T.P. Singh, editors *Geoinformatics for Climate Change Studies*, volume, 1, Nova Science Publishers, Hauppauge, NY Vienna, May 2010.
2. L. Abeti, P. Ciancarini, and R. Moretti. A Service Oriented Approach to Model a Grid System for the Civil Protection. In *Proceedings of Workshop on Complex Networks and Infrastructure Protection (CNIP2006)*, March 28 - 29, Rome, Italy, pages 489–498, Rome, Italy, Mar 2006.
3. L. Abeti, P. Ciancarini, and V. Presutti. An Ontology Driven Method for Designing Software Agents for Workflows across Organizations. In A. Cimitile, A. DeLucia, and H. Gall, editors, *Cooperative Methods and Tools for Distributed Software Processes*, Software Technologies, pages 162–175. Franco Angeli, Rome, Italy, 2003.
4. P. Ciancarini, V. Presutti, and L. Abeti. An ontology driven design method for inter-agent communication. In *SEKE*, pages 90–94, San Francisco, CA, Jul 2003. Knowledge Systems Institute.

Abstract

Context & motivation: The success in the adoption of innovative technologies and new software systems inside an organization is related not only to technical problems. Several authors pointed out the importance of business modelling in software development in order to understand for instance: how the organizational environment relates to its software infrastructure and how the development of a new software-intensive service for a company implies changes in the company business and in its processes. The growing complexity of organizations, such as: networked and information-dependent companies; globalization of the companies structures; evanescent services provided by companies, increase the uncertainty in software projects. Software engineering successes in turn became more and more bounded to the understanding of the organizational assets and their relation to the software.

Question/problem: How can we model the business in requirements engineering phases and, at the same time, enable business-level concepts (such as strategies, goals, trust, etc.) to relate to system-level artifacts?

Principal ideas/results: Our aim is to define a framework that helps software engineers to develop their systems in what we named the business-oriented approach. The business-oriented approach to software development has been defined as an approach that carefully consider the processes, actors and goal of the business in developing a software system and also analyzes the changes that the software will cause to the organization itself. In order to define such a framework we exploit model driven engineering that helps in managing high level abstractions for the software. In particular, our work relies on

the service architectural paradigm that is particularly suitable in modeling socio-technical systems.

Contribution: In order to define our framework we define: a method; a framework of technologies and a set of tools. Our method, named the Enterprise-Service-Implementation (ESI), starts from the requirements of the organization and the system, it derives a platform independent service model that can be in turn transformed into platform specific implementations. ESI exploits the service concept as an intermediate abstraction. Model driven technologies help us to translate concepts from the business domain, such as goals, into services concretely bounded to a specific implementation platform. A framework of technologies and languages helps us to implement the ESI method. We exploit and combine: a goal-oriented language named Si*; UML Use Cases and a business process modeling language named Business Process Modeling Notation (BPMN), in order to model the business from different perspectives. We develop two tools to put into practice our framework. The Wiki Requirements tool (WikiReq) is a collaborative system and business requirements management Web application based on a wiki. WikiReq uses semantic wiki features for requirements gathering and management and is able to export semantically annotated knowledge to the Eclipse Integrated Development Environment (IDE). The Service Modeling Tool for Eclipse (SMOTE) is an Eclipse IDE application that exploits the Eclipse model driven architecture in order to perform the models transformations needed in our framework. We present a case studio where we experienced our framework in the challenging domain of Civil Protection. We present first results on the adoption of our approach concerning both the stakeholders experience and quantitative data.

Chapter 1

Introduction

The work of this thesis is based on previous research activities carried out by the author in some fields of software engineering. We started our work in 2002 defining an UML-based ontology design system which allows to represent, manage and serialize UML models representing ontologies(7; 48). Ontologies modeled in UML allow on one hand the knowledge management in a human-readable form, on the other hand a general representation for knowledge used by agents and Web Services (BL98).

In all our research work it can be identified a general leitmotiv that is: the usage of software modeling languages to abstract away the complexity of enterprise software development. The ability to model software by means of a hierarchy of abstractions (i.e., a meta-modeling hierarchy) bring us to investigate how requirements and business modeling could be effectively related to the model-driven software development.

The growing consensus about Meta-Object Facility (MOF) (OMG02a) and Model Driven Architecture (MDA) (Obj01) standards, reinforces the opinion that software applications design can be carried out manipulating models and meta-models. MDA allows to abstract away software details and to produce software using a meta-level representation of its structures. This approach has noticeable consequences in software engineering. In particular, referring to the MDA approach, we have studied how the Service Oriented Computing (SOC) benefits from a meta-

modeling representation of software application. Thus, in 2004-2006 we have investigated how the "service" concept can be employed in the MDA technology to define and manage SOA's interoperability.

The general goal of the service oriented approach is to overcome current interoperability issues among middlewares, languages and hardware platforms (155). It is explanatory that this is also the main goal of the MDA technologies which attempt to reach it from a model-oriented perspective. Service Oriented Architectures (SOA's)(OAS06a) play a fundamental role in this scenario. They can be considered the middle-level abstraction, between the business and the software, that can bridge existing gaps among business requirements and system development artifacts.

The main goal of this thesis is to give an answer to the question "how we can formalize stakeholders-level requirements in order to connect them with software-implementable abstractions?". In order to give an answer to this question we have moved the focus of our research from a system-centric perspective to a business-centric perspective. Thus, we start investigating requirements engineering methods and frameworks that can be employed in modeling the business in order to connect, in some way, the model of the business with the model of the system.

In the rest of this Chapter we state the problem of developing a SOA in a global software engineering context (Section 1.1.1) and how this context require a business-oriented approach to treat development issues (Section 1.1.2). We outline the contribution of this thesis in Section 1.2. Sections 1.3 and 1.4 describe respectively our previous work on the model driven development of SOA's and the emergency management domain where we apply and compare our approach to commonly used requirements engineering practices. Finally, Section 1.5 gives an outline of the thesis organization.

1.1 Motivations

1.1.1 Global Software Engineering of SOA's

The globalization of the companies structures requires an high organizational flexibility both in time and in space. These companies take part in opportunistic joint-ventures; they rapidly change their organization to accommodate market changes or new strategies. Besides, they do not have centralized structures (77). In this context, requirements engineering should model the system-to-be (i.e., the system that must be developed) carefully considering the impact of such a system in the organization, the changes that it requires and how it affects the whole business.

Global Software Engineering

Global software development (GSD) (88) faces challenges additional to those of single-site development. Global software engineering practices are performed at geographically separate locations. This difference introduces challenges that have been reported in the literature (58; 89; 91), such as temporal, strategic, and sociocultural challenges. Thus, the globalization of software is pervasive in the IT and software industry and affects both the context where the software has to be used and the software development process.

Herbsleb and Moitra (89) attribute the rapid growth of the GSD field of research on five specific benefit they identify in multi-site software development:

1. capitalize on the talent pool and use resources wherever available;
2. business advantages of new markets;
3. quick formation of virtual teams to capitalize market needs;
4. improve time to market by utilizing "around-the-clock" development;
5. flexibility to capitalize on merger and acquisition opportunities globally.

As evidenced by Fred Brooks in the classic Mythical Man-Month (41) lack of communication in software development is the main cause of schedule disasters, functional misfits and system bugs. This statement has gained even more significance in GSD (29).

The diversity of culture and communication and the dispersion over time and space require novel techniques, tools, and practices from many disciplines to overcome numerous difficulties and to take advantage of rare opportunities such environments entail (88).

Some promising solutions to address these challenges have been identified, e.g. in (51; 94; 124). Despite the real contribution of such approaches cannot yet be fully demonstrated, it is useful to evidence that such solutions are all founded on splitting the decisions basing on the architecture of the software. The architecture can induce dependencies between software engineering tasks such as managing synchronization and meeting a release schedule (26). As with single-site development, architectures can help to overcome some of these challenges also in GSD(36; 51). Among the possible approaches based on architectures the service-oriented one seems a right compromise that allows us to consider also business and organizational issues (106). The SOA approach supports the componentization of the architecture into isolated functional components (i.e., the services) that can be then composed via a loosely coupled composition.

Service Oriented Architectures: a bridge between the software and the organization

Services are not only an interesting abstraction to overcome technical issues on global organizations and global software development. Economic statistics confirm that local, national, and global economies are increasingly based on information and services (22; 165). Service industries is the great majority of the Gross Domestic Product (GDP) of all most developed economies (1; 135). Services cover a very broad and diverse range of activities, including health care, education, transportation and logistics, utilities, financial services, government services, including national defense, civil protection, entertainment, and more. The demand for

new services using IT is also increasing, and the importance of services in inter-corporate competition is rising as well.

Because of their importance, the development and delivery of services have become a common discussion matter at universities and corporations. In particular, in finding a way to increase productivity and innovation in services-related industries and tasks by applying a scientific approach (1).

The Levitt claim "everything is a service" (110) evidenced the misunderstanding in believing that the service was people-intensive while the rest of the economy was capital-intensive. He argue that there are only industries more or less service-oriented than other industries. The more a product is technologically sophisticated, the more it success depends on the quality and availability of its accompanying services that defines a value for the customer (110).

In modern global economies the differentiation between products and services is increasingly blurring (160). The analysis of services and service value should not be distinguished from that of products; further "products themselves should be merely seen as vehicles for service delivery" (25).

Basing on this envision of services the analysis of the problems of services industries requires more than a purely technical (e.g., software engineering) solution. The human element is almost always present in every step of the service development and usage lifecycle. Thus, an effort involving multiple disciplines is required. A consensus is emerging that the cumulative and interconnected innovations in information and communication technology, industrial engineering, business strategy, economics, law, and elsewhere cannot be described and understood by a single academic discipline (151; 153).

1.1.2 The Need for a Business-Oriented Approach to Software Services Development

The need for a deep understanding of services and improved productivity is becoming unavoidable inside business and non-business organiza-

tions such as government structures. Either if manufacturing products depends on the related services, the services themselves were differentiated from other products on the basis of four characteristics: *intangibility*, *heterogeneity*, *inseparability*, and *perishability* (178).

A general definition of service can be as follow:

Service: "a change in the condition of a person, or a good belonging to some economic entity, brought about as the result of the activity of some other economic entity, with the approval of the first person or economic entity." (90)

The value of a service is created through a complex set of relationships among involved actors. The typical actors of a service value network involves: consumers, service providers and service enablers. Such a value is influenced by the social, technological, economic and political context in which the service is embedded (25). Aside from the value network also the services exchange is different from both the agricultural and manufacturing epochs (78). "Each party in the exchange needs the other's knowledge in negotiating the exchange" (46). The provider lacks the contextual knowledge of the customer's business, and the customer lacks the knowledge of the full capabilities of the provider's technologies and experience.

This leads to consider the nature of the knowledge involved in a services exchange. It can be given a fundamental distinction for this knowledge (130):

- Codified knowledge: that refers to knowledge transmittable in formal, systematic languages (e.g, a technical specification of a product).
- Tacit knowledge: that is difficult to transfer between people, between groups, and between organizations (e.g., the ability to ride a bicycle). This knowledge may also be difficult to explicit.

The nature of tacit knowledge complicates the services exchange, and limits the ability of each party to fully comprehend the needs and abilities of the other, thus to benefit from their value.

In this thesis we analyze the relationship among services (intended in their broad meaning) and software development. We evidence that the service has a double importance in this connection because in SOA's: the software is a service and IT services are enabled by the software.

The software is a service because it can be consumed as a service (31; 135; 155). Indeed, the software is a very unusual "product". It is one of the most service-oriented products because like the general notion of service (178) the software is intangible, heterogeneous and perishable too. Many current services and in particular IT services are enabled by software and new related technologies (67). In particular, the software affects the Business Processes (BP's) (83) that enable services to produce their values.

Thus, the problem of addressing global software engineering taking into consideration the business and organization issues can be addressed by means of a reengineering of the processes that enable the services. Our approach is aimed to adopt business reengineering techniques and methods in order to develop service-oriented software.

We need to consider a business-oriented approach to the software because GSD defines a complex scenario both in software development and in the company management. The requirements that influence the adoption of the software such as organizational assets, business strategies, business rules, political issues etc., must be carefully considered and analyzed in relation to the developed software. In an economy based on services, we need a business-oriented approach because also the value of a product is not only related to its physical manufacture. A product is a part of a service that is sold to the customer and the product is a result of a production BP usually heavily based on software systems and ICT.

Current Approaches to the Problem

New challenges in software development require new techniques, tool and practices in order to solve the issue described in this section. Current methods and tools try to address the described issues limited to a single point of view. They lack to use a comprehensive approach that allows to treat all the described problems in a coordinated way.

Requirements engineering, for instance (93; 171), center on requirements and their evolution during the system development process. In particular, for software systems such studies have concluded that one of the most problematic tasks in the development is understanding the requirements and correctly transforming them into code (145).

Despite requirements engineering recognizes that not only the functionality of the system itself, but also its environment needs to be considered during specification (163); it is yet more concerned with technical problems. Such as, the environment and the organization are investigated in order to understand only how they influence the software development. They use techniques like for example use case modeling (105) and business modeling (156) to better understand requirements for the system to develop (43; 69; 141).

The business modeling activities carried out in requirements and software engineering do not focus on business reengineering and are aimed to model the environment of the system in order to support project and change management activities and to better understand system requirements (82). A typical example is the Rational Unified Process (RUP) (105) approach. RUP is an iterative software development process that integrates many software engineering best practices. An important feature of RUP is that it uses software engineering techniques to perform the business modeling. The language, and thus the diagrams, used for business modeling are the same used for system modeling. This has many advantages in mapping processes concepts to system concepts in the requirements documents. RUP allows to derive parts of the system specification directly from the business models. RUP has several drawbacks: it does not support a cross-functional modeling of BP's; it is limited to the development process and it does not support interactions among different teams (15).

Some other approaches are mainly intended to represent all the knowledge about the enterprise and to reengineer the business by means of new technologies. They are concerned in understanding how the development of a new software system for a company implies changes in company business (15; 99; 102). It is the case of business process reengineering

(83) extensions of RUP named the Enterprise Unified Process(EUP) (15) that adds to RUP: the definition and analysis of the enterprise strategy; the identification of processes implementation options; the modeling of the organization and the domain besides the BPM. Another business-oriented approach is the goal oriented approach (99) like the Yu's *i** language (172). The approach is requirements driven i.e., concepts used in requirements modeling are used/mapped also in design phases. This feature allows to define a continuity from early requirements analysis to the detailed design and system implementation. In this way, conceptual models representing the system are obtained with an incremental refinement and extension of a model of the environment. The usage of use cases in EUP and goals in *i** to describe the business, are useful to define a business-oriented approach of software development. Nevertheless, both use cases and *i** do not take into consideration dynamic aspects of the business and in particular the BP's. Moreover, it is not clear how requirements described by means of use cases and goals can be concretely mapped in software implementable abstraction for instance components or services, assuring the traceability of requirements.

A description of the dynamic aspects of the business can be achieved by means of Business Process Modeling (BPM) (86). For instance the Business Process Modeling Notation (BPMN) (OMG06) standard synthesizes the BPM community best practices and it is aimed to give a standardized notation for BPM. BPMN defines a graphical notation for BP similar to flow-charts which is aimed to be understood by stakeholders, analysts, business users, developers etc. BPMN can be incrementally adopted and supports automated translation to BP execution languages. However, BPM is not useful to describe systems that are not process oriented (86; 164). The BPM is a complementary approach to the use case and *i** business modeling but these approaches are currently completely unrelated each other.

1.2 Contributions of the Thesis

In this section we describe the thesis contributes and how they relate with the issues described in Section 1.1.

Several authors have pointed out the importance of a business-oriented approach in software development (43; 69; 141). They found their approaches only on business modeling for software requirements engineering and business process modeling.

We have pointed out that global companies continually renew themselves, their products, and their processes. They have to survive to rapid changes in markets, user preferences and technologies and their products are increasingly service-oriented. It is meaningful to remark that those changes must be supported by new technologies (31). The development of such new technologies cannot be performed without considering how new systems affect the organizational processes of the companies where they are employed (83).

The products and the business organizations are more and more organized around the concept of service (44; 78; 111) and the service notion share many foundation characteristics with the software. In particular, the service can act both as an enabler of flexible rapid changes (e.g., in software SOA's) and as a way to connect business requirements to code (17; 71). Thus, information technology professionals are appointed to implement organizational reengineering efforts (43; 69; 83) by means of the technologies they are used to employ, in particular service-oriented technologies. The ability to model, represent and manage the service notion in a system engineering context, is the fundamental requirements in order to manage global software engineering complexity.

Among other technologies, the Model Driven Engineering (MDE)(33) is the software modeling approach that allows to model business-level concepts and relate them to the system by means of the service abstraction. Thus, this thesis is concerned to define a framework to perform business-oriented software engineering. We base our work on the results reached with our previously developed MOTO-GAS tool, that helps to manage the service concept by means of MDE technologies.

1.2.1 General Objectives for the Thesis

The overall goal of this thesis work is to define a single framework to perform business-oriented software engineering.

That is to define:

- a method (Chapter 3)
- a framework of technologies (Chapter 4)
- a set of tools (Chapter 5)

that can be used to reengineer the business and its processes by means of model driven engineering and service oriented architectures.

In this section we summarize the context of the thesis problem that we argue in previous sections (Section 1.1.1 and 1.1.2).

The problems the thesis try to overcome are:

- The organizations where the software has to be used are global;
- Requirements engineering and software development must be both performed globally;
- Modern economies are more and more based on the service paradigm: that is a notion, a unit of measure, a business-autonomous entity, an arrangement to define structures inside an organization, etc.;
- Services are different from other products: intangibility, heterogeneity, inseparability, and perishability. They resemble to the software;
- The more a product is technologically sophisticated, the more it success depends on the quality and availability of its accompanying services;

Why the problem is a problem?:

- Diversities in global software development requires new techniques, tool and practices;

- A new paradigm is needed in order to define a common understanding among software users, business needs, software architects, organizational processes and software developers;
- The service value network influence service development by: the social, technological, economic and political context in which it is embedded;
- The manufacturing industry is increasingly service-oriented: the product success depends by the related services (e.g., sales).

1.2.2 Summary of Contributions and Limitations of the Approach

In this Section we show how the exposed problems are faced in this thesis. Solutions:

- An architectural paradigm to be used in socio-technical problems: the service;
- A method that takes into consideration organizational and software issues: the business-driven approach;
- A framework that helps to manage high level abstraction for software by means of model driven engineering;
- Reengineering manufacturing business processes taking in consideration that "everything is a service"(110).

How can we model the business in requirements engineering and, at the same time, enable business-level concepts (such as processes, strategies, goals, trust, etc.) to relate to system artifacts?

Our idea is to explore the engineering of requirements by using the technology of wikis, more precisely Semantic Media Wiki (Wik07). The result is a collaborative system which stakeholders can use to write requirements using different notations and producing a model suitable for further transformations based on MDE and the service abstraction.

Thus we define a business-oriented methods to support the modeling of the knowledge about the business and the system. We name such a method the Enterprise-Service-Implementation (ESI) method (5). The ESI method is organized in three phases:

1. Enterprise Modeling phase: that models the organization, strategies, business rules, business domains, internal and external BP's etc.
2. Service Modeling phase: that models the internal BP's, business entities, the systems, the architectures etc. It is the perspective of the software engineers.
3. Platform Specific Implementation phase: that defines an executable model for BP's and concerns with system design and implementation.

The Enterprise Modeling is a requirements engineering phase. Enterprise Modeling requirements engineering is not only limited to the software system, it involves also the whole organization that have to be reengineered and modeled similarly to a system.

At the beginning of the Service Modeling phase the goals and the goal dependencies among the actors are analyzed to understand how they relate with the system. A set of transformations are used to transform requirements into a technological platform independent service model. Such Platform Independent Model (PIM) (OMG01) can in turn be transformed in a Platform Specific Model (PSM) (Obj01) in order to be implemented. The ESI method is presented in Chapter 3.

We define a framework of technologies, we named the Business Process Reengineering Framework (BPR Framework) in order to support the ESI method. We use a set of language to support requirements acquiring during the Enterprise Modeling phase, such languages are based on goal oriented formalism (e.g., i^*), Use Case and notations to describe BP's. They allows to define a Computational Independent Model (CIM) (Obj01; OMG01) such as a model of the business that is independent from

the software based manipulation of the some of its parts. The BPR Framework also define an architecture to support transformations performed by means of the MDE approach. We define two main transformations that are: the CIM2PIM transformation mapping the business concepts contained in the CIM into a service-oriented PIM; the PIM2PSM transformation that derives a platform specific service model (e.g., a model for Web Services) from the PIM service model. The BPR framework is presented in Chapter 3.

Finally, in order to support our approach, we develop two tools named WikiReq and SMOTE. The Wiki for Requirements (WikiReq) tool (6) is used to manage both the system and the organizational requirements, in the context of business process reengineering. Such a wiki is based on three main contributes: 1) using a semantic wiki for requirements gathering and management; 2) exploiting the wiki platform to define an argumentation system for both synchronous and asynchronous discussions among stakeholders; 3) achieving interoperability between the semantic wiki and an Integrated Development Environment (IDE) platform like Eclipse. The WikiReq tool is exploited in the Enterprise Modeling phase in order to support the requirements elicitation process. WikiReq allows sketching the non-functional requirements described in terms of goals and actors, it connects this knowledge with a more formalized version defined in the enterprise knowledge.

The Services MOdeling Tool for Eclipse (SMOTE) (3) defines an architecture that supports the management of PIM and PSM service models. SMOTE is based on a set of Eclipse plug-ins implementing the MDA architecture (OMG01). It defines a modellation environment for business and implementable services and concretely implements the CIM2PIM and PIM2PSM BPR Framework transformations. The WikiReq and the SMOTE tools are presented in Chapter 5.

Summing up the main contribute of this thesis are:

- The ESI method: a business-oriented method to derive a platform independent service model of the system starting from the business model;

- The BPR Framework: a framework of languages and technologies that support the ESI method;
- Two tools:
 - WikiReq that is a collaborative system and business requirements management semantic Web application based on wiki;
 - SMOTE that is a service models design and management application based on MDA and the Eclipse IDE.

Moreover, our thesis provide a detailed case studio on a real project in Civil Protection. Civil Protection is a very challenging business domain since it requires the rapid development of system infrastructure among a wide set of heterogeneous organizations. Chapter 6 describe such case of reengineering in Marche Region local government Civil Protection where our framework has been applied.

Limitations of the approach

The approach we present has both technical and application limitations we report in this section. First of all it is limited in its context of application. Indeed, it is not a framework intended to be adopted in large development teams and big software development companies. We define the framework presented in this thesis in order to provide an instrument to software development in Small Medium-sized Enterprises (SME's) which can use it to develop customer software taking into consideration business modeling and reengineering issues. It can also be used internally in those organizations that want to analyze how to revamp their processes and successfully introduce new software technologies in their work. Such as in the e-Government scenario we present. In an e-Government scenario is fundamental to analyze in advance implications and applicability of new software systems and then commit them to a software development team by means of a detailed call for tender.

In those contexts the usage of WikiReq together with an approach that derives services starting from a set of goals defined by stakeholder

(i.e., our framework) it is strategic. e-Government organizations are usually very complex organizations, where results not only depends by economics or measurable indexes. For instance, results could depend by politicians satisfaction or by the social utility of the services furnished to the citizens. By means of WikiReq it is possible to allow the e-Government organization stakeholders integrate their visions and relate technical choses to vague goals where the fulfillment condition is not or cannot be well defined (they are named soft-goals in i*).

Despite we have based our approach on GSD, our framework is not applied to a world-wide software development context. We adopt a multi-sited approach of requirements and software engineering. Indeed, distances need not to be global to be important(12; 104). In fact, being in another building or on a different floor of the same building, strongly reduces the communication among employers and teams (89). Solutions that improve globally distributed work will much more improve the work of teams "in the same zip code". Also regarding the GSD solutions to cultural differences issues our approach should not be regarded limited to ethnic diversities. Cultures may differ on many dimensions, for instance attitudes toward hierarchy, ICT backgrounds, generational differences, sense of time and communication styles.

We limited our investigation on SME's organizations and prove results on the e-Government context. Despite the term "global" could induce to think to a world-wide activity, usually managed by big enterprises, the SME's and e-Government contexts in which smallest and heterogeneous teams and technologies have to cooperate is more interesting for our research (8; 74). SME's have not only flourished in domestic economies. SME's evolve as multinationals either through their own investments or as a result of the formation of alliances with other SME's or institutions. Thus, SME's companies are entering in the GSD arena, but "their involvement is more often opportunistic than carefully planned" (142). The limitation to small teams and organization (both in SME's and public administration) is not restrictive in size. Indeed, more than the Europe 70% Gross Domestic Product derives from SME's (Mic05) thus it is fundamental to provide SME's with lightweight solutions to the new chal-

lenges of GSD requirements management and software development. In particular, the Italian public administration domain is very suitable for a business-driven and service-oriented approach. Indeed, the lacks in communication and sharing of knowledge in these organizations is one of the most important causes of its inefficiency (49; 63; 74).

As for the service-oriented approach we remark that our approach assign to the notions of service and SOA a very general interpretation. Thus, the service should be intended not only an implementable entity (e.g., a Web Service (W3C01)). For instance, a service can be also meant as an economic entity with an assigned and measurable value or as a computational independent manner to organize BP's outputs entirely performed by human beings.

From a software implementation point of view this thesis is not intended to concern automatic orchestration and choreography of SOA's and Web Services. We limited our research to the use of SOA's software platform as a middleware to define platform independent software. Thus, we exploit the service abstraction to define a new service or application plug-in defining a model where a set of services are statically related. Thus from a software engineering point of view we consider SOA's as a reference architectural middleware for the development of software components without providing support for service orchestration and choreography. The approach we plan use use relates service and to define new services starting from those services is described in Section 4.2. It is a future work where define a way to manage interactions in SOA's by means of the plug-in architecture that allows to define a user interface for a new developed services able to be exploited by the user inside a wrapper application.

1.3 Previous Work of the Author Related to the Thesis

In this section we discuss how technologies based on the Model Driven Architecture (MDA)(119) can be exploited to the design of service oriented applications. The presented approach is supported discussing the

design and use of the tool we have developed for service modeling, named MOdeling TOol for Grid and Agent Services (MOTO-GAS) (3). We start developing the tools and the frameworks described in this thesis founding on the previous research work carried out in MOTO-GAS.

Model Driven Development in SOA

Our work starts by considering three main fields of research: the Model Driven Architecture (MDA) (Obj01; OMG01), the Grid (72), and the Semantic Web Services (144). The goal was to understand how an effective degree of interaction among some service-oriented technologies could be obtained by means of an abstract definition of the "service" concept.

To reach a general understandable definition of the service concept, two standards have been investigated: the Web Service Description Language (WSDL) (W3C01) and the WS-Resource (OAS06d) .

WSDL simplifies the encapsulation behind a common interface of diverse implementations of the same services (virtualization). It defines services as collections of network endpoints, or ports. The WSDL specification provides an XML format for documents services endpoints.

The WS-Resource approach issue from the observation that there are many ways a Web Service might model, access and manage a state. The WS-Resource has been proposed as a means to codify the relationship between Web Services and stateful resources in terms of an implied pattern. The WS-Resource approach introduces support for stateful resources without compromising the ability to implement Web Services as stateless message processors. The relationship between Web Services and stateful resources in WS-Resource is codified in terms of an implied resource pattern (CFF04; K. 05). The implied resource pattern refers to the mechanism used to associate a stateful resource with the execution of message exchanges implemented by a Web Service. This support is performed by means of the Resource Property Document (RPD) (OAS06c) defined by the Web Service Resource Framework (WSRF) (CFF04). The RPD is the XML data structure representing the state for the (stateless) Web Service and is used in an implied pattern to define stateful services that can be instantiated.

The WS-Resource standard enables the convergence between the Grid and the Semantic Web Service technologies. In this context the notion of Web Service as defined for Grid systems (72) has important implications for the convergence of the Grid, Semantic Web and MDA technologies.

Several standard specifications have been proposed for semantic Web Services (W3C05; OMG03a; W3C02a). Among them, the Web Service Semantics (WSDL-S) (W3C05) seems to be more suitable to be used in this context. Similarly to WS-Resource, WSDL-S is a lightweight approach extending the WSDL metamodel to define an ontology for service concepts. Other approaches to the Semantic Web Services like, for instance, OWL-S, duplicate the WSDL input and output descriptions and are bound to specific ontology representations (W3C05). Despite its simplicity, WSDL-S is compatible with different ontology definition mechanisms such as the OMG Ontology Definition Metamodel (ODM) (OMG03a), the Web Ontology Language (OWL) and UML. It can be used both in stateless and stateful services and hence it complies with the general framework for services defined by WSDL and WSRF.

Despite the technological perspective, a general notion of services, can be also treated by means of a model-oriented viewpoint. In recent years the increasing demand of complex business applications and the growth of different distributed systems technologies have determined the emergence of new approaches to modeling standards. This situation led to the development of a more general system to drive software development lifecycle named Model Drive Engineering (MDE) (33).

MDE is a software development methodology which focuses on creating models that describe the elements of a system. One of the basic principles of MDE is that "Everything is a model" (33): models are the main entities that drive the software development process. Model Drive Architecture (MDA) (Obj01) accomplish the MDE principles by means a set of OMG's standards-based modeling languages that are used as formal model-based development languages. MDA is about using modeling languages as programming languages rather than merely as design languages. Programming with modeling languages can improve the productivity, quality, and longevity of software. OMG's Model Drive Architec-

ture (MDA) (Obj01) in this context must be interpreted as an alternative way to solve the middleware interoperability problem respect to the SOA approach.

MDA is based on a straightforward idea: simply exploiting the meta-modeling mechanisms in order to allow the development of software at an higher-level than it is commonly used to. It obtains this goal separating the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. An application business functionality or behavior can be engineered by using a Platform Independent Model (PIM) expressed in UML. This model is expected to remain stable as technologies evolve. An MDA development tool then converts this PIM in one or more Platform Specific Models (PSM's) and finally to (one or more) working implementation on virtually any middleware/framework platform. A set of rules and techniques is used to transform a model into another one. The MDA Metamodel Description (Obj01) can help us to describe different kinds of mapping which can occur in MDA: PIM to PIM; PIM to PSM; PSM to PSM; PSM to PIM.

In order to manage the WSDL and WSRF features together with their semantics, we define in MOTO-GAS a MOF compliant meta-model and we use this meta-model as a profile for modeling the services. From such WSDL/WSRF MOF meta-model we can represent services in a platform independent way by means of application service models describing both WS-Resources and Web Services. In this context, we use MDA-based transformations as an harmonizing glue. Such transformations consist of a set of rules and techniques used to transform an model into another model. The models in MDA may represent, for instance, parts of functions, structures or behaviors of a system at different levels of detail. Our framework automatically transforms a platform-independent service models in a platform-specific model which directly maps into code files implementing the services.

The general goal of MOTO-GAS is to make the design of an SOA application similar to the design of an Object Oriented (OO) application in UML. For instance, programmers and software engineers can find many

analogies with concepts of OO development. The service abstraction allows to manage new challenging situations (76; 77) where not only the applications and the infrastructures (e.g., the Grid) dynamically change, but also the business organizations and the business goals.

A number of tools, for instance, the UML-Based Ontology Toolset (UBOT) (24) and UML Data Binding (UDB) (53), have demonstrated the suitability of using UML for developing and represent the ontologies. According to this approach (48), the designer can model the ontology domain (e.g., in a class diagram) and then refer concepts described in this domain in the service application model.

We have introduced in MOTO-GAS the support for semantic Web Services in order to allow the service to represent and manage ontologies in Web and Grid infrastructures (4).

We implement MOTO-GAS by means of a set of MDA frameworks described in (3). We compare MOTO-GAS with other similar approaches. For instance the Uniframe (30) and the IBM Emerging Technologies Toolkit (ETTK) (a106) projects.

Uniframe is aimed to overcome distributed platforms heterogeneity using an MDA-based service-oriented view of the system. It uses a formal specification language to define the components and to support connection and Quality of Services (QoS) computation. Uniframe has investigated many component-based and service-oriented issues and define a good abstraction for a unified architecture by means of the service abstraction. It also uses UML and MDA respectively to design service models and to realize the Unified Meta-Component Model (UMM) (30) employed as a glue that puts different technologies together.

ETTK is a collection of technologies developed by IBM research labs. It provides experimental tools, for example code, documentation and executable demos that illustrate IBM's current approach on emerging technology topics. In particular, the "ETTK for Web Services and Autonomic Computing" (ETTK-WS) (a106) is a software development kit for designing, developing and executing Web Service technologies using open specifications such as SOAP, WSDL, WS-Resource and Web Services Distributed Management (WSDM)(OAS06b). ETTK-WS provides to developer basic

software components needed to experiment with and create Web services and autonomic programs. In particular it gives a Java implementation of all WSRF specifications and allows WS-Resource to be modeled with some plug-ins in the IBM WebSphere (IBM97) development environment.

Currently, the combined usage of MDA and SOA is a widespread approach in enterprise computing (73). MDA and SOA did not invent new middleware technologies or new interaction patterns rather than expand on existing frameworks and help them to work better. Despite the nice characteristics of the SOA's, their success heavily depends on surrounding conditions like for example: the vendor support; the standard evolution and the community acceptance (155). Beside using MDA to give an abstract representation of services, we can use MDA also to overcome the SOA's technology volatility. Such as to use a platform independent service model that allows an effective decoupling between the services represented at the business level and the specific technologies used to implement the services (e.g., WSDL, WSRF, CORBA etc.).

The goal of the model driven development of SOA is to provide an environment where technology-specific services can be produced in a way that is independent as possible from the particular implementing technologies and their changes. Also in design phases services should be considered a technology aware concept and modeled at a business-level. Business-level services concern mainly with business requirements rather than technological issues and should focus on the information the service need to manipulate and the functionalities the service must provide(73).

In order to give a more general representation of the service concept (actually the business service concept) the research presented in this thesis moves from the scenario discussed in this section and investigates how business services can be directly derived from requirements. Thus, we analyze how services relate to business models and business processes in order to define service models useful both for software development and business reengineering.

1.4 Services in e-Government and Civil Protection

Every business and government organization is engaged in delivering services. We can summarize this vision we expose in the previous sections, quoting Newcomer and Lomow: "the service orientation is an organizational principle that applies to business and government as well as to software"(128).

Different national governments are aware that the use of ICT represents a means to increase collaboration and cooperation between different public institutions. They aim to provide better and more efficient services to citizens. However most of the governmental bodies develop their own ICT systems independently from the other organizations and connecting them together is not a simple task.

The development of an e-Government system usually requires an effort in the integration of heterogeneous architectures and applications among heterogeneous organizational structures. Nevertheless, there are some situations, such as emergencies and crises, where systems have to be integrated in a few days and, sometimes, in a few hours. In these situations, many organizations and governmental departments, involved in order to face up the emergency and avoid further damages, are unaware to the structures and technologies of other organizations.

The main challenge in e-Government relies on integrating efficiently all those heterogeneous public ICT systems by providing an unified middleware (2; 23). The SOA approach has been studied to be a successful approach in defining such a middleware (122; 127). SOA's represent a suitable standardized integration architecture to achieve better flexibility while implementing new business processes across organisation divisions and ICT systems (139). The need for such integration is driven by the concern of process integration while maintaining different IT infrastructures across the whole organisations. SOA-based approaches to e-Government have been specified keeping in mind that business processes integration should be as well taken in account (23). As claimed in this thesis, also in the e-Government scenario social and organisational

aspects should be taken in account since the integration success depend heavily on the acceptance of the global model (139).

In Chapther 6 we show how our BPR framework and the SMOTE tool can be used together to implement an e-Government application useful in a Civil Protection scenario. We highlight how our business-oriented approach to the software development helps to overcome issues related to the management of emergency management volunteers organizations.

In particular, the Italian public administration domain is very suitable for a business-driven and service-oriented approach because it lacks in communication and sharing of knowledge, this is one of the most important causes of its inefficiency (49; 63; 74).

Thus in this section we discuss the challenging domain of Civil Protection e-Government scenario that it is also useful to understand the case studio described in Chapter 6. Moreover, we present a scenario of a service-oriented project named SISSI that has been developed in the Marche Region Civil Protection. The software developed in such a project has never be used due to some misjudging on the analysis BP's and organizational issues.

1.4.1 Services for Critical Infrastructure Protection and Emergency Management

The e-Government contexts in which smallest and heterogeneous teams and technologies have to cooperate(8; 74) is an interesting domain for our research .

The issues to solve for service integration in the e-Government context can be meant and compared to the "global context" we present in Section 1.1.1 for the Global Software Engineering. The e-Government context demands to many institutions to collaborate actively. Usually, these institutions are different because of their organizational structure and "business culture". This collaboration, in particular concerning the experience and the information exchange, is particularly complex in the civil protection domain.

Civil protection is the institutional function responsible over emer-

agency management in the Governments of European countries (Eur02). Especially in recent years, the Civil Protection had to face increasingly complex and varied emergencies, not only closely related to natural hazards. Also anthropic (i.e., induced from humans) and social emergencies had to be faced by the Civil Protections such as the Campania Italian region trash emergency in 2007 or the management of big events like the G8 in L'Aquila in 2009(Pre09). Moreover, during the last few years, more and more active collaborations between the European and the World Civil Protections makes this context an effectively global context where services have to be developed and deployed.

Civil protection is a "system" that operates in times of normality (also called "times of peace") to develop prediction, prevention and study of the various risks faced by the territory. In times of emergency, Civil Protection provides the best possible operative answers. The system of Civil Protection is a complex system in which organizations, institutions, and different structures have to operate in a synergistic way.

The activity of the Civil Protection system is governed by a set of laws and rules, both from national and regional authorities, which allows to govern and manage the different situations generally defining processes and tasks of each actor involved in emergency prevision and prevention.

In most European countries, Civil Protection is a function assigned to a single institution or a few public authorities. In Italy, both centrals and peripherals organizations of the State are involved in this function: local authorities (such as municipalities, provinces and regions) and also civil society fully participate in what is named "the National Service of Civil Protection", including voluntary citizen organizations as well.

The activities of the National Service of Civil Protection in Italy are not limited to attend in case of disasters and calamities. Indeed, the Civil Protection is employed mainly in prevision and prevention of various risk scenarios and in the management of potential emergency situations (including the development of intervention procedures to be used in emergency planning).

Prevention is intended as the study of the causes of the disasters phenomena, the identification of risks and identification of areas of the terri-

tory subjected to the same risks. Prevention activities are carried out by a system of networks linking the Civil Protection service to:

- national centers of scientific research (e.g., universities, research institutes, centers of excellence, etc.);
- technology systems of collecting and processing information about various types of risk and conditions that may increase the probability of danger for the community;
- information centers that are able to report the probability of catastrophic events.

This set of technical and scientific activities, ranging from the collection of land information to their preparation, allows to the entire system of Civil Protection to evaluate the possible risk situations and to alert the intervention system with the most useful advance. It is also useful to provide the authorities the necessary information to make reasoned and timely decisions.

The prevision consists of the activities directed to avoid or minimize the possibility of damage consequent the events also from knowledge acquired about the effect of prevision. It is an emergency management task aimed to reduce the probability of disastrous events or, at least, to limit the effects on population and settlements of a calamitous hazard. In particular, among the prevision activities, a special mention must be given to the emergency planning activity. Besides the plans prepared by each municipality, special plans are prepared for particular risks (e.g., sanitary risk, business risk, traffic, etc.). The planning must of course include operating processes of rapid intervention and through specific methods of work.

The offices of the National Civil Protection Department are responsible for predicting natural and anthropic risks. They are connected through a network organized at national and regional levels. Therefore is of vital importance the rapid and timely exchange of knowledge with the area where the disaster may occur. The use of technologically advanced networks, such as radar networks for weather forecasting, the national net-

work of seismographs, the sophisticated systems for monitoring the activity of volcanoes, and the best scientific and professional expertise available, help the Italian Civil Protection operate also basing on early warning and, when possible, adopting preventive measures such as evacuation of areas interested by a risk.

The aid activities consist in the implementation of emergency interventions to ensure, to the populations affected by natural disasters, first aid and assistance. For the purpose of civil protection activities, the law divides the field of natural disasters in three categories:

1. natural events or events connected with the human activity that can be faced through interventions implemented by individual institutions and competent authorities in ordinary way;
2. natural events or events connected with human activity which by their nature and extension involve the coordinated action of several institutions or competent authorities in ordinary way;
3. natural events, disasters or other large scale events that must be faced with extraordinary resources and powers due to their magnitude;

It is important to remark that the classification of an event in one of three categories is closely connected with the size and the resources held by individual institutions. Indeed, an event that occurs in a big city can be run directly by local technical services, while in a small town may have to be managed as a real emergency. The overcoming of emergency consists in the implementation of the actions needed to be postponed to remove obstacles to the resumption of normal living conditions.

With the Italian law (No. 225) of February 24th 1992, Italy has organized civil protection as a "national service". Such a service is coordinated by the President of the Council of Ministers (through the Department Head of Civil Protection) and composed, as the first Law comma "by the authorities in the central and peripheral, Regions, Provinces, the Municipalities, the National public and local entities and all other institutions and private and public organizations on national territory."

Therefore the Italian Civil Protection national service is a very complex organization that requires, in respect of independence and autonomy of its individual components a close collaboration among institutions, businesses, voluntary organizations and non-governmental organizations and even individuals. Moreover, the interaction between the civil protections organizations during a disaster must be established quickly and in a coordinated manner in order to reduce the lives loss and to mitigate the harmful consequences inherent in every disaster.

Finally, this scenario has been further complicated by the recent need to coordinate and set up systems of civil protection at a supranational level and, sometimes, worldwide level. As an example it was necessary in the tsunami in Thailand in 2004 (INE04) and Katrina Hurricane in 2005 (Nat05).

In recent years, the European Union is implementing a specific system of European Civil Protection (Eur02) named the EU Civil Protection Mechanism. It incorporates, basing on the involved countries, very different tasks, measures and organizations of national Civil Protection organizations. Such a meta-organization has the challenging mission of integrate system that are culturally, organizationally and of course technologically very different.

For these reasons we believe that the domain of Civil Protection is especially suited to test and demonstrate the validity of our business-oriented approach to the implementation of SOA's. In this domain more than it happens in others domain, the integration between technology and processes has a strategic role in the success of adopting a new software. The rapid and right integration among services and processes in civil protection are, sometime "matter of life and death".

1.4.2 The Marche Region SISSI Scenario

In January 2001, the Marche Region commit in outsourcing the design and implementation of a software named SISSI to the Company Design Services Srl, a software development company located on the same city of the Marche Region offices and with a total of 7 employees.

The SISSI project involved the creation of a Web application for the prevention activities relating to the protection of water resources in the Marche region area. The two main goals of the software were:

- the census of water resources in the area (i.e., source, group of springs, wells, derivations etc.) by means of forms filled online directly by the resources holders.
- to enable resource holders to send periodically data on the monitoring of water (e.g., minimum, maximum and other data to measure).

From a technical point of view, the tender documents required a software architecture based on Web Services, an immature technology, that was slightly used in 2001 to implement applied e-Government projects and that was not completely standardization at that time. In fact, this constraint was not motivated by special needs or organizational requirements rather by the possibility of claiming an innovative software that besides technical benefits gave prestige to the institution and the involved officers.

In order to collect requirements in the SISSI project, they have been possible only four 3-hours meetings with the technical officials who followed the contract. Only in 2 meetings there was the attendance of the executive managers of the Marche Region. The outcome of the meetings led to the following technical choices:

- The use of the Web Services Architecture;
- The login system of the Web application should use certificates, in the light of the fact that by 2003 the Marche region was equipped with a single signature system accessible by digital signature;
- The creation of a single card for both census and monitoring data of a water resource with a total of 74 fields distributed up to 12 forms for each resource.

Meetings only marginally was interested in the system users requirements. Such users worked into agencies external to the Marche Region

organization. Such agencies were named Optimal Territorial Area (ATO). The ATO were distributed over the entire Marche territory, they were 5 and employ an average of 5 persons per office. Both the ATO managers and the employers were not involved in the design of the system.

In light of the difficulties encountered in adopting a system based on Web Services, which also used digital certificates for logging, the development company limited the compatibility of the system to Netscape (Ame94) version 6.0. The constraint was accepted considering the consequences of this choice less important than the fact that project used a Web Service SOA.

The SISSI system has been verified and released on September 2003 and never used by ATO's. It was only used for a few months by ATO no. 3 and ATO no. 5 but then abandoned. In 2007 the SISSI system has been completely dismissed and the function of water resource monitoring and data acquiring has been entrusted to the Marche Regione Civil Protection Department (MR-CPD) (The09b) due to a drought emergency in central Italy during summer 2007. Officers and technicians of MR-CPD define a simple and synthetic spreadsheet in Microsoft Excel delivered by mail to all ATO directors to be returned in 20 days. Hence, they start defining a new Web application starting from the acquired data.

1.4.3 An Inquiry on the Causes of the SISSI Failure

We have made a telephonic interview to one referent for each ATO in order to understand the most relevant misunderstanding of SISSI requirements and goals. We further analyze the technical choices of SISSI with the Marche Region officer responsible for the project. We summarize the main causes of failure of the SISSI project as follow:

1. ATO's employees were not well skilled in the usage of computers and Internet navigation. Their age is comprised between 45-60 year old. They use computers rarely and only for office automation tasks. Thus, they were unaware to use an unknown Web browser (i.e., Netscape) and the login certification system;

2. Only 2 of the 5 ATO's dispose of at least one worker sufficiently skilled in ICT that could help other fellow worker in case of issues with the system. This is particularly important in the first phases of system adoption when the great part of data should be inserted. Besides, a continuous assistance is likewise important in the periodical sending of measured data because it is a task performed 3 or 4 times per year and some employers forget how to use the system;
3. The SISSI project foresees only few training activities that have been judged insufficient from both the ATO's and the Marche Region employers devoted to use the Web application;
4. None processes have been studied and applied in the verification of inserted data and to control the actual delivery of data;
5. None institutional support has been provided to legitimate the Marche Region to require data to the ATO's (e.g., laws, regulations, etc.);
6. The fact that each client required the installation of Netscape resulted in a limitation on the number of computers enabled to use the SISSI system;
7. The form fields used to register water resources are excessive and unnecessary in relation to the primary goal of defining a map of existing resources. This has discouraged the ATO's to fill such a long form for each resource. Since the ATO's cannot fill all the form fields they do not fill at all.
8. The usage of the Web Services Architecture (W3C04a) in the SISSI Web application, where the main task is form filling is rather inconvenient. Such a choice creates a development overhead and reduces the ability to modify the system.
9. The new versions of the Netscape Web browser are incompatible with the SISSI system.
10. A single signature system for the login to the Marche Region system was not operative in 2003 and it is currently (2009) not fully implemented in all the ATO's.

We use the SISSI scenario to show how our contributes can be applied in a real word project. It is obvious that the project requirements and description have been highly simplified. Many details have been omitted in order to use SISSI as a reference scenario.

We emphasize that the identified causes of failure are all derived by lacks in connecting requirements expressed as goals to processes and implemented services. Even if the SISSI project is completely based on the Web Service SOA's its services have not been defined considering requirements and business goals the software should solve.

1.5 Outline of the Thesis

The rest of the thesis is structured as follow.

In Chapter 2 we present the state of art of related technologies and researches. In particular, Section 2.1 presents a survey on BPM and reengineering considering both the economic and the computer science point of views; Section 2.2 discusses how GSD practices has been applied also to the requirements engineering discipline in order to address multi-site distributed requirements gathering; Section 2.3 presents MDE technologies relating them to the enterprise-centric computing perspective that considers technology evolution very bounded to business and organizational assets. At the end of each section of this chapter we present current related works of the topic: Section 2.2.4 discusses the goals oriented approach to requirements engineering and introduce the goal-oriented i* language and the Tropos software engineering method; Section 2.1.5 presents the ATHENA Framework that is a project that share many motivation with our approach; Section 2.2.3 gives an overview of Softwiki that applies a wiki to requirements engineering practices and can be compared to our WikiReq tool.

Chapters 3, 4, and 5 presents our three contribute: Chapter 3 presents the ESI method and its three phases; Chapter 4 presents the BPR framework describing the transformations that support the goal to service mapping and the plug-in based approach to service static orchestration. Chapter 5 details the WikiReq and SMOTE tools that support our method by

means of the BPR framework technologies.

In Chapter 6 we validate the hypothesis of our thesis in a real case studio developed in the Civil Protection Department of the Marche Region concerning the development of some Web applications that directly affects the organization BP's.

Finally, Chapter 7 draws up our conclusions and future works.

Chapter 2

The State of Art of Business-Oriented Model Driven Development

In this chapter we present the state of art of the technologies and method related to our work. Section 2.1 discusses business modeling techniques currently used in software engineering for requirements acquiring and the relations of such activity with the reengineering of the organizations. Software and information related services are a fundamental parts of the companies business: they influence both the companies products and the companies organization. Thus, we consider an essential issue to study the way companies and their business will be redefined by the software.

In Section 2.2 we present the new challenges due to the “globalization” of companies. The issue of consider business modeling in software engineering is further complicated by the global dimension of the organizations. The global and multi-site organizational style is a problem in software engineering, new methods and approaches have to be developed. In particular, the globalization of companies is a problem for requirements engineering. A global context requires software engineering methods that take into account both the multi-site development to be performed in global software development companies and the multi-

site requirements gathering that must be performed in order to develop software for companies with a global organization.

Section 2.3 gives the state of art of some technological approaches that take into account enterprise-level issues in the development of software. Basing on such technologies, for instance MDA, new high level abstractions can be defined in order to manage software components. In particular, we study the service abstraction that can be used to relate the software with the organization.

At the end of each section of this chapter we present current related works concerning the topic.

2.1 Business Modeling and Reengineering

The concept of process definition in order to improve human activities is an old idea, for instance, in the V sec. B.C. Sun Tzu describes the need for hierarchical structure and communications in "The art of war" that can be considered the basis for strategic planning of modern business processes (118). More generally the study of processes and the study of the organizational models are closely related concepts (102). Thus, to understand how the business can be modeled and improved we need to understand also how business organizations are changed over the past two centuries and how they are changing nowadays.

In this section we present business modeling and how this discipline is evolved in recent years. Currently, the business modeling study requires to consider both economic and computer science approaches. The technology-centered perspective was first introduced by Michael Hammer in 1990 with the Business Process Reengineering (BPR) approach (83). Hammer claims to use information technologies to radically redesign business process in order to obtain effective improvement of the business rather to embed "outdated processes in silicon and software". Usually, BPR imply to obliterate existing processes and to rapidly change the organization structure to accommodate and improve opportunities given by the new technologies. Reengineering trigger changes in many aspects of the business, not just in the business processes themselves.

Anything associated with the process must be refashioned: job designs, organizational structures, management systems etc.

Thus, in order to understand the whole business not only processes must be analyzed (i.e., business process modeling). A more general activity of business modeling must be performed in redefining software and non-software systems. In the rest of this section we distinguish among business modeling and business process modeling and present details about notations to be used in business process modeling.

2.1.1 A Historical Perspective of Business Modeling

The concept of process in the business was first analyzed by Adam Smith (1723-1790) in the book "An Inquiry into the Nature and Causes of the Wealth of Nations" (148). In a pin farm he showed how the division and specialization of work improves productivity of the 24.000% . Subsequently, Frederik Taylor (1856-1915), a mechanical engineer, identified "scientific management" as a new discipline to study and optimizes work processes analytically. This approach introduced in enterprises a scientific method to study information and data about processes in order to optimize the productivity of the work. A strict separation of planning (management) and doing (work) was defined in the organizations. The Taylor model well fitted the organizations of the early 20th century. However, this approach underestimated one important factor: the technology. One of the most critical points of the Taylor theories was that they are not concerned with technology. Scientific management "took tools and technology as givens" (67) while technology is not a way to enhance organization performances; it is the principal source of organizational change.

A more systematic study of the Business Process (BP) field come from the Business Process Reengineering (BPR). In 1990 Michael Hammer (a professor of Computer Science) claimed that the major challenge for managers is to obliterate non-value adding work, rather than using technology for automating it (83). This statement accused managers of having focused the wrong issues, namely that technology and information systems has been used primarily for automating existing work rather than

using it as an enabler for making non-value adding obsolete. The idea was that most of the work being done does not add any value for customers, and this work should be removed rather than accelerated through automation.

Hammer and Champy (84) claimed that modern enterprises are the product of the industrial revolution. The methods of work of such enterprises (based on BP's) were defined at a time when management methods prescribed a decomposition of work into a set of elementary tasks that could be performed without thinking as discussed for Smith and Taylor theories. Enterprises then needed to put all these tasks together again into a meaningful whole. To manage these processes such organizations needed to develop large bureaucracies that monitored the status of the tasks and relayed the result of each task to the next task. Hammer and Champy claimed that the enterprises crisis that began in the 1970s was largely due to enterprises focusing inward on the tasks that they were performing rather than outward towards their customers. It thus spelled the enterprises inability to adapt to a changing environment. Enterprises that could not adapt were doomed to disappear. In BPR the required change could not be performed by evolving the existing processes of enterprises but by rethinking them from the core. Thus BPR prescribes radical change, rather than the continuous change.

Hammer defines some principles for reengineering that we resume:

1. *Organize around outcomes, not tasks*: the design of a person job should be organized around an objective or a outcome, not around a task or a single task of a given process. Step by step sequential processing (e.g., of an order) causes problems. For instance all data must be passed through the process step even if they are useful also in the step *n*. Often, these causes numerous errors and misunderstandings, moreover any new question about requirements have to come back all the steps down to the customer causing delays and reworks.
2. *Have those who use the output of the process perform the process*: departments, units and individuals that need the result of a process should perform the process themselves. Specialized departments that han-

dle specialized processes allow the process to work anyway but the work is slow and bureaucratic.

3. *Subsume information-processing work into the real work that produces the information:* an organization unit that produces an information should be capable also to process it. Often, there are units inside the organization aimed only to collect and process information. Modern technologies allow information producers to process information in a semi-automatic way.
4. *Link parallel activities instead of integrating their results:* parallel functions should be coordinated and linked together while their activities are in process and not after. For instance product development usually develop sub-systems independently and then integrates its results. In this way problems and errors come out only at the end, in the testing phase. By means of interconnection systems it can be performed an ongoing coordination.
5. *Put decision points where the work is performed:* the control should be put into the process. The process should have a built-in control, who do the work should make the decisions. That principle imply to rethink the hierarchical structure of the organization, pyramidal management layers can be compressed in order to allow a self-managing and self-controlling hierarchy. The manager role changes from one of controller and supervisor to one of supporter and facilitator.
6. *Capture information once and at the source:* This is a simple rule, to integrate and centralize information. Technologies such as online databases, bar code, Radio Frequency Identification (RFID) and other systems help retrieving and sharing information eliminating redundancy and overhead costs.

These principles help us to understand how much the organizational study and the process analysis is affected by technologies, in particular by software technologies. The contribution of computer science in this

discipline is not only limited to process modeling. Reengineering triggers changes in many aspects of the business, not only limited to the business processes themselves. Also the organizational structures, the strategies and the management must be revamped. This is because, in this thesis, we generically refers to business modeling and reengineering rather than business process modeling and BPR.

Actually, business modeling is a topic that spans between two disciplines: economics and computer science. The work on reengineering developed by Hammer (that was partially anticipated by Peter Drucker in the 80th (67)) has influenced many other similar approaches. For instance, the Lean Thinking approach (132) that is an economic strategy centered on the notion of value where everything that do not generate value is obliterated, exposes principles very similar to the business reengineering principles. Thus, reengineering, that comes from the ICT discipline, can be more generally interpreted as a theory that actually has affected economic business modeling over the past two decades.

2.1.2 The Discipline of Business Modeling in Computer Science

Software development affect business processes. However, it is in turn a process (i.e., a production process) that could be reengineered in a way similar to BPR and Lean Thinking (132). Indeed, modern approaches to software engineering such as eXtreme Programming (XP) and agile methods share many principles with lean management (56).

Despite these mutual influences, current software engineering processes lack to give an adequate support to analyze how the software system they are developing affect the business processes and more generally the whole organization (126; 172; 174).

The emergence, since the 1960's, of software systems in the enterprises has created a situation in which enterprises have become increasingly dependent on the capabilities of their software and ICT systems. This realization led Zachman to define a framework for Information Systems Architecture (ISA)(176). The ISA framework was renamed as "The Frame-

work for Enterprise Architecture” to show to what extent software systems were important to the enterprise (150).

The ISA framework is defined by Zachman as providing a “neutral, unbiased, independent” representation of the enterprise and its IT system. For Sowa and Zachman (150) the ISA framework’s purpose is to provide an overall view of the IT system and its relationship to the enterprise and the enterprise’s surrounding environment. From this perspective, failure to provide such a view may lead to the optimization of only some capabilities of the IT system and/or the enterprise. This partial optimization may come at the expense of the optimization of the enterprise as a whole, thus leading to results that are viewed as negative.

The ISA framework seeks to integrate a number of different representations of the enterprise in a model that is independent of any of these representations. The idea is to provide a holistic view of the enterprise as a whole and at the same time segmenting this holistic view into independent viewpoints so that the each of these viewpoints can be reflected upon, in isolation from the other viewpoints.

The term architecture is used to show the analogy between “the construction of a computer system and the construction of a house” (150). Hence, the ISA framework is based on analogies between traditional building architecture and software systems requirements definition, design, and implementation.

We show the the ISA framework in Figure 1.

This ISA framework has six columns and 5 rows. The rows represent analogs of the architectural levels to an enterprise with a focus on IT systems. The rows are named: Scope description; Model of the business; Model of the Information System; Technology model and Detailed description. Zachman states that each row is not merely a more detailed description of the row above it. Rather, it is a different representation done for a different purpose by different disciplines.

The columns represent the kind of questions that can be asked about each of the rows. Thus, the cells of the table represent answers to the questions: “what entities are involved, how they are processed, where they are located, who works with the system, when events occur, and why these

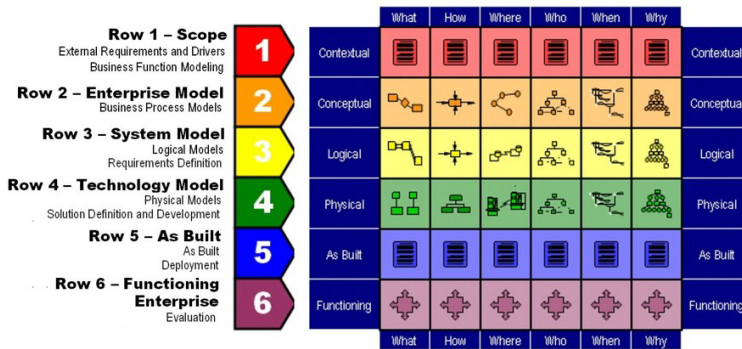


Figure 1: The ISA framework matrix

activities are taking place” (176) for each one of the views represented by a row.

Zachman states that the order of the columns does not represent a hierarchy of importance. The ISA framework requires that entities and their relationships, their functions, the people who manipulate them and their motivations be separated into four separate columns with four different notations. Thus, the end/means hierarchy (why) is separated from the entities and their relationships (what), from the activities performed on these entities (how), and from the people performing these activities (who). From our point of view these aspects can and should be modeled together. Also, the separation of the what and the who columns is quite artificial.

Computer Scientists’ business modeling and BPM is aimed to better understand requirements for the system to develop. Several software engineering processes include a BPM phase to better understand requirements of software. In order to give an overview of how BPM has been used in software engineering we present the Rational Unified Process (RUP) (105) approach.

RUP is an iterative software development process that integrates many

software engineering best practices, it is a process framework that can be adapted and extended to suit the needs of an organization. RUP uses two software engineering perspectives (named disciplines): the development discipline and the support discipline.

Business process modeling is the first development workflow of the RUP process, it does not focus on business reengineering and it is aimed to model the environment of the system in order to support project and change management activities and to better understand system requirements.

An important feature of RUP is that it uses software engineering techniques and languages to perform business modeling. The language, and thus the diagrams, used for business modeling are the same used for system modeling. This has many advantages in mapping processes concepts to system concepts in the requirements documents.

In RUP BP's elements are intended as follows:

- Business Actors: represent business users e.g., customers, vendors, or partners
- Business Use Case and Business Use Case realizations: represent BP's. They refer to business goals and business rules
- Business Workers: represent the roles that people play in an organization
- Business entities/events: represent the "thing" that an organization manages and/or produces. They are organized in business systems

As depicted in Figure 2, such elements are used to produce artifacts by two roles of BPM: the Business Process Analyst and the Business Designer. The former leads BPM activities outlining the organization being modeled, for instance it captures business goals and determines business actors; the second details the specification of a part of the organization, for instance it describes one or several business cases, it determines the business workers and business entities defining their responsibilities, operations, attributes and relationships.

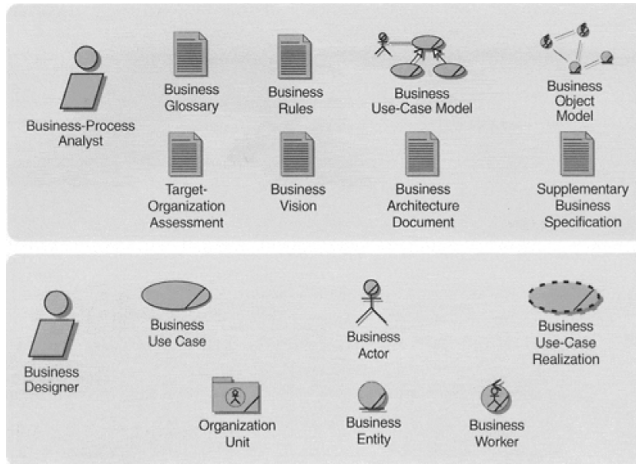


Figure 2: Roles and Artifacts of business modeling in RUP

The most relevant artifacts of the RUP business modeling workflow are:

- **Business Use Case Model:** it is a model of the business intended functions used as an input to identify roles and deliverables in the organization.
- **Business Analysis Model:** it is an object model describing the realization of the business use cases.
- **Target-Organization Assessment:** it describes the status of the organization in which a system is to be deployed.
- **Business Architecture Document:** it provides an overview of architecturally significant aspects of the business from a number of perspectives. It is used in reengineering some processes or to describe the business to other parties.

One of the advantages of using the same formalism to model both the BP's and the system, is to define a mapping among elements of the organizational domain and elements of the system to be.

RUP defines this mapping and allows to derive parts of the system specification directly from the BP models. System Use Cases can be identified starting from the business object model: for each business worker a candidate system actor is identified and for each business use case the business actor participates a system use case is created.

If the goal of the designer is to automate a BP, then the business worker is not mapped into the system actor, but the business actor is mapped into the system actor and communicates directly with the system.

Business entities will correspond to entities in the analysis model of the information system and other not specified mappings can be performed in order to extract other system requirements (i.e., user profiles, legacy considerations, system architecture constraints etc.)

RUP has several drawbacks: it does not support a cross-functional BPM; it is limited to the development process and it does not support interactions among different teams. In order to meet the "real-world needs of typical organizations". Thus, RUP has been extended with a further discipline named the Enterprise Discipline by the Enterprise Unified Process(EUP) (15).

In particular, EUP extends the business modeling workflow of RUP to support the enterprise view of BP's. The Enterprise Business Modeling is mainly intended to represent all the knowledge about the enterprise and to reengineer the business by means of technology.

EUP produces artifacts at a higher-level than RUP business modeling. It includes: the definition and analysis of the enterprise strategy; the identification of processes implementation options; the modeling of the organization and the domain besides the BPM.

BP's can be shared and used for different development projects. They are more stable and allow an evaluation of business strategies, rules and organization. The business modeling activity it is intended to model the vision and the mission of the business. It identifies 3 main dimensions for BP's:

1. External Environment: it models the environment which surrounds BP's. Customers and their needs are identified, as well as suppliers,

vendors, partners and competitors.

2. Business Processes: they identify BP's for marketing, administration and sales. An important part is the identification of the offerings (services and products) the organization provides to customers.
3. Critical Business Rules: the critical business rules, policies and fundamental ideas of are also identified and modeled. These BP's allow to observe and validate the adequacy of such rules.

Some overlaps exist between the EUP and RUP activities. However, the BPM scope of each process is different. Enterprise Business Modeling (EBM) looks at the entire business, whereas business modeling typically looks only at the business related to a single system. Thus, EBM can be used in business modeling of several projects, sharing a unique vision for the business, the organization and the main goals.

Despite the EUP can be considered a meaningful advance in extending software engineering to business modeling it does not furnish an effective support to analyze how business processes are affected by the system to be developed and how such a system will affect the goals and strategies of the entire business. EUP tries to perform a more accurate requirements engineering in order to "deliver systems that meet all the needs of today's businesses" (15). It allows a more accurate requirements and domain engineering, it is about to better relate project management with software development. Nowadays, EUP, RUP and many other software development processes lack for a business driven approach that consider software as an instrument to rethink and redesign the enterprise (a leaner enterprise if possible) rather to automate some tasks.

2.1.3 Avoid confusion: Business Process Modeling is not (only) Process Modeling

The studies on BPM are based on the observation that each product that a company provides to the market is the outcome of a number of activities performed. BP are a way to model and organize such activities in order

to improve them and understand their interrelationships. Software and IT play a fundamental role in BPM, because more and more activities that a company performs are supported by information systems. While at an organizational level, BP's are essential to understanding how companies operate, BP's also play an important role in the design and implementation of flexible information systems. As we discuss in the previous sections BPM is influenced by concepts and technologies from both business administration and computer science.

In order to understand the differences in how BPM is performed at an organizational level and to support information system development, we present BP's basing on the organizational vs. operational dimension (162). As depicted in Figure 3 they can be identified five levels of abstraction in BPM. At the highest level *business strategies* of the organization are identified. In the second level business strategies are divided into a set of *goals* and sub-goals that contribute to satisfy the related strategy. The third level concern with *organizational BP's*, such as high-level BP's sometime described in textual form only considering process inputs, outputs and expected results. *Operational BP's* describe in detail activities and relationships among processes needed to support organizational BP's. Finally, *implemented BP's* contain information on the execution of BP's activities and information on the environment where such activities have to be executed.

There are multiple ways to implement a BP. For instance it can be specified by means of written procedures and policies to be adopted by workers in the organization or it can use a process management software platform. Also formalism to describe such processes can be various: ranging from informal and semiformal techniques used for the three higher level depicted in Figure 3 to more formal and consistent languages adopted in the remaining BP's levels.

Despite all the levels described in Figure 3 can be defined in terms of processes, a separation of concerns is needed here. In particular process models are not a good starting point for identifying business stakeholder requirements. For instance in a e-commerce scenario a projects start with the design of a business model stating what is offered by whom to whom,

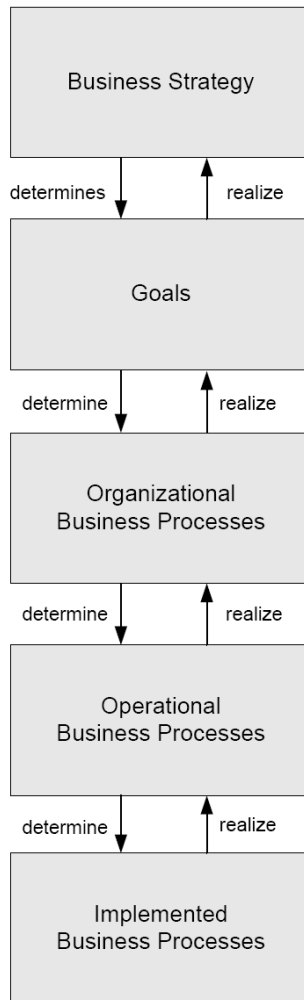


Figure 3: Modeling levels in BPM.

rather than with the design of a process model stating how these offerings are selected, negotiated, contracted and fulfilled operationally (82).

Thus is evident that different levels of the BPM require different formalisms to describe their meaning. In particular the first three levels are different in content to the remaining two levels. In this thesis we will refer to Business Modeling to the representation and management of the strategies, goals and organizational BP's and to Process Modeling to the representation of operational ad implemented BP's.

A business model represents the way of doing business in terms of the stakeholders strategic goals in order to create and exchange objects of value. It represent "the way of doing business" exposing the "what" and the "who" of the business. BPM further specify the "how" of the business defining how resources and other entities identified in Business Modeling must be used in order to produce a value for the organization.

Gordijn et al. identify several differences among these two levels of concerns (such as, Business Modeling vs. process modeling)(82):

- Business Modeling is aimed to define common understanding between stakeholders regarding the business, its organization, and the way the business is carried out (involved actors, resources, tasks etc.). Rather process model is aimed to clarify how processes should be carried out, and by whom.
- Contents in Business Modeling are centered around the notion of value, while in Process Modeling concepts focus on how a process should be carried out and the related issues.
- In Business Models value exchanges represent a transfer of ownership, while in a process model a flow of information or goods implies a change of state.
- Decomposition rules of model are different. Business Models use decomposition of value-adding activities as a way to discover new value-adding activities. Process Models serves the goal of clarity, or studying various resource allocations to activities.

- In Business Models, object properties are used by the stakeholders to determine the value of an object. In Process Model properties are used to determine the state transitions.

Thus, both Business Modelling and process modelling are forms of conceptual modelling but they differ both for the type of information they describe and in the way such information are described.

Usually, Business Modeling in the first three levels of Figure 3 are performed by requirements engineering formalism (172) and regards only marginally BP's representation. In particular, strategies and goals can be represented by goal oriented requirements engineering formalism like *i** and Tropos (126) or by not-process-oriented formalisms like the UML Use Case diagrams(105).

Instead, operational and implemented BP's can be represented with process-oriented (i.e., workflow-like) formalism (164). Such formalism share a theoretical underpinning on the pi-calculus and Petri Net (86) and allow choreography, orchestration and high level design of processes (e.g., BPEL4WS (BEA03), WS-CDL (W3C04b) and BPMN (OMG06) are all comprised in this category). In Figure 4 we report how the most noticeable process modeling standard relate to pi-calculus and Petri Nets.

Obviously, we can apply also process modeling formalisms to the business modeling. In particular it could be useful to exploit the BPMN, that allows to define an high level representation of processes, to represent some organizational BP's. However, not all business modeling concepts can be represented in terms of processes. Also from the implementation point of view there can be some problem to define all the software system by means of BP's. Process modeling is concerned only with process-oriented applications (86) such as applications whose processes have the following characteristics (86):

- Atomicity: processes are executed in an atomic way such as "all or nothing"; they are completely executed or none.
- Consistency: a process when executed terminate in a valid terminal state of the system.

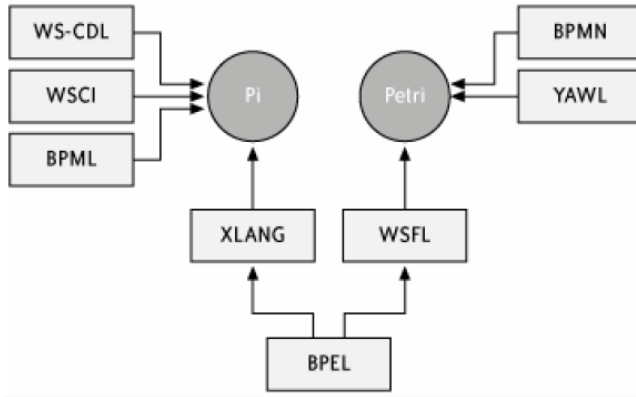


Figure 4: Theory family tree for BPM standards as envisioned in (86)

- Isolation: a process does not influence a not related process either if they are concurrently executed.
- Durability: the result of a completed process must not be discharged.
- Long running: from start to finish the process require hours, weeks, month or more to be performed.
- Persistent state: because the process is long running the state reside in a database.
- Bursty: the process sleep most of the time and waits to triggering events to wake up.
- System/Human Orchestration: the process manage the communication of many system and human actors

Thus there can be application that are more or less process-oriented also at an implementation level. For instance, Process Modeling is suitable for the modeling of a booking process in a travel agency but is less suited for a On Line Transaction Process (OLTP) application, such as ATM's, because of lacks longevity and a persistent state in the latter example.

In this thesis we will refer BPM and Business Modeling as synonymous, that is as the discipline that comprises all the levels depicted in Figure 3. In order to clarify the terms we used and the way we use them in Table 1, we furnish a general glossary for BP and related terms.

2.1.4 Notations for Business Process Modeling

In this section, we present an initiative about BPM aimed to connects the economic and the computer science perspective on BP's design. It is the development of the Business Process Modeling Notation (BPMN) (OMG06) standard that is aimed to give a standardized notation for BPM for these two research domains. For these reasons, it is particularly suited to the modeling of organizational and operational BP's.

BPMN synthesizes the BPM community best practices. It defines a graphical notation for BP similar to flow-charts which is aimed to be understood by stakeholders, analysts, business users, developers etc. BPMN can be incrementally adopted and supports automated translation to BP execution languages (such as BPEL4WS(BEA03) and ebXML).

In BPMN there is a core set and a full set of elements. The core set is useful for many BP's while the full set is needed only for complex BP's. The core elements are divided in 4 categories:

1. Flow Objects: they are the main graphical elements (Events, Activities, Gateways)
2. Connecting Objects: they allow to connect flow objects (Sequence Flows, Message Flows, Associations)
3. Swimlines: they group elements and processes (Pools, Lanes)
4. Artifacts: they are used to add (graphically) further information to the process (Data Object, Groups, Annotations)

Figures 5 and 6 represent respectively, the Flow Objects and the remaining BPMN core elements. For each BPMN element it is defined: a graphical notation; a semantic description in natural language and a set of standard attributes. BPMN further defines mechanisms to exchange




Element	Description	Notation
Event	An event is something that “happens” during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.	
Activity	An activity is a generic term for work that company performs. An activity can be atomic or non-atomic (compound). The types of activities that are a part of a Process Model are: Process, Sub-Process, and Task. Tasks and Sub-Processes are rounded rectangles. Processes are either unbounded or a contained within a Pool.	
Gateway	A Gateway is used to control the divergence and convergence of Sequence Flow. Thus, it will determine branching, forking, merging, and joining of paths. Internal Markers will indicate the type of behavior control.	

Figure 5: BPMN core: flow objects

BP models connection rules for some elements patterns for complex associations; standard flows and ways to perform the BPEL4WS mapping.

A process in BPMN is an activity performed in the organization. It is represented as a graph of Flow Objects that are sets of other activities and connection objects. BPMN business processes, more generically, are a set of activities which are performed within an organization or across organizations. A BP thus consists in a set of one or more separate Processes.

If in a BP Diagram more than one process is used, the usage of Pools is mandatory. Pools are a way to distinguish a business entity or a business role that interact in the BP by means of of Message Flows. The artifacts are used to facilitate analysis and reading of the BP but they do not influence the flows.

BPMN is an important attempt to define a standard way to represent BP's. It specifies only the notation for business modeling (and not an entire process as described for RUP and EUP). However, the opportunity to use a standardized notation for a BP will be a strategic feature in order

to define a unique framework for reengineering shared among software engineers and managers.

2.1.5 The ATHENA Model-Driven Interoperability Framework

The Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications (ATHENA) project (The09a) is an integrated project sponsored by the European Commission in support of the Strategic Objective "Networked businesses and government" set out in the IST 2003-2004 Workprogramme of FP6. ATHENA aims to make a major contribution to interoperability by identifying and meeting a set of inter-related business, scientific and technical, and strategic objectives.

We consider ATHENA a work related to the approach presented in this thesis since it also attempts to develop software components taking into account the impact of such software in enterprise processes and interoperation.

The ATHENA consortium currently comprises 19 leading organizations in research, academia, industry and other stakeholder communities including SME's, working collaboratively in pursuit of a common set of objectives in interoperability.

The ATHENA Model-Driven Interoperability (MDI) Framework provides guidelines for how MDE approaches can be applied in developing interoperable enterprise software system. As showed in Figure 7, the ATHENA project merges three research areas supporting the development of Interoperability of Enterprise Applications and Software:

1. Architecture and Platforms: to provide implementation frameworks,
2. Enterprise Modelling: to define interoperability requirements and to support solution implementation,
3. Ontology: to identify interoperability semantics in the enterprise.

ATHENA adopts an "holistic" perspective on interoperability in order to achieve interoperation between enterprises. It identifies a structured approach to collect, identify and represent the current state of the



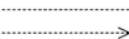





Element	Description	Notation
Sequence Flow	A Sequence Flow is used to show the order that activities will be performed in a Process.	
Message Flow	A Message Flow is used to show the flow of messages between two participants that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two participants (e.g., business entities or business roles).	
Association	An Association is used to associate information with Flow Objects. Text and graphical non-Flow Objects can be associated with the Flow Objects.	
Pool	A Pool represents a Participant in a Process. It is also acts as a “swimlane” and a graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.	
Lane	A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organize and categorize activities.	
Data Object	Data Objects are considered Artifacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process, but they do provide information about what activities require to be performed and/or what they produce.	
Group (a box around a group of objects for documentation purposes)	A grouping of activities that does not affect the Sequence Flow. The grouping can be used for documentation or analysis purposes. Groups can also be used to identify the activities of a distributed transaction that is shown across Pools.	
Text Annotation (attached with an Association)	Text Annotations are a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram.	

Figure 6: BPMN core: connecting objects, swimlines and artifacts

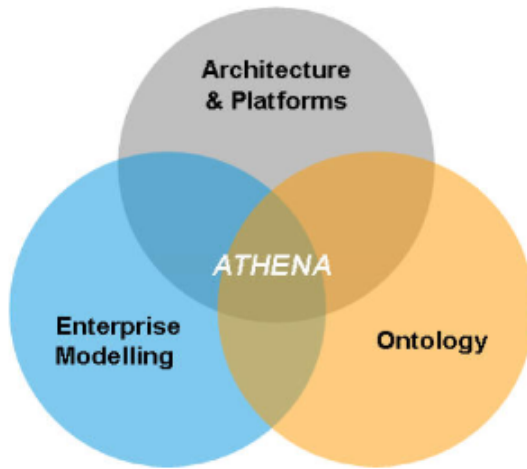


Figure 7: Research areas of the ATHENA Interoperability Project

art, vision statements, and research challenges. It defined a framework for capturing and inter-relating this information from many perspectives consisting in three layers:

- The business layer is located at the top of the framework. In this layer, all issues related to the organisation and the operations of an enterprise are addressed. Amongst others, they include the way an enterprise is organised, how it operates to produce value, how it takes decisions, how it manages its relationships (both internally with its personnel and externally with partners, customers, and suppliers).
- The knowledge layer deals with acquiring a deep and wide knowledge of the enterprise. This includes knowledge of internal aspects such as products, the way the administration operates and controls, how the personnel is managed, and so on, but also of external aspects such as partners and suppliers, laws and regulations, legal obligations, and relationships with public institutions.

- The ICT systems layer focuses on the ICT solutions that allow an enterprise to operate, make decisions, exchange information within and outside its boundaries, and so on.
- The semantic dimension cuts across the business, knowledge and ICT layers. It is concerned with capturing and representing the actual meaning of concepts and thus promoting understanding.

To achieve meaningful interoperability between enterprises, interoperability must be achieved on all layers. In particular in business layer the business environment and business processes; the knowledge layer organisational roles, skills and competencies of employees and knowledge assets; in the ICT layer applications, data and communication components and finally, in the semantics support mutual understanding on all layers.

The interoperability framework integrates principles of model driven, service oriented and adaptive software architectures. Model driven architectures focus on design-time aspects of system engineering. Model driven development methodologies describe how to develop and use models as an active aid in the analysis, specification, design and implementation phases of an ICT system. Service-oriented architecture (SOA) specifies systems composed of services offered by various service providers, which provides the basis for supporting new business models, such as "virtual organisations". Adaptive software architectures focus on run-time aspects of system engineering. Agent and P2P technologies enrich an ICT system with dynamic and adaptive qualities.

The ATHENA model-driven interoperability framework builds on the vision: "Enterprises are able to flexibly develop and execute interoperable applications based on model-driven development approaches to service-oriented and adaptive software solutions" (The09a)

The framework is structured in three main integration areas:

- Conceptual integration which focuses on concepts, metamodels, languages and model relationships. It provides us with a foundation for systemising various aspects of software model interoperability.

- Technical integration which focuses on the software development and execution environments. It provides us with development tools for developing software models and execution platforms for executing software models.
- Applicative integration which focuses on methodologies, standards and domain models. It provides us with guidelines, principles and patterns that can be used to solve software interoperability issues.

For each of these three areas a reference model to describe and support the application of model-driven development of software systems is specified.

The reference model for conceptual integration has been developed from a MDE point of view focusing on the enterprise applications and software system. A CIM corresponds to a view defined by a computation independent viewpoint. It describes the business context and business requirements for the software system. A PIM corresponds to a view defined by a platform independent viewpoint. It describes software specifications independent of execution platforms. A PSM corresponds to a view defined by a platform specific viewpoint. It describes the realisation of software systems. The models at the various levels may be semantically annotated using reference ontologies which help to achieve mutual understanding on all levels. ATHENA also uses interoperability patterns for horizontal and vertical integration.

ATHENA identifies four categories of system aspects where specific software interoperability issues can be addressed by conceptual integration. These four aspects can be addressed at all three CIM, PIM and PSM levels. *Service aspects* are an abstraction and an encapsulation of the functionality provided by an autonomous entity. *Information aspects* are related to the messages or structures exchanged, processed and stored by software systems or software components. *Processes aspects* describe sequencing of work in terms of actions, controlflows, information flows, interactions, protocols, etc. *Non-functional aspects* represents extra-functional qualities that can be applied to services, information and processes.

The architecture of the enterprise applications and software systems

can be described according to a 4-tier reference architecture where each tier provides different software services required by the enterprise. The software system itself is coupled to an ICT infrastructure illustrated by a service bus that provides the necessary communication infrastructure. Infrastructure services such as composition, mediation, matchmaking and transformation that enables interoperability between software systems should be provided. The service bus will make use of infrastructure services, and registry and repository.

The MDE methodology needs to follow a structured approach where interoperability requirements from business operations in a networked enterprise drive the development of software solutions. This means that MDE methodology needs to be related to enterprise architectures. A specific part needs to address how the MDE concepts and the technical software components are reflected in a model world of the enterprise. The Figure 8 shows how the model world, reflecting the applicative integration, is related to the reference models for conceptual and technical integration. Enterprise and software models can be built to understand and analyse the interoperability requirements of an enterprise.

An enterprise model describes a set of enterprise aspects, which includes business models capturing actors and stakeholders, business services, business information and business processes. These business models provide a context for the software solutions that needs to be developed and integrated. Software models describe how software systems are used to support the businesses of an enterprise. The software models further refines the business models in terms of software realisation models. All these models should include descriptions of the four system aspects identified in the reference model for conceptual integration. The software models can be classified as CIM, PIM or PSM models according to a MDE abstraction.

The realisation of the framework focuses on providing guidelines and existing method chunks for creating or customizing the development and integration methodologies based on these principles, and guidelines and existing assets for creating or customizing your own MDI tool set.

The MDI framework aims at providing guidelines for the following

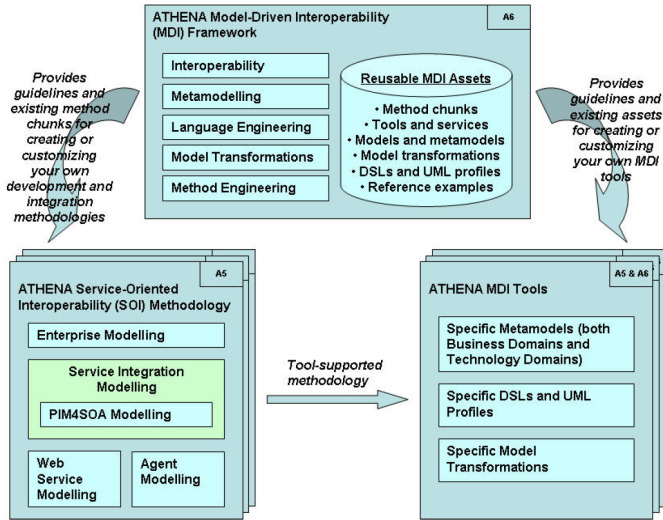


Figure 8: The ATHENA Realization Framework

topics: Model-driven architecture (MDA) and interoperability; Metamodelling; UML profiles and domain-specific languages (DSLs); Model transformations and Method engineering. In addition it provides reusable assets in terms of method chunks, tools and services, models and metamodels, model transformations, DSLs and UML profiles and reference examples. The assets in the form of examples and tutorials will focus on applying Eclipse technologies for implementing your own MDI methodology and tool suite.

2.2 Global Requirements Engineering

Despite the modeling of the process-oriented part of the business, it is of fundamental importance in our framework to exploit also other formalisms that allow to model the business from others perspectives. A great effort in this direction has been performed in the requirements engineering disciplines (171). Requirements engineering is an activity that

is not only restricted to software engineering, but that can be performed in the engineering of a generic system (157). In particular, we are interested in global requirements engineering, such as the study of how to gather requirements for a system in a distributed organizational and development context. Studies and techniques for the global context are an interesting starting point to address issues due to modern enterprises organizations. Actually, requirements engineering issues identified for distributed projects (93) are likewise important in a single-site software development context (147) in particular relating to the needs of collaboration and information sharing.

In this section we give an overview of some aspects of requirements engineering that are related with our work and describe how the requirements engineering practices has been applied in a global context. We discuss in detail the wiki-based approach and the goal-oriented approach to the management of requirements and they relationships with our contribute.

2.2.1 Requirements Engineering

Requirements engineering (RE) is concerned with producing a set of specifications for software systems that satisfy their stakeholders and can be implemented, deployed and maintained (93). A classical definition of requirements is: "A specification of what the system should do without specifying how it should do it" (20; 107). This is a quite sibylline definition since in practice it is difficult to separate the "what" from the "how" (108).

The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. Broadly speaking, software systems RE is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. There are a number of inherent difficulties in this process.

An assessed definition of requirements engineering (131; 157) is the Zave one (177): "Requirements engineering is the branch of software en-

gineering concerned with the real world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families”.

This definition highlights the importance of “real-world goals” that motivate the development of a software system. These represent the “why” as well as the “what” of a system we discuss in the ISA framework and help us to connect RE with the BPM discipline.

We can summarize the reason that make the RE process so complex (93; 107; 157):

- Usually stakeholders cannot express what they need. They may not share the same perceptions of their problems with an external observer.
- Many users have great difficulty explaining what tasks they perform, and even more difficulty in explaining why they carry out these tasks.
- Often stakeholders specify a solution instead of a demand.
- The target system is not just a piece of software, but also comprises the environment that will surround it: is made of humans, devices, and/or other software.
- Stakeholders find it difficult to imagine new ways of doing things, or imagine the consequences of doing a familiar task in a proposed new way.
- There are multiple concerns to be addressed beside functional (e.g., safety, security, usability, flexibility, performance, cost, etc.) These non-functional concerns are often conflicting.
- Often different stakeholders have conflicting views because of different background, skills, knowledge, concerns, perceptions, and expression means.

- Stakeholders will often reject proposals due to a general resistance to change.
- Demands change over time. External factors change and priorities change. Once a demand is met, new ones turn up as a result.
- Requirements specifications may suffer a great variety of deficiencies (121).
- RE covers multiple intertwined activities in business and engineering contexts and it is not just a single phase that is carried out and completed at the outset of product development. (93).

Requirements engineering has emerged as a field of research independent from the disciplines in which it was incepted, software engineering, systems engineering, computer science, in an effort to manage all aspects of requirements in order to specify complete and correct requirements despite problems such as those listed above.

The question of RE thus becomes: "what is the environment of the IT system and how it can be described?". The environment of an IT system is usually thought of as being the enterprise for which it is built. The definitions of RE given above reflect the point of view, largely shared among requirements engineers, that the enterprises and stakeholders that constitute the environment of an IT system are mainly motivated by the satisfaction of goals.

According to Nuseibeh and Easterbrook (131) "the context in which RE takes place is usually a human activity system, and the problem owners are people". The fact that a RE activity should take place implies that some new computer-based system could be useful or needed, but such a system will change the activities that it supports. Thus, at this point is clear that RE and BPM are very bound activities.

Due to the nature of the object that RE should model and the involved stakeholders, RE methods needs to be sensitive to how people perceive and understand the world around them, how they interact, and how the sociology of the workplace affects their actions. RE should thus consider cognitive and social sciences issues in defining practical techniques for

eliciting and modelling of requirements (131). In particular Cognitive Science can help in understanding the difficulties people may have in describing their needs; Anthropology provides a methodological approach to understand how computer systems may help human activities and if they will be accepted by employers and, finally, Sociology can help in understanding the political and cultural changes caused by the introduction of a new software system (e.g. how they affect the structure and communication paths within an organization).

Thus, RE can be considered as a discipline that include business modeling in particular regarding to the requirements related to the organization and its processes (171). Indeed, the Zachman ISA framework can be considered a way to organize the various requirements types (87). In particular, business goals are the abstraction that connect RE and business modeling. Business requirements are the essential activities of an enterprise that bring to the need of developing systems and software. Great part of business requirements are derived from business goals (i.e., the objectives of the enterprise or organization). In order to identify business goals, all the business and organizational activities should be considered and thus BPM is an essential activity to be performed in all RE processes (171).

An interesting aspect analyzed by RE is the requirements traceability. In RE context, traceability is about understanding how high-level requirements (such as objectives, goals, aims, aspirations, expectations, needs etc.) are transformed into low-level requirements. Traceability relates low and high level requirements many-to-many: one lower level requirements may be referred to several higher level requirements and vice versa. Traceability is important not only for requirements management but also in RE analyses (93):

- *Impact analysis* allows to determine the impact of requirements change on other system artifacts.
- *Derivation analysis* helps in understanding the change impact starting from low level artifacts.
- *Coverage analysis* can be used to determine that all requirements do

trace downwards to lower layers and thus that all elicited issues has been considered.

The RE point of view brings to identify goals to be achieved by enterprises and stakeholders and to transform these goals into requirements for the IT system. For those methods to be successful, they need, as input, goals that the enterprise seeks to satisfy. Thus they implicitly assume, or sometime explicitly state, that goals are to be found in the enterprise.

2.2.2 Requirements in Distributed Projects

A multi-sited distributed context adds a new dimension in the complexity of RE. In particular, effective knowledge sharing platforms and relationship building practices are needed (57). In global RE stakeholders operate in geographical distribution. This situation not implies only a two-site distribution (i.e., customer stakeholders located differently from software engineers) but it is usually complicated by GSD. Client organizations, such as a government department, outsource a software to a geographically remote vendor that may rely on other software development teams closer to the client location. Such a situation described in Figure 9 explain the global context motivation of our thesis that we introduce in Section 1.1.1.

In GSD the global challenges are referred to coordination and management of development teams. The distributed dimension of requirements engineering introduces a more challenging situation that require to consider also a multi-sited distribution of stakeholders. Damian recognize that the challenges introduced by global RE concern mainly knowledge and change management, she claim a clear separation between: the knowledge related to the application/user needs and the knowledge related to the project development. In a investigation on 28 distributed projects Smite recognize that the five top threats in distributed project derive from a poor global RE (147). In Table 2 we report the result of such Smite's survey.

However, the solution to both application/user needs and the project development needs is the same (57; 147). They are needed new platforms

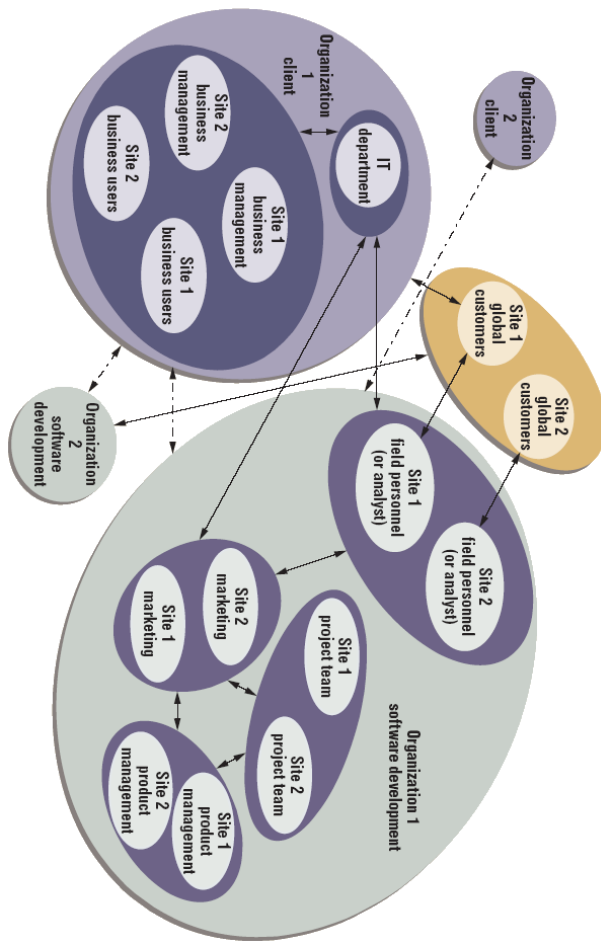


Figure 9: Global Requirements Engineering main stakeholders as described in (57)

to support GSD and Global RE that allow:

- A rapid adoption by development teams: because the process maturity and expertise of involved development teams is usually different and their work practices may differ, thus they have to adapt the system in their contexts.
- Easy to learn for involved stakeholders: because they "do not have sufficient human resources to validate all software requirements specifications" (147).
- Both synchronous and asynchronous collaborations: because global virtual collaborations requires to collaborate mostly asynchronously (e.g., in case of worldwide development it is a must due to timezone differences). Asynchronous collaborations also help to overcome language differences since many stakeholders are not able to speak fluently other languages and misunderstandings can arise.
- An integrated and easy to use/learn version management system: because the set of involved stakeholders, project managers, developers and users may be large and culturally conflicting.
- The rapid share and update of the knowledge about the project: because all changes must be known early and by everyone.
- The definition of standard template for requirements gathering and analysis and the definition of a common glossary.
- A reduction of the face to face meetings: because they are expensive for all involved organizations.
- A constant communication: because there are small doubt and questions that can be replayed soon or more appropriately with advanced and innovative communication tools. The e-mail exchange is not sufficient in global RE (58). Messaging tool, chats, videoconference systems enrich the communication possibility.

- Synchronization among the development teams: because GSD requires commonly defined milestones and clear entry and exit criteria (89). Good tools supporting collaboration among time and space are required.
- The use of a general and clear architectural software abstraction: because the absence of a clear trace of what software components address a requirements can result in misalignment and rework.

Summing up, Smite's survey shows that requirements management in global projects is one of the essential challenges that shall be paid adequate attention. The "virtual product development" is considerably more complex than projects managed entirely in-house(98). Communication plays an important role and the dimension of the GSD problem is sometime misunderstood. Solutions that help collaboration in globally distributed context will help in-house development: "we probably think of *distributed* work in much too limited a way. Distances need not be global to be important. In fact, being in another building or on a different floor of the same building, or even at the other end of a long corridor, severely reduces communication." (89). Also cultural factors are not exclusively related to worldwide contexts but mainly to the organizational culture, such as differing attitudes toward hierarchy and communication styles. Finally, from a technical point of view, the process differences inherent in inter-organizational partnerships cause problems in aligning RE processes and supporting tools because each group has different processes for requirements analysis, documentation, and change management (58).

Thus, the principal requirements for tools and platforms supporting GSD and global RE are: a strong support of multi-channel multi-modal communications; a lightweight and easy to learn introduction of applications, languages and new practices supporting global RE; and a software architecture abstraction that allows a synchronous development of components that is consistent with requirements.

The Wiki Way to Requirements Engineering

A recent approach to requirements engineering is based on the usage of wikis for the gathering of requirements(109). Wikis are systems that allow the collaborative creation and edition of Web page content using a Web browser. They follow a text-oriented model using titles, paragraphs, lists, tables, figures, etc. as building blocks for creating hypertexts. They provide the user with a simple syntax, allowing any word to be easily turned into a hyperlink, hence supplying a flexible mechanism for organizing contents. Wikis are particularly appropriate in collaboration scenarios, they are capable of effectively presenting and editing Web-based information, using a very simple markup language, a powerful dynamic-linking mechanism based on lexical conventions, and support the notion of adaptive Web pages.

The simplicity of wikis makes them easily accessible to a wide range of possible users whenever a generalized availability of the produced contents is desired. Due to their simplicity and effectiveness as a medium for collaborative authoring, wikis are now widely extremely popular, spread all over on the Web and can be successfully adopted in requirements engineering. A large amount of the software documentation produced today is Web-based. Since wikis provide a nice environment for collaborative authoring of Web-based documents, wikis can be used as a tool to support the edition, organization and storage of software requirements. They can be effectively used to solve both issues related to the global context than the issues of relating the software with the organization (11; 140).

Indeed, the use of wikis in software engineering dates back to 1995 at the beginning of this technology. Ward Cunningham(109) created the first ever wiki as a platform for discussing patterns and software development efforts, namely in the Portland Pattern Repository(Cun96). The simplicity and effectiveness of wikis as a medium for collaborative authoring has led to their vast popularity across many domains.

They are often used to support software development, in particular in the area of open source software. In these contexts, wikis are seen as a lightweight platform for exchanging reusable artifacts between and

within software projects. Indeed, they facilitate communication through a basic set of features and delegate the actual way of coordination to the people using the wiki. From the point of view of the author, these basic features are (140):

- single place publishing: there is only one version of a document available that is regarded as the current version;
- simple and safe collaboration: the versioning and locking mechanisms that most wikis provide;
- easy linking: documents within a wiki can be linked by their title using a simple markup;
- description on demand: links can be defined to pages that have been not been created yet, but might be filled with content in the future.

Nowadays, there are a lot of wikis specifically used to support code development and documentation. They offer specific functionalities for software engineering like the creation and maintenance of software documentation, taking into account also needs like consistency maintenance, contents integration, reuse and process integration. We present an overview of them with their most important features (52; 140):

- Xiao et al. (169) propose the Galaxy Wiki project which integrates code and textual editing directly into the wiki. The per-page organization of the wiki architecture is exploited developing a one-to-one relation between a wiki page and a object oriented class. This straightforward relation helps not only in ease code production but also in documentation production and maintainance.
- The Trac system developed by Edgewall Software (Edg03) is a software project management tool based on wiki that allows to relate wikis pages to issues and vice versa. It also allows the integration between documentation and the software development artifacts, like source code files, issue tracker items and project milestones, etc.

- FitNesse (R. 08) integrates a wiki and an acceptance testing framework in order to create acceptance tests in a collaborative way (e.g., including stakeholders in the process of defining inputs and executing the tests). As for Galaxy Wiki the integration is at the page level, wiki pages may include both textual descriptions and tests, which can be ran from within the wiki page context.
- WikiDoc (M. 05) allows a collaborative effort among stakeholders to create code documentation. It consists in a wiki interface that allows to add help to the Java code.
- Aguiar et al. (10) propose the XSDoc system, one of the first experiments of source code and models integration in wikis. XSDoc does not allow source code and model editing directly inside the wiki but integrate the pages with other IDE's. In particular, it manages documentation and code in different page file allowing a more rich correlation among artefacts.
- TeamWeaverWiki (H. 08), similarly to Trac, integrate the wiki in the full software development process. It empower the viki with specific source code editors and search engine. It further integrate wiki pages with tools commonly used in software development like Concurrent Versions System (The98) and eclipse (The08a).

In recent years there are also interesting experiences in using wikis as requirements engineerings tools. In particular the wikis lightweight "philosophy" address many issues of the global requirements engineering context. Ward Cunningham confirmed this vision in a recent interview (May 2009) (War09) where he speaks about the wiki as an approach oriented to a wide active participation in software engineering. Such as, it is indeed the wiki "philosophy" that can be applied to the software and requirements engineering, allowing the opening of requirements elicitation to a wide range of stakeholders and permitting a more effective collaboration among software developers, also in the physical World.

Beside the practical reasons we expose above, wikis features help in addressing the GSE challenges presented in the previous sections. In

global software development, developers are located in different cities, different countries, or different hemisphere, communication and coordination is the major concern in the management of distributed teams.

Al-asmari and Yu (11) have published a study evidencing how wikis facilitate communication, coordination, and documentation writing in distributed software development. They analyze the communication system commonly used in GSD, such as:

- Travel. Globalization involves that people interact with each other more frequently and using many different methods, including travel. Working geographically dispersed should be just like working in the same building. Therefore, travel is still a necessary in order to bring team members face-to-face. However, "travel is one of the most expensive and time-consuming communication methods; the more you travel, the more you reduce the benefits of distributed development" (11). Therefore, travel should be used only if necessary.
- Phone. Conversations by Phone are the traditional communication technique in GSD. Since it is a very rapid communication, it has many limitations, for instance a phone call is more suited for a step-by-step instruction; involved stakeholders need to be available at the same time; time zone can be a problem; etc.
- Email. Email is widely used in collaborative work by distributed teams. However, email is an informal communication method, it does not support the structuring of contents, an effective system for versioning and backup.
- On-line project management software. Such software can provide information like project overview, project organization, project plan, time schedule, and work assignments. It is a formal communication method but most project management software only support one-way communication, that is from manager to developers; they do not support multi communication channels among developers.
- Content management software. Content management software allows members in distributed teams to work together on the same

document, such as the system design specification, at the same time. Some CMS allow also more complex documents, including PDF and Microsoft Word editing. However, most of the content management software is complex, need training to be understood and it does not support communications among users in an effective way.

All these tools have been employed to facilitate GSD communication and coordination. However, all these methods have their limitations. On the other hand, wikis are tools that simply allow users to freely create and edit Web pages contents using any Web browser. With the same simple architecture, wikis allow also to manage page history, email notification, passwords, messaging chats, collaborative drawing, etc. Therefore, wikis are an ideal tool for distributed software engineering. However, it does not replace completely the communication channel we discuss but integrate them and empower collaboration efficiency: wikis do not constrain the time schedule of the developers like face-to-face meetings and phone; message posts in a wiki are more easy to be noticed, shared and discussed by all users.

We remark that wikis do not introduce an innovative technology or a particular powerful instrument respect to other existing systems. Wikis introduce a different approach in the way stakeholders collaborate in software engineering, in a way that is, at the same time, easy to be understood and used by all stakeholders. Indeed, wikis are not more sophisticated than a database or a content management system, but, for instance: a database requires the people who uses it to have a prior knowledge of its working language, its restrictions and its internal organization (e.g., tables, relationships, primary keys, etc.) and if the project is large, it is unlikely that all people involved in it have the necessary skills to contribute actively to the development of documentation.

Wikis has all the advantages of a database (i.e., many implementations store its contents in a relational database), with the ease of access of a Web page (i.e., its contents are published through a Web browser) (14). The use of wikis in a working environment in GSD, solve also two of the main drawbacks of this technology such as: lacks in monitoring of the quality of content and vandalism. A comparison with the content

management systems, that are very similar to wikis regarding to the content management, has been given in one of our works (6) and reported in Table 3

Obviously, also the wiki technologies has some limitations (11): they currently do not support collaboration on complex document type editing, such as Microsoft Word and Microsoft Excel files; many wikis do not support a visual editing but a minimum of tag learning is required; an integration with other presentation tools such as video conference tools and phone services will be useful.

In any case, we think that the wiki-based approach is the best fitted choice in order to face the complexity of the business-oriented software development scenario. Wikis from one hand address the GSE issues due to a multi-sited distribution of development teams, from the other hand, the wiki approach guarantee a wide participation of all stakeholders in requirements and software engineering, that is a fundamental need in business modeling (68).

2.2.3 The SOP-wiki and Softwiki Projects

There are various works that emphasize the advantages of exploiting wikis as requirements engineering tools (62; 137; 140). In particular, the work carried out for the Software Organization Platform wiki (SOP-wiki) (161) and the SoftWiki (114) systems are interesting related works we analyze.

The SOP-wiki is an evolution of a software previously developed by Decker et al. (62) for the exchange of information and collaborative work on software artifacts. SOP-wiki is mainly based on the MediaWiki (Wik08) wiki platform that is a popular wiki engine used in many collaborative projects, such as Wikipedia (Wik09). The GUI of the wiki has been implemented by means of the Adobe Flex technology (Ado09) that allows at the same time to structure the requirements and give them a semantics. The Flex user interface is built around a browser control mechanism (i.e., a widget or a component in the context of the Flex framework), which allows browsing and editing wiki articles. It further allows to create sev-

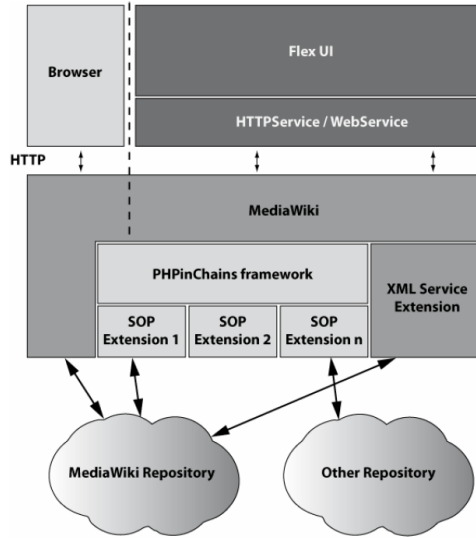


Figure 10: The SOP-wiki Architecture

eral user interfaces for several input devices. The SOP-wiki architecture is depicted in Figure 10.

SOP-wiki authors claims that it will keep track of all activities throughout the entire software process once integrated into other tools for software development by means of import and export capabilities.

The result of the SOP-wiki application experiences (62) are likewise important and useful in our work. SOP-wiki has been tested in small project involving from ten to thirty stakeholders. However, the result of this wiki are important in our work. Besides, we focus our attention in the application of the framework for SME's and in project that involve no more than one hundred stakeholders. Decker et al. identify six main drawbacks in using wikis for requirements engineering (62):

1. Remembering page names: in order to define page links, the user has to remember the page name and sometime interrupt page editing;

2. Versioning across several pages: wikis versioning features are usually per-page based, thus it is difficult to define a release based on the state of several pages in a particular moment;
3. Page structuring: most wikis support page structuring only by categories or namespaces, there is the need for more rich semantics in order to manage link between pages;
4. Page reclassification: wikis do not provide a way to classify and re-classify groups of pages, thus sometime repetitive operations are needed to re-classify pages when a requirements schema changes
5. Missing replication of contents: some stakeholders need to work offline from time to time and wikis does not support content replication and change management for offline changes.

They overcome a great part of these issues improving the wiki with a specific semantic support and by means of the usage of other developing frameworks. Thus, since a wiki can be used as is for requirements engineering purposes, small and not overloading adjustments are useful to solve such limitations and further improve the effectiveness of the application of wikis in requirements engineering.

In Figure 11 we report a table comparing the pros and cons of a general wiki platform and a wiki specifically adjusted for requirements engineering.

Another interesting project similar to SOP-wiki is SoftWiki (137). SoftWiki is a work aimed to support the collaboration of software requirements stakeholders in a very large and spatially distributed scenario. SoftWiki is based on an explicit requirements engineering ontology which comprise, among the others, the goal and the stakeholder concepts (114). Each requirement gets its own URI, then it is linked to other resources using Semantic Web standards such as RDF and OWL. SoftWiki exploit an explicit ontology for semantical annotation named SoftWiki Ontology for Requirements Engineering (SWORE)(170).

The aim of SoftWiki is to support the collaboration in potentially very large and spatially distributed user groups in order to give rich seman-

	General wikis	RE-specific wiki (for example, Software Operations Platform wiki)
Description	■ Requirements are stored in general-purpose wiki systems	■ Requirements are stored in RE-specific semantic wikis such as SOP
Pros	<ul style="list-style-type: none"> ■ Support for collaboration ■ Support for grouping requirements into individual documents (for example, for specific releases) ■ Support for requirements versioning ■ Low cost (mostly open source software) ■ No tool licenses required ■ Also applicable in other development phases, such as testing 	<ul style="list-style-type: none"> ■ Support for collaboration ■ Support for grouping requirements into individual documents ■ Support for requirements versioning and baselining ■ Low cost (mostly open source software) ■ No tool licenses required ■ Also applicable in other development phases, such as testing
Cons	<ul style="list-style-type: none"> ■ Untyped links (because regular wikis can't capture the nature of a link between documents) ■ No explicit baselining of requirements; versioning content across several pages is difficult ■ Users must remember page names to be linked ■ Difficult to export page content ■ Missing features for restructuring content, particularly for classifying and reclassifying pages ■ Missing replication of wiki content 	<ul style="list-style-type: none"> ■ Increased complexity of wiki syntax to denominate metadata ■ Missing replication of wiki content

Figure 11: Pros and Cons of wikis used in requirements engineering

tics to requirements. SoftWikis thus uses wikis focusing on its support in semantically annotated requirements rather than effectively gathering and structuring requirements. This project aims to enable semantic interoperability with further tools for instance exporting the requirements in RDF-format and manage them with other IDE.

2.2.4 The Goal Oriented Approach to Requirements Engineering

The need for Requirements Engineering (RE) activities implies that a new computer-based system has to be developed. Such a system will change the working activities that it supports thus it is clear that exists interesting links to study between RE and BPM. Early-requirements engineering techniques focused on the later phase of RE, which concern completeness, consistency, and automated verification of requirements (171). In contrast, the later phase of requirements engineering models and analyzes stakeholder interests and how they might be addressed, or compromised, by various system-and-environment alternatives. (173). Yu, Mylopoulos and van Lamsweerde (37; 126; 159) recognized the importance of goals in order to provide for a modelling and reasoning support in the early phase of requirements engineering. Requirements engineering research has in-

creasingly recognized the leading role played by goals in the RE process (158).

Goals, in this context, are understood as giving the rationale for stakeholders actions and hence, serve as the rationale for software system requirements. Thus, van Lamsweerde in the 2000 propose to (re)defines the RE basing on goals: "Requirements engineering is concerned with the identification of the goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints, and the assignment of responsibilities for the resulting requirements to agents such as humans, devices, and software" (157).

RE is concerned with producing a set of specifications for software systems that satisfy their stakeholders and can be implemented, deployed and maintained. Goal oriented requirements engineering takes the view that requirements should initially focus on the why and how questions rather than on the question of what needs to be implemented. Traditional RE analysis and design methods focused on the the functionality of the system to be built and its interactions with users. Instead of asking what the system needs to do, goal oriented methods ask why a certain functionality is need and how it can be implemented. Thus goal oriented methods give a rationale for system functionality by answering why a certain functionality is needed while also tracking different implementation alternatives and the criteria for the selection among these alternatives.

The goal oriented approach models human organizations considering both humans and organizations as goal-seeking entities. This approach derives from a model of software engineering that consider new systems to be built in order to help people and organizations achieving their goals (92). From this perspective all the RE discipline is now defined in the RE community as goal driven. Anton explains what differentiate goal-oriented methods from more "traditional" RE techniques in the following way: "Traditional systems analysis focuses on what features (i.e. activities and entities) a system will support. Goal-based approaches focus on why systems are constructed, providing the motivation and rationale to justify software requirements. The notion of focusing on the why is not new; organizing requirements around goals is new" (18)

Goals oriented methods can be seen as a subset of RE methods, which propose techniques for defining the complete requirements for a software system starting from stakeholders goals. However, most other RE and enterprise architecture methods give goals a very important place.

Goals are important in several respects of RE (59): they lead to identify the system components which support the requirements; they help to obliterate useless system components or justify the presence of components (that are not necessarily for the clients); they may be used to assign and evaluate the system components development of the system (according to the capabilities, reliability, cost, load, motivation of the related goal); finally, goals provide basic information for detecting and resolving conflicts that arise from multiple viewpoints stakeholders (143).

In particular the motivation that brings to goals oriented methods for RE can be summarized as follow (158):

- Goals are more stable than the requirements that implement them. (18). The higher level the goal is the more stable it will be (158).
- Goal refinement techniques give traceability from organizational goals to IT systems requirements. Goal refinement tree provides traceability links from high-level strategic objectives to lowlevel technical requirements. In particular, for business application systems, goals may be used to relate the software-to-be to organizational and business contexts.
- Goals enable to verify that the requirements are complete. If the requirements can be proved to satisfy all the stakeholders goals, then the requirements are complete.
- Goals enable requirements engineers to define which requirement is irrelevant and which is not, avoiding irrelevant requirements.
- Goals enable requirements engineers to better explain requirements to stakeholders; to manage conflicting requirements and to consider alternative design decisions.
- Goals and scenarios are considered to be a basic driving force behind requirements.

- Goals are used to understand stakeholders issues and negotiating them, they are used to systemically search for intentional keywords relating to: enterprise policies, enterprise mission statements, enterprise goals, BP diagrams, interview and scenarios written with stakeholders.

Goal oriented methods are mainly built on the notion that goals can be arranged in a hierarchical order, from high-level to low level goals. Such method are mainly based on problem solving techniques developed in Artificial Intelligence (AI) (166). The main technique used in these methods is goal refinement. Goal refinement is a technique used to reduce a goal into sub-goals by means of AND/OR relations.

There is a number of definition of the goal concept in the RE literature each targeted for the method or approach based on this concept. In this thesis we will refer to the Mylopoulos and Yu perspective that consider a goal as a state to be achieved. A goal is a "condition or state of affairs in the world that the stakeholders would like to achieve." (38). They introduce also the concept of soft-goal that is very useful in early requirements modeling and to the hypothesis of our research work: "Soft-goals are goals that do not have a clear-cut criterion for their satisfaction. We will say that soft-goals are satisfied when there is sufficient positive and little negative evidence for this claim, and that they are unsatisfiable when there is sufficient negative evidence and little positive support for their satisfiability." (126).

Goal oriented RE methods are connected to BPR and sometimes to business strategic planning like the "organizational theory" of choice (102). Goals, in this vision, are implicitly and explicitly considered as the ultimate explanation of human behavior.

The most noticeable goal oriented methods in RE are the following:

- The Knowledge Acquisition in automated Specification (KAOS) (59; 159) method that consists of a formal framework based on temporal logic and AI refinement techniques where all terms such as goal and state are consistently and rigorously defined. The main emphasis of KAOS is on the formal proof that the requirements match the goals

that were defined for the envisioned system.

- The Non-Functional Requirements (NFR) approach (47; 125; 126) is based on the notion of soft-goals rather than (hard) goals. A soft-goal is satisfied rather than achieved. Goal satisfying is based on the notion that goals are never totally achieved or not achieved. We enclose in this category techniques and method like Tropos (79), i* (47) and the the Goal-oriented Requirements Language (GRL) (113).
- Goal-Based Requirements Analysis Method (GBRAM) (18) defines a top-down analysis method refining goals and attributing them to agents starting from inputs such as corporate mission statements, policy statements, interview transcripts etc.
- The ESPRIT CREWS approach (146) focuses more on goal definition and the linking of goals to stakeholders actual needs by linking goals and scenarios.
- Pohl and Haumer (85) and Kaindl (96) also define methods for relating goals with scenarios.

KAOS is a formal approach for analyzing goals and produce requirements based on pre-stated goals (59). KAOS is largely the product of van Lamsweerde at the Catholic University of Louvain, Belgium. The KAOS approach is mainly oriented toward insuring that high-level goals identified by stakeholders to concrete system requirements. The method is composed of:

- A specification language based on concepts such as object, action, agent, goal, constraint, etc. This language also used a so called real-time temporal logic to represent constraints on past and future states. The temporal primitives are, for example: in the next state; in the past state; always in the future; always in the past.
- An elaboration method for transforming stakeholders goals into requirements for the software system. This method includes classical questions such as how and why to refine and abstract goals in the

goal-reduction graph: the identification of pre, post and trigger conditions of goals, the identification of agents to which goals are to be ascribed, identification and resolution of conflicts etc.

- A meta-level knowledge base used for guiding decisions during the elaboration process. This meta-level knowledge base contains: a classification of goals; rules for insuring the consistency and completeness of requirements; rules for insuring the consistency and completeness of requirements; tactics and heuristics for driving the elaboration and selecting among alternative goals.

The most interesting aspect of KAOS is the classification of goals. KAOS classifies goals into: achieve, cease, maintain, avoid and optimize goals. Achieve and cease goals are said to generate behaviors. Maintain and avoid goals are said to restrict behaviors. Optimize goals are said to compare behaviors (59). KAOS uses domain knowledge that is considered as objective knowledge, to reduce goals into sub-goals. Also, KAOS does not encourage the challenging of goals given expressed by stakeholders with the exception of conflict resolution (159). Thus, KAOS provides tools for transforming stakeholders goals into requirements but without making sure that these are the right goals to base the requirements on.

The Goal Based Requirements Analysis Method (GBRAM) has been proposed by Anton in 1996 (18). It attempts to reach more accuracy in the identification of high-level goals that it was in KAOS and i*. GBRAM does not assume that high-level goals have been previously specified. It identify and abstract goals from all available sources of information. Thus, in GBRAM, the origin of goals is considered to be the available information sources, scenarios, etc. Stakeholders identification proceeds from the identification of goals. One of the main preoccupations in GBRAM is the need to create models that are understandable by stakeholders.

The GBRAM inquiry process follows the following activities(18):

- Extracting goals from natural language documents, interviews policy statements, etc.
- Identifying goals and stakeholders and matching stakeholders with goals;

- Organizing goals by considering their precedence relationships and classifying these goals into different types of goals, mainly into maintenance and achievement goals;
- Refining goals, eliminating redundancy and reconciling synonymous goals;
- Elaborating goals, uncovering hidden goals and requirements by identifying goal obstacles and scenarios;
- Operationalizing goals, transforming goals into a software requirements document by formalizing goals into goal schemas and identifying the actions necessary to support the goals.

The GRAMB source of goals is the document that helps to identify the goal. It is important to understand the stakeholders application domain and goals before concentrating on the actual or current system so that the system requirements may be adequately specified (18).

Anton recognize that customers tend to express their goals within the context of their application domain, not in terms of an existing or desired system. This is an important issue to consider in our research since it is connected directly with BPR issues and the way RE can be used in reengineering the organization (19; 174).

GBRAM presents some lacks in its approach to goal analysis. For instance goals are presumed to exist, goals derived from documentation artifacts are not questioned as to their validity. The goal is classified, reconciled, expanded etc. but is not fundamentally challenged. The fundamental question about the need of the identified goal is not asked. Moreover, goals are not considered in relation to the organization, no enterprise model is proposed and related to the stakeholder goals.

The main point in GBRAM is the process and heuristics that enable analysts to elicit goals from stakeholders and specify scenarios for their achievement. This is lacking in both KAOS and NFR. However, GBRAM is very focused on goals of the IT system and therefore it lacks to relate goals with the organization. The result is an IT system that automates

existing business processes without attempting to provide innovative solutions hence it does not bring to an effective BPR and business-oriented software development.

The i*, Tropos and GRL Projects

The i* modelling framework (172; 175) offers primitive concepts of (social) actors, goals and actor dependencies, which allow to model both software systems and organizational settings. The i* framework aims at modeling "strategic relationships" between actors that represent stakeholders and their goals. i* can be used to explore alternative business processes by showing how the actors depend on each other for the achievement of goals and to evaluate the merit of different alternative non perfect solutions for the satisfying of not clearly defined non-functional requirements.

In i*, actors have freedom of action, but operate within a network of social relationships. Specifically, they depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. These dependencies are intentional in that they are based on underlying concepts such as goal, ability, commitment, belief, etc. The actors are defined as strategic because they evaluate their social relationships in terms of opportunities that they offer, and vulnerabilities that they may bring. Strategic actors attempt to protect or fulfill their interests. i* provides a higher level of modelling rather conventional modelling techniques such as data flow diagramming, workflows and object-oriented analysis (e.g., UML). Such high level of modeling allow engineers to reason about opportunities and vulnerabilities of both the organization and the system.

The i* framework includes the *strategic dependency model* for describing the network of inter-dependencies among actors, as well as the *strategic rationale model* for describing and supporting the reasoning that each actor goes through concerning its relationships with other actors. The strategic dependency model provides an intentional description of a process in terms of a network of dependency relationships among actors (173). It aims at capturing the underlying motivations and intents behind the process that it is modeled. The strategic rationale model provides an intentional description of process in terms of process elements and ra-

tionales behind them. Thus, the strategic dependency model describes the external dependencies among actors while the strategic rationale model describes the intentional relationships from the internal perspective of actors. The strategic rationale model explicits the means-ends relationships that relate process elements, providing an explicit representations of "why" and "how" alternatives.

Tropos (39) is a software development method that includes *i** as its requirements framework. Thus, Tropos is not limited to goal oriented requirements engineering but it is a full featured software development method founded on the notions of actor, goal and actor dependency which are used as a foundation to model requirements and design the system (38; 79). More specifically, Tropos is an agent-oriented methodology founded on the organizational theory and the *i** framework. It extends the *i** notation with *actor*, *goal*, *task/plan*, *soft-goal*, *resource*, and *dependency* as basic modelling constructs.

A number of organizational styles and social patterns were proposed (102) to guide the development of the organizational model for an information system. Organizational styles describe the overall structure of the organizational context of the system or its architecture, while social patterns focus on the social structures necessary to achieve one particular goal.

The Tropos project consists in 3 mains efforts: the Tropos methodology that is a software development method; formal Tropos that is a formal language aimed to support the Tropos Methodology and some social and intentional models used in the development process (78; 152)

The Tropos method covers four phases of software development:

1. Early requirements: where the relevant stakeholders are identified along with their goals, this phase is concerned with the understanding of a problem by studying its organizational setting;
2. Late requirements: introduce system-to-be as another actor who can accommodate some of these goals, in this phase the system-to-be is described within its operational environment, along with relevant functions and qualities;

3. Architectural design: more system actors are added and are assigned responsibilities, in this phase the system's global architecture is defined in terms of subsystems, interconnected through data, control, and other dependencies;
4. Detailed design: completes the specification of system actors and the behavior of each component of the system architecture.

A key feature of Tropos is the usage of *i** concepts through the entire software development process. The approach is requirements driven i.e., concepts used in requirements modeling are used/mapped also in design phases. This feature allows to define a continuity from early requirements analysis to the detailed design and system implementation. In this way conceptual models representing the system are obtained with an incremental refinement and extension of a model of the environment.

The principal concept of Tropos is the Actor(Uni02). Each phase of the Tropos methodology treats the Actor/Agent concept and its related notions (e.g., goals, task, plans) to design a specific aspect of software.

The Actor/Agent concept used in Tropos is a generalization of the agent notion used in Agent Oriented Software Engineering (AOSE) (167) and AI (166). Software development is tackled as a Multi Agent System (MAS) planning where the delegation of goals to other actors and intentionality of actors dependencies are the basis for the requirements design.

In Table 4 the main concepts used in Tropos are presented.

Tropos models software using a semi-formal visual language, this language has been specified by: an ontology; a meta-model; a graphical notation and a set of usage rules. In Figure 12 we show a part of the Tropos metamodel relating to the actor concept taken from (152).

The method uses a development process that is based on 5 diagrams capturing static and dynamic aspects of the system:

1. Actor Diagram: It is a static representation of the system that points out actors, goals and actor dependencies. It is the fundamental product of the early requirements phase where the environment that will surround the system-to-be is described but it is used in all phases of the Tropos method. Starting from a an arbitrary level

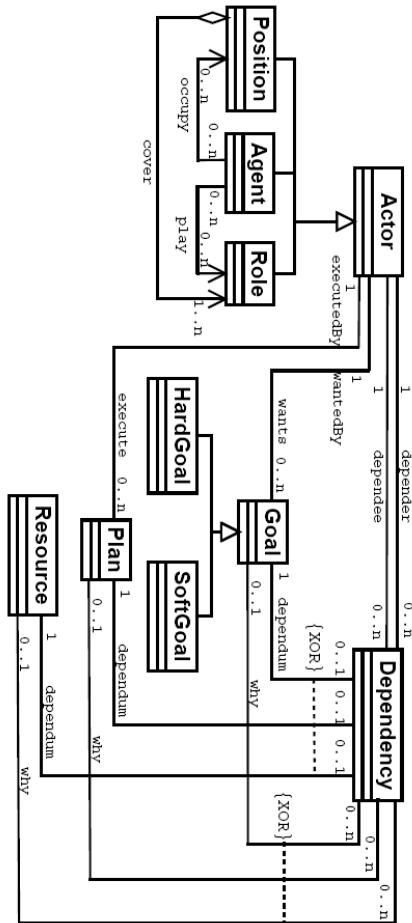


Figure 12: The Tropos metamodel of the actor concept and the dependency relation specified by means of UML.

of abstraction (but focusing on the identification of stakeholders and their dependencies) the analyst analyzes the domain to identify main actors, actor dependencies and actor goals. This diagram is refined and incrementally increased especially in late requirements.

2. Goal Diagram: It allows the means-end and AND/OR analysis of goals and dependencies. It is also a static diagram. It is based on the Actor Diagram and allows the goal decomposition, to find alternative solutions for goals and how different solutions contribute positively or negatively to fulfill goals. It is used mainly in early and late requirements where it is used to analyze respectively social actor and system actors.
3. Plan Diagram: It is an AUML (28) activity diagram that analyzes internal states of a plan inside a capability. It is realized in the detailed design phases.
4. Capability Diagram: It is an AUML activity diagram used to model a capability from the point-of-view of an actor. It is realized in the detailed design phases.
5. Agent Interaction Diagram: It is an AUML sequence diagram which model the asynchronous agent interaction. It is realized in the detailed design phases.

Requirements diagrams notation is inspired by i* and KAOS while design diagrams notation uses Agent UML techniques. We are mainly interested in requirements phases and thus in Actor and Goals diagrams because of it strictly relates to BPM. Differently from prescriptive (late) requirements which capture the "What" and "How" of the system to be, early requirements captures "Why" the system is developed.

Goals are desired by actors and are delegated to other actors for fulfillment. Early requirements involve identifying stakeholders (social actors) and their goals. Late requirements phase exploits goal diagrams to describe and support the reasoning about actor relationships. In this phase the system-to-be comes in the model as one or more actors that satisfy

stakeholders needs and it is given additional responsibilities. Further, such a system actor is decomposed into several sub-actors which take on some of these responsibilities. The Late Requirements analysis results in the definition of all functional and non-functional requirements (i.e., resource, task and soft-goal dependencies) and the identification of how the system goals can actually be fulfilled exploiting the other actors.

Goal-oriented Requirements Language (GRL) is an international standardization effort of i*. There is an abundant literature on this family of methods that comprise also some ideas of the Tropos approach. GRL is part of a standard draft of United Nation International Telecommunication Union (ITU) (ITU09a) that is called User Requirements Notation (URN) (ITU09b). There are two parts to this standard: URN-FR refers to the Functional Requirements part and URN-NFR refers to the Non Functional Requirements. GRL is the language proposed to describe URN-NFR.

GRL has evolved from the basic idea that there are two kinds of requirements Functional Requirements (FR) and Non Functional Requirements (NFR). FR are understood as those requirements that describe what the system should do for its stakeholders. NFR are understood as whatever is non FR, i.e., issues such as, customer satisfaction, increase of market share, availability, security, adaptability etc. (96; 125; 126).

GRL supports the analysis of strategies, which help reach the most appropriate trade-offs among (often conflicting) goals of stakeholders. A strategy consists of a set of intentional elements that are given initial satisfaction values. These satisfaction values capture contextual or future situations as well as choices among alternative means of reaching various goals. These values are then propagated to the other intentional elements through their links, enabling a global assessment of the strategy being studied as well as the global satisfaction of the actors involved. A good strategy provides rationale and documentation for decisions leading to requirements, providing better context for standards/system developers and implementers while avoiding unnecessary re-evaluations of worse alternative strategies. GRL also provides support for reasoning about

scenarios by establishing correspondences between intentional GRL elements and non-intentional elements referring to scenario models of URN-FR. Modelling both goals and scenarios is complementary and may aid in identifying further goals and additional scenarios (and scenario steps) important to stakeholders, thus contributing to the completeness and accuracy of requirements.

The basic idea of GRL is that these issues can be captured with the concept of soft-goal. While goals are defined as: "a condition or state of affairs in the world that the stakeholders would like to achieve" (ITU09b). A soft-goal is defined as a goal for which: "A soft-goal is a condition or state of affairs in the world that the actor would like to achieve, but unlike in the concept of (hard) goal, there are no clear-cut criteria for whether the condition is achieved, and it is up to subjective judgment and interpretation of the developer to judge whether a particular state of affairs in fact achieves sufficiently the stated soft-goal" (ITU09b).

The main modeling elements in GRL are: Actor, Goal, soft-goal, task, resource, and belief. A task specifies a particular way of doing something. Goals, soft-goals, tasks, and beliefs are called intentional elements in GRL. GRL defines a number of relationships between the modeling elements including: *Means-end links* reflect how goals are achieved; *Decomposition links* show what are the component of a task; *Contribution links* show how one intentional element influences the achievement of another intentional element; *Dependency links* establish a relationship between two actors. They contain what are called dependums. A dependum can contain a goal, soft-goal, task or resource.

In GRL, as in KAOS, goals are considered to have been predefined. The distinction between goals, soft-goals, and tasks is also considered as non problematic. However, i* makes fewer assumptions about the need for an IT system than methods such as KAOS and GBRAM (99), it is thus useful in an earlier phase when the debate centers about how to resolve some problem rather than what the goals of the IT system should be.

The purpose of GRL is to model strategic relationships and analyze their needs but it does not offer tools for representing the constraints imposed by the context of an enterprise and the different interpretations that

stakeholders may have of these constraints. However, it provides both an abstract and concrete grammar for its classes that can be exploited in tools and method that will adopt such a standard.

Si* and Secure Tropos

Tropos is currently conceived as a "Toolset" such as a collection of methodologies, frameworks, and models tackling different aspects of the software development process (Uni02) (38). The Tropos toolset adopts the i* modeling language (extended with concepts for security and dependability) for requirements specification

The Tropos Toolset consists of 5 main efforts (117):

- Tropos: the main software engineering methodology which can be used from the early requirements phase down to the implementation (see Section 2.2.4);
- Secure Tropos: a security requirements engineering methodology;
- Goal-Risk Tropos: a risk analysis framework;
- P-Tropos: a framework for the automatic selection and evaluation of design alternatives;
- Formal Tropos: a formal language aimed to support the Tropos Methodology;

Each framework in the toolset uses a fragment of the i* modeling language on the basis of its purpose: Tropos uses i*; Secure Tropos uses Si*; GR (Goal-Risk)Tropos uses GRI*; P-Tropos uses PI* and Formal Tropos uses a subset of i*.

In order to use Tropos and i* in our framework we consider to use Si*, and hence the Secure Tropos methodology, as a basis for our business-oriented approach.

Si* (81) is an extension of i* notation that consider the notion of service and security. The basic idea of Secure Tropos is to distinguish the trust relationships among goals and actors.

Secure Tropos is tailored to model and analyze security and privacy aspect of systems and their organizational setting from the early phases of the software development process. Beside adopting the SI* modeling language for requirements acquisition, Secure Tropos exploits formal analysis techniques for security requirements verification and validation.

Si* is used to introduce security requirements analysis in the early phases of the software development process. This allows us to elicit security requirements from the organizational environment, to analyze security requirements within the organizational environment in which the software will operate and to motivate the use of specific security mechanisms.

Thus Si* add the possibility to delegate and manage further concepts such as the trust and distrust notion, ownership, permission, etc. that are fundamental notions in the organizational context(80).

Thus, Si* introduces some relations and changes in order to define the security degree of each element of the model for modeling and analysis of functional and security requirements. Three new relationships are introduced as follow:

- Ownership (among an actor and a goal/resource) represents the fact that an actor is the legitimate owner of a goal/resource;
- Trust (among two actors and a goal/resource), marks a social relationship that indicates the belief of one actor that another actor will not misuse the goal/resource he has been granted. A trust relationship can be: a delegation of trust permission (Tp) when the trustor believes that the trustee will not misuse the goal/resource and a delegation of trust execution (Te) when the trustor believes that the trustee will achieve/deliver the goal/resource.
- Delegation (among two actors and a goal/resource), refines the delegation relationship of Tropos. It marks a formal passage of permission. A delegation relationship can be: a delegation of permission (Dp) when the delegatee has the permission to fulfill/use the goal/resource (but it does not need to) and a delegation of execution when the delegatee should fulfill the service.

Actually, the Trust relationship is the mental counterpart of delegation. The Delegation relationship is used to model formal passage of responsibility, it is an action due to a decision. The Trust relationship is a mental state driving the decision. While in case of Delegation the delegator becomes vulnerable, because it has no warranty that the delegatee will achieve/not misuse the goal/resource, in case of Trust the delegator (in this case named trustor) expects the delegatee (named trustee) achieve/not misuse the goal/resource.

Security requirements are social requirements, we need to capture the key social requirements for security and model at the same time functional requirements and security requirements. Indeed, Secure Tropos involves two different levels of analysis: Social level analysis where the structure of organizations are defined associating to every role objectives and responsibilities and Individual level analysis where agents are not only defined with their objectives and responsibilities, but also they are associated to roles they can play.

Secure Tropos involves modeling activities similar to Tropos, such as:

- Actor Modeling: where actors are modeled with their objectives, entitlements, and capabilities;
- Functional Requirements Model: where delegation of execution relationships are defined;
- Trust Model: where trust of execution and permission relationships are defined;
- Trust Management Implementation: where delegation of permission relationships are defined;
- Goal modeling: where to elicit actor social relationships

Secure Tropos defines also an extension of the formal model of Tropos in order to allow automated analysis of trust and delegation relationships (80).

In the context of our work, we base our requirements engineering process on the Si* language in order to give a more refined delegation model

and to add the trust model. Thus, we use Si* instead of i* because it allows to model and analyze security and privacy aspects along with the organizational setting, in the early phases of the software development process. In a business oriented approach, it is very useful to take into consideration social relations concerning trust, permission and execution delegation because they help to identify actors and goals considering security and privacy constraints. However, only a small part of the Secure Tropos methodology and the Si* language have been exploited in our framework. The need of maintain our requirements framework lightweight and easy to be understood by the stakeholders bring us to adopt only the trust and enriched delegation model of Secure Tropos. Thus, we use the Trust and Delegation model of Secure Tropos and define the Trust Model and Trust Management Implementation Model leaving out more complex -though powerful- mechanisms such as the Si* formal model.

2.3 Modeling Enterprise-centric Computing

In this section we give an overview of MDE and the support provided to this technology by the Eclipse Integrated Development Environment (IDE) (The08a). Eclipse is a platform for building integrated application development tooling. The basic Eclipse environment does not provide a great deal of end user functionality by itself. The platform encourages the rapid development of integrated features based on a plug-in model. Eclipse provides a common user interface (UI) model for working with tools. It is designed to run on multiple Operating Systems (OS's) while providing robust integration with each underlying OS. Plug-ins can program to the Eclipse portable API's and run unchanged on any of the supported operating systems. At the core of Eclipse is an architecture for dynamic discovery, loading, and running of plug-ins. The Eclipse platform defines an open architecture so that each plug-in development team can focus on their area of expertise. It uses the model of a common workbench to integrate the tools from the end user's point of view. Developed tools can plug into the workbench using well defined hooks called extension points.

The MDE approach is normally implemented as a set of modeling standards and languages. In particular, the MDA specification defines a set of OMG modeling standards to represent and transform models. Specific implementations of the MDE approach, such as the MDA, contemplate the ability to apply different technologies in order to implement model driven standards. Thus, the Eclipse IDE modular architecture is a fitted choice to manage a changeable and not fully developed technology like MDE.

In this section we will present an overview of MDE and MDA in the context of enterprise computing. Then, we describe how the Eclipse IDE currently support MDE by integrating a set of specific plug-ins.

2.3.1 MDE and MDA in Enterprise Computing

The history of software development is a history of raising the level of abstraction. These abstractions arise in all the most meaningful fields of computer science and are not completely independents in their evolution. Abstractions have been defined in programming languages (e.g., assembler, C, C++, Visual C); in architectures (e.g., client-server, n-tier, SOA's); in operative systems (e.g., Virtual Machines, Middlewares, Grid systems, Internet operating systems); in data representation (e.g., file systems, databases, XML files) and many other research fields. Despite the advantages of such abstractions they define new software platforms that are more and more complex to manage and program. This complexity is worsened by recents trends of software platforms that try to overcome modern global enterprises organization models. Modern company structures require high flexibility both in time and in space. They take part in opportunistic joint-venture, they rapidly change their organization to accommodate market changes or new strategies, they do not have centralized structures (77). The exposed scenario has been referred as the Enterprise-centric Computing (73) (also named Enterprise Computing). Beside this scenario there are two cross platform tendencies in Enterprise Computing to take into account, such as: a wide array of end-user systems and clients (e.g., fat clients, Web clients, mobile phones, iPods, etc.)

and the ubiquity of information (e.g., mobile and opportunistic networking, sensor networks, radio frequency identification (RFID) applications, etc.).

Platform volatility in Enterprise Computing can be simply resumed saying that "the only thing we can predict with confidence about the future of platforms is that things we can't predict will happen" (73).

In this thesis we underline how the MDE approach can be applied to face up platform volatility and new challenges of software development in the context of Enterprise Computing. The basic principle of MDE is that "Everything is a Model" (33). This principle, very similar to the object oriented principle where "Everything is an Object" (101), has two interesting properties: the *representedBy* and *conformsTo* relationships among models. The word model comes from the Latin *modulus* through the Italian *modello*, a model consist of sets of elements that describe some physical, abstract, or hypothetical reality. There are plenty of practical usages of models: statistical model, meteorological model, biological models, ecological models, economical models, etc. Computer science may be mainly described as the science of building software models. A model is not intended to capture all the aspects of a system, but mainly to abstract out only some of these characteristics. Instead a system is usually represented by a set of different models, each one capturing some specific aspects. A graphical map is a typical example of this principle. Different maps of the territory should be used if one wish to ride a bike or visit a museum but the represented system (the territory) is the same. It is clear in the map example the meaning of the *representedBy* relation among the system and the models (i.e., the maps). A truly expressive example of this concept is given by Bezvin (33) by means of the Magritte painting "This is not a pipe". As represented in Figure 13 Magritte insisting on the fact that the painting of a pipe may be useful for several purposes, but certainly not to smoke tobacco. Similarly we may have a painting of the painting, and so on.

The second important relation among models is the *conformsTo* relation. Lets think to the graphical maps legends, they indicate how to

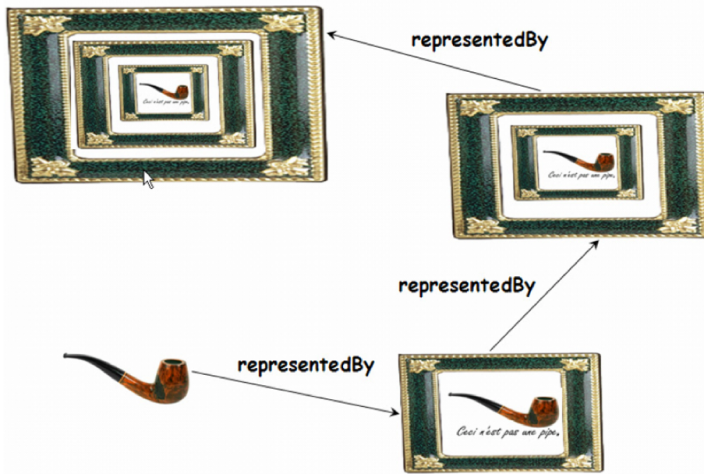


Figure 13: The paint "The Treachery of Images", René Magritte, 1929.

interpret the maps themselves. In this case we say that the maps conform to their legends, such as the maps are written in the graphical language defined by their legends. The conformsTo principle generalizes this concept saying that a model conform to its metamodel. A metamodel can be simply defined as a model of a modelling language. It defines the structure, semantics, and constraints for a family of models (i.e., a group models that share common syntax and semantics). The relation between a model and its metamodel is also related to the relation between a program and the programming language in which it is written, defined by its grammar, or between an XML document and the defining XML schema or DTD (73; 119). Coming back to the Magritte example in Figure 13 it is useful to remark that representedBy define a chain of models of models and not metamodels (i.e. models of modelling languages).

MDA may be defined as the realization of the MDE principles around a set of Object Management Group (OMG) standards like MOF, XML, OCL, UML, etc. MDA uses such standards-based modeling languages as formal development languages in order:

- to allow the definition of models and metamodels
- to allow model transformations
- to define system in a way that is independent from:
 - platforms
 - platforms evolution
 - application domains

The basic usage of metamodels in MDA is to facilitate the separation of concerns. We can characterize each perspective of the system we are modeling by means of different metamodels. The OMG define the so named "4-layer metamodeling architecture" to define metamodeling as represented in Figure 14. A model or terminal model (M1) is a representation of a part of the real system (M0) such as, for instance, a representation of a pipe. This M1 model conform to its metamodel defined at the M2 level. The M2 metamodel is the representation of a set of modeling construct (that can be used in M1) that conform itself to a further meta-model named meta-metamodel at level M3. Finally, the meta-metamodel conforms to itself.

OMG furnish different standards to instantiate the 4-layer metamodeling architecture. For instance the classical usage of UML is one of these. In this case we use UML metamodel conforming to MOF as the modeling language (M2); a UML model conforming to the UML metamodel to represent a class of object (M1). MDA fits the MDE postulate to have a unique meta-metamodel by means of MOF specification (33). This is an essential postulate to comply with, since the number of domain specific metamodels is rapidly growing with the danger of a fragmentation in MDE. Without a unique, or a small set of, meta-metamodeling languages, comparisons, transformations, merges and other operations among models and metamodels would be impossible to be realized.

The MOF specification defines both an abstract language and a framework to specify, develop, manage and exchange metamodels. The framework allows to implement metamodeling repositories that store a persistent representation of metamodels. However, what concretely helps in

defining models in MDA is UML. UML in this sense has a double role in MDA: 1) Its specification is an instantiation of the 4-layer metamodelling architecture; 2) it is the modeling language used to define all the models used in all the 4-layer metamodelling architecture. Despite using UML to model class diagrams we can use UML constructs also to represent metamodelling and meta-metamodelling concepts. A MOF meta-model normally consist in a formal definition of the abstract syntax of the model concepts conforming to the MOF meta-metamodel plus an informal description of the metamodel semantics usually defined in natural language. The abstract syntax can be expressed in a graphical notation or not. Usually, to express MOF metamodels it is used UML as concrete syntax graphical notation. Other possible concrete syntaxes for MOF can be used such as the XML Metadata Interchange (XMI) (OMG07b) that is a XML specification to metamodel interchange and transformation. A MOF metamodel described in the abstract syntax or by means the UML concrete syntax can be "serialized" to a XMI model, such as can be rendered in a textual structured Semantic Web file in order to be exchanged among modeling tools or managed by MDA transformation engines. Thus the unicity of MOF as a meta-metamodeling language can be exploited by means of concrete syntaxes in order to define bridges between the MDA technical space and other spaces like XML documents (trough XMI) or the Java technical space (trough the JMI (Sun02) specification that defines a correspondence among MOF and Java classes) or adding other specifications.

The MDA has demonstrated the realism of the MDE approach. MDA can be used in different context from metaprogramming to dynamic architecture management (27; 33; 119). For the sake of this thesis we are interested in the classical way MDA is used in software platforms. In the traditional MDA approach, the objective is to be able to generate Platform Specific Models (PSM's) from Platform Independent Models (PIM's). A PIM is a model of a system that does not consider technical detail of a specific platform. On the contrary a PSM is a model that represents details about a particular platform. Both PIM and PSM conform to a metamod-els that define the level of abstraction we are dealing with. Automatic or

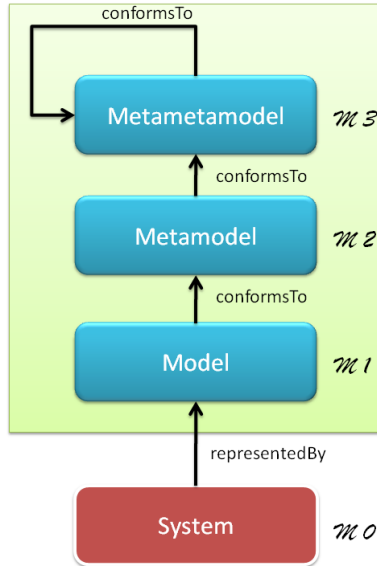


Figure 14: The OMG 4-layer Metamodeling Architecture.

semi-automatic generation imply the existence of a transformation language for MDA. In order to unify these languages or at least a family of such languages, OMG defines a request for proposal for transformation domain specific languages named MOF Query/View/Transformation- (MOF/Q/V/T) (OMG02b). In MOF/Q/V/T transformation generates a target model M_b from a source model M_a as described in Figure 15. A model conforms to a metamodel. The source and target models conform to metamodels M_{Ma} and M_{Mb} . Similarly the transformation $M_t: M_a \rightarrow M_b$ (i.e. the transformation program itself) conforms to a metamodel M_{Mt} defining the common model transformation language. MOF Q/V/T apply the MDE principle since the transformation M_t itself is a model. The PIM to PSM transformations are used to map in a semi-automatic way a PIM to different PSM's and thus to give independence both from platforms technologies and from platform evolution of the systems described in PIM's. However, other transformations are possible such as: PIM to

PIM transformations used to refine models without concern with platform specific details; PSM to PSM transformations used to deploy components in the context of a specific platform and PSM to PIM transformations used to abstract PIM from existent models (e.g., extraction of models from legacy systems). Platform independence is however a relative concept (73). It means independence from some specific execution and development domains, thus it has meaning only with respect to some specified platform or platforms. When we use the term platform independent or PIM; we have to specify the domains from which independence is being asserted.

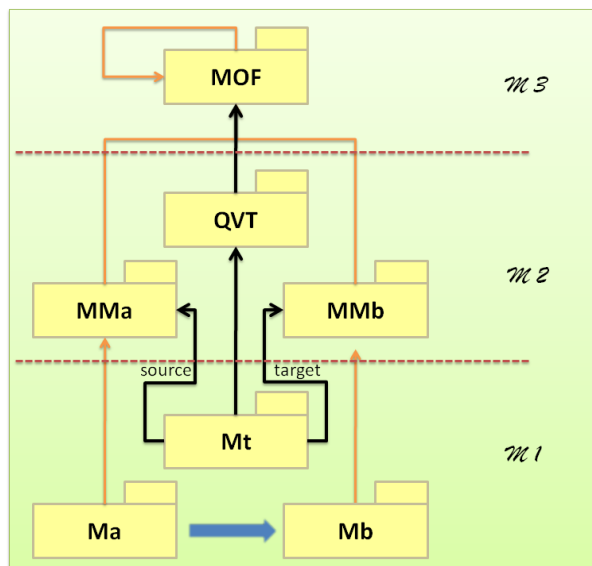


Figure 15: Model to model transformation.

Nowadays, the MDA technology is still in development, and some of the technologies need to be developed further and standardized, while others need further definition. One important difference between the old modeling practices and modern MDE is that the new vision is not

to use models only as simple documentation but as formal input/output for computer-based tools implementing precise operations. As a consequence model-engineering frameworks have progressively evolved toward solid proposals like the OMG MDA. We may clearly see the three levels of principles (general MDE principles), of standards (e.g. the OMG-/MDA set of standards), and of tools (like the Eclipse Modeling Framework (EMF) (The08b) or the Microsoft Visual Studio system (Mic08)).

It is useful to remark that MDA is not aimed to substitute other technologies or abstractions. The problem is not that we use different abstractions to describe different aspects of software components. The problem is that we have no overall architecture for integrating specifications made in different languages. MDA defines a single architecture for the integrated management of such metadata, it expands on existing technologies and abstractions in order to help them working better.

MDA is not a completely new approach, it has notable ancestors, such as database schemes, generative programming, CASE tools and many other model oriented approach to software development(50; 54; 73). Nowadays this technology is applied to the Enterprise Computing scenario using a set of standard languages in a generalized way. It attempts to move from solution space abstractions used in programming languages to problem space abstractions to be used in Enterprise Computing.

2.3.2 MDA Tools

Beside general principles and specification levels of model-engineering frameworks, represented respectively by MDE and MDA, the tools level is what concretely allows to put in practice the MDE proposal. Currently, there are not tools that fully implement MDA as envisioned by the OMG (35). Existing tools are usually limited to a single aspect such as: a specific platforms; a specific languages or some specific transformations. Usually, such existing MDA tools are evolutions of tools born in other research fields especially in generative programming (54) and Metaprogramming (27).

At the top level, a rough classification of the MDA tools is based the on

their model transformation approaches (55; 138). We can distinguish between model-to-model and model-to-code transformation support. The two approaches are significantly different since, usually, tools that manage model-to-code transformations are evolutions of generative programming tools while tools that support model-to-model transformations have been more likely developed by the MDE/MDA community. Anyway, model-to-code transformations can be considered as special cases of model-to-model transformations; we only need to provide a metamodel for the target programming language since the source code is an executable model of a system described in terms of a programming language (55). However, from a practical point of view, model-to-code tools are significantly different by mode-to-model tools. In order to use existing programming language compilers, model-to-code tools simply generate code as text, which is then fed into a compiler. For this reason, model-to-code transformation would be better described as model-to-text since non-code artifacts such as XML may be generated. Some tools offer both model-to-model and model-to-code transformations (e.g., OptimalJ (Com08)).

In the model-to-code category, we can distinguish between:

- Visitor-based approaches: consists in a visitor mechanism to navigate the internal representation of a model and write code to a text stream.
- Template-based approaches: template usually consists of the target text containing splices of metacode to access information from the source. Metacode performs an iterative code selection on the source model. The structure of the template is very similar to the code that will be created, and thus more intuitive for transformation developers. Templates combines untyped string patterns with executable logic for code selection and iterative expansion. The logic accessing the source model may have different forms, it could be simple Java API provided by the internal representation of the source model (e.g., JMI (Mic02)) or it could be declarative queries (e.g. OCL (OMG03b) or XPath (W3C99a)).

The majority of currently available MDA tools support template-based

model-to-code generation. Differently, model-to-model transformations translate between source and target models, allowing OMG Q/V/T PIM to PSM mappings. Usually, this type of MDA tools allow also PIM to PIM and sometime PSM to PIM mappings. All of these approaches support syntactic typing of variables and patterns.

In the model-to-model category, we distinguish among:

- Direct-manipulation approaches: that offer internal model representations for models plus some API to manipulate them. They usually provides a minimal infrastructure to organize the transformations (e.g., an abstract class for transformations). Users have to implement transformation rules mostly from scratch.
- Relational approaches: are based on mathematical relations. They state the type of the source and the target element of a model and specify it by using constraints. Such declarative constraints can be given executable semantics by means of predicates of logic programming.
- Graph-transformation approaches: are based on theoretical work on graph transformations. These approaches operate on typed, attributed, labeled graphs. Graph transformation rules consist on graph patterns. The graph patterns can be rendered in the concrete syntax of their respective source or target language (e.g., in VIATRA) or in the MOF abstract syntax.
- Structure-driven approaches: distinguish two phases, the first phase is concerned with creating the hierarchical structure of the target model, the latter phase sets the attributes and references in the target. It is the framework that determines rules the scheduling and application strategy; users are only concerned with providing the transformation rules. The OptimalJ (Com08) model-to-model transformations use a structure-driven approach.
- Hybrid approaches: combine different techniques from the previous categories.

There is another approach that could be used to model-to-model transformations that MDA tool do not implement for efficiency reasons. Since models can be serialized as XML using XMI transformations could be implemented by means of XSLT (W3C99b), which is a standard technology for transforming XML documents. Unfortunately, this approach has severe scalability limitations. XMI and XSLT suffer of maintainability problems due to their verbosity. Model transformations in XSLT quickly leads to non maintainable implementations. Even defining more declarative XSLT rule descriptions, this approach suffers from poor efficiency because of the copying required by the pass-by-value semantics of XSLT and the poor compactness of XMI (55).

As mentioned before there are not tools that fully implement MDA as envisioned by the OMG (i.e., with a CIM, PIM and PSM). Currently, many products that are sold as MDA tools actually are not (35; 103). In particular, tool developed in the context of generative programming and code generation tools developed prior to the MDA specification are not specifically MDA tools, although they may be applicable to an MDA process. Also model-to-model tools usually implement only partial functionalities of MDA or do not conform to MOF or the Q/V/T request for proposal. Basing on these considerations, we can characterize MDA tools basing on their limitations in: 1) tools form models representation and management; 2) tools for models transformation; 3) tools for models-to-code generation.

In order to give an overview of MDA tools currently present in the market, in Table 5 we describe tools that are succeeding both in the model-to-code and in model-to-model context.

2.3.3 How Eclipse supports MDA

Eclipse is a general purpose IDE aimed to be an universal tooling platform (i.e., it provides an open platform for setting up application development tools). Eclipse is an open, extensible architecture based on plugins. All the tool presented in Table 5 can be employed in the Eclipse IDE

in order to define a MDA-based development environment.

As we discuss in the previous section, none of the presented tools support all the features defined in the MDA specification (Obj01). This is because two reasons: 1) MDA is composed by a set of standard not completely defined yet; 2) the different levels in MDE (such as MDE principles, MDA specifications and tools implementations) allow specific implementations of the MDE approach, such as the MDA, to apply different technologies in order to implement model driven tools. For these reasons, the Eclipse IDE plug-in architecture is a fitted choice to manage a changeable and not fully developed technology like MDE and thus MDA.

A plausible strategy to define a framework for MDA in Eclipse is to combine different plug-ins in order to obtain a MDA IDE targeted for specific needs. It also helps in managing the evolution of the specific technologies whose impact are limited to single plug-ins. This is the approach we follow in our researches (4) and the same chosen by the Eclipse Foundation with the Eclipse Modeling Tools package (The08c) that is a pre-bundled version of Eclipse comprising most relevant and free open source plug-ins for modeling and metamodeling.

In this section we present two interesting technologies that are emerging as de-facto standard in Eclipse and are supported by great part of current MDA Eclipse tools. The Eclipse Modeling Framework (EMF) is the analogous of MOF in the Eclipse IDE Platform, it consists of an extended set of API, a metamodeling framework named Ecore and an Eclipse plug-in. The Atlas Transformation Language (ATL) is a language specifically targeted for model transformations, it supports MOF 1.4 and Ecore while conforming to MDA Q/V/T transformation request for proposal.

EMF is a framework and a code generation facility. It relies on a meta-metamodel named Ecore and allows to manage models represented in code and graphs. In particular, EMF is aimed to unify Java, XML and UML in order to bridge the gap between modelers and Java programmers. Models can be defined using a UML modeling tool, an XML Schema, or by specifying simple annotations on Java interfaces. EMF is a framework and code generation facility that allows to define a model in any of these forms (that is, Java interfaces, UML diagram, or XML

Schema) and then automatically generate the others and also the corresponding implementation classes. For instance, if we want to build an application to manipulate some specific XML message structure, EMF let us to automatically get a UML class diagram starting from the XML schema. At the same way starting from the XML schema or the UML class diagram EMF allows to generate a set of Java implementation classes for manipulating the XML. EMF is truly integrated with and tuned for efficient programming, the claim is that "in EMF, modeling and programming can be considered the same thing" (42).

The meta-metamodel used to represent models in EMF is called Ecore. MOF and Ecore have many similarities in the management of classes. Nevertheless Ecore is comparable to MOF it is not correct to define it a MOF implementation. Ecore and more generally EMF are more focused on tool integration, rather than metamodel management. However, the EMF experience is substantially influencing the MOF specification evolution thus we can foreseen a future alignment of these two approaches on meta-metamodeling.

The initial Ecore model can be created both in UML, XML and Java. In order yo create the model in UML it can be used any UML modeling tool that support Ecore such as Omondo (Omo07) or IBM Rational Rose (IBM08); to create the model in XML it can be used an Ecore XMI file that is a standard XML serialization of the exact metadata that EMF uses; finally to create a model from Java it can be used some tags to mark classes interface such as `@model` . Once we have the model, EMF allows to automatically generate code from it. For each "class" defined in the model EMF generates an interface and an implementation class. The implementation class defines a set of API providing accessor methods to allow to get and set values for each attribute and reference that belong to the class together with reflective methods to manage structural features. Two further classes named factory and package are created. Factory includes a `create` method for each class in the model that can be used to create objects; the package class provides static constants and a set of method to access model metadata. EMF can also generate other code such as a skeleton for adapter class factories and a plug-in manifest file to use the

model as a plug-in.

EMF is not limited to the base generator we described. There is also the EMF.Edit extension that generates adapter classes to enable model viewing and editing. Such an extension can even generate a working editor for the model. The EMF.Edit framework code is divided into two parts: 1) the EMF.Edit is a plug-in containing the user interface independent editing support classes; 2) the EMF.Editor is a plug-in containing user interface dependent editing support classes. The EMF.Edit generator generates a complete plug-in containing the UI independent portion of a model editor. The EMF.Editor defines a very functional multipage editor provided with wizards for creating new model instance documents; action bars, toolbars; menu bars items and icons. Finally, EMF provide also interoperability features such as notifier classes that send notifications whenever an attribute or a reference change; a generic XMI serialized not limited to Ecore models and the support for dynamic generation of model instances.

Starting from a model, that is actually a metamodel, EMF.Edit produces:

- A set of typed item provider classes, one for each class in the core model;
- An item provider adapter factory class that creates the generated item providers for the package. It extends from the model-generated adapter factory base class;
- A plug-in class that includes methods for locating the plug-in's resource strings and icons;
- A plug-in manifest file, `plugin.xml`, specifying the required dependencies;
- A property file, `plugin.properties`, containing the externalized strings needed by the generated classes and the framework;
- A directory of icons, one for each model class.

The generated editor is a very functional multipage editor. The following is generated in the editor plug-in (EMF.Editor):

- An integrated Eclipse workbench editor;
- A wizard for creating new model instance documents;
- An action bar contributor that manages the popup menus, and toolbar and menu bar items;
- A plug-in class that includes methods for locating the plug-in's resource strings and icons;
- A plug-in manifest file, `plugin.xml`, that specifies the required dependencies and extensions of the editor, wizard, and action workbench extension points;
- A property file, `plugin.properties`, containing the externalized strings needed by the generated classes and the framework;
- A directory containing icons for the editor and model wizard;

Thus, EMF is not just a generator tool, it is also a powerful runtime framework to model management. Like MOF in MDA, EMF is considered only the starting point in the MDE support given by the Eclipse IDE. Almost all tools exposed in Table 5 are based on EMF and Ecore models.

The EMF.Edit generate a minimal graphical plug-in that helps defining models basing on Ecore metamodel used in the generation process. The editor generated by EMF is based on the tree representation of the model that could be uncomfortable for people experienced with UML tools. Another framework named Generative Modeling Framework (GMF) (The05a) is focused on feature-rich and extensible graphical model editor. It defines a generative infrastructure to graphical editor production bridging EMF and the Graphical Editing Framework (GEF). GEF allows Eclipse developers to create generic rich graphical editors providing a layout and rendering toolkit for displaying graphs, manage palettes, handle and resize graphical objects. As for EMF, the GMF framework (The05a) do not implement a complete plug-in yet. It creates the skeleton framework with

minimal editing capabilities, more advanced editing and visualization components need to be instrumented in code.

The second de-facto standard MDA technology in Eclipse is ATL. ATL is a model transformation language specified both as a metamodel and as a textual concrete syntax. Relating on the taxonomy given in Section 2.3.2, ATL is a hybrid approach (55). A transformation rule in ATL may be fully declarative, hybrid, or fully imperative. Declarative style is the default choice for simple transformations, source model elements are navigated by means of a set of rules that create target model elements. However, ATL also provides imperative constructs in order to ease the specification of mappings that can hardly be expressed declaratively. An ATL transformation program is composed of rules that define how source model elements are matched and navigated to create and initialize the elements of the target models. ATL relies on a ATL Virtual Machine that allows transformations. A transformation from the ATL metamodel to the virtual machine code enables to "execute" ATL transformations. ATL mainly focus on the model to model transformations and comply with the OMG Q/V/T request for proposal. Its transformation engine currently provides support for both the MOF 1.4 (OMG02a) and the Ecore meta-metamodels.

The ATL model transformation language enables to specify how one (or more) target model can be produced from a set of source models. The transformation language is specified both as a metamodel and a textual concrete syntax. Model to model transformations can be specified by means of ATL modules. Besides modules, the ATL transformation language also enables developers to create model to primitive datatype programs (ATL queries) aimed to compute a primitive value, such as a string or an integer. ATL also allows to define ATL libraries that can be imported from the different types of ATL units, including libraries themselves. An ATL transformation file has the .atl extension. Modules are structured in three sections: 1) the headers that define some attributes relative to the transformation; the helpers that are similar to Java methods and the rules that define the way target models are generated from source mod-

els. Rules can be of two types: matching rules and called rules. Matching rules are used for ATL declarative specifications both to define for what kind of source elements target elements must be generated and to define the way the generated target elements have to be initialized. Called rules are used in ATL imperative programming, they are a particular type of header that can generate target model elements (we can also invoke called rules in imperative code sections).

The ATL language has been developed by the ATLAS project team of INRIA (ATL08). It has been implemented as an Eclipse plug-in in 2006 and in 2007, it has been recognized a standard component in the the Eclipse Modeling Tools package (The08c). In particular, ATL is the bases for other MDA tools and plug-in such as the AM3 and AMW described in Table 5 and general modeling tools such as TopCased (Top08) and Papyrus UML (Pap08).

Obviously, there are other emerging technologies in Eclipse concerning MDA, we present EMF and ATL because of their general acceptance and proved efficacy. We expect that further implementations of EMF, such as the GMF (The05a), and the integration of ATL in widely used modeling tools like TopCased will define in a few year a strong MDA framework that enable the Eclipse IDE to fully support the MDA/MDE approach.

Term	Definition
Process	a series of actions or operations conducting to an end; especially : a continuous operation or treatment especially in manufacture (Merriam-Webster on-line dictionary)
Business Process (BP)	a structured, measured set of activities designed to produce a specific output for a particular customer or market. (60)
Business Process Reengineering (BPR)	encompasses the envisioning of new work strategies, the actual process design activity, and the implementation of the change in all its complex technological, human, and organizational dimensions. (60)
Business Process Management/Modeling (BPM)	includes methods, techniques, and tools to support the design, enactment, management, and analysis of business strategies, goals, organizational, operational and implemented business processes. (156)

Table 1: A glossary for business processes related terms

Threats	Risk	Frequance	Mangitude
Poorly defined or inconsistent software requirements specifications	3	4 (62%)	3
Faulty effort estimates	3	4 (62%)	3
Diversity in process maturity and/or inconsistency in work practices between the partners	3	3 (52%)	3
Increased level of unstructured poorly-defined tasks	3	3 (45%)	3
Poor or disadvantageous distribution of software development activities between the customer and supplier(s)	3	3 (41%)	3

Table 2: TOP 5 threats faced by distributed projects identified by Smite (147)

Wiki	CMS
It requires less effort by the stakeholders in order. to learn it. It requires to understand only the "edit page" tab.	It has a backoffice that has to be understood before the pages editing.
All stakeholders are page editors	It has some editors (e.g., few stakeholders; only requirements engineers) and many readers.
There is no page structure. Everyone can modify the Web site structure.	Pages are structured. The Web site structure is managed by an administrator.
The revisions are kept for every editing.	The versioning is usually supported worse than a wiki.
The page is a primary conceptual unit for content management.	It usually distinguishes among pages, sections, contents items, news, notes etc.

Table 3: A comparison among wiki platforms and CMS platforms features in GSE

Name	Description
Actor	It is an intentional entity with strategic goals inside the system. It can be a person, an animal a machine as well as a software component. It has 3 specialization (agent, role, position)
Goal	It represents a strategic interest of an actor inside the system or organization. It has 2 specialization (hard- and soft-goal)
Plan	It represents a way (set of actions) to satisfy a goal. Used as synonymous of task (i.e., action+intention)
Resource	It represents a physical or informational entity that one actor wants and another can deliver
Dependency	It indicates that one actor depends on another one in order to obtain some goals, execute its plan or obtain a resource.
Capability	It represents the ability of an actor to define, choose and execute a plan to fulfill a goal in a given environment. It is activated by an event
Belief	It represents the actor's knowledge of the world

Table 4: The Tropos main concepts

Name	Description	Tool Type	Transformation
Acceleo	A MDA based code generator for various target middleware and languages: Java EE, C#, Python, PHP etc.	3	x
Atlas Megamodel Management (AM3)	It defines a management environment for MDE repositories based on the "megamodel" approach. A megamodel is a registry of model resources available in a given scope (a zone) In order to manage megamodels users may use metamodels from a library or invent their own ones for new kinds of artifacts.	1	x
Atlas Model Weaver (AMW)	A tool for representing correspondence between models by means of a model (named weaving model). Common weaving use cases are: data exchange, data integration, model merging, etc	1	x
Atlas Transformation Language (ATL)	A model transformation language that provides ways to produce a set of target models from a set of source models. It also provide a toolkit (the ATL IDE) with a number of tools aimed to ease the development of ATL transformations.	2	x
Epsilon	It can be used to manage models of different modeling technologies using a family of integrated model management languages. It can be used for model navigation, modification, transformation, validation and comparison. It also provides tools for defining and executing wizard and for code profiling and monitoring.	1	x
IBM Model	Is a set of tools that helps to make comparisons, check consistency, and implement transformations Transformation Framework between Eclipse Modeling Framework (EMF) models.	2	x
Kermeta	A metaprogramming environment based on an object oriented executable meta-modeling paradigm: a Domain Specific Language (DSL) optimized for metamodel engineering. It allows model and metamodel management, weaving and transformation.	2	x
open ArchitectureWare	It is suite of tools and components supporting in model driven software development. It is built upon a modular model-to-code generator framework implemented in Java. It supports arbitrary import (model) formats, meta models, and output (code) formats It is "a tool for building MDA tools"	1,3	x
AndroMDA	AndroMDA is an extensible generator framework that adheres to MDA. Models from UML tools are transformed into deployable components different platforms: J2EE, Spring, .NET. It provides patterns for Axis, Struts, JSE, Spring, Hibernate and other toolkits	3	x
QiQu	It transforms an UML-model into source-code (Java, C#, Cobol etc.). It relies on XMI and allows to build a domain-specific generator that transform models into code or anything else (XML, HTML, Scripts, Excel etc.)	3	x

Table 5: An overview on tools that currently support MDA

Chapter 3

My proposal: the Enterprise-Service- Implementation (ESI) Design Method

The reengineering of a company BP's is more than an attempt to automate or speed up the existing processes (83). Usually, the reengineering implies a deep rethink of the entire business: the job design, the organizational structure, the company strategies, the management systems, etc. These activities require a continuous collaboration among engineers and stakeholders. Such a collaboration goes beyond a simple stakeholder-producer relationship but is an effective collaborative work which should continue during the entire business lifecycle (9).

Nowadays, to developing a new software system for a company does not mean only to provide a new system but, in many cases, it means also to develop a new product or to define new services. Software engineering in these contexts cannot be carried out without taking into consideration a reengineering of the business. This need originate because modern companies businesses are intimately bounded with software and technologies. Thus, also the nature of the product to be developed is changed. The

services, such as the abstraction we identify to concretely connect business modeling and system development, are quite different from other commodities. The service development is more complex, information-centric and economic-related. In the service development context the quality of the involved BP's is an essential part of the service quality, that is the service quality is not only a simple consequence of the quality of its development process as for commodities and goods (46; 60). The service quality depends on many other factors and related services, for instance, the marketing services, the BP's that have to be used in order to exploit the service, possible cultural aspects influencing the service usage, etc.

However, the need to reengineering the business by means of software, in particular service-oriented software, arises also in more traditional manufacturing industry. Modern companies structures require high flexibility both in time and in space. They are rapidly changing organization due to market changes and usually they do not have centralized structures (77). In these contexts, an approach to business modeling and BPM based only on software engineering is not enough. The final goal of software engineering is the software development, while the final goal for business modeling is the reengineering of the business by means of the software (46).

In order to study and foresee the effect of the software in the business, we need BPR techniques that allow to consider the entire company domain and to relate it directly with the technologies. We must understand *why, what, who, when* and *where* a process can or should be reengineered (32) and the way these changes affect the business strategies, the business rules and the business goals as envisioned by Zachman in his framework (150).

The *how*, which is usually addressed by software engineering, is only a part of this more complex scenario. Software engineering alone does not address all the business reengineering requirements. Current software engineering processes miss to adequately treat the way the software system they are developing affects the business processes and more generally the organization.

The Standish Group (Sta00) have studied that about the 28% of soft-

ware projects fail and over the 50% of software project have problems because of the misunderstanding of the context where the software have to be used (such as the users, the management and their requirements).

There is the need for a unique method taking into consideration the engineering of the entire business as a whole. Technologies must be developed to drive the revamp of business processes and to refashion the whole business. It means to consider: the business issues, the human factors, markets and society paradoxes that are not commonly modeled in a software engineering process. We named this approach the business-oriented approach to software modeling.

This Chapter detail the methodological contribute of this thesis. We present the Enterprise-Service-Implementation (ESI) design method that is a way to employ the framework described in Chapter 4 and the tools described in Chapter 5 in order to develop software considering the business-oriented approach.

3.1 A Business-oriented Approach to Software Modeling

The studies about enterprise organizational processes have brought deep changes in the economy and in the society (60). It is important to point out how the evolution of the business management discipline has been strongly influenced by the new technologies. Indeed, as we discuss in Section 2.1, the new technologies are the main harbingers of the business innovation in the context of business process reengineering(83).

Despite the close relationship between the business process modeling approach in technology and economy, processes are considered differently and for different goals by the software engineers and the managers (OMG06).

The Business Processes (BP's) of modern networked enterprises (e.g., Amazon and IBM) have to consider the changes brought by the new technologies in their processes and organization(135). On the other hand, developers of new technologies and the software engineers, have to consider the importance of the overall enterprise business in the develop-

ment of new software systems (77). Besides, the emergence of new enterprise models, such as networked and service-oriented enterprises, requires open and interoperable technologies supporting their processes and changes. Summing up these issues, there is the need for a unique framework that allows to manage business modeling and reengineering in a technology-centric way. This unique framework could be the enabler of new business opportunities provided by the new technologies, which allow many new services. This scenario has been sometime referred as the Service Science (135) that is a discipline focusing on finding new ways to increase the productivity and innovation in services-related industries .

In order to propose such a framework that helps to address business-oriented software development challenges we develop:

- a method (presented in this chapter 3)
- a framework of technologies (presented in Chapter 4)
- a set of tools (presented in Chapter 5)

The ESI method guides the usage of the framework of technologies we employ and the tools we develop and allow us to define such a proposed unique framework for business-oriented software development.

In order to understand the contexts of our contributes, we have identified three point of view (see Figure 16) representing three main perspectives for Business Modeling and its goals:

1. Enterprise view: models concern with organization, strategies, business rules, business domains, internal and external BP's etc. It is the business modeling perspective commonly used by economic business analysts. It corresponds to the first two perspective of the Zachman ISA framework showed in Figure 1.
2. System view: models the requirements of the system by means of the business understanding. The models concern with organization, internal BP's, business entities, systems, architectures etc. It is the perspective of the software engineers. It corresponds to the logical and physical perspective of the ISA framework.

3. Execution view: defines an executable model for BP's and systems. It is the view concerning with the system implementation, execution and evaluation. It corresponds with the last two perspectives of the ISA framework.

Our work focuses in particular on understanding and clarify the relationship between the Enterprise view and the System view. The contribute presented in this thesis take into account such three perspective of business modeling we depict in Figure 16.

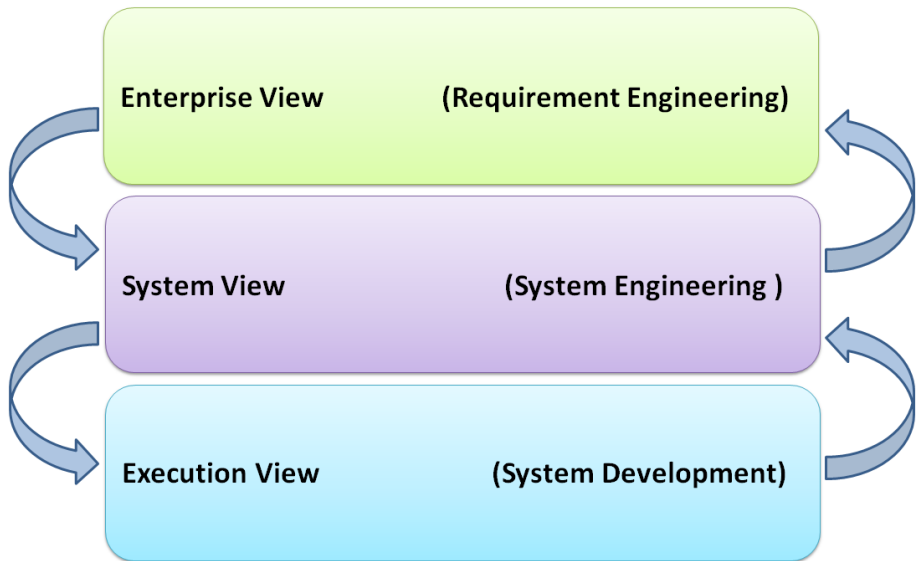


Figure 16: Our three views for Business Modeling.

In the context of this chapter, Figure 17 presents the three main phases of the ESI method and how they model the knowledge about the business and the system from various perspectives.

3.1.1 The Enterprise Modeling phase

The goals of the Enterprise Modeling phase is to wrap the current management with a technology support and to constantly share the knowledge about the enterprise among the economists, the managers, the users, the system engineers and other stakeholders.

The focus of the Enterprise Modeling phase is on collecting and maintaining all the knowledge about the company organization, the strategies, the goals, the risks and the management-related issues. In order to face these aspects, this phase is decomposed into two sub-phases: the *informal brainstorming* and the *enterprise knowledge formalization*. The former sub-phase is a collective learning activity (123) trying to define a shared knowledge-base about the company organization, the strategies and the goals.

Usually different stakeholders assign different meanings to the constructs of the organizational knowledge (e.g., actors, goals, strategies, organizational units) based on their mental models. Morecroft (123) recognizes that the collaboration in business modeling cannot be solely assessed in purely teleological terms (such as, the production of an acceptable model), but it has to be seen as a collective learning exercise that augments the organizational knowledge base of the organization (9). The model serves as a transitional object for mental models in a very similar way we present the Magritte's paint of Figure 13 in Chapter 2.3.1.

In such an informal-brainstorming sub-phase, the models are mainly used to structure the problems and the organization. In the latter sub-phase, such as the enterprise knowledge formalization sub-phase, the business process analysts and the managers try to formalize the knowledge in order to check its consistency and to discuss with the stakeholders. Thus, these sub-phases are cyclic Enterprise Modeling sub-phases performed many times taking into account the specific company, project or stakeholders needs. These sub-phases enable to move from a tacit knowledge of the company into a codified and consistent knowledge consisting of Si* diagrams and BP's describing the represented organization.

In the Enterprise Modeling phase the degree of competency and the

background of each participant may vary significantly. Thus, it does not make sense to propose a standard sets of steps and restrictive guidelines for business modeling. Besides, the language constraint provided by Si* (117), UML (OMG07a) and the Business Process Management Notation (BPMN) (OMG06) are enough to guide the enterprise knowledge formalization. This set of guidelines and languages is not restrictive because usually the stakeholders and managers are reluctant to spend time on brainstorming, process formalization, learning, training and becoming confident with formal specification languages and methods.

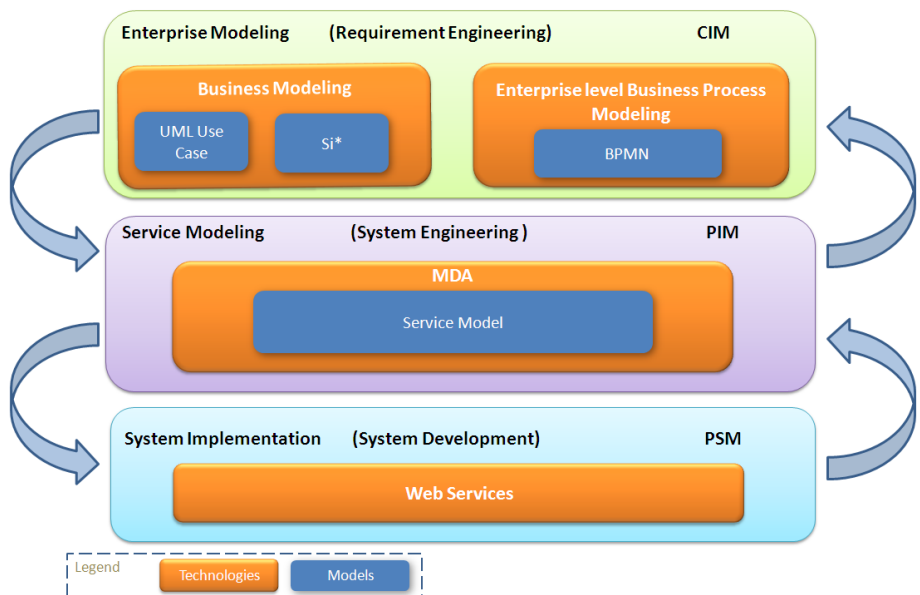


Figure 17: The three main phases of our ESI method.

The WikiReq tool is exploited in this phase in order to support the acquirement process. Internet and Web-based collaboration tools have a proven efficacy in the development of business modeling (9; 97). WikiReq allows to sketch the non-functional requirements described in Si* and connect this knowledge with a more formalized version defined in the

enterprise knowledge formalization sub-phase. WikiReq is a suitable instrument in order to perform the collective learning exercise in the organizational knowledge formalization recognized by Morecroft and other authors such as Adamides and Karacapilidis (9).

We distinguish between functional and non-functional specifications. The functional specifications are formalized using the UML Use Case diagrams as regards the static aspects of BP's, and BPMN as regards the dynamic interactions among the BP's concepts. The non-functional specifications closer to tacit knowledge are formalized by using Si* diagrams. The informal brainstorming sub-phase allows to identify an unorganized and not-formalized set of early-requirements giving a first sketch of the domain and the knowledge concerning the organization.

The modeling of goals and strategies (that are not caught by Use Cases and BPMN models) are essential for the design of the business models. Si* helps to model strategical and operational aspects of the business. By means of the Si* concepts, we are able to connect systems and actors to, for instance, business units, manager aims, practices and company policies.

The enterprise knowledge formalization sub-phase is the first phase trying to obtain a formal and analyzable representation of the business models. The cyclic informal brainstorming sub-phase helps the stakeholders in understanding their implicit knowledge and defining a shared knowledge base of the business and the BP's. Due to the general meaning of the Si* concepts, we start using them without a prescriptive formalism in the informal brainstorming sub-phase. Thus, starting from the beginning of our practice, the tacit knowledge is acquired in terms of the concepts that we will use in the Enterprise knowledge formalization sub-phase and in the Service Modeling phase.

The output of the Enterprise Modeling phase consists in three Si* models derived by the WikiReq tool in a semi-automatic way: (1) the Actor Model derived by the Actor Viewpoint page; (2) the Goal Model derived by the Goal Viewpoint page; (3) the Dependency Model derived by the Relational Viewpoint page. Moreover, it has been developed a model for each process and Use Case of the organization that has been identified during this phase. BP and Use Case models are described in a require-

ments document.

Thus at the end of the two sub-phases of the Enterprise Modeling phase it is expected to have:

- *An Actor Model*: that identifies the actors and their objectives, entitlements, and capabilities. The agents are also described in terms of the roles they play.
- *A Goal Model*: that models the goals from the point of view of an actor. The impact of the goals on the achievement of other goals is analyzed and modeled in order to refine the requirements models and to elicit new social relations among the actors.
- *A Delegation Model*: allows to model an actor delegating to other actors the achievement of the goals, the execution of tasks, the access to the resources and the ownership of trust. This model enables the transfer of rights among the actors and the assignments of responsibilities and trust among the actors. It is used mainly in the Enterprise Modeling phase in order to lead stakeholders choices in the definition of the goal and actor model.
- *A Set of BPMN models*: that details the steps and the states of the BP's workflows and their interactions by means of messages.
- *A requirements document with a set of Use-Cases*: the document is edited by all stakeholders in the WikiReq tools to clarify the main needs, goals and opportunities expected in business reengineering. Use Cases give a very high-level structuring of BP's and business actors interactions. They help to coherently connect processes and goals with concrete cases (in top-down analyses) and to maintain a general representation of the business (in bottom-up analyses). Also contents from the WikiReq argumentation system are used in order to identify general rules of the business and undefined constraint (e.g. budget, time limits, etc.).

The structure of the first three models is borrowed from the Tropos methodology (39). As showed in Figure 17, we categorize the Enterprise Modeling as a requirements engineering phase. What we perform

in Enterprise Modeling is a requirements engineering not only limited to the software system, but to the whole organization that must be reengineered, that is modeled like a system. We do not model only the part of the organization bounded to the system, but the ESI method requires to model the entire business of the company. In a business reengineering perspective, we force stakeholders to discuss and think about their organization and BP's in order to analyze together, first of all, the *Why* of the system. It should not be confused with a Project Management activity, such as a cost-benefit analysis. In project management, the *whys* of the system and its relations with a model of the organization are not analyzed systematically and involving all stakeholders in the debate. The Enterprise Modeling phase helps in thinking over the real need of a new system and also to foresee the changes that the system will bring in the organization and in the business (e.g., creation of new business units, development of new services to support the system product, opening of new markets, etc.).

Concerning the languages used in the Enterprise Modeling phase, the UML Use Case diagrams have been chosen because of their proved efficacy in stakeholder-analyst collaboration (66). Even if we do not introduce this formalism in Chapter 2, UML Use Cases diagrams use intuitive and quick to understand concepts (105) such as: *actor*, *use case*, *inclusion*, *extension* and *system boundary*.

BPMN (OMG06) is the OMG and Business Process Management Initiative (BPML.org) specification we introduce in Section 2.1.4. It synthesizes the best practices of the BPM community and defines a graphical notation for the BP's similar to a flow-chart. Such a notation is both consistent with UML and easy understandable by the stakeholders, analysts, business users, developers, etc. Moreover, BPMN is widely extensible and can be incrementally adopted.

Si* (79) exploits the i* set of primitive concepts in order to model a socio-technical system as described in Section 2.2.4. We choose Si* instead of the original i* because of the managing of the trust relation that is very useful in modeling the business. However, we do not exploits all the Si* capabilities, such as formal analysis techniques used for re-

quirements verification and validation. The usage of Si* is only limited to the definition of the trust relations in the Delegation Model, that is not exported in the other phases of the ESI method. Indeed the Delegation Model is used like the UML Use Cases to lead stakeholders choices in the definition of the Goal and Actor model. Thus, the use of Si* in the framework presented in this thesis is limited to an early requirements analysis, such as Si* helps us to correctly define Actor and Goal models in the informal brainstorming phase taking into consideration the trust relationships. However, we do not perform further considerations on security and formal checking of trust relationships either if Si* allows to perform powerful formal analysis.

In this thesis we refer only to the *actor*, *goal*, *soft-goal task*, and *resource* Si* concepts. Indeed there are other concepts and characterization of such concepts used both by i* and Si* (e.g., plans and roles as we see in Table 4) that we do not consider in our framework.

The meaning we give in our framework to the the Si* concepts is the same of Tropos and i* (79), it is composed by: *actor*, *goal*, *soft-goal*, *task*, and *resource* (79). An *actor* is an active entity having strategic goals and performing actions to achieve these goals. The actors can be modeled by using two types of sub-units: *agents* and *roles*. An agent is an actor with concrete, physical manifestations. A role is the abstract characterization of the behavior of a social actor within some specialized context. A *goal* is a strategic interest of an actor. A *soft-goal* is similar to the goal but the fulfillment condition is not clearly defined. A *task* specifies a sequence of actions that can be executed to achieve a goal. A *resource* represents a physical or an informational entity. The graphical notation of the concepts of Si* we exploit in the ESI method is reported in Figure 18.

Goals, tasks, and resources are often related among them in many ways. In particular, three relations have been identified, namely *AND/OR decomposition*, *means-end*, *delegation*, and *contribution* relations (79). The AND/OR decomposition combines AND and OR refinements of a root goal into sub-goals. Means-end and contribution relations analyze how goals are delegated among actors and how actors act so that their goals can be fulfilled. The contribution relation is used to expose goals con-

tributes to the soft-goals fulfillment. Since an actor may not have enough capabilities to achieve its objectives by themselves, they can delegate its execution to other actors as described in the delegation relation. Si* enriches the delegation relation defining a delegation of permission (Dp) when the delegatee has the permission to fulfill/use the goal/resource (but it does not need to) and a delegation of execution (De) when the delegatee should fulfill the service. Moreover, Si* allows us to define in the same model the trust relations, such as to model trust permissions (Tp) when the trustors believes that the trustees will not misuse the goal/resource and a delegation of trust executions (Te) when the trustors believes that the trustees will achieve/deliver the goal/resource.

From a MDA point of view, the framework of language we define (i.e., composing Si*, Use Case and BPMN), allows to define a Computational Independent Model (CIM) (Obj01; OMG01) that defines the business context and business requirements for the system that have to be realized. A CIM in MDA is a view of a system from the computation independent viewpoint. A CIM does not show details of the structure of systems. In other requirements engineering contexts (171) CIM is sometimes referred as a domain model. CIM's have to bridge the gap between those that are experts about the domain and its requirements on the one hand, and those that are experts of the design and construction of the artifacts that together satisfy the domain requirements, on the other hand such as the main goal of the ESI Enterprise Modeling phase.

We resume all the formalisms of our CIM language in a single images (see Figure 18). Such an image is also the image we use also to instruct stakeholders to the graphical semantics of the language employed in the ESI method.

The modeling of goals and strategies (that are not caught by Use Cases and BPMN models) is essential in the Enterprise Modeling phase for the knowledge acquisition about the organization. For this purpose, our approach uses the Si* notation. Si* helps to model strategical and operational aspects of the business. By means of the Si* concepts, we are able to connect formally represented systems and actors to, for instance, business units, manager aims, practices and company policies.

In particular, Si* is useful in business modeling because it exploits the same small set of concepts both in the organizational context and in the system context. In this way, Si* allows the system that must be developed, the system-to-be, to be represented as a further actor that can be inserted in the organizational scenario. Thus, after a first phase of early requirements elicitation, based on the description of the goals and the related BP's and Use Cases, the system-to-be appears as a new actor of the scenario together with its goals, tasks and resources. Means-end and contribution analyses allow to understand how the actor representing the system-to-be relates to the other actors and goals. In this way, it is possible to analyze how the business and its processes are affected by the system functionalities. For instance, we can represent how the system contributes in the fulfillment of a goal or the way a BP is affected by the system-to-be. Stakeholders can define their knowledge about BP's and business requirements inside the wiki by means of Si* concepts.

The modeling and analysis of goals and strategies (that are not represented by means of Use Cases and BPMN models) are essential in order to represent and study the business models. Si* helps to model strategical and operational aspects of the business. By means of Si*, we are able to connect systems and actors to: business units, manager goals, practices and company policies. Si* and the Tropos methodology indeed enable to model the organizational theory and the strategic alliance of the business (37; 102) (i.e., the functional and cross-functional aspects of the organization). At the same time, Si* is quite intricate when used to represent processes and dynamic evolving activities. Either if Si* allows such representation (136), in order to represent processes in a simple way rapidly understandable by stakeholders we introduce BPMN. Finally, modeling Use-Cases allows to further complements the other models giving a high level representation of system interaction and functional requirements. Thus, like the Si* Delegation Model, Use Cases are used to support the reasoning on Goals and Actors models and to enrich the descriptions in the Enterprise Modeling phase.

3.1.2 The Service Modeling phase

At the beginning of the Service Modeling phase, we use the outputs of the Enterprise Modeling phase in order to derive an analysis similar to the Tropos early-requirements analysis (102).

The Enterprise Modeling phase provides three inputs to the Service Modeling phase:

1. A formalized knowledge about the business and business processes represented in a set of diagrams (we named these diagrams the CIM's of the business).
2. Some indicative choices and purposes of managers to develop new systems and products.
3. A first analysis concerning the technologies that can be used to define new goals or improve the existing business (described in the requirements document)

In the Service Modeling phase, further technical choices are made in technical brainstorming. The requirements document and the Use Cases are analyzed to better define what type of technologies can be employed in the organization (i.e., also considering management issues such as times, deadlines, budgets etc.).

In particular, we consider the non-functional requirements defined by means of Si* inside the WikiReq tool. Besides, the goals and the soft-goals and their relationships with the system can be deeply analyzed by exploiting the Tropos early-requirements analysis process (79; 102).

The goals and the goal dependencies among actors are analyzed to understand how they relate with the systems actors. This is a fundamental step in our framework since it allows us to relate the business with the systems. In Si* and in Tropos the "system-to-be", that is the system (or the systems) that drive the reengineering of the business, are described as actors, such as they are represented as workers, offices and business units. Since Si* is a goal-oriented language, in this way we relate goals and tasks concerning the company with the system actor.

These models enable to define a rigorous representation of the enterprise knowledge. Besides, they enable to perform some analysis on such a knowledge by means of the Goal model. The analyses that can be carried out are:

- **Means-end analysis:** aimed to identify tasks, goals or resources that provide means for achieving a specific goal.
- **Goal/Resource refinement:** analyzes and decomposes the goals and/or resources in terms of AND/OR decompositions.
- **Contribution Analysis:** studies the impact of the tasks and the achievements of goals on the achievement of other goals.

After the redefinition of the models performed by System Engineers, in the Service Modeling phase a MDE transformation allows to derive a Service Model starting from the CIM set of models. Such a Service Model represents the platform independent vision of an implementable SOA to be used to develop the new software system.

Goals identified in the Enterprise Modeling phase and refined in this phase are related to BPMN activities and transformed into set of services. We give a detailed description of these MDA transformation processes in Chapter 4.

3.1.3 Platform Specific Implementations

The last phase of our practice is the System Implementation phase whose goal is to model the concrete executable support for the BP's and hence to define the business reengineering technologies.

The aim of this phase may change depending on the specific structure of the Enterprise Modeling and Service Modeling phases. The System Implementation phase is not necessarily performed because in the Service Modeling phase it is possible to decide that no changes have to be made in the current BP's and that no new systems or software have to be realized (such as, that a reengineering activity is not necessary in the organization at the moment).

The output of the Service Modeling phase is a service model. Such model can be used to obtain a detailed design of the system. Depending on the system nature (e.g., Web applications, centralized systems, Grid systems, agent-based systems, etc.), a specific system implementation abstraction is used in order to model the service.

We could decide to define a set of Web Services starting from the system-related services that appears in the service model. In this case the resulting SOA is very similar to the one defined in our previous works (3) described in Section 1.3. However, the MOTO-GAS service model defines a set of services without considering their mutual relationships and the way those services are related with business models.

In the System Implementation phase we can exploit also other architectures rather Web Services such as the the plug-in architectural style. Plug-ins allow to define a more complete SOA, starting from the service model defined in the Service Modeling, the plug-in abstraction can be used to define patterns for service orchestration. Actually, the plug-in implementation is only a work in progress, we give further details on this approach in Section 4.2.1.

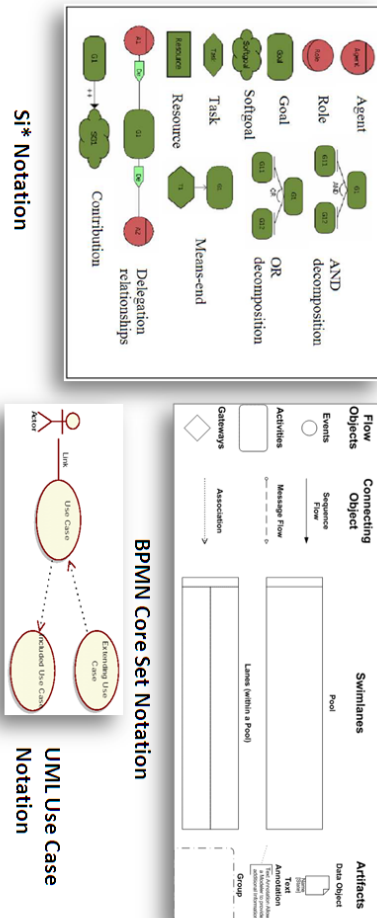


Figure 18: The Si*, BPMN and UseCase graphical syntaxes exploited by the ESI method.

Chapter 4

My Technological Framework

In this chapter we describe the framework of technologies we exploit to implement the ESI method. We present how the technologies that compose our framework work together in order to put into practice the three phases we report in Chapter 3. The framework has been named the BPR framework. It is shown in Figure 19. We describe the details of the two tools we develop, such as Wikireq and SMOTE, in Chapter 5. This chapter gives details on the approach used to perform the transformations from goals to services and from services to implementable platforms such as Web Services and Portlets. In particular, Section 4.2 exposes our research in progress regarding portlets and plug-ins architectures as a target for the ESI System Implementation phase. Such a subject will be the basis for our future works on the presented framework.

4.1 The Framework Transformations

4.1.1 The Goals 2 Services Transformation

The most important activities of the Service Modeling phase is the mapping from the business models concepts into the concepts useful in the

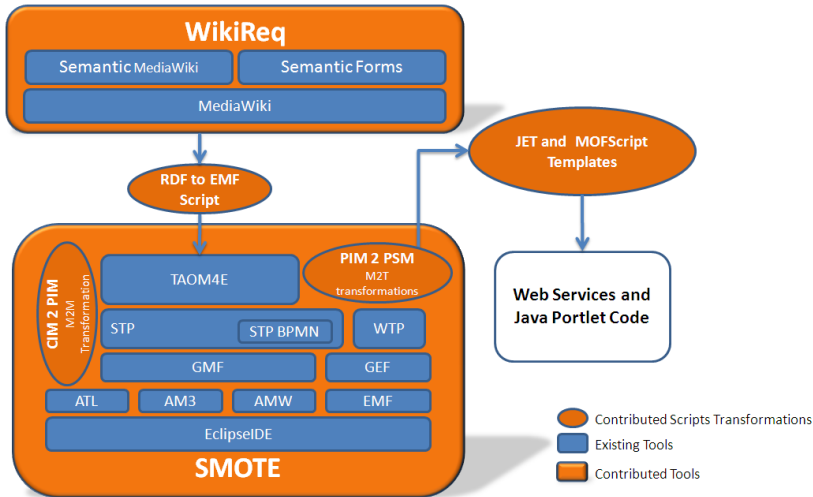


Figure 19: Our BPR framework architecture

Platform Specific Implementation phase. In our framework, these concepts are not mapped directly into programming languages concepts (e.g., objects, classes or functions), but we exploit the service concepts as intermediate abstractions enabling to move from business concepts, represented in the CIM, to implementation concepts, in the PSM. Starting from the Si* and BPMN models, a service model is defined.

We named the transformation the Goal to Service transformation since the goals represented in Si* are analyzed and transformed into a set of services. The transformation is performed taking into consideration both BPMN processes and Use Cases. The output service model represents a platform independent model for the system. In our SMOTE tool that implements the transformation presented in this section we implement it as a CIM2PIM Transformation.

The Goals2Service transformation is based on some considerations:

1. A service is needed in order to satisfy a goal: the nature of the service implies that there are two or more economic entities (See Sec-

tion 1.1.2) that operate in order to change the condition of a good or a person. This change of condition correspond to one or more goals that can be extracted from Si* diagrams and analyzed and connected to concrete services.

2. A goal is satisfied by an end event (or a specific intermediate event) of a process: thus BP's and goal diagrams have an explicit point of contact in end events that represents the end of the change of condition defined by the services interacting in order to fulfill the goal.
3. Goals are fulfilled by tasks, and processes are executed by tasks (named activities in BPMN): it means that goals and BP's have a further point of contact in tasks. Tasks are a unit of execution in processes and a way to fulfill goal we have to consider designing the services.
4. Actors perform processes in order to satisfy their goals: It is a syllogism of the previous point. Goals are fulfilled by tasks and set of tasks (such as processes), and actors want their goal to be satisfied, thus we can relate the actors and goals identified in Si* also by means of BPMN processes.
5. Tasks and resources are the enablers of a services: tasks can be considered the elementary steps useful to clarify the boundary of a services. At the same way, resources can be used to identify services but they gives a further information about the fact that services contain or not a persistent state.

The strategy of executing the transformation starts from goals as they are described in the goal model after the Tropos-based analyses (i.e., Means-End, Goal/Resource and Contribution analyses) performed in the Enterprise and System Modeling phases of the ESI method. For each goal a high-level service is defined (we have named this type of service the Business Service). If some tasks are necessary for the fulfillment of the goal related to the Business Service, the Business Service is replaced by a set

of service (one for each task linked to the considered goal). We have to remember the Business Service that originates these services so we can use a specific attribute or a meaningful name (such as `BusinessService-Name.ServiceName`). This transformation of task is performed only for those tasks that appear in a BP (other goal tasks are discharged). At this point we have a -big- set of services and Business Services. We identify those services and Business Services related to the system actor. For each Business Service related to the system actor a BPMN process have to be defined. So, if any, we have to perform a cyclic phase and return in the Enterprise Modeling phase in order to understand how that business service relate to the BP's (in this case relations have not been defined).

At this point in the service model will appear both services (some of them related to the system) and Business Services. For each Business Service we have to consider if it can be transformed into an implementable service, if it has to be obliterated or if there are other services that can replace it. If it is not the case, the goal related with the Business Services is suggested to be debated in a cyclic phase and return, if needed, in the Enterprise Modeling phase in order to understand if the Business Service is useful or it has been wrongly analyzed.

The final version of the service model do not contain Business Services. However, it contains two type of services: system-related and not-system-related services. All these services derive from a goal that is fulfilled by tasks that appear in at least one BP.

The concept of service is considered as a very general abstraction in the context of our framework and can be used to represent a wide range of interacting software components (155). The service model is represented in diagrams similar to the UML Class Diagrams (OMG07a). We propose this type of diagrams since they are very intuitive and have an easy to remember semantics (48).

Besides, the service abstraction is useful for a logical division of the software (134). Such a division is more coarse-grained than the division obtained by components or objects. The selection of the service abstraction represents a meeting point between the designer and developer re-

quirements in the translation from business models to services. The granularity of the services helps to define software components that can be changed and reused in a way that is understandable by the stakeholders. Moreover, the service notion itself (OAS06a) is not bounded to a specific computer-dependent abstraction (like for instance the object and component concepts). Thus, this help us to use services in a straightforward way inside MDA as a PIM, let zooming in and out models maintaining or discharging properties and details about the modeled system.

We named our framework showed in Figure 19 the BPR framework since the effect of the Goals2Services Transformation is to think about the organization and how its goals relate to the outcome of the organization. One of the main principles for reengineering identified by Hammer and Champy (84) and described in Section 2.1.1 is that the design of a the organization job should be arranged around an goal or a outcome, not around a task or a single task of a given process. Goals in the business context connect the customer and the producers, in a business and more generally in a organizational context, we have a goal because we have a customer need to satisfy (and make profits in this activity). For this reason, the goals that we transform into services have to be identified among what we named the Business Goals. The service model that outputs from this transformation is thus technologically neutral, it is implemented as a PIM in the SMOTE tool. Business Goals are what effectively relate the business and the system contexts; thus processes are only a way to perform the goals that rapidly change basing on the implementing technologies.

4.1.2 The Services 2 Web Services & Portlets Transformations

Basing on the PIM, such as the service model we developed in the Service Modeling phase by means of the Goals2Service transformation, we can produce a plug-in PSM and a Web Services PSM. Both the PSM's are created because a plug-in is intended a set of services wrapped in the plug-in content infrastructure.

These PIM2PSM's transformations are more related to the metamodels of the specific technology employed. We describe the technical details of the MDA transformation implemented in the SMOTE tools in Section 5.2.2. However, it is an evolution of our previous work on WSDL and WSRF mapping introduced in Section 1.3.

The transformation to Web Services is very similar to the one implemented in the MOTO-GAS tool (3). It has been simplified not considering different types of services (e.g., Grid service, Web Service and WS-Resource).

We analyze the service model manually, then decide what service have to be implemented by means of a software system and then we identify if there are already implemented Web Services that can be used to perform the service work. If there are not already existent services, we transform the service specification in the related WSDL code.

The transformation to plug-in architectures is more complex and is actually a work in progress. Basing on the information obtained by the BP models defined in the Enterprise Modeling phase, we group services that appears in the service model basing on BP's flows. For each service group we evaluate if the group is useful to support the considered BP's. If it is the case those groups of services will define a more general service named plug-in. Notice that is a normal situation that plug-ins share one or more services. Events, resources and flows information can be used at this point in order to define how the plug-in must relate with the other plug-ins while performing their tasks. Since the plug-in is composed also by non-system-related services, we can also extract useful information on human-computer interaction needs in the concrete development of Web Services and Plug-ins.

In other words, the plug-in abstraction helps in treating services and services compositions bridging the gap between business modeling represented by means of the business goals and business processes. In an analogous way, in the implementation level, the plug-in abstraction bridges the gaps between Business Services and the fine grained service composition defined in Web Service SOA.

4.2 Plug-in Architectures as Patterns for SOA Interaction

In the previous section we present how concretely the business knowledge acquired in the Enterprise Modeling phase can be translated to a service-based PIM.

We have implemented the Services2WebServices transformation that is a basis also for the Services2Portlets transformation that we will develop as a future work. In this section we give an overview of the research we carried out so far in this topic and the reason that lead us to expand our framework whit plug-ins architectures.

Starting from the system-related services defined in the service model of the Service Modeling phase, the Service 2 Web Service transformation allows us to define a set of Web Services obtaining a SOA model similar to the one defined in the MOTO-GAS tool (3; 5). However, this SOA define a set of services without considering their relations and the way such services interact together.

In the Web Service context, the Business Process Execution Language for Web Services (BPEL4WS) (BEA03) is a general accepted language for specifying business process behavior. BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that facilitates the expansion of automated process integration.

Since our framework defines a set of unrelated services, we could use BPEL4WS to define the interaction relation among those services. Besides, we could also exploit the BPMN to BPEL4WS mapping defined in the BPMN specification (OMG06). However, implementing effectively a SOA requires, besides service and composition description, also applications and run-time infrastructures that support the SOA principles. The Enterprise Service Bus (ESB) is emerging as a middleware infrastructure component that is needed in order to implement SOA's within enterprises (44).

ESB's generally provide an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. The need for an ESB in implementing SOA is fundamental in order to (100):

- Decouple the business view of a service from the actual implementation of the service
- Decouple technical aspects of service interactions
- Integrating and managing services in the enterprise

In this section we propose an approach to enhance our framework in order to support also service interaction by means of the Service2Portlets transformation. This approach differs from BPEL4WS and other workflow-based approaches. We propose to pay more attention on the standardization of ESB's rather to exploit general notations for service interaction like in BPEL4WS. This approach is based on two considerations:

1. The limits of the workflows-oriented approaches. BPEL4WS is suited only in process oriented applications (86), such as applications with an essential sense of the state and process that are intimately part of the application.
2. The emergence, in the Internet, of new wrappers for software and Web Services that overcome many interoperability issues and can be considered enablers for a convergence toward a standard model for ESB's.

4.2.1 SOA Infrastructures

The idea of assembling components into a network of services in building distributed applications is the essence of service-oriented computing (133). In particular, Web services have been broadly accepted in the industry; they are supported by products of many vendors and can thus be defined as the "incarnation" of the service oriented technology envision (111).

SOA's rely upon orchestration to assemble business processes from loosely-coupled services. The orchestration enables businesses to shift the focus from data centric to business centric applications with several advantages such as: improved flexibility; better alignment with business; cost reduction and increased manageability of the architecture (116).

The standard set of Web Service technologies (i.e., XML, SOAP, WSDL) provides the means to describe, locate and invoke a Web Service as a single and platform independent entity. The Web Service Description Language (WSDL) (W3C01) describes each atomic and low-level functions of the service. However, the basic technologies do not give SOA's enough behavioral details to describe the role the service plays as a part of a complex collaboration scenario. Orchestration technologies are aimed to describe services collaboration activities when they are designed to accomplish a specific business goal.

Currently, there are many languages and frameworks to support service orchestration. Early work in Web Services orchestration included CommerceNet eCo (Dum99) for e-commerce service integration; Web Services Conversation Language (WSCL) (W3C02b) to standardize language conversation among services; Microsoft XLANG (Tha01) to create business processes in the Microsoft BizTalk Server, and the IBM Web Services Flow Language (WSFL) to describe both public and private process flows. The Web services workflow specifications outlined by XLANG and WSFL have been superseded by BPEL4WS (BEA03). The BPEL4WS specification combines WSFL and XLANG specifications to model the behavior of Web services in a business process interaction. The specification provides an XML-based grammar for describing the control logic required to coordinate Web services participating in a process flow. This grammar can then be interpreted and executed by an orchestration engine, which is controlled by one of the participating parties. The engine coordinates the various activities in the process, and compensates the system when errors occur.

SOA applications collaborate by invoking each other services, and services are composed into larger sequences to implement business processes. From this perspective, SOA is an architectural approach to define

integration architectures that are based on services. In this context, a service can be defined as a discrete function that can be offered to an external user. The function can be an individual business function or a collection of functions that together form a process. Successfully implementing a SOA requires applications and run-time infrastructures that can support the SOA principles. Applications can be developed by creating service interfaces to existing or new functions that are hosted by the applications. The service interfaces should be accessed using an infrastructure that can route and manage service requests to the correct service provider.

ESB's generally provide an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code.

Despite the importance of ESB's components in SOA there is not in the current services architecture implementations a standardization effort similar to the one carried out for the basic set of Web services technologies (i.e., XML, SOAP, WSDL). Probably, this standardization effort will never occur. As for other middleware technologies, software companies like IBM, Microsoft, BEA and SUN Microsystem find more profitable to develop ESB framework bound to their architectures (e.g., programming languages, Web servers, development suites and other software) rather than searching for a high level interoperability among services. Moreover, ESB's software require a lot of training to be understood and usually they are expensive software suites. Thus many small companies prefer to adopt less sophisticated solutions to deploy and run Web services, like Apache Tomcat used together with JAX-PRC API and Java Web Service Development Pack (JWSDP) (Sun08) or to adopt none while recognizing the value of SOA's ¹.

The gaps between the business models and the service composition bring engineers to exploit SOA's principally for integration purposes (116), such as using Web services as an alternative technology for implementing distributed systems rather than to exploit their orchestration features.

These lacks in standardization must be took care in defining a pat-

¹Forrester Reseach. SOA Adoption in European and North American Enterprises. <http://www.forrester.com/Research/Document/Excerpt/0,7211,41686,00.htm>

tern oriented service composition design. Service oriented applications should run inside ESB's, much like Web applications run in Web Servers. In order to understand how to define patterns for service composition in ESB, we must agree on a simple and common model for the ESB's architectures.

A further underestimated drawback in SOA design is the business modeling vs. business process modeling misunderstanding (82), we have discussed this topic in Section 2.1. This misunderstood separation of concerns is the sake of many issues in mapping business requirements in business processes and hence in service composition structures. In particular, process models are not a good starting point for identifying business stakeholder requirements. They are too workflow-oriented to gather the generic concepts and relationships needed to model the business (82). Also analysis and decomposition techniques are soundly different: business modeling decomposition is aimed to analyze and discover goals and actors, while business processes study the way actors manage services and resources to fulfill goals. Finally, workflows-oriented approaches are a fitted choice only for process-oriented applications that are applications with the following characteristics (86):

- Long running: The process spans hours, days, weeks etc. from start to finish
- Persisted state: Because the process is long-lived the state have to be stored (e.g., in a database)
- Bursty: The process spends most of the time waiting for the next triggering event.
- Orchestrator: The process is responsible for managing and coordinating the communications of various systems or human actors.

Obviously, not all applications are process-oriented, and also process-oriented applications can exhibit different degree of process-orientation. Without criticism on the BPEL4WS and workflow-based approach to service interaction, the exposed scenario bring us to investigate a different approach to treat service orchestration.

4.2.2 Plug-ins as a Reference Architecture for ESB's

In order to obtain the goal of an effective modularization of software, interactions must be treated as first-class elements. We must be able to dynamically manage interactions upon components (17; 64), in other words a separation of *computation* from *coordination* is needed (75). This goal has been partially obtained in SOA's by means of loosely-coupling and just-in-time binding. These SOA features help to replace the identity-based model of the Object Oriented paradigm with the service-based model in module interactions (17). Thus, in a SOA a change in the service interconnection does not necessary imply code-level changes.

SOA's satisfy the first key requirement of the programming in-the-large envision (64) such as decoupling of interaction and implementation. However, there is a second requirement of the programming in-the-large vision that is not completely solved by the SOA abstractions. It is the need for a high-level orchestrator component (64; 70; 120) helping in coordinating and managing software components. Despite there are many efforts concerning languages and specifications to formally check and manage components orchestration, less attention has been paid to the standardization of ESB's and more generally to orchestrator engines. Thus, we focus on understanding how the emerging architectures and frameworks can support a better orchestration of software services.

An important drawback of SOA's is the lack of a standardized architecture for ESB components. There are interesting emerging SOA models to study in the Web 2.0 (61). In such models, the services are developed, deployed and provided not only relying on a service interconnection architecture based on standard protocols (OAS06a). In such models there is an high level orchestrator of services. This orchestrator can be considered as a lightweight ESB describing design time service compositions. We give some example of such architectures:

- *Portlets* are pluggable user interface components managed and displayed in Web portals. They are based on the OASIS Web Services for Remote Portlet (WSRP) standard protocol that allows the remote plug-and-play of remote portlet components and a set of interopera-

ability API's named Java Portlet Specification (JSR186).

- *Yahoo! Konfabulator* is the Widget Engine of Yahoo! Inc. A Widget is a small XML wrapped application that runs locally inside the konfabulator widget engine. Konfabulator is a Javascript runtime engine for Windows and MacOS X that allows to manage (download, set and run) widget components.
- *Wordpress* is one of the most popular content management system over the World ². It is used for Blogs and Web Site creation. Wordpress is based on plug-ins that are programs or functions in the PHP scripting language. A plug-in adds a specific set of features/services to the content management system. Plug-ins are hooked in the Wordpress core platform by means of a Javascript & PHP API's plus a set of standard triggered functions and header information.
- *GoogleGadgets* are small applications built by means of HTML, Javascript and Flash. Gadgets run inside some lightweight environment provided by Google Inc. such as: iGoogle (a personalized Ajax Web page); Google Desktop (a hosting application); Google Toolbar (a Web browser plug-in); Google Map (a georeferenced Web application).

These sample architectures are based on different SOA orchestrators: WebPage-based (e.g, iGoogle, Portlets, Wordpress); WebBrowser-based (e.g., Google Toolbar, YouOS) and ProprietaryEnvironment-based (e.g., Yahoo! Konfabulator, Java Virtual Machine, Macromedia Flash).

In our opinion, the described orchestrators are a specialization of a more general emerging model that is the plug-in Architectural style (21; 40). The applications and the development context in plug-in architectures relies on the assumptions going further a SOA defined as a service interconnection architecture based on standard protocols. The Plug-in model is significantly different from the SOA reference architecture (OAS06a) and the Web Service SOA implementation (W3C04a). It assumes the existence of an hosting environment which is able to host, to manage, to load

²Alexa - <http://www.blogswweek.com/en/most-visited-cms-site>

and to present ad-hoc wrapped services (either developed by means of Web Services, PHP code, RSS feeds, C programs etc.).

Our claim is that this emerging general model tries to overcome the second key requirement of a programming in-the-large envision, such as a reference architecture for an orchestrator of distributed components.

Nowadays many distributed applications are adopting the plug-in architecture model. At the same time, the effective openness of the underlying components, made possible by means of SOA's and Semantic Web standards, helps in opening further scenarios for this paradigm.

There are two basic features that differentiate plug-ins from SOA's architectures such as the Web Service Architecture (WSA)(W3C04a):

1. The existence of an explicit host that acts as an orchestrator of components. It means that it is possible to centralize policies, security, verification of consistency features in these hosts (sometimes considered as an upper-level service).
2. The presentation oriented aims of plug-ins. They go further the (standardized) business logic method presented by SOA services. Plug-ins enable a very quick plug-and-play of components without code rewriting needs.

The first feature allows to give a more consistent environment for SOA services in spite of what is possible in the current WSA. The second feature allows what is currently named "component mashup" (61) that is a concept very similar to service composition but at an higher architectural level.

Other benefits of plug-in architectures can be summarized as follows (21):

- Application features can be implemented and incorporated very quickly in the system
- Problems can be quickly isolated and solved at a business modeling level, because plug-ins are separated modules with well-defined interfaces
- Custom versions of an application can be created without source code modifications

- Third parties can develop additional features without any effort on the part of the original application developer
- Plug-in interfaces can be used to wrap legacy code written in different languages
- The application user interface can be managed and simplified by customers basing on their preferences
- Application customization can reduce memory footprint and improving performance

All these considerations on plug-in architectures are useful to define a framework for plug-in-based composition of software. Plug-ins are meant as special types of SOA services bound to a particular orchestration architecture (i.e., specific ESB's). The plug-in itself can be either a set of composed services (e.g., a set of Web Services or other plug-ins) imitating the recursive architecture of SOA's compositions, inside a specific environment.

The Plug-ins introduce a further level of abstraction in service composition. They can be defined, for instance, as patterns to host Web services, they can thus act themselves as recursive orchestrators of services. Since plug-ins themselves can be composed and they represent very high level functionalities for the system.

Chapter 5

My Tools Supporting the ESI Method

The framework we have defined by means of the ESI method and the transformations presented in the previous chapters, is supported by two tools: WikiReq and SMOTE. This chapter gives a detailed description of these tools, their architecture and their features.

5.1 Semantic Wikis for Requirements Engineering

In the context of a national project named TOCAL.IT¹ (TOC08), we have developed a wiki to support our BPR framework (5). The BPR framework we present in Chapter 4 formalizes the enterprise knowledge by means of a set of models connected with software requirements and also provides a partial automation of the system implementation. The ESI method drives the usage of WikiReq according to the BPR framework. In this context, a clear and consistent requirements elicitation is a fundamental task in order to analyze how the system functionalities relate and

¹TOCAI: Tecnologie Orientate alla Conoscenza per Aggregazione di Imprese in Internet (Knowledge-oriented technologies for enterprise aggregation in Internet)

affect BP's. Wikis are eligible platforms for complex requirements gathering. They easily allow collaboration in document redaction and both synchronous and asynchronous debates among a very large number of spatially distributed stakeholders (62). Our BPR framework provides a wiki named Wiki for Requirements (WikiReq) in order to collaborate in defining a first common knowledge base about the organization, its goals and its processes. WikiReq is based on the Semantic Mediawiki (SMW) platform (Wik07) and the Si* goal-oriented language (79). Such a wiki allows to acquire requirements by means of a set of pre-defined forms. It also enables to automatically define semantic relationships among the Si* main concepts. The SMW has been enhanced to support an argumentation debate about requirements (9) in a specific tab page. Finally, semantically annotated requirements are managed by a PHP script in order to be transformed in an Eclipse Modeling Framework (EMF) (The08b) instance to graphically represent Si* concepts in the Eclipse IDE (The08a).

WikiReq is based on three ideas: 1) using a semantic wiki for requirements elicitation and management; 2) exploiting the wiki platform to define an argumentation system for both synchronous and asynchronous discussions among stakeholders; 3) achieving interoperability between the semantic wiki and an Integrated Development Engineering (IDE) platform like Eclipse.

Requirements acquired by the Semantic Mediawiki can be exported in the Eclipse IDE in order to partially automate the rendering of organizational business processes and system artifacts from the requirements description. We present the WikiReq tool highlighting its relationships with model driven engineering and business process modeling in order to show how WikiReq can be used in the context of a full featured business reengineering framework.

5.1.1 The WikiReq tool

Despite the availability of a great number of notations, methods and tools for requirements engineering, Small and Medium-sized Enterprises (SME) are unfamiliar with these techniques (129). Most SME's use only a few

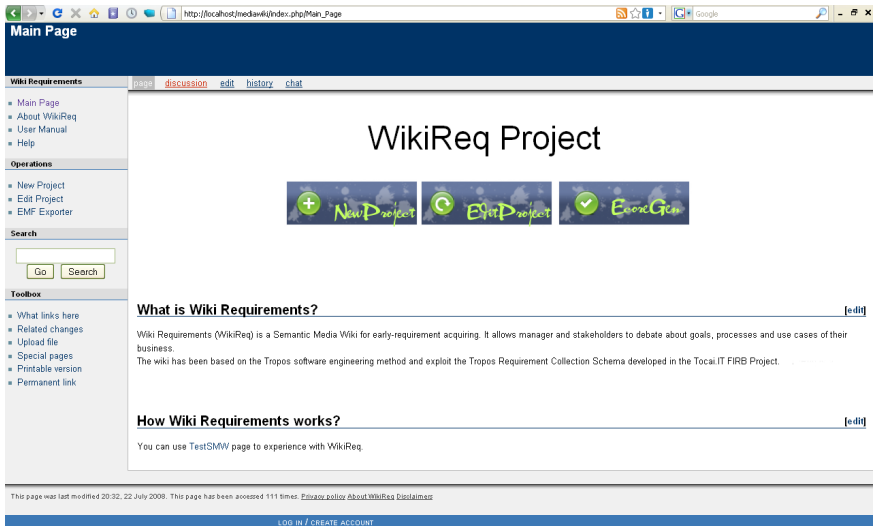


Figure 20: WikiReq tool: Main Page

well known set of tools for this activity such as office suites and general collaborative tools (62). An analogous problem is suffered in software houses where the effort to adopt a complex notation, such as the full UML specification, is considered to be "of insufficient value to justify the costs" (66). In this thesis we have defined a lightweight framework for Business-Oriented Software Engineering that allows business modelers, managers, engineers, users and other stakeholders (such as company employees and customers) to reduce the effort needed for a deep understanding of specific language syntaxes and semantics. In WikiReq we combine the benefits of the three notations used in ESI: Si*(79); UML Use Cases (OMG07a) and Business Process Management Notation (BPMN) (OMG06). Stakeholders can define their knowledge about BP's and system requirements inside WikiReq by means of Si* concepts. Such a knowledge is then exported in the Eclipse IDE to be detailed and connected to BPMN and Use Cases diagrams.

WikiReq has been developed in order to support the stakeholders in

Figure 21: WikiReq tool: the Actor semantic form

acquiring the requirements in terms of the Si* concepts. These requirements are acquired by means of a set of semantic schemes that helps to describe actors, goals, tasks and resources. The WikiReq system is based on the SMW. SMW allows to relate Resource Description Framework (RDF) (Wor08) semantic notation to the wiki content in order to better organize the search and browsing of the information. We actually exploit also a SMW extension, named Semantic Forms (Wik08) that allows to manage the semantics of the edited pages by means of simple Web forms.

We have developed three form templates for the WikiReq pages to be filled by all stakeholders:

- **Actor Viewpoint.** It is used to identify and detail the actors involved in the business and the actors that interact with the system. Also the system-to-be is represented as an actor (or a set of actors in case of systems that require complex interactions). Actors are described and commented by the wiki users. All the goals that an actor attempts to fulfill are selected by the existing goals or edited as red links (i.e., in a new page that must be created).
- **Goal Viewpoint.** It is used to identify and detail the goals in an

analogous way it is done for the agents. In addition to descriptions, comments and actors involved in the goal fulfillment the stakeholders have to specify whether the goal is a soft-goal and/or what sub-goals contribute to its fulfillment.

- **Relational Viewpoint.** It is used to define the delegation relationship. This page is used to model the transfer of duties to do something, from an actor who delegates a goal (the delegator) to another actor who receives the delegation (the delegatee) (79).

In Figure 21 and Figure 22 we show how these requirements scheme are filled by means of respectively: the semantic form, and the resulting Actor Viewpoint page. In Figure 21 a new actor named *Sales Manager* is added by means of the actor semantic form. Figure 22 presents the Actor Viewpoint page where the Sales Manager actor is listed together with the *Define offers* red link goal and the other information.

The Relational Viewpoint actually is used exclusively in the context of WikiReq and it is not imported in the Eclipse IDE. The delegation and trust relations are used mainly to elicit the actors (i.e., to recognize agents and roles) and to understand how goals have to be fulfilled and/or decomposed according to the security and trust scenario. On the contrary, the Goal Viewpoint is useful in refining the Si* models in Eclipse, since the sub-goals relationships with goals (or soft-goals) help to define the AND/OR decomposition and the contribution analysis. Thus, further details on the Si* goal-oriented analysis are performed and discussed in Eclipse by the technical managers, engineers and analysts. If doubts or incoherences appears, the analysts can come back to WikiReq and discuss the issues with the other stakeholders in order to modify the requirements.

We point out that the semantic expressiveness of SMW is an enabling feature to WikiReq. It is the basis for transforming and structuring requirements into EMF models. It is also fundamental in order to make easier the selection of semantically related concepts by means of the semantic autocompletion of fields supported by the Semantic Forms extension. The autocompletion allows an easy way to overcome one of the

Wiki Requirements									
page	discussion	edit	history	delete	move	protect	watch	refresh	chat
<div><div><div><div><div></div><div>■ Main Page</div></div><div><div></div><div>■ About WikiReq</div></div><div><div></div><div>■ User Manual</div></div><div><div></div><div>■ Help</div></div></div><div>Operations</div><div><div><div></div><div>■ New Project</div></div><div><div></div><div>■ Edit Project</div></div><div><div></div><div>■ EMF Exporter</div></div></div><div>Search</div><div><div><div>Go</div><div>Search</div></div><div></div></div><div>Toolbox</div><div><div><div></div><div>■ What links here</div></div><div><div></div><div>■ Related changes</div></div><div><div></div><div>■ Upload file</div></div><div><div></div><div>■ Special pages</div></div><div><div></div><div>■ Printable version</div></div><div><div></div><div>■ Permanent link</div></div></div></div></div>									
<div><div><div><div></div><div>Actors</div></div><div>Sales Manager</div></div></div>	<div><div><div><div></div><div>Is a role?</div></div><div>true</div></div></div>	<div><div><div><div></div><div>Goals (Objectives)</div></div><div>Define offers</div></div></div>	<div><div><div><div></div><div>Actor Description</div></div><div>A sales manager have to maintain accurate offers, in terms of content, price and margins. It manages all related to sales chain, for instance: if a product change/upgrade can be sold to existing customers, too</div></div></div>				<div><div><div><div></div><div>Comments</div></div><div>no</div></div></div>		
<div><div><div><div></div><div>CIO</div></div></div></div>	<div><div><div><div></div><div>true</div></div></div></div>	<div><div><div><div></div><div>Avoid additional systems costs</div></div><div>Simplify system development and deployment</div><div>Architectures integration</div></div></div>	<div><div><div><div></div><div>The Chief Information Officer (CIO) is a job title for the board level head of information technology within the company. The CIO reports to the chief executive officer. The prominence of this position has risen greatly as information technology has become a more important part of business. The CIO is a member of the executive board of the organization.</div></div></div></div>				<div><div><div><div></div><div>The executive board membership have to be discussed</div></div></div></div>		
<div><div><div><div></div><div>CEO</div></div></div></div>	<div><div><div><div></div><div>true</div></div></div></div>	<div><div><div><div></div><div>Comply to ISO 9000</div></div><div>Increase Productivity</div><div>Increase Product Success</div><div>Financial Improvement</div></div></div>	<div><div><div><div></div><div>It is the Chief Executive Officer (CEO) of the enterprise. It is the highest-ranking corporate officer or administrator in charge of total management of the corporation, company, organization, reporting to the board of directors.</div></div></div></div>						
<div><div><div><div></div><div>Buyer</div></div></div></div>	<div><div><div><div></div><div>false</div></div></div></div>	<div><div><div><div></div><div>Awareness on requirement changes</div></div><div>Easy access product information</div><div>Compare supplier offers</div></div></div>	<div><div><div><div></div><div>The buyer is the company customer.</div></div></div></div>				<div><div><div><div></div><div>no</div></div></div></div>		

Figure 22: WikiReq tool: the Actor viewpoint page

principal drawbacks of using a wiki for requirements engineering such as the user ability to remember the page names (62).

5.1.2 WikiReq argumentation feature

The second feature of WikiReq is the customization of the SMW talk page (also named discussion page) in order to support an argumentation approach to the knowledge acquiring. WikiReq allows the collaborative work among the stakeholders not only for software requirements elicitation but also in BP's understanding. It is a fundamental activity in order to clarify how the company works and how the system-to-be will change its processes (9). We use the Toulmin schema for expressing arguments (154). It consists in defining for each argument: a *Claim* (such as the conclusions whose merit must be established); the *Data* (the facts we appeal to as a foundation for the claim); a *Warrant* (the statement authorizing our movement from the data to the claim); a *Backing* (that must be introduced when the warrant itself is not convincing enough in order to certify the statement expressed in the warrant); the *Rebuttal* (that consists of statements recognizing the restrictions to which the claim may legitimately be applied); and, finally, the *Qualifier* (that consists of words or phrases expressing the speaker's degree of force or certainty concerning the claim).

The WikiReq arguments are partially connected to the goals and actors by the RDF semantic statements. The SMW Social Rewarding (Wik08) extension is used in order to define a metrics to evaluate the argument validity and the weight basing on the claiming stakeholder rewards. The WikiReq argumentation allows an asynchronous conversation by means of the argumentation system. Moreover, synchronous conversations among the stakeholders are provided by means of the Wiki Chat extension (Wik08) that allows a rapid interaction especially in clarifying the arguments. The chat contents are not volatile, so that other people can read the previous conversations about the page (i.e., the goals or the actor described by the page). Both synchronous and asynchronous conversations are a fundamental enhancement with respect to sporadic brainstorming where not

all stakeholders are present. They also represent a continuously updated record of the work done to elicit the requirements.

5.1.3 WikiReq to Eclipse export

The most important feature of WikiReq is the ability to export the acquired knowledge in the Eclipse IDE. We allow to use the Si* concepts without a prescriptive formalism. The elicitation of the requirements is based on filling a WikiReq form. Therefore, the stakeholders are not required to understand the graphical notations. The Si* requirements are then automatically transformed into graphs that can be used by the developers as a basic input in order to define a formalized version of these specifications. These can be done by means of a rigorous usage of Si* and other languages such as UML Use-Case (OMG07a) and BPMN (OMG06). Even though the UML Use Cases and BPMN are intended to be used by a small subset of the stakeholders to validate and verify the consistency with WikiReq requirements, we choose these two formalisms because of their simplicity. The Use-Case diagrams use intuitive and easy to understand concepts (such as: actor, use case, inclusion, extension, system boundary etc.) and are the most used diagrams for stakeholder-analyst interaction (66). BPMN is a notation to describe BP's easy to understand by all business stakeholders (OMG06).

In order to specify how WikiReq relates to the other components of this framework we show in Figure 19 its overall architecture (our contributes are in orange). The WikiReq output is managed by a set of Eclipse IDE plug-ins. The Tool for Agent Oriented Modeling (TAOM4E) (Cen08) plug-in allows to graphically represent the Si* model and to detail an AND/OR decomposition analysis for system goals. The remaining components allow to manage the Si*, UML Use Cases and BPMN models according to the MDA transformations inside the Eclipse IDE. The BPR framework manages Si*, UML Use Case and BPMN artifacts in Eclipse by means of a set of EMF models. By means of EMF, we are able to define an instance model conforming to a meta-model and more generally to manage knowledge by means of models and model transformations. We have

developed a PHP script for SMW named RDFtoEMF which allows to export the semantically annotated RDF representation of the requirements in a EMF model conforming to the Si* meta-model. The TAOM4E Eclipse plug-in is then used to represent the Si* model in a graphical format.

Thus, starting from the beginning of our requirements engineering process, the tacit knowledge about system requirements and the company BP's is acquired by the stakeholders in terms of the same general concepts that we will use also in system modeling and implementation. Combining the wiki functionalities with MDA modeling is a fundamental task to maintain coherence between system artifacts and requirements. The MDA functionalities allow to partially automate the knowledge management by means of model transformations. These functionalities enables also to define software system taking into consideration how the BP's relate to requirements as it is expected in BPR.

5.1.4 WikiReq: an Example Scenario

This section presents an example scenario developed in the context of the TOCALIT project ² in order to show how WikiReq works and the way the stakeholders can use it to elicit requirements. A stakeholder can add a new actor to the Actor Viewpoint page by clicking on the Add Actor link at the end of the page. WikiReq requires to insert the name of the new actor that will become the name of the semantically annotated page describing the actor. As shown in Figure 21 the stakeholder can create the *Sales Manager* actor and adds goals and descriptive information indicating that the actor represents actually a role. When the page is saved, the new actor is showed in the scheme of the Actor Viewpoint page. The *Define offers* goal is a red link meaning that the page corresponding to the goal has not yet been created. The stakeholder can then easily create the *Define offers* goal by clicking on the red link. In the goal form we can add information about the goal and its sub-goals. We model the *Define offers* goal as a soft-goal and relate it with two sub-goals: *Understand the buyer*

²TOCAI: Tecnologie Orientate alla Conoscenza per Aggregazione di Imprese in Internet (Knowledge-oriented technologies for enterprise aggregation in Internet)
- <http://www.dis.uniroma1.it/focai>

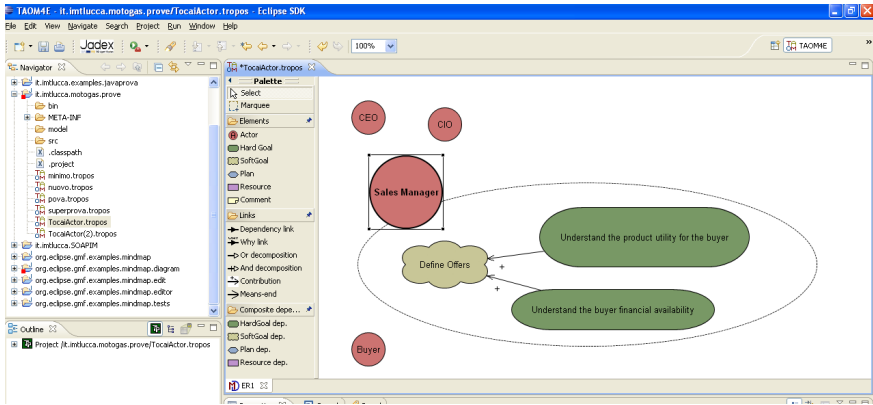


Figure 23: The actor model exported from WikiReq by the RDFtoEMF script and loaded in the TAOM4E plug-in

financial availability and *Understand the product utility for the buyer*. In the main page of WikiReq, the stakeholder (for instance an analyst) can in any moment open the RDFtoEMF wiki page and instantiate the script to export the Actor and Goal viewpoints of a specific project. The output in the TAOM4E (Cen08) Eclipse plug-in is represented in Figure 23. As described in this figure, we have zoomed the *Sales Manager* actor which contains the *Define offers* soft-goal and the related sub-goals. Then, the model can be further detailed in Eclipse by the stakeholders in order to define a more refined formalized representation of the BP's requirements. The BPR framework allows to relate the WikiReq model elements with both Use Case and BPMN model elements in order to connect functional requirements with non functional requirements represented as Si* soft-goals (5).

5.2 The SMOTE Tool

In this section we describe the Services MOdeling Tool for Eclipse (SMOTE), the tool we have developed in order to implement the framework transformations we present in Chapter 4.1. SMOTE is conceptually similar to our previous MDE tool, the MOTO-GAS tool (3), we have introduced in Section 1.2.2. However, SMOTE has been developed in the Eclipse IDE and manage models described by using the Ecore metamodel (The08b) rather than OMG MOF (OMG02a).

SMOTE exploits some plug-ins that provide for an MDA support in the Eclipse IDE as represented in Figure 24. We describe these plug-ins and their role in Section 2.3.3. In the Eclipse IDE, differently from other MDA-enabled IDE like Microsoft Visio (Mic08) and IBM Rational Rose (IBM08), the support of MDA feature is carried out by means of a set of community developed plug-ins rather than by a specific predefined component. Despite the Eclipse Modeling Tools package, we originally start using a set of component we set up independently from such distribution. This is not a problem since the models and code developed in SMOTE has been also tested in the Eclipse Modeling Tools environment without problems.

The SMOTE tool consists in two contributes:

1. The CIM2PIM ATL Transformation
2. The PIM2PSM ATL Transformation

We details each contribute in the following sections.

5.2.1 Goals2Service as a CIM2PIM MDA Transformation

The Goals to Service transformation of our framework described in Section 4.1.1 has been implemented in SMOTE in a CIM2PIM MDA transformation (OMG01). The model exported by the PHP script of WikiReq allows to represent a detailed goal and actor models in the TAOM4E plug-in. Indeed, TAOM4E represents models both graphically than in the Ecore XML notation.

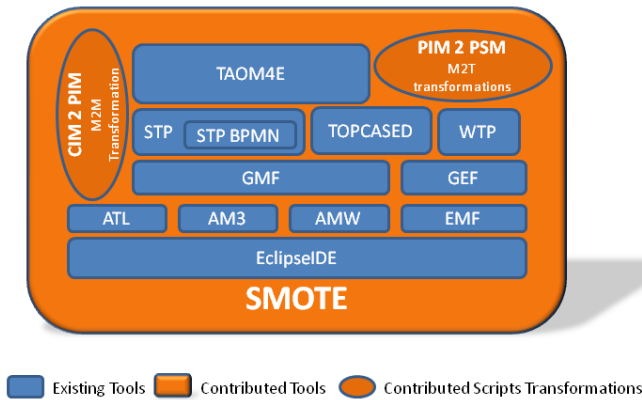


Figure 24: The SMOTE Architecture

BP's are described in SMOTE by means of BPMN in the STP BPMN plug-in. STP BPMN allows to graphically represent the BP described in the wiki document as an output of the Enterprise Modeling phase of the ESI method. Also STP BPMN outputs an Ecore version of its models that we use for the SMOTE transformation.

Actually, SMOTE does not use directly the Ecore models produced by TAOM4E and STP BPMN. We transform such models by means of a further PHP script that cleans some notations and make them comply with the SMOTE metamodels.

We define a SMOTE metamodel for the subset of Si* we use in our framework (reported in Figure 25) and the SMOTE metamodel of BPMN (reported in Figure 26). All model elements have to be meant with an attribute that defines their name (it is inherited by a generalized element that is not showed in the models). We also develop a SMOTE version of the UML Use Case metamodel. Either if SMOTE transformation currently do not involve Use Cases (they are only used by the stakeholders as a reference in the documentation) we develop the SMOTE Use Case metamodel because a future work about WikiReq is to implement Use Case modeling directly into the wiki.

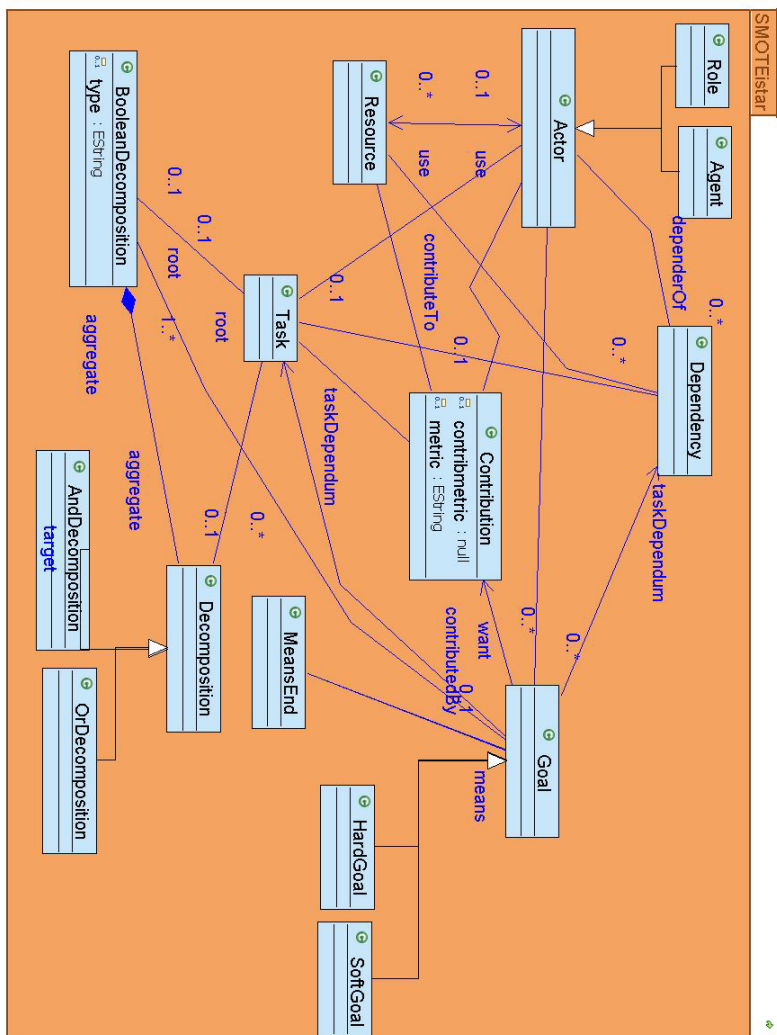


Figure 25: SMOTE tool: the Si* metamodel used by SMOTE

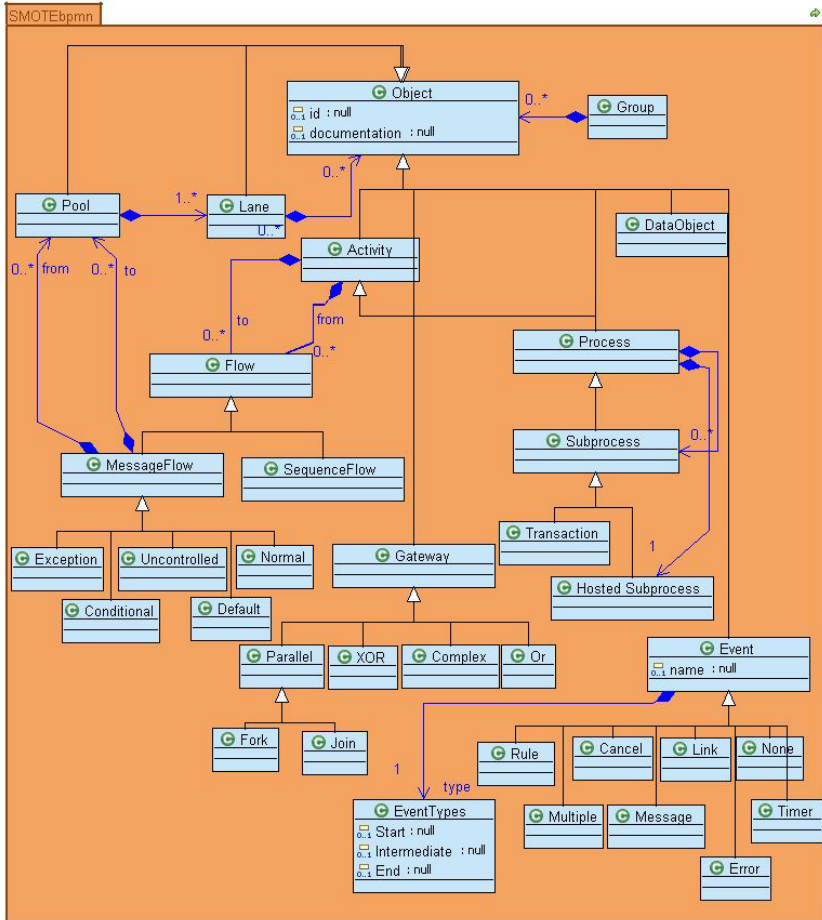


Figure 26: SMOTE tool: the BPMN metamodel used by SMOTE

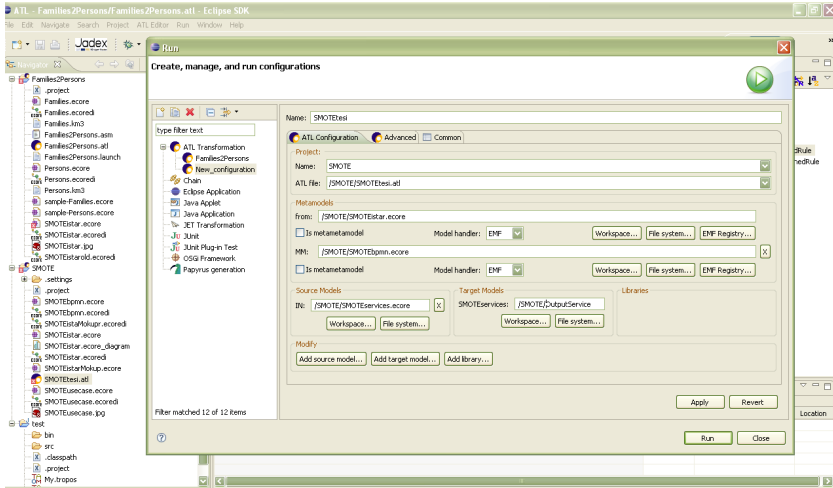


Figure 27: SMOTE tool: a screenshot of the SMOTE ATL configuration

SMOTE uses the ATLAS Transformation Language (ATL) (13; 95) and the ATLAS MegaModel Management (AM3) (34) in order to perform the transformations we generally describe in Section 4.1. ATL is both a declarative and imperative language. We adopt the declarative style in our transformations. Source model elements are navigated by means of a set of rules that create target model elements on the basis of source model elements. ATL relies on a ATL Virtual Machine that allows transformations to be translated in virtual machine code and executed by the ATL engine as described in Section 2.3.3.

We define ATL modules (such as model-to-model transformations) for each concept treated in our metamodels. Modules enable ATL developers to specify the way to produce a set of target models starting from a set of source models. Models are composed by: an *Header section* that defines some attributes related to the transformation; *Helpers* that are very similar to Java methods; *Rules* that define the way target models are generated from source models.

We show in Figure 27 the ATL interface used to apply the transforma-

tions of SMOTE. We set `SMOTEistar.ecore` and `SMOTEbpmn.ecore` as source metamodels and `SMOTEservice` as the target metamodel, while in the Target Model field we insert `SMOTEservices.ecore`, that is the name of the model produced by the transformation (that will conform to the `SMOTEservice` metamodel).

In order to give an example, we report a part of the `Goals2BusinessService` module of the `CIM2PIM` transformation as follows:

```

module Goals2BusinessServices;
create OUT : BusinessService from IN : Goal;

rule Goal {
from
a : SMOTEistar!Goal
to
p : SMOTEServices!BusinessService (
name <- a.name
)
}

```

For instance, the Task2Service module is very similar to the listed code, with the exception of the fact that it calls a method that verify the existence of a link of the Task with the goal in a context of a process.

5.2.2 Services2WebServices&Portlets as PIM2PSM MDA Transformations

The second transformation of SMOTE, such as the PIM2PSM transformation, allows to transform the Service Model `SMOTEServices` produced as an output of the CIM2PIM transformation in a model able to be implemented. Thus, the target model of the CIM2PIM transformation became the source model of this further transformation (and consequently the complying `SMOTEServices.ecoremetamodel` is used here as the source metamodel).

We show the `SMOTEServices.ecore` metamodel in Figure 28. In such diagram we distinguish four type of services. Actually, this is only a distinction needed in order to help the transformations to work because the metamodel concept actually needed is only `service`. The remaining three entities are used because the CIM2PIM transformation does not map directly into the final services. In order to obtain the service model, engineers must operate manually to decide if Business Service should be transformed into Services, if they are implemented by existing services or if they should be obliterated. Sometime the complexity of the analyzed

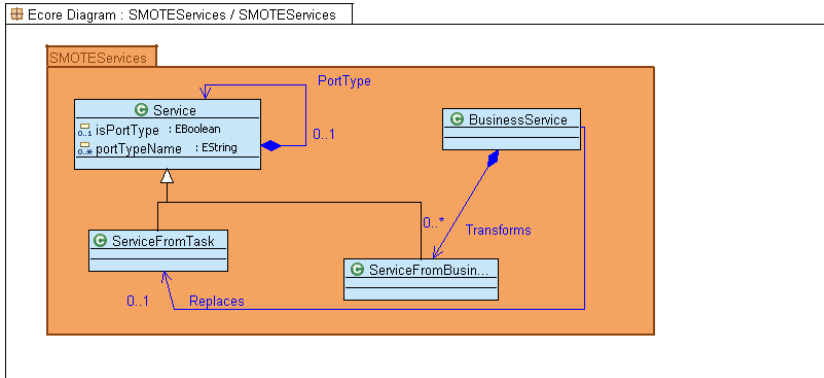


Figure 28: SMOTE tool: the metamodel for Service PIM's

Business Service lead to come back to analyze better the goal that generates the Business Service or, in other cases, to debate the goal again in WikiReq.

Actually, in PIM2PSM's transformations, the term "model" can mislead the reader and bring to think to a graphical model. Indeed, platform specific model can be both model graphically representable (e.g. in EMF) than instances of a language metamodel (e.g., Fortran). Thus the output of the transformation can be, beside a graph, also a textual PSM that can be compiled in order to be executed.

Thus the Services2WebServices transformation has been implemented both by means of an ATL Model to Model (M2M) transformation and as a Java Emitter Templates (JET) (The03) Model to Text (M2T) transformation.

The M2M transformation performed by means of ATL is simply an aggregation of some services. In the manual operation that define the service model the engineers instantiate the `isPortType` boolean attribute. That attribute helps the ATL transformation to identify a service that actually represent a function that does not need to be implemented as an independent Web Service but that can be used as a PortType of another Web Service. Thus the engineer sets the `isPortType` attribute true for

the services that will become Port Types and connects them with the service that will contain the PortType once transformed to a Web Service. At this point the ATL PIM2PSM transformation is run and the result is a new service model, very similar to the source model but with some services transformed in PortTypes attributes of other services. Instead to define another source and destination metamodel for this transformation we use the same service metamodel. The drawback of this choice is that we have some platform specific implementation details on the `SMOTEServices.ecoremetamodel`. This is not an issue since such details (i.e. the `isPortType` and `portTypeName` attributes of `Service`) are optionals and not instantiated in the CIM2PIM transformation.

The M2T part of our PIM2PSM transformation has been implemented in JET. JET is a generic template engine that can be used to generate SQL, XML, Java source code and other output from templates (The03). It is a part of the EMF Eclipse plug-in (The08b). In JET generating text takes two steps: a translation step and a generation step. The first step translates the template to a template implementation class. The second step uses this template implementation class to generate the text. Thus in the first step we have defined the template for the generation of the XML text of WebServices WSDL that takes in input the Ecore file of the source model. The JET API automatically has generated the related Java classes able to perform the transformation. Thus, these classes can be used by the command line interface to generate the Web Services WSDL definitions together with Port Type where defined from the service model.

Despite JET is included in EMF, it is used mainly to manage generic XML documents. It is a powerful but generic system to manage transformations that do not exploit the full capabilities of Ecore and MDE. In order to improve the M2T part of the PIM2PSM transformation and to extend it to support plug-ins, we are using another code generation tool named MOFScript (The05b). MOFScript is a complete framework for supporting model to text transformations, it supports the generation of implementation code directly from Ecore models providing support for parsing, checking, and the execution of the transformation. The MOFScript architecture will allow to improve the M2T transformation cur-

rently defined in JET and to implement the Plug-in PIM2PSM transformation.

Chapter 6

The Civil Protection Case Study

In Section 1.4 we have presented the issues related to the software development to be used in e-Government organization and in particular, in the Civil Protection domain.

In order to illustrate the use of our approach in a case study of real-world software project, this chapter presents a project we have developed in the Marche Region Civil Protection Department (MR-CPD) (The09b). We show how our framework has been applied by the MR-CPD in the Identification for Emergency Administration (IDEA) project (Mar07) concerning the development of a Radio Frequency Identification (RFID)-based system for the management of civil protection volunteers in emergency situations. The project started in October 2007 and terminate in May 2009.

6.1 The Case Study

The IDEA system issues from the general need (such as the general goal) of managing the civil protection volunteers during emergency operations. In particular, the MR-CPD needs to manage the accesses of voluntaries inside some emergency areas (e.g., camps, forest fires, garrisons, etc.) and the administrative operation connected with the participation of volun-

teers to the activities together with their vehicles.

The problem presents various complications, for instance:

- one goal is that volunteers data have to be acquired without loss of time in asking each person and, if possible, completely avoiding writing on papers or forms.
- the attendance of worker volunteers in an activity implies to perform a rapid BP where the MR-DPC has to communicate to the employer the participation of the worker to the emergency as a volunteer. Then, the MR-CPD will refund the employer for the working-day directly to its bank account.
- there is the need to estimate the number of meals that have to be prepared for the volunteers considering their turns in the camp, if they are away for operation and if they will have only lunch or if they also sup.

The MR-CPD contracts the project to two companies: The Proietti S.r.l. and Informasistemi S.p.a. The Proietti S.r.l. company is far 120 kilometers from the MR-DPC head office and employs in the project 1 programmer and 2 electronic engineers. The Informa Sistemi S.p.a. company is far 320 kilometers from MR-DPC employs 5 programmers, 1 software engineer and 2 managers in this project. We will name them company A and company B hereafter.

6.1.1 The IDEA Enterprise Modeling Phase

We start the Enterprise Modeling phase in order to model the civil protection organization structure and to better understand the stakeholders, their competences and their responsibilities. We design BP's for the entire voluntary service of the MMR-CPD since a good solution to the access control problem could also be applied to other contexts. Indeed, during the analysis we realize that two other processes can be automated relying on the same identification system: the management of the camps canteen service and the management of the emergency vehicles of the volunteers.

The requirements in the Enterprise Modeling phase have been acquired by means of 29 stakeholders. The MR-CPD team is composed by 18 people: a project manager, two software engineers, four expert of the volunteer domain, one administrative expert, five volunteers, two designed users of the system, two executive managers and one general director. The remaining 11 stakeholders are the workers of the two involved companies. We planned five brainstorming meetings and adopted the usage of WikiReq. WikiReq helps us to develop the collaborative specifications and to have the argumentation brainstorming. The Enterprise Modeling phase started in January 2008 and end in May 2008, some other meeting and work has been necessary until the end of the project (May 2009). The schedule and argument of the meeting has been as follows:

1. Jan 2008: The first meeting has been spent mainly to expose the usage of WikiReq and the Si*, UML and BPMN languages to all stakeholders. In this meeting we also start the informal brainstorming sub-phase. We require to all stakeholder to start writing their goals and involved actors or to rewrite the one described by others. We suggest to model all the organization structures by means of Si* giving more details for the volunteers system and the office that administer the volunteers.
2. Feb 2008: The second meeting has helped to solve some issues related to the understanding of the organization both by companies A and B, and among the MR-CPD employees. The latter also did not have a clear and shared envision of the processes in the volunteers context. For instance, MR-DPC project managers and software engineers worked in a office that manages hydro-geological distasters so they had a vague and wrong supposition on he way volunteers were managed. In this meeting a first cyclic step of the enterprise knowledge formalization sub-phase is performed, some goals and relations among actors described in the delegation model were discussed and clarified.
3. Apr 2008: In this meeting the scenario become more clear, we understand that also the management of the camps canteen service

process and the management of the volunteers emergency vehicles process can be reengineered by means of the same technologies exploited for the volunteers management, thus we start informal brainstorming also for these parts of the project. Before the meeting, the engineers performed a first exports of the goals and actors diagrams from WikiReq to TAOM4E. They graphically analyzed the defined relations and compare them with Use Cases and delegation models. Since there were some inconsistencies, the issues have been discussed and explained to MR-CPD managers.

4. May 2009. In the fourth meeting the actors, goals and relations scenario was quite complete. Soft-goals in particular has been discussed in order to clarify their fulfillment conditions if existent. The diagrams exported in TAOM4E have been presented and analyzed together with the other stakeholders. Actually, it has been a more technical meeting more focused on the Enterprise Knowledge Formalization sub-phase.
5. May 2009. The last meeting held after 20 days from the fourth. It has been a meeting where the first sketch of the service model produced from the CIM2PIM transformations has been presented to the MR-CPD stakeholders in order to analyze the services that the goals produced.

Since the project is relatively large (it is a SME's-sized project as we expect for our method) we consider unnecessary a further iteration from the Service Modeling phase to the Enterprise Modeling phase at the end of requirements engineering activities. However, if important incoherences and services produced, further iteration would have performed.

Quite all the stakeholders have taken part to the first two meetings, however from the third meeting the participants go down to the 50 % or less. On the contrary, the WikiReq usage increase from March 2008 to April 2009 such as in the most important period for the requirements engineering activities performed in the Enterprise Modeling phase.

For the sake of brevity we do not report all the actors and their description. Some of them has been shown in Figure 29 that reports a part



Figure 30: The *To speed up administrative processes* goal page in WikiReq

3. *To refund the employers*

4. *To know if there are ongoing changes in the number of meals*

Goals 1 and 4 are shared with the Canteen Volunteers actor. About these goals we give an example of argumentation. The panel discussion in the WikiReq talk page originate because of the different and opposed meanings of the two goals for the two actors. The Volunteer Office mean to spend less by reducing the number of meals prepared and not served, while Canteen Volunteers was only interested in reducing meals not served since they were refunded basing on the number of meals prepared. This originate a different AND/OR Structuring of the sub-goals that has been discussed in meeting 3 and 4. The goal *to speed up administrative processes* that we report expanded in Figure 30 has been structured in four sub-goals: *to have the bank account of each volunteer; to be sure of the identity of the volunteer; to automate refundings*.

Other goals such as *Do not wait for meals*, *Do not wait for registration* and *To use a contactless technology* bring to the choice of adopting the Radio Frequency IDentification (RFID) technology and to endow each volunteer with a RFID badge. In particular, the system need to be used in emergency situations thus it requires quick and easy interactions with a

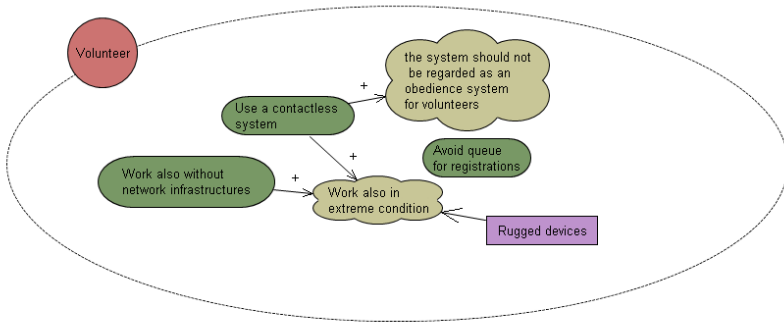


Figure 31: A subset of the IDEA stakeholders goals

high number of volunteers (such as between 200 and 1.500 peoples). Another factor that brings to the adoption of the RFID is that other identification technologies such as Bar Codes and Magnetic Bands deteriorate in case of extreme condition (e.g., mud, sun exposition, contact with cellular phones etc.).

A small subset of the stakeholders goals are represented in Figure 31. The goal *the system must not be regarded as an obedience system for volunteer* has determined the choice to use free-hands gates instead of automated doors in monitoring the accesses of the volunteers. Other identification technologies (i.e., a Bar Code and a Magnetic Band) are combined with RFID in order to assure redundancy of infrastructure in case of failures. Beside free-hands gates also rugged and wearable palmtop computers are used to read the volunteer badge.

6.1.2 The IDEA Service Modeling Phase

In order to comply with the stakeholder deadlines, in the Service Modeling phase we choice to split the system realization into three parts. The full project is named IDentification for Emergency Administration

(IDEA) (Mar07). The first part of IDEA concerns with the free-hand gate access control and the access control software development. The second part is an upgrade of the access control technology to support the canteen management and the palmtop usage instead of free-hands gates. The third part concerns with volunteer vehicles management in emergency situations (vehicles tracking and monitoring, driver identification, itineraries, etc.). We perform 7 CIM2PIM transformations, two for the first two parts of IDEA and four for the third part.

In particular, we present the Service Modeling activities related to the *palmtop software* actor and the related transformation.

The most relevant goals concerning the palmtops non-functional requirements are: *to work with and without network infrastructures; to manage of both centralized and distributed data; to use rugged devices; the rapid display of reports to support decision making*. Beside these goals, also functional requirements for the software have to be considered. They are represented by the same goals of the *Registration system* actor such as: *To register accesses; To register volunteers that take part to the event or emergency; To register meals; To manage security with the accesses*.

Here is the service and business services produced by the CIM2PIM transformations directly from the goals exposed above (we rename them for a human comprehension):

- **NetworkSupport** : it is a Business Service that have to manage the sending of data from the palmtops to the servers considering the presence or not of network infrastructures (e.g., Wireless networks, cellular phone data networks, etc.). If there are not networks available then data must be collected locally and then send to the server. Since conflicts can occur, then the service should also solve such data inconsistencies;
- **DataStore** : it is a Business Service that works with **NetworkSupport** in order to manage data storage and sending of data to the distributed architecture. It is bounded to some tasks that appear also in BP's that describe how the data must be send and received basing on the presence of a network;

- `RuggedDevice` : is a Business Service that indicates that the palm-top device has to be a rugged device;
- `Reports` : it is a Business Service that manages the display of reports about the activities;
- `Registration` : it is a Business Service that manages the registration of a volunteer that takes part in a event or emergency;
- `AccessMonitoring` : it is a Business Service that manages the accesses (i.e., entrances and exits) of volunteers from a camp or an emergency area;
- `CanteenManagement` : it is a Business Service that accounts the number of meals needed and served both for lunch and dinner.
- `SecurityManagement` : it is a Business Service in charge of managing security issues.

There are also other services derived from the tasks needed by the presented goals and that appear in BP's, such as:

- `AccessMonitoring\DefinitiveExit` : this service manages the definitive exit of a volunteer from an emergency or an event. It is the counterpart of the `Registration` service but is managed in the context of `AccessMonitoring` business service.
- `AccessMonitoring\ExitforActivities` : this is a service derived from a task of the *To register accesses* goal that indicates a different way to manage exits from the camp or the area of activities. The volunteers exit for a particular objective, such as an activity like succors, garrison or surveillance. The related BP's defines a further activity beside the one expected for the exit such as to register the type of activity.
- `Registration\RegistrationWithoutBadge` : this service manages the task related to the registration of volunteers that do not have the RFID badge. They might do not have the badge because

of they miss it or they have not already received it from the MR-CPD. So, a different registration process is needed that has to be supported by this service.

- `SecurityManagement\Identification` : this is a service related to the *To manage security with the accesses* goal task. The RFID badge allows also to identify the volunteer and to know information about him, for instance: if his organization is authorized to participate to the event/emergency, if he is an active volunteer, etc.

At this point the CIM2PIM's transformations end its work. A manual adjustment has been performed by the engineers of companies A & B, and by the engineers of MR-CPD. As we present in Section 4.1.1 all the Business Services has to be transformed in Services or obliterated.

The `Report Business Services` is actually a service since it has been already identified as a task of a goal in the transformation relating the access control part of the project. `RuggedDevice` is actually a Business Service that is not related to systems services (such as, services derived from a goal of the system actors) and thus it has been obliterated. In order to obliterate a Business Service does not mean to discard it. On the contrary, for instance *to use rugged devices* identifies a fundamental non-functional goal, the relating service `RuggedDevice` will thus be "implemented" outside the system context, and hence in the organizational context, that requires rugged palmtops. Thus, the service `RuggedDevice` will be implemented by the rugged palmtop devices. It is also interesting to note the argumentative discussion in the talk page about this goal: technicians pound away at buying rugged devices; the Administrative Office is averse because the cost of fulfilling this goal is four time the cost of buying not-rugged palmtops (there was a soft-goal added by the Administrative Office that influenced negatively the other goal that has been deleted in Enterprise Modeling). The Head of Department decide for a win-win solution. He claims that "it is true that rugged devices are expensive (in the future we will buy 50 devices) but it is also true that in some situations rugged devices are needed", so the decision has been to buy the first 6 devices rugged and the future 50 not rugged. All the

remaining Business Services exposed above has been transformed into Services.

The service model of the palmtop software part of the IDEA system modeled in the SMOTE service model editor developed by means of GMF (The05a) has been reported in Figure 32

6.1.3 The IDEA Platform Specific Implementation Phase

In the System Implementation phase the PIM2PSM Web Service transformation produces a service-oriented architecture derived from the application service model developed in the System Modeling phases. For each Service in the service model a Web Service is defined.

The goals of data decentralization and the uncertain on the existence of network infrastructure, deeply influence the design of the architecture. All devices involved in the IDEA architecture (RFID gateways, rugged and wearable palmtop) access a shared database by means of three alternative network technologies: WiFi 802.11b, GPRS and Wired Ethernet. However, each device can manage data locally and then transfer data to the centralized database or to other devices that it "senses". The `NetworkSupport` and `DataStore` have to manage the switching among networks and the conflicts of opportunistically transferred data.

The Graphical User Interface (GUI) for the the palmtop software has been developed in the System Implementation phase. As depicted in Figure 33, it corresponds approximatively to the `AccessMonitoring` ; `CanteenManagement` ; and `SecurityManagement` services of the system by means of three large buttons. This approximation is a simplification used in this thesis to better explain the Implementation Phase, however, functions in the GUI do not necessary match with one single service. For instance `Registration` is implemented in the same GUI function of `AccessMonitoring` and `SecurityManagement` is actually implemented by the `SecurityManagement\Identification` service. Thus, here is evident how the service abstraction is useful in splitting system functionalities in a right way for the platform specific architecture (maintaining the traceability with requirements). At the same time, a further

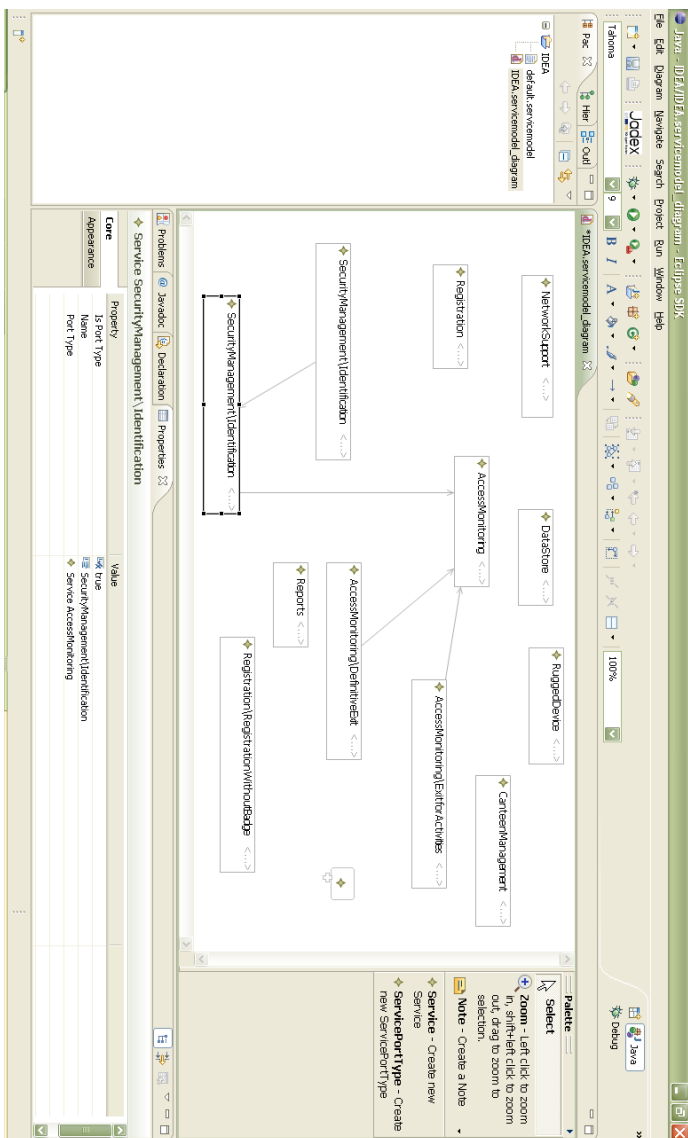


Figure 32: A screenshot of the IDEA Service Model in SMOTE



Figure 33: A screenshot of the IDEA palmtop software

abstraction can help in grouping service together, and define GUI functionalities. For these reasons we are investigating plug-ins as a plausible way to group services in order to define GUI functionalities as exposed in Section 4.2.

Once the user pushes a button the RFID reader starts reading in a continuous way in order to search for tags. A continuous reading allows to perform multiple tag reads without further human interactions. Such a continuous and usable reading strategy has been tested to be very useful in emergency but it conflicts with the soft-goal of a *long mobile device battery endurance*. In order to fulfill this non-functional requirement during the IDEA Implementation phase we took the decision of developing a specific function that manages the RFID reader activity accordingly with palmtops screensavers states.

A goal-oriented design of the early requirements helps in performing a better user-centred engineering of the application also during the System Implementation phase. Goals such as *the system should be usable by aged and technology-unskilled volunteers*, bring to reduce the number of functions presented in the GUI to three big buttons.

6.2 Critical Evaluation and Results of the Experiment

We have submitted a questionnaire to all the 29 stakeholders that have used WikiReq during the Enterprise modeling phases of IDEA. The questionnaire investigates how the wiki is perceived by the stakeholders considering their skills and competencies about business modeling and software engineering.

The objectives of the study are: (a) to measure the degree of satisfaction; (b) to measure the degree of participation of the stakeholders.

The questionnaire comprises eleven questions, 2 about personal skills and 9 about the experience with WikiReq. The stakeholders can replay to the 9 questions with a five-point Likert scale (112) , such as: 5 = "strongly agree" , 4 = "agree" , 3 = "neutral" , 2 = "disagree" and 1 = "strongly disagree".

The questions are:

1. What is your knowledge about software engineering?
2. In how many ICT development projects have you been involved in the last 6 year?
3. WikiReq caused the write of unnecessary requirements.
4. The time spent in learning the ESI language framework has been useful in define correctly the requirements about the project.
5. WikiReq collaborative features (such as, the live chat, the argumentation system, the versioning system) are useful to clarify the requirements about the system and the organization.
6. WikiReq help to save time.
7. Requirements gathered by means of WikiReq are qualitatively better than requirements gathered in other projects.
8. The resulting system satisfy the requirements.

9. WikiReq helps in selecting a better solution for an issue in spite of a meeting.
10. WikiReq does not help to learn new facts about the organization that will use the software.
11. The services that fulfill the requirements expressed by means of goals are evident in the service models.

The panel of the 29 stakeholders is composed by: 6 programmers; 5 engineers; 12 users; 6 managers. We obtain 22 compiled questionnaire (such as the 75.8%). The items 3 and 10 are negative items, thus their answers have been inverted. It has been assigned a value from 1 to 10 to the first question basing on the answers.

In Figure 34 we report the data acquired by means of the questionnaire. The interviewees are more pleased by their experience with the wiki relating to the organization than to the system. In particular, this attitude is clear in question 4, 5 and 9. Also the general question 8 (i.e., The resulting system satisfy the requirements) reach a kind result with a fairly good standard deviation.

The results derived by this analysis conform to similar researches on wikis for requirements engineering. In particular, Eric Ras has analyzed two case studies (140) in the application of the SOP-Wiki platform (see Section 2.2.3). He shows that the highest effort savings has been observed for corporate knowledge and users experience management, such as to improve the knowledge about new facts on the organization and the system that were unknown before and which would not have been communicated by verbal communication.

Majchrzak et al. (115) performed a survey with 168 corporate wiki users. The users mentioned that reputation is enhanced, that work is made easier and that it helps to improve the organization processes.

Chau and Maurer (45) also performed a case study in a medium-sized software company, where they used a wiki-based experience repository called MASE for formal and informal knowledge acquiring. Their evaluations focused on read access (80% of read access were concentrated on

People	Answers											
	3	4	5	6	7	8	9	10	11	1	2	
Programmers	Stefano	3	4	5	3	5	3	5	4	4	8	15
	Fabio	2	4	4	4	2	4	5	3	2	7	20
	Maurizio	4	4	5	5	4	3	4	4	4	7	35
	Matteo	3	4	5	4	5	3	5	3	3	5	5
	Pietro	4	5	4	4	4	4	5	3	4	4	8
Engineers	Marco	5	4	5	5	4	4	4	3	4	8	6
	Luca	4	5	4	4	4	5	5	4	4	10	7
	Paola	5	5	4	5	4	5	5	5	5	9	8
	Luigi	2	5	5	5	5	5	5	5	5	6	25
	Simone	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Users	Andrea	5	4	5	3	5	4	4	4	4	9	N/A
	Cesarina	4	3	5	4	3	3	5	5	3	N/A	N/A
	Mauri	3	4	5	3	3	4	5	5	4	2	3
	Caterina	4	3	3	4	3	5	5	5	3	2	1
	Paolo	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Serena	3	4	5	3	3	5	2	4	3	N/A	1
	Stefano	3	4	3	3	4	4	5	5	3	N/A	N/A
	Ettore	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Marco	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Susanna	4	3	4	3	3	5	5	5	3	2	5
	Enrico	2	2	4	3	4	3	4	4	3	N/A	N/A
	Gianluca	3	4	3	3	3	3	5	5	3	8	10
Managers	Francesco	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Maurizio	3	4	5	4	3	5	4	2	3	3	15
	Andrea	5	5	5	2	3	5	4	3	3	8	15
	Giuseppe	5	3	4	4	4	4	3	2	4	2	8
	Pierpaolo	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mean	Luigi	4	4	4	4	4	5	5	3	7	N/A	N/A
	Roberto	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Standard Deviation	Mean	3,64	3,95	4,36	3,73	3,73	4,14	4,50	3,91	3,50		
	Standard Deviation	1,00	0,79	0,73	0,83	0,83	0,83	0,80	1,02	0,74		

Figure 34: The results of the questionnaire about the WikiReq system.

20% of the pages), amount of structured (42%) vs. unstructured (58%) information and amount of synchronous vs. asynchronous (more than 90%) communication.

Beside the evaluation of WikiReq by means of the questionnaire we also analyze the impact of the IDEA system in the BP's that it reengineers. The registration of volunteers on camps in normal situations (e.g., with more that 800 volunteers) need four people that spent about four hours to manage this task. Moreover, the volunteers have to wait in queues from 3 to 30 minutes to be registered. The IDEA system optimize and speed up this BP that is now performed by one person in few second per volunteer that has to wait a maximum of 5 minute to be registered. Post emergencies BP's, such as the refunding of vehicles fuel and employers require about the 30 per cent of the time of the previous administrative processes.

The analysis of the IDEA system shows that such a system fulfill also unforeseen goals, such as it helps to evaluate the effective contribution of volunteer organizations in an emergency or an event, and hence to refund these organizations accordingly with meritocratic criteria.

A final remark about the IDEA project and its results concern its efficacy. The Italian Minister of Public Administration indicate this project as one of the best-practices in e-Government ICT projects ¹. The inexpensive budget for the development of the IDEA system lead other e-Government organizations (such as the Rome Municipality, the Veneto and Piemonte Regions) to ask for reuse such projects. Actually, it is an ongoing process, but the knowledge acquired by means of WikiReq is going to be easily shared and reused by the other organizations. In particular, issues and goals discussed and solved by means of WikiReq in IDEA are analyzed by the engineers of the reuse projects and directly presented to their stakeholders to focus on such most challenging issues. Thus, we foresee a further WikiReq benefit to study, such as the ability to easily reuse and share "solutions". These solutions are not given as software components or reengineering practices but as goals binded to software-implementable

¹Ministero della Funzione Pubblica - Non Solo Fannulloni - Progetto IdEA - <http://www.nonsolofannulloni.forumpa.it/100-storie/segnalazioni-dalla-pa/?id=583>

services and requirements structured and unstructured knowledge (e.g., discussions, chats, arguments, etc.).

6.2.1 Lessons Learned and Threats to Validity

The application of the ESI method; the tools and the framework presented in this thesis to the IDEA project, helps us to realize some limitation and future extensions of our work.

We become aware that the framework work better in projects that expect a long time for requirements engineering activities. There is a trade off between the size of the project, in particular regarding the requirements elicitation, and the effort needed by all stakeholders to understand the three languages we use and the WikiReq platform (even if they are simple to learn). However, the time saved in software maintenance due to the right and quite complete identification of system functionalities will compensate costs for requirements engineering. In fact, in IDEA, ESI has reduced both maintenance time than cyclic phases of development if compared with similar project in the Marche Region. Consider, for instance, the time spent in the SISSI project presented in Section 1.4.2 for iterative phases in the definition of the interface form field. In order to support this claim, we require to apply ESI to at least another project.

From the point of view of the application of the ESI method, in IDEA we noticed that a great part of the goals have been identified during the discussion about the actor, such as when we ask stakeholders to describe in detail actor feature and characteristics. We also noticed an interesting correlation among tasks that originate a service in the CIM2PIM transformation and the Use Cases, such as between BPMN activities that satisfy a goal and Use Cases. In particular, the most relevant Use Cases (i.e., cases that originate services independent from other goals functionalities) are represented as Si* tasks. Thus, such tasks are quite all described by a Use Case that can be used to give implementation details to the developers. Generally speaking, the fact that requirements engineering is an activity that should be carried out in all phases of system engineering is not a novelty as reported by Young in his Handbook (171). It is interesting

to note in ESI that there are requirements acquired in business modeling that can be used in different phases of the system engineering. This is an unexpected effect that we will further study in future application of the method.

Another unexpected lesson learned by the application of the ESI method to the IDEA project concerns the usefulness of Si^* in late phases of analysis and development. Besides the goals and actor model, we also define the trust and delegation relations in the WikiReq relational viewpoint. In ESI the relational viewpoint has been defined to help stakeholders to define accordingly actors and goals viewpoints and their related models. Since relations are described at a requirements-level and defined considering goals instead of service, developers have found very useful the knowledge described in the wiki. This has been especially true for trust and security policies to adopt in services that implement functionalities for the volunteers' access and exit BP's. They have been also useful to define the security levels in the palmtop and Web Application software.

Finally, the IDEA experience has confirmed our hypothesis about the application of plug-ins architectures as patterns for the orchestration of services. We observe that services once decomposed from goals in the CIM2PIM transformation and further decomposed in the PIM2PSM transformation, need to be grouped again in a way that is consistent with the implementation architecture and the GUI functionalities (e.g., IDEA server, palmtop software functionalities, etc.). We mean that plug-in architectures help in defining such a composed services.

Chapter 7

Conclusions and Future Works

In this thesis we have presented the Enterprise-Service-Implementation (ESI) method and a framework of technologies aimed to implement this method. Our aim is to define a framework that helps software engineers to develop their systems in what we named the business-oriented approach. The business-oriented approach to software development has been defined as an approach that takes into account new challenging issues of software engineering, such as: the complexity of globals and multisited organizations; the changes made by new technologies and software in the organization BP's; and the services related to the products the company or the organization supplies.

We present a systematic approach that enclose in a single framework all the technologies needed to address the problem of modeling the business in requirements engineering and, at the same time, to relate business-level concepts (such as strategies, goals, trust, etc.) to system artifacts. Our work involves a broad spectrum of researches and technologies we analyze to define a fairly lightweight approach.

We explore the engineering of requirements by using the technology of wikis in order to define a collaborative system which stakeholders can use to write requirements. We also provide the stakeholder with a model-

ing language that helps to define a computationally independent model of the entire business domain. We use different notations and produce a model suitable for further transformations based on MDE and the service abstraction.

Thus, our method starts from what Mylopoulos et al. (102) name the “early-requirements”, such as the requirements of the organization, and its domain. The ESI method manages models of both the domain and the system down to the implementation phases. The service abstraction and the ability to manage such an abstraction with MDE technologies help us to translate concepts from the business domain, such as goals, into services more and more concretely bounded to a specific implementation platform. Also the *i** and Tropos role is decisive since they allow to model non functional aspects of the organization and to relate them with the systems.

The effectiveness of our approach has been tested in the Civil Protection case studio reported in Chapter 6. The Civil Protection domain and this case studio presents all the issues we want to address with our framework, for instance: heterogeneity and global asset of the involved organizations; strong correlation among BP’s and the system to develop; many soft-goals to debate and analyze. The results of this case studio evidence the fact that our framework improves the requirements engineering activities for all stakeholders, their work is made easier and they also improve the knowledge about new facts on the organization and the system.

The impact of a wide participation of stakeholders, in particular users, in requirements engineering has been empirically tested in (68) to be critical. The results indicate that as uncertainty increases, greater user participation alleviates the negative influence of uncertainty on the quality of requirements engineering. This also results concerning the stakeholder involvement (62). Complex organizations, multisited development and evanescent services provided by companies, together increase the uncertainty in software projects. The software projects in turn, are more and more bounded to the organizational assets: the software may change the organization BP’s, the software can create new services to be used by

workers or to be sold, and hence a new business or a new piece of the organization, etc.

In order to solve these interwoven issues we interweave three instruments: a wiki exploited as a platform for requirements engineering; a set of languages easy to be understood by all stakeholders; a MDA to manage stakeholder-understandable abstractions of the software.

Many related works confirms the effectiveness of the wiki-based approach we adopt, in particular in the design of software taking into consideration both business processes and the organization (11; 52; 62; 114; 137; 140; 161). WikiReq together with the languages we exploit for requirements engineering, helps us to define a framework that permits stakeholders to collaborate sharing their knowledge in terms of a small set of concepts.

The need for a predefined structure for requirements acquiring in wikis is an essential pre-condition to avoid inconsistencies and uncontrolled content sprawls as evidenced by Decker et al. (62). However, WikiReq gives more emphasis on this pre-condition than it is done in the Decker et al. SOP-wiki. In fact WikiReq uses the explicit semantics of Semantic MediaWiki RDF notations and imposes a template also for the discussions structure. Different stakeholders usually assign different meanings to the constructs of the organizational knowledge and to the notation used to represent such knowledge (e.g., actors, classes, tasks etc.). The wikis collaborative features are particularly important in these contexts. Our approach reduces and simplifies the concepts involved in the requirements and BP's structuring and, at the same time, complies with more technical artifacts (such as UML Use Cases and BPMN models). In this way, WikiReq forces stakeholders to reason and expose their requirements in terms of the few set of concepts presented for Si* that are then connected with the other language by means of the transformations. Our wiki also improves the debate management if compared with SoftWiki and SOP-wiki. The argumentation feature described in Section 5.1.2 has been successfully experienced by Karacapilidis et al. (97) for both requirements and collaborative BP's debates (9). We exploit argumentation also in the wiki context combining the results of both research areas (such as

wikis for requirements engineering and Web-based argumentation and collaborative decision making). WikiReq offer features to "detect unexpressed conflicts" (62) that is one of the six drawbacks in using wikis for requirements engineering we report in Section 2.2.3. Also others drawbacks are solved by WikiReq such as: "page name remembering" that we solve with the semantic autocompletion of fields, "page structuring and reclassification" that are solved by WikiReq Semantic Forms exploiting and "versioning across several pages" that can be solved by backups of contents in the RDF format. Semantic Forms are also useful to maintain the collaboration platform easy to use and maintain, unlike SOP-wiki that uses Adobe flex to create forms and manage semantics and thus requires the installation and understanding of other components in order to use or customize the platform if needed.

The WikiReq semantically-annotated and machine-processable forms allow both to export the requirements in Eclipse and to bind the discussions to a specific concept. We give a one-to-one relation between pages and goals/actors. As we discuss in Section 2.2.2 this is one of the main advantages of using wikis to structuring knowledge rather than content management systems.

Further work in WikiReq is required to model Si* tasks directly inside the MediaWiki platform. This will be useful to show to the stakeholders how the Si* task concepts will be related to the BP's activities. We are also working on integrating an extension of MediaWiki that helps to directly design UML diagrams into the wiki and export them in RDF. Such an extension will be based on the MetaUML language (Sol09). A more challenging issue is the ability to manage a two-way relations among the Eclipse IDE and WikiReq, such as to re-import models from Eclipse to WikiReq. This feature will allow to semantically represent in the WikiReq platform the enhancements of models carry out by analysts in the Eclipse plug-ins (such as TAOM4E).

Also the SMOTE tool will be improved in many ways. In general, the PIM2PSM transformations that maps the service model to a platform specific implementation of the model, has to be enhanced to support the production of more details in the WSDL code. The PIM2PSM transforma-

tion is going to be translated in MOFScript(The05b) in order to support the generation of implementation code considering the Ecore models element instead of the XML file as it is in JET. MOFScript will provide a complete support for parsing, checking, and the execution of the transformation of the model to the code. In order to graphically model the services in SMOTE, we currently use the EMF.Editor editor generated by means of GMF. We made only few changes to the editor automatically generated, so it such editor is very coarse (see Figure 32) and can be refined adding richer graphical elements and tools.

As we state in the Introduction, our work is not intended to concern automatic orchestration and choreography of SOA's and Web Services. We limited our research to the use of SOA's software platform as a middleware to define platform independent software. Thus, we exploit the service abstraction to define a new service or application plug-in defining a model where a set of services are statically related. Thus, from a software engineering point of view we consider SOA's as reference architectural middlewares for the development of software components without considering providing support for service orchestration and choreography. This is a limitation we think to overcome by the approach described in Section 4.2. We retain such approach the main future extension of our work, in particular in the use of plug-in architectures to define a user interface for a new developed services. The future works described until now are based on methods and tools we have already developed, the approach of using plug-ins as a service wrapper instead is a completely new part of our research. Either if it can be implemented as a further extension of the PIM2PSM transformation made by means of MOFScript, the plug-in paradigm is particularly interesting from a methodological point of view. In particular, we plan to study plug-ins architectures relating them with Feature Oriented Programming (FOP) (149) that is the study of feature modularity in product-lines. A feature is a characteristic that significantly distinguishes a member within a family of similar systems (i.e. a product-line). Features can be considered increments in product functionalities of large-scale software design and construction. Indeed FOP moves from step-wise refinement (65), a paradigm for developing com-

plex programs from a single program by incrementally adding details. FOP proposes to specify systems as a composition of features and "pattern" of features (named collaborations (16)). Some approaches to FOP treat composition synthesis of programs. Features and multi-featured applications are treated as algebraic expressions. Algebraic approaches define basic programs and features respectively as a set of constants and refinements, they use this "domain model" as an algebra for the domain to be modelled (such as the product-line) (149). The general goal of the FOP approaches is to make the modular design of applications as simple as possible. The existence of an underlying algebra and the ability to compose programs by means of declarative languages (50) constitute a practical and powerful framework to base our work on plug-ins composition.

FOP and the base research fields underling FOP represent valid contributes to be applied in SOA's structuring their ESB as plug-in architectures. Indeed, the plug-in architectural style:

- It is very similar to a software product-line: the plug-in host defines a family of similar systems (149) that can change the application component plug-ins basing on the stakeholder needs.
- It builds applications following a step-wise refinement approach. Starting from the host application, features are added or subtracted in order to obtain the desired system.
- It is based on an orchestrator component that is the scenario expected in SOA's and hierarchical structures of architectural metaprogramming (17).
- It can be easily manipulated by model driven engineering due to its very high-level representation of software components.

The refinement and synthesis approach of FOP (currently studied for the object oriented abstraction) can be compared with the operations implemented in the plug-in approach (e.g. hooks, extension points etc.). Thus, specific refinement operations can be studied for plug-in architectures to define refinement and synthesis of service composition patterns.

Basing on the FOP work, a core algebraic framework can be defined to be implemented in plug-in hosts (for instance treating the hosts themselves as base programs for feature refinements). Declarative Domain Specific Languages (DSL) (50) can be studied for plug-ins composition. They can be derived from general real domains or single plug-in architectures (e.g. a DSL for Civil Protection, a DSL for the Yahoo! widget architecture). Declarative specifications can improve optimization and give further simplification and abstraction of service composition .

Finally, the application of the ESI method to the Civil Protection case studio evidenced a lack in the management of data. A critique move by engineers and programmers of IDEA to our method and tools is that it is currently too oriented to the functionalities of the system and do not take into account data concepts. We have underestimated the importance of data in requirements engineering because we was convinced that data would have been useful only in implementation phases (e.g., when developers implements Web Services). Instead, IDEA show us that data is important both in modeling goals and BP's, also at a requirements engineering level. We plan to cover this lack of our method exploiting the resource element of Si^* and defining a specific viewpoint page in the WikiReq tool. Such a viewpoint will be treated by other MDA transformations and managed as an ontology model (24) in order to be implemented in database schemes or other data repositories. In this way we will improve also the service model that currently does not manage nor service attributes neither PortTypes parameters.

Bibliography

- [1] T. Abe. What is Service Science? Technical Report No.246, Fujitsu Research Institute., Mar 2005. 4, 5
- [2] L. Abeti, P. Ciancarini, and R. Moretti. A Service Oriented Approach to Model a Grid System for the Civil Protection. In *Proceedings of Workshop on Complex Networks and Infrastructure Protection (CNIP2006)*, March 28 - 29, Rome, Italy, pages 489–498, Rome, Italy, Mar 2006. 23
- [3] L. Abeti, P. Ciancarini, and R. Moretti. Service oriented software engineering for modeling agents and services in grid systems. *Multiagent and Grid Systems*, 2(2):135–148, Apr 2006. 14, 18, 21, 130, 137, 138, 157
- [4] L. Abeti, P. Ciancarini, and R. Moretti. Model driven development of ontology-based grid services. In *WETICE*, pages 229–234, Los Alamitos, CA, USA, 2007. IEEE Computer Society Press. 21, 105
- [5] L. Abeti, P. Ciancarini, and R. Moretti. Business process modeling for organizational knowledge management. In Pierpaolo Degano, Rocco De Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models*, volume 5065 of *Lecture Notes in Computer Science*, pages 301–311, Berlin / Heidelberg, 2008. Springer-Verlag. 13, 138, 147, 156
- [6] L. Abeti, P. Ciancarini, and R. Moretti. Wiki-based requirements management for business process reengineering. In A. Aguiar, U. Dekel, and P. Merson, editors, *ICSE Companion Wikis4SE Workshop Proceedings*, volume CFP0935G, pages 14–25, Los Alamitos, CA, USA, May 2009. IEEE Computer Society. 14, 73
- [7] L. Abeti, P. Ciancarini, and V. Presutti. An Ontology Driven Method for Designing Software Agents for Workflows across Organizations. In A. Cimitile, A. DeLucia, and H. Gall, editors, *Cooperative Methods and Tools for Distributed Software Processes*, Software Technologies, pages 162–175. Franco Angeli, Rome, Italy, 2003. 1

- [8] Z. J. Acs and L. Preston. Small and medium-sized enterprises, technology, and globalization: Introduction to a special issue on small and medium-sized enterprises in the global economy. *Journal Small Business Economics*, 9(1):1–6, Feb 1997. 16, 24
- [9] E. Adamides and N. Karacapilidis. A knowledge centred framework for collaborative business process modeling. *Business Process Management Journal*, 12(5):557–575, 2006. 115, 120, 121, 122, 148, 153, 188
- [10] A. Aguiar, G. David, and M. Padilha. Xsdoc: an extensible wiki-based infrastructure for framework documentation. In Ernesto Pimentel, Nieves R. Brisaboa, and Jaime Gómez, editors, *JISBD*, pages 11–24, San Vicente del Raspeig, Alicante, Spain, Oct 2003. Universidad de Alicante. 70
- [11] K. Al-Asmari, R. Batzinger, and L. Yu. Experience distributed and centralized software development in ipdns project. In H. R. Arabnia and H. Reza, editors, *Software Engineering Research and Practice*, pages 46–51, Athens, GA, USA, Jun 2007. CSREA Press. 68, 71, 73, 188
- [12] T. J. Allen. *Managing the Flow of Technology*. The MIT Press, Cambridge, MA, USA, Oct 1977. 16
- [13] F. Allilaire and T. Idrissi. ADT: Eclipse development tools for ATL. Technical Report 17-04, University of Kent, Sep 2004. 161
- [14] J. Alonso, J. Olmeda, and J. Rodriguez. Documentation Center Simplifying the Documentation of Software Projects. In *Online proceedings of the International Symposium on Wikis, September 8 - 10, Porto, Portugal*, Porto, Portugal, Sep 2008. http://www.wikisym.org/ws2008/index.php/Wikis4SE_-_Wikis_For_Software_Engineering. 72
- [15] S. W. Ambler, J. Nalbhone, and M. J. Vizdos. *The Enterprise Unified Process: Extending the Rational Unified Process*. Prentice Hall PTR, Upper Saddle River, NJ, USA, Feb 2005. 8, 9, 44, 45
- [16] E. P. Andersen and T. Reenskaug. System design by composing structures of interacting objects. In O. L. Madsen, editor, *ECOOP*, volume 615 of *Lecture Notes in Computer Science*, pages 133–152, London, UK, 1992. Springer-Verlag. 191
- [17] L. F. Andrade and J. L. Fiadeiro. Service-oriented business and system specification: Beyond object-orientation. In H. Kilov and editors K. Bacłwaski, editors, *Practical Foundations of Business System Specifications*, pages 1–23. Kluwer Academic Publishers, Norwell, MA, USA, 2003. 10, 143, 191

- [18] A. I. Antón. Goal-based requirements analysis. In C. Chang and C. Shekaran, editors, *Second International Conference on Requirements Engineering (ICRE 96)*, 15-18 Apr 1996, Colorado Springs, CO, USA, pages 136–144, Los Alamitos, CA, USA, Apr 1996. IEEE Computer Society. 77, 78, 80, 81, 82
- [19] A. I. Antón, W. M. McCracken, and C. Potts. Goal decomposition and scenario analysis in business process reengineering. In G. Wijers, S. Brinkkemper, and A. I. Wasserman, editors, *CAiSE*, volume 811 of *Lecture Notes in Computer Science*, pages 94–104. Springer, Jun 1994. 82
- [20] A. I. Antón and C. Potts. The use of goals to surface requirements for evolving systems. In K. Torii, K. Futatsugi, and R. Kemmerer, editors, *International Conference on Software Engineering (ICSE 98)*, pages 157–166, Los Alamitos, CA, USA, Apr 1998. IEEE Computer Society Press. 60
- [21] Apple Inc. *Code Loading Programming Topics for Cocoa*. Cupertino, CS, USA, Aug 2007. <http://developer.apple.com/documentation/Cocoa/Conceptual/LoadingCode/LoadingCode.html>. 144, 145
- [22] U. Apte, U. Karmarkar, and H. K. Nath. Information services in the U.S. economy: Values, jobs, and management implications. *California Management Review*, 50(3):12–30, Sep 2008. 4
- [23] A. Bechina Arntzen and A.-M. Krosgrud. Web-services architecture: The solution for e-government applications. In H. R. Arabnia, editor, *CSREA EEE*, pages 49–54, Las Vegas, Nevada, USA, Jun 2006. CSREA Press. 23
- [24] K. Baclawski, M. M. Kokar, P. A. Kogut, L. Hart, J. E. Smith, W. S. Holmes III, J. Letkowski, and M. L. Aronson. Extending UML to support ontology engineering for the Semantic Web. In Martin Gogolla and Cris Kobryn, editors, *UML*, volume 2185 of *Lecture Notes in Computer Science*, pages 342–360, Berlin / Heidelberg, 2001. Springer-Verlag. 21, 192
- [25] R. C. Basole and W. B. Rouse. Complexity of service value networks: conceptualization and empirical investigation. *IBM System Journal*, 47(1):53–70, 2008. 5, 6
- [26] M. Bass, V. Mikulovic, L. Bass, J. D. Herbsleb, and M. Cataldo. Architectural misalignment: An experience report. In R. Nord, N. Medvidovic, R. Krikhaar, J. Stafford, and J. Bosch, editors, *WICSA*, pages 1–17. IEEE Computer Society, 2007. 4
- [27] D. Batory. Multilevel models in model-driven engineering, product lines, and metaprogramming. *IBM System Journal*, 45(3):527–539, 2006. 98, 101

- [28] B. Bauer, F. Bergenti, P. Massonet, and J. Odell. Agents and the UML: A unified notation for agents and multi-agent systems? In Wooldridge et al. (168), pages 148–150. 87
- [29] A. Begel and N. Nagappan. Global software development: Who does it? In S. Sowmyanarayanan, F. Lanubile, and B. Sengupta, editors, *IEEE international conference on Global Software Engineering (ICGSE08) 17-20 Aug*, pages 195–199, Los Alamitos, CA, USA, Aug 2008. IEEE Computer Society. 4
- [30] B. Benatallah, R. Dijkman, M. Dumas, and Z. Maamar. Service Composition: Concepts, Techniques, Tools, and Trends. In Z. Stojanovic and A. Dahanayake, editors, *Service-Oriented Software System Engineering: Challenges and Practices*, chapter 3, pages 68–87. Idea Group Publishing, Hershey, PA, 2005. 21
- [31] K. H. Bennett, P. J. Layzell, D. Budgen, P. Brereton, L. A. Macaulay, and M. Munro. Service-based software: the future for flexible software. In D. Poo, J. S. Dong, J. He, and M. Purvis, editors, *APSEC*, pages 214–221. IEEE Computer Society, 2000. 7, 10
- [32] A. T. Berztiss and J. A. Bubenko. A software process model for business reengineering. In *Proceedings of Information Systems Development for Decentralized Organizations (ISDO95), an IFIP 8.1 Working Conference*, pages 184–200, Norwell, MA, USA, Aug 1995. Chapman & Hall - Kluwer Academic Publishers. 116
- [33] J. Bézivin. On the unification power of models. *Software and Systems Modeling*, 4(2):171–188, May 2005. 10, 19, 95, 97, 98
- [34] J. Bézivin, F. Jouault, P. Rosenthal, and P. Valduriez. Modeling in the large and modeling in the small. In U. Aßmann, M. Aksit, and A. Rensink, editors, *MDAFA*, volume 3599 of *Lecture Notes in Computer Science*, pages 33–46, Berlin Heidelberg, 2004. Springer-Verlag. 161
- [35] J. Bézivin, F. Jouault, and D. Touzet. Principles, standards and tools for model engineering. In *ICECCS*, pages 28–29, Los Alamitos, CA, USA, 2005. IEEE Computer Society. 101, 104
- [36] M. Boasson. The artistry of software architecture. *IEEE Software*, 12(6):13–16, Nov 1995. 4
- [37] A. Borgida, S. Greenspan, and J. Mylopoulos. Knowledge representation as the basis for requirements specifications. In C. Rich and R. C. Waters, editors, *Artificial Intelligence and Software Engineering*, pages 561–570, 1986. 76, 127

- [38] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Modeling early requirements in tropos: A transformation based approach. In Wooldridge et al. (168), pages 151–168. 79, 84, 90
- [39] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004. 84, 123
- [40] F. Bronsard, D. Bryan, W. Kozaczynski, E. S. Liongosari, J. Q. Ning, Á. Ólafsson, and J. W. Wetterstrand. Toward software plug-and-play. *SIGSOFT Software Engineering Notes*, 22(3):19–29, 1997. 144
- [41] F. Brooks. No Silver Bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, Apr 1987. 4
- [42] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose. *Eclipse Modeling Framework: A Developer's Guide*. E. Gamma, L. Nackman, and J. Wiegand. The Eclipse series. Addison-Wesley Professional, Boston, MA, USA, Aug 2003. 106
- [43] J. Champy. *X-engineering the corporation: Reinvent your business in the digital age*. Warner Business Books, Clayton VIC, Australia, Feb 2002. 1st edition. 8, 10
- [44] D. Chappell. *Enterprise Service Bus*. O'Reilly Media, Inc., Sebastopol, CA, USA, Jun 2004. 10, 138
- [45] T. Chau and F. Maurer. A case study of wiki-based experience repository at a medium-sized software company. In P. Clark and G. Schreiber, editors, *K-CAP*, pages 185–186, New York, NY, USA, Oct 2005. ACM. 181
- [46] H. Chesbrough and J. Spohrer. A research manifesto for services science. *Communication ACM*, 49(7):35–40, Jul 2006. Special Issue: Service Science. 6, 116
- [47] L. Chung, B. A. Nixon, and E. S. K. Yu. Using non-functional requirements to systematically support change. In *Second IEEE International Symposium on Requirements Engineering, March 27 - 29, 1995, York, England*, pages 132–139, Los Alamitos, CA, USA, 1995. IEEE Computer Society. 80
- [48] P. Ciancarini, V. Presutti, and L. Abeti. An ontology driven design method for inter-agent communication. In *SEKE*, pages 90–94, San Francisco, CA, Jul 2003. Knowledge Systems Institute. 1, 21, 135
- [49] A. Ciliberti. Changes in discursive practices in italian public administration. *Journal of Pragmatics*, 27(2):127–144, Feb 1997. 17, 24

- [50] T. Cleenewerck. Component-based DSL development. In F. Pfenning and Y. Smaragdakis, editors, *GPCE*, volume 2830 of *Lecture Notes in Computer Science*, pages 245–264, New York, NY, USA, 2003. Springer-Verlag. 101, 191, 192
- [51] V. Clerc, P. Lago, and H. van Vliet. Global software development: are architectural rules the answer? In F. Paulisch, P. Kruchten, and A. Mockus, editors, *Second IEEE International Conference on Global Software Engineering (ICGSE 2007) August 27-30*, pages 225–234, Washington, DC, USA, Aug 2007. IEEE Computer Society. 4
- [52] F. Correia. Extending and Integrating Wikis to Improve Software Documentation. In *Online proceedings of the International Symposium on Wikis, September 8 - 10, Porto, Portugal*, Porto, Portugal, Sep 2008. http://www.wikisym.org/ws2008/index.php/Wikis4SE-Wikis_For_Software_Engineering. 69, 188
- [53] S. Cranefield. UML and the Semantic Web. In I. F. Cruz, S.n Decker, J. Euzenat, and D. L. McGuinness, editors, *The Emerging Semantic Web*, volume 75 of *Frontiers in Artificial Intelligence and Applications*, pages 113–130, Amsterdam, The Netherlands, 2001. IOS press. 21
- [54] K. Czarnecki and U.W. Eisenecker. *Generative programming: methods, tools, and applications*. Addison-Wesley Publishing Co., New York, NY, USA, Jun 2000. 101
- [55] K. Czarnecki and S. Helsen. Classification of model transformation approaches. In R. Crocker and G. L. Steele Jr., editors, *OOPSLA*, pages 361–378, New York, NY, USA, 2005. ACM Press. 102, 104, 109
- [56] M. Dall’Agnol, A. Janes, G. Succi, and E. Zaninotto. Lean management-a metaphor for extreme programming? In M. Marchesi and G. Succi, editors, *XP*, volume 2675 of *Lecture Notes in Computer Science*, pages 26–32, Heidelberg, Germany, 2003. Springer. 39
- [57] D. Damian. Stakeholders in global requirements engineering: Lessons learned from practice. *IEEE Software*, 24(2):21–27, Apr 2007. x, 64, 65
- [58] D. Damian and D. Zowghi. Re challenges in multi-site software development organisations. *Requirements Engineering*, 8(3):149–160, Aug 2003. 3, 66, 67
- [59] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993. 78, 79, 80, 81

- [60] T. Davenport. *Process Innovation: Reengineering work through information technology*. Harvard Business School Press, Boston, 1993. 111, 116, 117
- [61] D. Lopez de Ipina, J. I. Vazquez, and J. Abaitua. A Web 2.0 platform to enable context-aware mobile mash-ups. In B. Schiele, A. K. Dey, H. Gellersen, B. E. R. de Ruyter, M. Tscheligi, R. Wichert, E. H. L. Aarts, and A. P. Buchmann, editors, *AmI*, volume 4794 of *Lecture Notes in Computer Science*, pages 266–286, Heidelberg, Germany, 2007. Springer. 143, 145
- [62] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth. Wiki-based stakeholder participation in requirements engineering. *IEEE Software*, 24(2):28–35, Mar-Apr 2007. 73, 74, 148, 149, 153, 187, 188, 189
- [63] C. Demmke, G. Hammerschmid, and R. Meyer. Decentralisation and accountability as a focus of public administration modernisation: Challenges and consequences for human resource management. Technical report, European Institute of Public Administration (EIPA), Maastricht, NL, Dec 2006. EIPA Code 2006/01 <http://www.eipa.eu/en/publications/show/&tid=1765>. 17, 24
- [64] F. DeRemer and H. Kron. Programming-in-the large versus programming-in-the-small. In *Proceedings of the international conference on Reliable software*, pages 114–121, New York, NY, USA, 1975. ACM Press. 143
- [65] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, Oct 1976. 190
- [66] B. Dobing and J. Parsons. How UML is used. *Communications of the ACM*, 49(5):109–113, May 2006. 124, 149, 154
- [67] P. F. Drucker. The coming of the new organization. *Harvard Business Review*, 1(88105):45–53, Jan 1988. 7, 36, 39
- [68] K. El Emam, S. Quintin, and N. H. Madhavji. User participation in the requirements engineering process: An empirical study. *Requirements Engineering*, 1(1):4–26, 1996. 73, 187
- [69] A. Fairchild. *Reengineering and Restructuring the Enterprise: A Management Guide for the 21st Century*. Computer Technology Research, Charleston, SC, USA, Jan 1998. 1st edition. 8, 10
- [70] J. L. Fiadeiro and T. Maibaum. A mathematical toolbox for the software architect. In *Software Specification and Design, 1996., Proceedings of the 8th International Workshop on*, pages 46–55, Washington, DC, USA, Mar 1996. IEEE Computer Society Press. 143

- [71] P. Fingar. Component-based frameworks for e-commerce. *Communications of ACM*, 43(10):61–67, Oct 2000. 10
- [72] I. T. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, 2002. 18, 19
- [73] D. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. T. Hudson and J. H. Russel. John Wiley & Sons, Inc., Indianapolis, IN, USA, Jan 2003. 22, 94, 95, 96, 100, 101
- [74] J. Garnett. *Handbook of Administrative Communication*, volume 63 of *Public Administration and Public Policy*. CRC Press, Broken Sound Parkway, NW, USA, Mar 1997. 1th edition. 16, 17, 24
- [75] D. Gelernter and N. Carriero. Coordination languages and their significance. *Communications ACM*, 35(2):97–107, 1992. 143
- [76] C. Ghezzi. Flexible processes for evolvable products. In *IEEE METRICS*, page 1, Washington, DC, USA, Sep 2005. IEEE Computer Society. 21
- [77] C. Ghezzi. Software engineering: Emerging goals and lasting problems. In Baresi L and Heckel R, editors, *FASE*, volume 3922 of *Lecture Notes in Computer Science*. Springer, 2006. Fundamental Approaches to Software Engineering, FASE 2006, Vienna, Austria, March 27–28, 2006. 3, 21, 94, 116, 118
- [78] P. Giorgini, M. Kolp, J. Mylopoulos, and A. Perini. Multi-agent and software architectures: A comparative case study. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *AOSE*, volume 2585 of *Lecture Notes in Computer Science*, pages 101–112, Berlin / Heidelberg, 2002. Springer-Verlag. 6, 10, 84
- [79] P. Giorgini, M. Kolp, J. Mylopoulos, and M. Pistore. The Tropos methodology: an overview. In M.-P. Gleizes F. Bergenti and F. Zambonelli (Eds), editors, *In Methodologies And Software Engineering For Agent Systems*, chapter 5, pages 89–105. Kluwer Academic Publishing, Norwell, MA, USA, Dec 2004. 80, 84, 124, 125, 128, 148, 149, 151
- [80] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Requirements engineering for trust management: model, methodology, and reasoning. *International Journal of Information Security*, 5(4):257–274, 2006. 91, 92
- [81] P. Giorgini, F. Massacci, and N. Zannone. Security and trust requirements engineering. In A. Aldini, R. Gorrieri, and F. Martinelli, editors, *FOSAD*, volume 3655 of *Lecture Notes in Computer Science*, pages 237–272, Berlin Heidelberg, 2005. Springer-Verlag. 90

- [82] J. Gordijn, H. Akkermans, and H. van Vliet. Business modelling is not process modelling. In S. W. Liddle, H. C. Mayr, and B. Thalheim, editors, *ER (Workshops)*, volume 1921 of *Lecture Notes in Computer Science*, pages 40–51, Heidelberg, Germany, 2000. Springer-Verlag. 8, 48, 142
- [83] M. Hammer. Reengineering work: Don't automate, obliterate. *Harvard Business Review*, 68(4):104–112, Jul 1990. 7, 9, 10, 35, 36, 115, 117
- [84] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperCollins Publishers, New York, NY, USA, May 1994. 37, 136
- [85] P. Haumer, K. Pohl, and K. Weidenhaupt. Requirements elicitation and validation with real world scenes. *IEEE Transactions on Software Engineering*, 24(12):1036–1054, Dec 1998. 80
- [86] M. Havey. *Essential Business Process Modeling*. Theory in practice. O Reilly, Sebastopol, CA, USA, Aug 2005. x, 9, 49, 50, 139, 142
- [87] J. Hay. *Requirements Analysis From Business Views to Architecture*. Prentice Hall PTR, Englewood Cliffs, NJ, USA, Sep 2002. 63
- [88] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions in Software Engineering*, 29(6):481–494, Jan 2003. 3, 4
- [89] J. D. Herbsleb and D. Moitra. Guest editors' introduction: Global software development. *IEEE Software*, 18(2):16–20, Mar 2001. 3, 16, 67
- [90] T. P. Hill. On goods and services. *The Review of Income and Wealth*, 23(4):314–339, 1977. 6
- [91] H. Holmstrom, E. O. Conchuir, P.J. Agerfalk, and B. Fitzgerald. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In F. Paulisch, P. Kruchten, and A. Mockus, editors, *IEEE international conference on Global Software Engineering (ICGSE06)*, pages 3–11, Washington, DC, USA, Oct 2006. IEEE Computer Society. 3
- [92] S. Holwell and P. Checkland. An information system won the war. *IEE Proceedings - Software*, 145(4):95–99, Aug 1998. 77
- [93] E. Hull, K. Jackson, and J. Dick. *Requirements Engineering*. Springer-Verlag, Amsterdam, The Netherlands, 2nd edition, Dec 2005. 8, 60, 61, 62, 63
- [94] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In R. Nord, N. Medvidovic, R. Krikhaar, J. Stafford, and J. Bosch, editors, *WICSA*, pages 109–120. IEEE Computer Society, 2005. 4

- [95] F. Jouault and I. Kurtev. Transforming models with ATL. In J.-M. Bruel, editor, *MoDELS Satellite Events*, volume 3844 of *Lecture Notes in Computer Science*, pages 128–138, Berlin Heidelberg, Oct 2005. Springer-Verlag. 161
- [96] H. Kaindl. Combining goals and functional requirements in a scenario-based design process. In H. Johnson, L. Nigay, and C. Roast, editors, *BCS HCI*, pages 101–121. Springer-Verlag Telos, Sep 1998. 80, 88
- [97] N. I. Karacapilidis and D. Papadias. Computer supported argumentation and collaborative decision making: the HERMES system. *Information Systems*, 26(4):259–277, Jun 2001. 121, 188
- [98] D.W. Karolak. *Global Software Development: Managing Virtual Teams and Environments*. Matt Loeb Wiley-IEEE Computer Society Pr, Los Alamitos, CA, USA, Dec 1998. 67
- [99] E. Kavakli and P. Loucopoulos. Goal modeling in requirements engineering: Analysis and critique of current methods. In J. Krogstie, T. A. Halpin, and K. Siau, editors, *Information Modeling Methods and Methodologies*, pages 102–124. IGI Global, Hershey, PA, USA, 2005. 8, 9, 89
- [100] M. Keen, S. Bishop, A. Hopkins, S. Milinski, C. Nott, R. Robinson, J. Adams, and P. Verschueren; A. Acharya. *Patterns: Implementing SOA using an Enterprise Service Bus*. IBM Redbooks, New York, NY, USA, Jul 2005. 139
- [101] W. Kent. Object-Orientation and interoperability. In Manfred Broy, editor, *NATO ASI OODBS*, pages 287–305, 1993. 95
- [102] M. Kolp, P. Giorgini, and J. Mylopoulos. Organizational patterns for early requirements analysis. In J. Eder and M. Missikoff, editors, *CAiSE*, volume 2681 of *Lecture Notes in Computer Science*, pages 617–632, New York, NY, USA, Jun 2003. Springer-Verlag. 8, 35, 79, 84, 127, 128, 187
- [103] M. Kontio. Architectural manifesto: Choosing MDA tools. Technical report, IBM developerWorks, Sep 2005. <http://www.ibm.com/developerworks/webservices/library/wi-arch18.html>. 104
- [104] R. E. Kraut, C. Egidio, and J. Galegher. *Intellectual Teamwork: Social Foundations of Cooperative Work*, chapter Patterns of Contact and Communication in Scientific Research Collaborations, page 552. Lawrence Erlbaum Associated, Hillsdale, NJ, USA, May 1990. 16
- [105] P. Kruchten. *The Rational Unified Process: An Introduction*. The Addison-Wesley Object Technology Series. Addison-Wesley Longman Publishing, Boston, MA, USA, 3rd edition, Dec 2003. 8, 41, 49, 124

- [106] J. A. Laredo and R. Ranjan. Continuous improvement through iterative development in a multi-geography. In S. Sowmyanarayanan, F. Lanubile, and B. Sengupta, editors, *International Conference on Global Software Engineering*, pages 232–236, Los Alamitos, CA, USA, Aug 2008. IEEE Computer Society. 4
- [107] S. Lauesen. *Software requirements - styles and techniques*. Addison-Wesley Professional, Boston, MA, USA, Jan 2002. 60, 61
- [108] S. Lauesen and O. Vinter. Preventing requirement defects: An experiment in process improvement. *Requirements Engineering Journal*, 6(1):37–50, 2001. 60
- [109] B. Leuf and W. Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison Wesley Longman Publishing, Boston, MA, USA, pap/cdr edition edition, Apr 2001. 68
- [110] T. Levitt. Production-line approach to service. *Harvard Business Review*, 50(5):4152, Sep 1972. 5, 12
- [111] F. Leymann. The (service) bus: Services penetrate everyday life. In B. Benatallah, F. Casati, and P. Traverso, editors, *ICSOC*, volume 3826 of *Lecture Notes in Computer Science*, pages 12–20, Heidelberg, Germany, 2005. Springer-Verlag. 10, 139
- [112] R. Likert. A technique for the measurement of attitude. *Archives of Psychology*, 22(140):55, 1932. 180
- [113] L. Liu and E. Yu. Designing information systems in social context: a goal and scenario modelling approach. *Information Systems*, 29(2):187–203, Apr 2004. Special issue: The 14th international conference on advanced information systems engineering (CAiSE*02). 80
- [114] S. Lohmann and P. Heim. Semantifying requirements engineering the soft-wiki approach. In S. Auer, S. Schaffert, and T. Pellegrini, editors, *Proceedings of the 4th International Conference on Semantic Technologies (I-SEMANTICS '08)*, TRIPLE-I, page 182185, Sep 2008. 73, 75, 188
- [115] A. Majchrzak, C. Wagner, and D. Yates. Corporate wiki users: results of a survey. In D. Riehle and J. Noble, editors, *International Symposium on Wikis*, pages 99–104. ACM, Aug 2006. 181
- [116] D.-A. Manolescu and B. Lublinsky. Service orchestration patterns: graduating from state of the practice to state of the art. In Ralph E. Johnson and Richard P. Gabriel, editors, *OOPSLA Companion*, pages 148–149. ACM, 2005. 140, 141

- [117] F. Massacci, J. Mylopoulos, and N. Zannone. An ontology for secure socio-technical systems. In IGI Global, editor, *Handbook of Ontologies for Business Interaction*, volume 1, page 469. Information Science Reference, Hershey, PA, USA, Dec 2007. 90, 121
- [118] M. R. McNeilly. *Sun Tzu and the Art of Business: Six Strategic Principles for Managers*. Oxford University Press, New York, NY, USA, Apr 2000. 35
- [119] S. J. Mellor, K. Scott, A. Uhl, and D. Weise. *MDA Distilled: Principles of Model-Driven Architecture*. G. Booch, I. Jacobson, and J. Rumbaugh. The Addison-Wesley Object Technology Series. Addison-Wesley Professional, Boston, MA, USA, Mar 2004. 17, 96, 98
- [120] D. Le Métayer. Software architecture styles as graph grammars. *SIGSOFT Software Engineering Notes*, 21(6):15–23, 1996. 143
- [121] B. Meyer. On formalism in specifications. *IEEE Software*, 2(1):6–26, Jan 1985. 62
- [122] S. Mokhtar and H. Harudin. Interoperability in e-Government: Adopting the service oriented architecture (SOA) framework for a transparent malaysian public delivery system. In G. Kotsis, D. Taniar, E. Pardede, and I. K. Ibrahim, editors, *iiWAS*, volume 229, pages 463–469. Austrian Computer Society, Dec 2007. 23
- [123] J. Morecroft. Mental models and learning in system dynamics practice. In Michael (Ed.) in Pidd, editor, *Systems Modelling: Theory and Practice*, chapter 7, pages 101–26. John Wiley & Sons, Hoboken, NJ, USA, Feb 2004. 120
- [124] N. Mullick, M. Bass, Z. Houda, Paulish Paulish, and M. Cataldo. Siemens global studio project: Experiences adopting an integrated GSD infrastructure. In F. Paulisch, P. Kruchten, and A. Mockus, editors, *IEEE international conference on Global Software Engineering (ICGSE06)*, pages 203–212, Washington, DC, USA, Oct 2006. IEEE Computer Society. 4
- [125] J. Mylopoulos, L. Chung, and B. A. Nixon. Representing and using non-functional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, Jun 1992. 80, 88
- [126] J. Mylopoulos, L. Chung, and E. S. K. Yu. From Object-Oriented to Goal-Oriented requirements analysis. *Communications of ACM*, 42(1):31–37, Jan 1999. 39, 49, 76, 79, 80, 88
- [127] Z. Nanping and L. Yuan. Study and application of the SOA based e-Government system. In X. Wang, H. Hu, P. Liu, and X. Cao, editors, *International Conference on Information Management, Innovation Management and*

Industrial Engineering (ICIII 2008), pages 476–479, Los Alamitos, CA, USA, Dec 2008. IEEE Computer Society Press. 23

- [128] E. Newcomer and G. Lomow. *Understanding SOA with Web Services*. Addison-Wesley Professional, Boston, MA, USA, Dec 2004. 23
- [129] U. Nikula, J. Sajaniemi, and H. Kalviainen. A state-of-the-practice survey on requirements engineering in small- and medium-sized enterprises. Technical Report Research Report 1, Telecom Business Research Center, Lappeenranta University of Technology, Lappeenranta, Finland, 2000. <http://www.lut.fi/TBRC/>. 148
- [130] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company*. Oxford University Press, Virgin Islands, USA, May 1995. 1st edition. 6
- [131] B. Nuseibeh and S. M. Easterbrook. Requirements engineering: a roadmap. In *22nd International Conference on Software Engineering the Future of Software Engineering Track, June 4-11*, pages 35–46, New York, NY, USA, Jun 2000. ACM. 60, 62, 63
- [132] T. Ohno. *Toyota Production System: Beyond Large-Scale Production*. Productivity Press, New York, NY, USA, Mar 1988. 39
- [133] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the art and research challenges. *Computer*, 40(11):38–45, Nov 2007. 139
- [134] M. P. Papazoglou and J. Yang. Design methodology for Web Services and Business Processes. In A. P. Buchmann, F. Casati, L. Fiege, M. Hsu, and M. Shan, editors, *TES*, volume 2444 of *Lecture Notes in Computer Science*, pages 54–64, London, UK, Aug 2002. Springer-Verlag. 135
- [135] L. D. Paulson. Services science: A new field for today’s economy. *IEEE Computer*, 39(8):18–21, Aug 2006. 4, 7, 117, 118
- [136] A. Perini, M. Pistore, M. Roveri, and A. Susi. Agent-oriented modeling by interleaving formal and informal specification. In *AOSE*, pages 36–52, 2003. 127
- [137] K. Pohl. Softwiki: User-oriented, distributed requirements engineering for evolutionary development processes, Jul 2008. <http://www.sse.uni-due.de/wms/en/index.php?go=236#web>. 73, 75, 188
- [138] N. Prakash, S. Srivastava, and S. Sabharwal. The classification framework for model transformation. *Journal of Computer Science*, 2(2):166–170, Feb 2006. 102

- [139] T. Puschmann and R. Alt. Enterprise application integration systems and architecture - the case of the Robert Bosch Group. *Journal of Enterprise Information Management*, 17(2):105–116, Feb 2004. 23, 24
- [140] E. Ras. Investigating wikis for software engineering - results of two case studies. In A. Aguiar, U. Dekel, and P. Merson, editors, *ICSE Companion Wikis4SE Workshop Proceedings*, volume CFP0935G, pages 47–56, Los Alamitos, CA, USA, May 2009. IEEE Computer Society. 68, 69, 73, 181, 188
- [141] G. Regev and A. Wegmann. Defining early it system requirements with regulation principles: The lightswitch approach. In M. Glinz, editor, *RE*, pages 144–153. IEEE Computer Society, 2004. 8, 10
- [142] I. Richardson, G. Avram, S. Deshpande, and V. Casey. Having a foot on each shore - bridging global software development in the case of SMEs. In S. Sowmyanarayanan, F. Lanubile, and B. Sengupta, editors, *IEEE international conference on Global Software Engineering (ICGSE08) 17-20 Aug*, pages 13–22, Los Alamitos, CA, USA, Aug 2008. IEEE Computer Society. 16
- [143] W. N. Robinson. Integrating multiple specifications using domain goals. *ACM SIGSOFT Software Engineering Notes*, 14(3):219–226, 1989. 78
- [144] D. De Roure, N. R. Jennings, and N. R. Shadbolt. Research Agenda for the Semantic Grid: A Future e-Science Infrastructure. Technical Report UKeS-2002-02, National e-Science Centre, Dec 2001. Report for EPSRC/DTI e-Science Core Programme <http://eprints.ecs.soton.ac.uk/6350/1/semgrid.pdf>. 18
- [145] P. Sawyer, I. Sommerville, and S. Viller. Capturing the benefits of requirements engineering. *IEEE Software*, 16(2):78–85, Apr 1999. 8
- [146] S. Si-Said and C. Rolland. Formalising guidance for the crews goal-scenario approach to requirements engineering. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *8th European-Japanese Conferences on Information Modelling and Knowledge Bases (EJC 98), May 26-29, 1998, Vammala, Finland, Information Modelling and Knowledge Bases*, pages 172–190, Amsterdam, The Netherlands, 1998. IOS Press. 80
- [147] D. Smite. Requirements management in distributed projects. *Journal of Universal Knowledge Management*, 1(2):69–76, May 2006. xii, 60, 64, 66, 112
- [148] A. Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. University Of Chicago Press, Chicago, IL, USA, Feb 1977. First Published 1776. 36

- [149] P. Sochos, I. Philippow, and M. Riebisch. Feature-oriented development of software product lines: Mapping feature models to the architecture. *Object-Oriented and Internet-Based Technologies*, 3263/2004:138–152, 0302-9743 2004. 190, 191
- [150] J. F. Sowa and J. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616, 1992. 40, 116
- [151] J. Spohrer. The opportunities and challenges of doing business in today's global services economy. <http://www.services-science.de/Downloads.html>, Apr 2006. 5
- [152] A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini. The Tropos Metamodel and its use. *Informatica (Slovenia)*, 29(4):401–408, 2005. 84, 85
- [153] J. M. Tien and D. Berg. A case for service system engineering. *Journal of Systems Science and Systems Engineering*, 12(1):13–38, Mar 2003. 5
- [154] S.E. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, Jan 1985. 153
- [155] W. T. Tsai. Service-oriented system engineering: A new paradigm. In *Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop*, volume 0, pages 3–8, Washington, DC, USA, Oct 2005. IEEE Computer Society. 2, 7, 22, 135
- [156] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske, editors, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12, Berlin / Heidelberg, Jun 2003. Springer-Verlag. 8, 111
- [157] A. van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *22nd International Conference on Software Engineering*, June 4–11, pages 5–19, New York, NY, USA, Jun 2000. ACM. 60, 61, 77
- [158] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In S. Easterbrook and B. Nuseibeh, editors, *RE*, pages 249–262, Los Alamitos, CA, USA, 2001. IEEE Computer Society. 77, 78
- [159] A. van Lamsweerde, R. Darimont, and P. Massonet. Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt. In *Second IEEE International Symposium on Requirements Engineering*, March 27 - 29, 1995, York, England, pages 194–203, Los Alamitos, CA, USA, 1995. IEEE Computer Society. 76, 79, 81

- [160] S. L. Vargo and R. F. Lusch. The four service marketing myths: Remnants of a goods-based, manufacturing model. *Journal of Service Research*, 6(4):324-335, May 2004. 5
- [161] S. Weber, L. Thomas, O. Armbrust, E. Ras, J. Rech, O. Uenalan, M. Wessner, M. Linnenfelser, and B. Decker. A software organization platform (SOP). In A. Jedlitschka and O. Salo, editors, *10th international Workshop on: Learning Software Organizations Methods, Tools, and Experiences*, (LSO 2008), *Product-Focused Software Process Improvement*, volume 5089 of *Lecture Notes in Computer Science*, pages 421–443, Berlin Heidelberg, 2008. Springer-Verlag. 73, 188
- [162] M. Weske. *Business Process Management Concepts, Languages, Architectures*. Springer-Verlag, Amsterdam, The Netherlands, Nov 2007. 46
- [163] R. Wieringa. Postmodern software design with NYAM: Not Yet Another Method. In Manfred Broy and Bernhard Rumpe, editors, *Requirements Targeting Software and Systems Engineering*, volume 1526 of *Lecture Notes in Computer Science*, pages 69–94. Springer, 1997. 8
- [164] K.M. van Hee W.M.P. van der Aalst. *Workflow Management: Models, Methods and Systems*. The MIT Press, Cambridge, MA, USA, Jan 2002. 1st edition. 9, 49
- [165] A. Wolfl. The service economy in OECD countries. STI Working Paper 3, Organisation for Economic Co-Operation and Development, 3 2005. <http://www.cepii.fr/anglaisgraph/pagepers/wolfl.htm>. 4
- [166] M. Wooldridge. *An Introduction to Multiagent Systems*, chapter 1, page 320. John Wiley& Sons, Chichester, England, Feb 2002. 79, 85
- [167] M. Wooldridge and P. Ciancarini. Agent-oriented software engineering: The state of the art. In Paolo Ciancarini and Michael Wooldridge, editors, *AOSE*, volume 1957 of *Lecture Notes in Computer Science*, pages 1–28, Heidelberg, Germany, 2000. Springer-Verlag. 85
- [168] M. Wooldridge, G. Weiß, and P. Ciancarini, editors. *Agent-Oriented Software Engineering II, Second International Workshop, AOSE 2001, Montreal, Canada, May 29, 2001, Revised Papers and Invited Contributions*, volume 2222 of *Lecture Notes in Computer Science*, Heidelberg, Germany, 2002. Springer-Verlag. 196, 197
- [169] W. Xiao, C. Yan Chi, and M. Yang. On-line collaborative software development via wiki. In A. Désilets and R. Biddle, editors, *International Symposium on Wikis*, pages 177–183, New York, NY, USA, 2007. ACM. 69

- [170] Y. Yang, F. Xia, W. Zhang, X. Xiao, Y. Li, and X. Li. Towards semantic requirements engineering. In P. Sheu and H. Yu ed., editors, *WSCS '08: Proceedings of the IEEE International Workshop on Semantic Computing and Systems, 14-15 July 2008*, pages 67–71, Washington, DC, USA, Jul 2008. IEEE Computer Society. 75
- [171] R. Young. *The Requirements Engineering Handbook*. Artech House, Norwood, MA, USA, Nov 2003. 8, 59, 63, 76, 126, 184
- [172] E. Yu. Modeling organizations for information systems requirements engineering. In M. Jarke S. Jacobs and K. Pohl, editors, *Proceedings of the First IEEE International Symposium on Requirements Engineering (RE'93) San Diego, Jan. 46, 1993*, pages 34–41, Los Alamitos, CA, USA, Jan 1993. IEEE Computer Society. 9, 39, 49, 83
- [173] E. S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In C. Heitmeyer and J. Mylopoulos, editors, *3rd IEEE International Symposium on Requirements Engineering (RE'97), January 5-8, 1997, Annapolis, MD, USA*, pages 226–235, Los Alamitos, CA, USA, Jan 1997. IEEE Computer Society. 76, 83
- [174] E. S. K. Yu and J. Mylopoulos. Using goals, rules, and methods to support reasoning in business process re-engineering. In S. Trevor, B. El-Rewini, N. Hesham, S. Jay, R. Hunter, and L. Mudge, editors, *27th Annual Hawaii International Conference on System Sciences (HICSS-27), January 4-7, 1994, Maui, Hawaii.*, volume 4 of *Information Systems: Collaboration Technology, Organizational Systems and Technology*, pages 234–243, Los Alamitos, CA, USA, 1994. IEEE Computer Society. 39, 82
- [175] E. S. K. Yu and J. Mylopoulos. From E-R to A-R - modelling strategic actor relationships for business process reengineering. In P. Loucopoulos, editor, *ER*, volume 881 of *Lecture Notes in Computer Science*, pages 548–565, Berlin / Heidelberg, 1994. Springer-Verlag. 83
- [176] J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 38(2/3):454–470, 1999. 39, 41
- [177] P. Zave. Classification of research efforts in requirements engineering. *ACM Computing Surveys*, 29(4):315–321, Dec 1997. 60
- [178] V. A. Zeithaml, A. Parasuraman, and L. L. Berry. Problems and strategies in services marketing. *Journal of Marketing*, 49(2):3346, Aug 1985. 6, 7

URL's

- [Ado09] Adobe. Adobe flex 3, 2009. <http://www.adobe.com/it/products/flex/>, Retrieved December 2009.
- [aI06] alphaWorks IBM. The Emerging Technologies Toolkit (ETTK) Project, 2006. <http://alphaworks.ibm.com/ettk>.
- [Ame94] America On Line. The netscape archive, 1994. <http://browser.netscape.com/>, Retrieved December 2009.
- [ATL08] ATLAS. Atlas: Complex data management in distributed systems, Nov 2008. <http://www.inria.fr/recherche/equipes/atlas.fr.html>.
- [BEA03] BEA Systems and International Business Machines Corporation and Microsoft Corporation. Business Process Execution Language for Web Services (BPEL4WS) specifications, May 2003. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp>.
- [BL98] T. Berners-Lee. Semantic Web Road map, Sep 1998. <http://www.w3.org/DesignIssues/Semantic.html> Draft. Plan based in discussions with the W3C team, and various W3C member companies.
- [Cen08] Centro per la ricerca scientifica e tecnologica (ITC-irst) di Trento. The Tool for Agent Oriented Modeling for Eclipse (TAOM4E), Jul 2008. <http://sra.itc.it/tools/taom4e/>.
- [CFF04] K. Czsjkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, D. Snelling, and S. Tueke. Modeling Stateful Resources with Web Services. version 1.1, Mar 2004. <http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf>.
- [Com08] Compuware. Optimalj, Jan 2008. http://www.compuware.com/press-room/news/2001/1268_ENG_HTML.htm.

- [Cun96] Cunningham & Cunningham, Inc. Portland pattern repository, 1996. <http://c2.com/ppr/>, Retrieved December 2009.
- [Dum99] E. Dumbill. Examining commercenet's eco framework, Oct 1999. <http://www.xml.com/pub/a/1999/10/eco/index.html>.
- [Edg03] Edgewall Software. The trac project integrated scm & project management, 2003. <http://trac.edgewall.org/>, Retrieved December 2009.
- [Eur02] European Commission. Environment directorate general - community mechanism for civil protection, 2002. <http://ec.europa.eu/environment/civil/prote/mechanism.htm>, Retrieved December 2009.
- [H. 08] H. J. Happel. Teamweaver wiki, 2008. <http://www.teamweaver.org/>, Retrieved December 2009.
- [IBM97] IBM. WebSphere software, 1997. <http://www-306.ibm.com/software/websphere/>.
- [IBM08] IBM. Rational rose product line, Nov 2008. <http://www-01.ibm.com/software/awdtools/developer/rose/>.
- [INE04] INET Inc. Tsunami relief information, 2004. <http://www.inet.co.th/tsunami/>, Retrieved December 2009.
- [ITU09a] ITU. International telecommunication union web site, 2009. <http://www.itu.int/net/about/index.aspx>.
- [ITU09b] ITU. URN goal-oriented requirement language specifications draft z.151 standard, 2009. <http://jucmnav.softwareengineering.ca/twiki/bin/view/UCM/DraftZ151Standard>.
- [K. 05] K. Czsjkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, D. Snelling and S. Tueke. The WS-Resource Framework, Mar 2005. <http://www.globus.org/wsrf/specs/ws-wsrf.pdf>.
- [M. 05] M. Gibson. Wikidoc, 2005. <http://www.wikidoc.org/>, Retrieved December 2009.
- [Mar07] Regione Marche. The IDentification for Emergency Administration (IDEA) project, Aug 2007. www.protezionecivile.marche.it/viewdoc.asp?co_id=522.
- [Mic02] Sun Microsystems. Java metadata interface (jmi) specification, 2002. <http://jcp.org/aboutJava/communityprocess/final/jsr040/index.html>.

- [Mic05] Microsoft EMEA Press Center. Q&a: Microsoft hosts sme day to foster support and dialogue for europe's small and medium-sized enterprises, 2005. <http://www.microsoft.com/emea/presscentre/pressreleases/SMEDayQA.06062006.msp>, Retrieved December 2009.
- [Mic08] Microsoft. Visual studio 2008 express edition, Jan 2008. <http://www.microsoft.com/express/default.aspx>.
- [Nat05] National Oceanic & Atmospheric Administration (NOAA). Hurricane katrina - most destructive hurricane ever to strike the U.S., 2005. <http://www.katrina.noaa.gov/>, Retrieved December 2009.
- [OAS06a] OASIS. Reference Model for Service Oriented Architecture v 1.0, July 2006. Official Committee Specification - <http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf>.
- [OAS06b] OASIS. Web services distributed management (wsdm) version 1.1, 2006. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm.
- [OAS06c] OASIS. Web Services Resource Properties, 2006. Working Draft - <http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceProperties-1.2-draft-04.pdf>.
- [OAS06d] OASIS. The Web Services Resource Properties (WSRF) specifications, Apr 2006. Standard <http://www.oasis-open.org/committees/wsr/>.
- [Obj01] Object Management Group (OMG). MDA guide version 1.0.1, Jun 2001. <http://www.omg.org/cgi-bin/doc?ormsc/05-04-01>.
- [OMG01] OMG. Model Driven Architecture (MDA) architecture board, Jul 2001. <http://www.omg.org/cgi-bin/doc?ormsc/05-04-01>.
- [OMG02a] OMG. Meta-object facility MOF versione 1.4, Apr 2002. <http://www.omg.org/technology/documents/formal/mof.htm>.
- [OMG02b] OMG. MOF 2.0 Query / Views / Transformations RFP, Feb 2002. <http://www.omg.org/docs/ad/02-01-06.pdf>.
- [OMG03a] OMG. Ontology Definition Metamodel - request for proposal, Jun 2003. Document ad/2003-03-40.
- [OMG03b] OMG. Uml 2.0 ocl specifications, Nov 2003. <http://www.omg.org/docs/ptc/03-10-14.pdf>.
- [OMG06] OMG. Business Process Modeling Notation (BPMN) 1.0 adopted specification, Feb 2006. <http://www.bpmn.org/Documents>.

- [OMG07a] OMG. Unified Modeling Language specification v.2.1.2, November 2007. http://www.omg.org/technology/documents/modeling_spec_catalog.htm.
- [OMG07b] OMG. XMLMetadata Interchange (XMI) version 2.1.1, Dec 2007. <http://www.omg.org/cgi-bin/doc?formal/2007-12-01>.
- [Omo07] Omondo. Omondo uml eclipse plug-in free edition, Dec 2007. <http://www.eclipsedownload.com/download.free.eclipse.3.3.html>.
- [Pap08] Papyrus. Papyrus uml editor, Nov 2008. <http://www.papyrusuml.org/scripts/home/publigen/content/templates/show.asp?P=55&L=EN&ITEMID=2>.
- [Pre09] Presidenza del Consiglio dei Ministri - Dipartimento della Protezione Civile. Grandi eventi ed ordinanze g8 2009, 2009. http://www.protezionecivile.it/legislazione/ordinanze_dettaglio.php?id=1403, Retrieved December 2009.
- [R. 08] R. C. Martin and M. Martin and P. Wilson-Welsh. Fitnessse, 2008. <http://fitnessse.org/>, Retrieved December 2009.
- [Sol09] C. Soltenborn. Meta uml, Jul 2009. <http://metauml.sourceforge.net/old/index.html>, Retrieved December 2009.
- [Sta00] Standish Group. Chaos report 2000. Technical report, The Standish Group Report, chaos2000 2000. <http://www.cs.nmt.edu/cs328/reading/Standish.pdf>.
- [Sun02] Sun Microsystems. The java metadata interface (JMI) jsr-000040 specifications - final release, Jun 2002. <http://java.sun.com/products/jmi/index.jsp>.
- [Sun08] Sun Microsystems. Java Web Service Development Pack (JWSDP) 2, 2008. <http://java.sun.com/webservices/downloads/previous/index.jsp>.
- [Tha01] S. Thatte. Xlang: Web services for business process design, Oct 2001. http://www.gotdotnet.com/team/xml_wsspecs/clang-c/default.htm.
- [The98] The Free Software Foundation Inc. Concurrent versions system, 1998. <http://www.nongnu.org/cvs/>, Retrieved December 2009.
- [The03] The Eclipse Funtation. Java Emitter Template, 2003. http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html, Retrieved December 2009.

- [The05a] The Eclipse Funtation. Graphical Modeling Framework (GMF), 2005. <http://www.eclipse.org/modeling/gmf/>, Retrieved December 2009.
- [The05b] The Eclipse Funtation. Mofscript, 2005. <http://www.eclipse.org/gmt/mofscript/about.php>, Retrieved December 2009.
- [The08a] The Eclipse Funtation. The Eclipse Platform, Jul 2008. <http://www.eclipse.org/>.
- [The08b] The Eclipse Funtation. Eclipse Modeling Framework (EMF), Jul 2008. <http://www.eclipse.org/emf/>.
- [The08c] The Eclipse Funtation. Eclipse modeling tools package, Jun 2008. <http://www.eclipse.org/downloads/packages/eclipse-modeling-tools-includes-incubating-components/ganymedesr1>.
- [The09a] The ATHENA Consortium. Athena interoperability framework (aif), Apr 2009. <http://modelbased.net/aif/>.
- [The09b] The Marche Region . Civil protection department, Jul 2009. <http://www.protezionecivile.marche.it>, Retrieved December 2009.
- [TOC08] TOCAI.it. Tecnologie orientate alla conoscenza per aggregazioni di imprese, Oct 2008. <http://www.dis.uniroma1.it/focai/>.
- [Top08] Topcased. Topcased uml editor, Nov 2008. <http://topcased.gforge.enseeiht.fr/index.php>.
- [Uni02] University of Trento. The tropos modeling language. a user guide., 2002. <http://eprints.biblio.unitn.it/archive/00000208/01/61.pdf>.
- [W3C99a] W3C. Xml path language (xpath) version 1.0, nov 1999. <http://www.w3.org/TR/xpath>.
- [W3C99b] W3C. Xsl transformations (xslt) version 1.0, Nov 1999. <http://www.w3.org/TR/xslt>.
- [W3C01] W3C. Web Services Description Language (WSDL) specifications, 2001. W3C Note - <http://www.w3.org/TR/wsdl>.
- [W3C02a] W3C. OWL. Web Ontology Language, reference version 1.0, 2002. <http://www.w3.org/TR/owl-ref/>.
- [W3C02b] W3C. Web services conversation language (WSCL) 1.0, Mar 2002. W3C Note - <http://www.w3.org/TR/wscl10/>.

- [W3C04a] W3C. Web services architecture (WSA), Feb 2004. W3C Working Group Note - <http://www.w3.org/TR/ws-arch/>.
- [W3C04b] W3C. Web services choreography description language version 1.0, Apr 2004. <http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/>.
- [W3C05] W3C. Web Services Semantics (WSDL-S), 2005. W3C Member Submission - <http://www.w3.org/Submission/WSDL-S/>.
- [War09] Ward Cunningham. Interview given to wiki4se @ icse2009 workshop, 2009. http://www.youtube.com/watch?v=I.75NoC85TE&feature=player_embedded, Retrieved December 2009.
- [Wik07] Wikimedia Foundation. Semantic Media Wiki project, Jul 2007. http://meta.wikimedia.org/wiki/Semantic_MediaWiki.
- [Wik08] Wikimedia Foundation. Mediawiki Extensions, Jul 2008. http://www.mediawiki.org/wiki/Extension_Matrix.
- [Wik09] Wikimedia Foundation Inc. Wikipedia web site, 2009. <http://www.wikipedia.org/>, Retrieved December 2009.
- [Wor08] World Wide Web Consortium (W3C). Resource Description Framework (RDF) model and syntax specification, Jul 2008. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>.