PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Engineering Business Processes with Service Level Agreements

From Early Requirements towards Business Processes

Ganna Frankova

Advisor:

Prof. Marco Aiello

University of Groningen

March 2010

Abstract

Web services' features of autonomy, platform-independence, readiness to be described, published, discovered, and orchestrated are increasingly exploited by companies to build massively distributed and loosely coupled interoperable applications. Enterprises not only export their functionalities as Web services, but also develop their business process to be Web service-based. Since services may be offered by different providers, non-functional properties, which go from execution time, costs, up to trust and security, become of paramount importance in defining the usability and success both of services and of Web service-based business processes. Ideally, the requestor of a service wants guarantees over the behavior of the services involved in the process. These guarantees are the object of service level agreements. The objective of a company is to align the service level agreements it negotiates as much as possible with its business goals. Establishing a service level agreement that favors the business objectives requires significant commitment of resources from the enterprise side, therefore any automation and support that can be obtained for this task is greatly beneficial for the enterprise.

This thesis addresses the problem of engineering secure Web service-based business processes with service level agreements from early requirements. The present work fills the gap between the requirements engineering methodologies and the actual generation of business processes based on Service-Oriented Architectures with particular emphasis on the security aspects.

We propose a methodology for deriving secure Web service-based busi-

ness processes together with service level agreements, that guarantee a certain quality of execution, from the informally specified early business requirements. Starting from early requirements modelled in the Secure Tropos formalism, we provide a set of user-guided transformations and reasoning tools the final output of which is a set of executable Web service-based secure business processes. Secure features of business processes are implemented in Secure BPEL. We propose the Secure BPEL language as a specification language for secure business process. Related service level agreements, to be signed in order to guarantee certain quality of service, are specified by the extended WS-Agreement. We propose an extension of the WS-Agreement specification and supporting environment to made an agreement more robust and long lived.

To derive service level agreements, we propose a new algorithm and we provide a prototype implementation in the constraint solving environment ECLiPSe. The implementation uses constraint programming system to satisfy user preferences against reference business processes. The IC Hybrid Domain Solver is used to solve the constraint problem. We conducted experimentation to show the feasibility of the warning strategy. In the experimentation, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings. To show the feasibility of the approach, we evaluated the functioning of the methodology on an e-business banking scenario, more specifically, from a typical loan origination process, inspired by an actual research project use case.

Keywords

[Business Process, Service Level Agreement, Requirements Engineering, Web Service, Service-Oriented Computing]

Acknowledgements

It is a pleasure to thank the people who made this thesis possible.

I express my sincere gratitude to my supervisor Marco Aiello, Professor of Distributed Information Systems at the Johann Bernoulli Institute of the Rijksuniversiteit Groningen, who has been my supervisor since the beginning of my study. He provided me with many helpful suggestions, sound advices and continuing encouragement during the course of this work.

I also thank all my friends and colleagues for their support and advice, for creation a friendly and joyful atmosphere, where it has been a great pleasure for me to work.

I am very grateful to my external thesis committee members, Vincenzo D'Andrea, Serge Chernyshenko and Pierluigi Plebani, who despite their busy schedule found the time to read, evaluate and give valuable comments and suggestions to improve my dissertation.

Lastly, and most importantly, I am infinitely grateful to my parents for their unconditional love and for years of support and encouragement. I appreciate very much everything you have done for me. I thank my husband Matteo and my little son Daniele for their understanding, endless patience and encouragement when it was most required. I love you very much.

Contents

List of Figures

Chapter 1

Introduction

The construction of massively distributed and loosely coupled applications is becoming evermore a reality thanks to the introduction of Web services. Web services are characterized by a set of technologies which cover the issues of describing, publishing, and finding individual services, as well as describing messaging and coordination mechanisms, quality of service parameters and many more facets tied to the realization of widely distributed information systems.

One of the key issues in Web services is that of automatically composing operations of individual services in order to build more complex added-value functionalities typified by business processes. Every day more and more organizations incorporate Web services as part of their business processes [42]. The research on service composition is well under way, while most of the focus is on functional properties of the composition, that is, how does one automatically compose? How does one enrich the services with semantic self-describing information? How does one discover the available services to use for the composition? If, on the one hand, this is crucial, on the other one, it is not enough. Non-functional properties or quality of service [73] of the composition are also of paramount importance in defining the usability and success of Web service-based business process.

When having repeated interactions with a service provider, a service consumer might desire guarantees on the delivery of the service. These guarantees involve both functional and non-functional properties of the offered service over a number of invocations. The non-functional properties of a service can be agreed by the procedure of negotiation a priori between the Web service provider and consumer by explicitly defining guarantee terms in a document, specifying a Service Level Agreement (SLA) [151]. Quoting business researchers: "drafting a contract and verifying that you're getting what you've paid for are real and valid expenditures of time and money" [24]. WS-Agreement [13] is an industrial standardized language and protocol for the establishment of service level agreements among loosely coupled service providers and requesters. If on the one hand, WS-Agreement is being adopted widely, on the other hand, it lacks a precise definition of the meaning of its constructs. The protocol does not contemplate the negotiation of the agreement itself, furthermore, there is no checking of how close a term is to being violated and, even more, breaking one single term of the agreement results in terminating the whole agreement, while a more graceful degradation is desirable.

Service level agreement is a tool to pair such business partners as service provider and consumer. The pairing as well as the process that need to interact with certain SLAs have to be designed. Requirements engineering is being increasingly adopted as a key step in the software development process and so new challenges and possibilities emerge. Designing of Web service-based business processes and workflows is one of the most thought challenging issues in requirements engineering [209]. We noticed that there is a gap between the requirements engineering methodologies and the actual production of software and business processes based on a Service-Oriented Architecture. When designing a Web service-based business process employing loosely-coupled services, one is not only interested

in guaranteeing a certain flow of work, but also in how the work will be performed. This involves the consideration of non-functional properties which go from qualities of services as execution time and availability up to trust and security. Business processes and security issues [160] are developed separately and often do not follow the same strategy [159]. The existing design methodologies for Web services do not address the issue of designing secure Web service-based business processes. Ideally, a designer would like to have guarantees over the behavior of the services involved in the process, i.e., obtain SLA of the business process. Service level agreement is considered to be a key component in service engineering [20]. If on the one hand, experts from the industry state that enterprise business objectives should form the fundamental basis of the SLA [109], on the other hand, developing an appropriate SLA supporting business goals of an enterprise it is not a trivial task and requires great deal of design by an expert human operator.

The present work fills the gap among the requirements engineering methodologies and the actual generation of business processes based on Service-Oriented Architectures. We propose the methodology for deriving *executable* Web service-based secure business processes with service level agreements from the informally specified early business requirements. The hierarchy of business processes is expressed in WS-BPEL and the related SLAs are specified in terms of the extended version of WS-Agreement. The proposed extended WS-Agreement allows for early warnings before agreement violation, and negotiation and possibly renegotiation of running agreements. As the proposed methodology focuses on security and trust aspects, secure features business processes are implemented in the proposed language Secure BPEL, a specification language for secure business processes.

1.1 Deriving Business Processes with Service Level Agreements from Early Requirements

This thesis deals with the issues related to engineering secure Web service-based business processes with service level agreements from early requirements. Precisely, the contributions of the thesis are five-folded and respond to the issues raised above.

1. We answer the question: "How to obtain a secure workflow from the early requirements analysis?" We address the issue of secure Web service-based business processes modelling based on the analysis of early requirements, namely, Secure Tropos [95, 144], by presenting a refinement methodology that bridges the gap between early requirements analysis and secure Web service-based workflows development.

2. We introduce a specification language for secure business processes that allows the workflow engine to automatically enforce trust and delegation requirements. The language is a dialect of WS-BPEL v2.0 [9] for the functional parts and which abstracts away low level implementation details from WS-Security [129] and WS-Federation [134] specifications. The workflows are then to be distributed; the security aspects being enforced dynamically at runtime accordingly to the identified requirements.

3. We address the question "What's in an Agreement?" In particular, we propose a formal analysis of WS-Agreement by resorting to finite state automata, we provide a set of formal rules that tie together agreement terms and the life-cycle of an agreement.

4. From the proposed analysis, some shortcomings of the protocol become evident. Most notably, the protocol does not contemplate the

negotiation of the agreement itself, furthermore, there is no checking of how close a term is to being violated and, even more, breaking one single term of the agreement results in terminating the whole agreement, while a more graceful degradation is desirable. To overcome these shortcomings, we propose an extension of WS-Agreement for which we provide appropriate semantics that allows (i) early warnings before agreement violation, and (ii) negotiation and possibly renegotiation of running agreements. We conducted experimentation to show the feasibility of the warning strategy. In the experimentation, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings.

5. We propose a methodology to design Web service-based business processes together with service level agreements that guarantee a certain quality of execution, with particular emphasis on the security aspects. Starting from an early requirements analysis modelled in the Secure Tropos formalism, we provide a set of user-guided transformations and reasoning tools the final output of which is a set of processes in the form of Secure BPEL together with a set of SLAs to be signed by participating services. The constraint algorithm for service level agreements is implemented in the constraint solving environment ECLiPSe. The implementation uses constraint programming system to satisfy user preferences against reference business processes. The IC Hybrid Domain Solver is used to solve the constraint problem.

The peer-reviewed publications by the author are presented in Section 1.5 are directly based on and are derived from the material presented in this thesis. We refer to the works wherever appropriate through the thesis.

1.2 Global View on the Thesis Contributions

The work proposed in the dissertation addresses the problem of engineering secure Web service-based business processes with service level agreements from early requirements. The present work fills the gap between the requirements engineering methodologies and the actual generation of business processes based on Service-Oriented Architectures with particular emphasis on the security aspects.

The present work proposes the methodology for designing Web service-based business processes together with SLAs. The proposal fills the gap that exists between the informally specified early business requirements the user provides and the executable Web service-based business processes. The idea is to enrich business processes with service level agreements which are favorable for the enterprise in order to achieve its business objectives with specific quality of service. As the activities about assignment of responsibilities on business processes need to be carefully considered from the security point of view, the proposed methodology focuses on security and trust aspects.

The issue of secure workflows modelling based on the analysis of early requirements is addressed by presenting a first part of the methodology that bridges the gap between early requirements and secure workflows for Web services development. The methodology allows a designer of a business process to derive the skeleton of the concrete secure business processes based on the early requirements. Furthermore, the secure business processes are refined in order to obtain the appropriate secure workflows.

Judging what is the appropriate service level agreement to sign after having defined the business objectives is far from being a straightforward task. With the second part of the proposed methodology, we provide means to go from a high-level analysis of the business requirements all the way to

the definition of the processes to be executed and the SLAs to be signed in order to guarantee certain quality of service.

We employ the Secure Tropos [95, 144] modelling framework and a methodology, an extension of the well established Tropos software engineering methodology [35], to derive and analyse both functional dependencies and security and trust requirements, i.e., early *requirements engineering*. The end-user or domain expert provides informal requirements that form the seed for developing formal processes. The output of this phase is an early requirement model. The process of the early requirements model acquisition starts from the early requirements, goes thought actor, functional dependency, permission delegation and trust modelling and ends with actor, functional dependency, authorization, and trust diagrams, i.e., the requirements model that is obtained by an expert, e.g., software engineer. The model is far from being an executable entity, but rather it is a conceptual description of the actors involved in the business, their goals and their trust and security relations.

The first part of the proposed refinement methodology aims to obtain an appropriate coarse grained business process and workflow at the workflow level based on early requirements. The refinement is processed by diagrams created in the early requirements engineering phase. The methodology takes the components of the diagrams and derives a secure business process constructs from them. The obtained secure business processes are described by a specification language for secure business processes that allows the workflow engine to automatically enforce trust and delegation requirements. The language we introduce is called Secure BPEL. Secure BPEL is a dialect of BPEL for the functional parts and which abstracts away low level implementation details from WS-Security and WS-Federation specifications. The process of the diagrams refinement and coarse grained secure business processes specification is late requirements engineering. Fur-

thermore, the secure business processes are refined in order to obtain the appropriate secure workflows, i.e., detailed design. As the Secure BPEL language is an extension of the well established WS-BPEL language [9], a business process designer, familiar with WS-BPEL processes, simply needs to understand the additional constructs introduced by Secure BPEL.

In the second part of the proposed refinement methodology, one navigates automatically the model and asks user intervention every time that an unambiguous choice is necessary. The results of the refinement of the early requirements are an intermediate model necessary to perform the reasoning on qualities of services, the business process hypergraph, and a hierarchy of business processes, specified by Secure BPEL and ready for execution. The business process hypergraph then is further analysed to build a constraint problem which represents the relationships among the various elements of the processes regarding quality of service and security properties of the processes. By reasoning with these constraints it is possible to derive the appropriate service level agreements to be signed in order to guarantee a certain quality of service when executing the process. Each of the obtained SLAs is specified by the extended WS-Agreement we propose. The extension of the WS-Agreement specification and supporting environment aims to made an agreement more robust and long lived.

The final output of the methodology is a set of executable secure business process, in the form of Secure BPEL, together with service level agreements, in the form of an extended version of WS-Agreement, to be signed by participating services fulfilling a specific business goal.

Relating the proposal to the current Web service technologies, the proposed methodology touches the following two major standards:
1) WS-BPEL is used to express the hierarchy of business processes, and
2) WS-Agreement is used to express the service level agreements.
Additionally, Secure BPEL, a specification language for secure business

processes, used in the proposal for the functional parts and abstracts away low level implementation details from WS-Security, WS-Trust and WS-Federation standards.

1.3 E-business Case Study

Let us now give the details of a typical loan origination process, that we use through the thesis for demonstrating purposes. The general environment in which the proposed scenario takes place is the e-business organization domain. The scenario is provided by SAP[1] and is a working scenario of the IST-FP6-SERENITY project[2]. The running example is abstracted from an e-business banking scenario, more specifically, from a typical loan origination process, in the context of which the activities about assignment of rights, roles, and tasks need to be carefully considered from a security point of view.

Scenario description

John is a single 25 years old man who wants to buy a flat and needs a loan. After visiting several banks, he decides to apply for a loan of 90,000 euros to his time-proved the BBB bank.

Scene 1. John goes to the bank to ask for a loan - Peter, the pre-processing clerk receives John, checks his identity, receives clients's data for identification from the Internal Computer System and matches them with the identity of John.

Scene 2. The bank double checks the credit worthiness of John - When the identity is checked, Peter introduces John to Maria, the post-

[1]Systems Applications and Products in Data Processing company, http://www.sap.com

[2]SERENITY (System Engineering for Security and Dependability) is a R&D project funded by the European Union. SERENITY aims to provide security and dependability in Ambient Intelligence systems, http://www.serenity-project.org.

processing clerk. Maria obtains several external (conducted by the Credit Bureau) and internal (conducted by the Internal Computer System) ratings in order to check the credit worthiness of the customer.

Using a Credit Bureau - The credit worthiness is checked querying the Credit Bureau. The Credit Bureau is a third party business partner of financial institution that processes, stores and safeguards the credit information of physical individual and industrial companies. In the case of John, the Credit Bureau does not return any negative information about credit worthiness and Maria continues to process John's loan.

Using internal rating - For the internal check, the post-processing clerk analyses results of calculation of the internal rating. The internal credit scoring application assigns a low risk level to John's application and the loan origination process moves to the third phase.

Scene 3. The bank calculates the loan price - Maria queries the Pricing Engine service to compute a price of the loan taking into account the score. The result in terms of original price, customer segment special conditions, customer company special conditions, asset limit for price, is then returned to Maria. Maria is able to make a proposal to John.

Scene 4. The bank and John sign the form - If John is satisfied by the proposed product, he is going to discuss the loan in more details and to finalize the process. The representative of the bank may be Maria or Caterina (the manager) according to the loan amount or the customer type. In this case, John and Maria are involved in the negotiation and signing of the contract.

The loan origination process is a business process that can be easily refined to the workflow. The different steps of the loan origination case study are depicted at the workflow diagram in Figure 1.1. The process starts from the Customer request for a loan, the request is elaborated by the Pre- and Post-proceeding clerks and Credit Bureau and then, if the

answer is positive, the Manager and Customer signs the contract and the loan is provided. Several Web services are associated with the processes such as authentication, store loan request, Credit Bureau, internal rating, loan calculation.

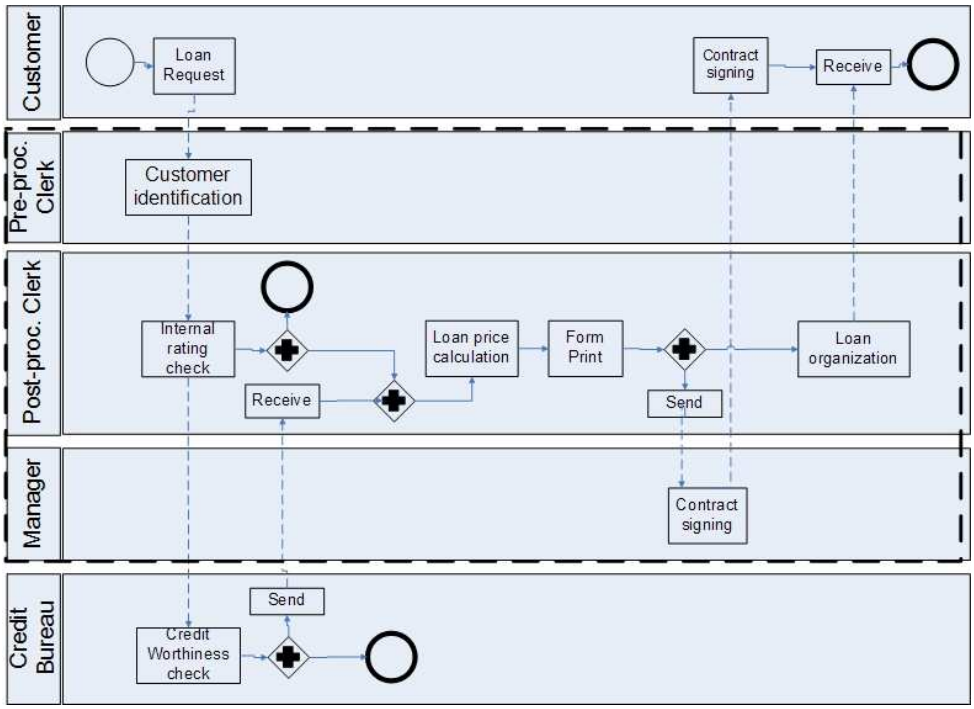

Figure 1.1: Loan origination workflow.

The presented scenario leads to several challenging security issues. There are two different security aspects here. The first one is related to intra-organizational perspective, i.e., separation of duties [177], the second aspect focuses on the extra-organizational point of view, i.e., authorization [3]. The first group of security issues are mainly: "Four Eyes" principle, message confidentiality, message integrity, authentication in a non-trusted environment, logging and auditing. In this work, we focus on the human aspects. This is a challenging issue. To prove this we refer to the French trader who was charged by Société Générale trading loss incident in January 2008, the total lost value was approximately 4.9 billion euros. In the loan origination case study, it may arise some errors and fraud if, for

instance, a clerk and a dishonest client collude together to steal money of the organization. Separation of duties can be seen as a mean of preventing errors and fraud through the limitation of a principal authority by requiring more than one person to complete a task. This is sometimes referred to as a dual control or the "Four Eyes" principal since two or more people are needed for the execution of a critical process. The same actor should not be assigned two different roles as post processing clerk and preprocessing clerk in the loan origination case study. The identification of the customer and the check of the credit worthiness should be done by two different clerks. Nevertheless, one actor may be assigned two roles in case he doesn't activate them at the same time. For example, an actor should be able to be assigned two different roles, but during two different loan origination processes. The second group of security issues are the issues this work is focused on, the issue of authorization across domains. In the loan origination case study, it may arise some errors and fraud, if, for instance, a clerk uses its legitimate rights to ask for credit worthiness of principles without their consent, in order to perform insider trading. Web service security permits to tailor the authorization of principles thanks to policies. However, grasping the context in which a principle makes a request is a difficult task to automate, often leading to over-permissive policies being deployed. In a context where clerks are interacting with banks over Web services accesses or in similar cross-organizational scenarios, is a need to ensure the least privilege principle. This principle refers to the concept that all users and systems at all times should run with as few privileges as possible. Our approach offers a mean to use augmented BPEL workflows in order to expose the context in which the Web service operations are performed. This approach enables on-the-fly delegation of authorization, further reducing the privileges of principals and preventing certain fraud scenarios.

1.4 Thesis Organization

The thesis is organized in the following way. In Chapter 2, we overview the state of the art of the research pertaining to the thesis. First, we recall the main notions that appears in the thesis such as Service-Oriented Computing, Web service, Business process/Workflow and the related standards as SOAP, UDDI, WSDL, BPEL. Then, we name works on quality of service for Web services dimensions and metrics, models. A definition of service level agreement and a description of the approaches aimed to its specification, negotiation and monitoring are provided. We overview the specifications developed in order to build secure Web services, then we review works on trust issues for Web services. The approaches on design of Web services and Web service-bases business processes conclude the chapter. We also provide the theoretical background that is employed in the remainder of the chapter. We present the Secure Tropos methodology that is an enhancement of the software development methodology Tropos.

Chapter 3 answers the question is "How to obtain a secure workflow from the early requirements?" The issue of secure workflows modelling based on the analysis of early requirements is addressed by presenting a methodology that bridges the gap between early requirements and secure workflows for Web services development. We introduce a specification language for secure business processes. At the end, the deployment of a Secure BPEL process is described.

We answer the question "What's in an Agreement?" by providing a formal definition of WS-Agreement by resorting to finite state automata. We provide a set of formal rules that tie together agreement terms and the life-cycle of an agreement in Chapter 4. From the analysis, some shortcomings of the protocol become evident. Most notably, the protocol does not contemplate the negotiation of the agreement itself, furthermore, there

is no checking of how close a term is to being violated and, even more, breaking one single term of the agreement results in terminating the whole agreement, while a more graceful degradation is desirable. To overcome these shortcomings, we propose an extension of WS-Agreement for which we provide appropriate semantics, that allows (i) early warnings before agreement violation, and (ii) negotiation and possibly re-negotiation of running agreements. Furthermore, we compare service level agreements and service licenses. Although an agreement is rather different from a license, they both regulate the activities of collaboration services. A basic difference is the fact that an agreement involves at least two parties, while a license is a unilateral statement. Nevertheless, for a license to be enacted, there must be at least a consumer of the service: this is the starting motivation to relate SLA and service licenses. We apply the proposed analysis to service licenses and propose the phases of a service license lice cycle.

In Chapter 5, we propose a methodology to design Web service-based business processes together with service level agreements that guarantee a certain quality of execution, with particular emphasis on the security aspects. Starting from an early requirements analysis modelled in the Secure Tropos formalism, we provide a set of user-guided transformations and reasoning tools the final output of which is a set of processes in the form of Secure BPEL together with a set of service level agreements to be signed by participating services.

Chapter 6 summarizes the thesis work and provides an overview of new research directions.

1.5 Published Material

The thesis is based on the peer-reviewed publications listed in the following co-authored with Marco Aiello, Fabio Massacci, Magali Seguran, Daniele

Malfatti, Artsiom Yautsiukhin, G.R. Gangadharan and Vincenzo D'Andrea.

Part of the material of the thesis has been published as articles in various workshops, conferences and journals and as technical reports. The work presented in the thesis is primarily concerned with the problems of engineering secure Web service-based business processes with service level agreements from early requirements.

We address the issue of secure workflows modelling based on the analysis of early requirements by presenting a methodology that bridges the gap between early requirements and secure workflows development and introduce a specification language for secure business processes, named Secure BPEL, in [81, 82] and then described in details together with the deployment of a Secure BPEL process in [176].

In [4, 5, 80] we provide a formal definition of an agreement and analyse the possible evolutions of agreements and their terms over an execution. As a result we identify a number of extensions which involve the initial negotiation, the monitoring of running agreements, and the possibility of re-negotiating agreements in executions. We apply the proposed analysis to service licenses and propose the phases of a service license lice cycle in [91].

A method that helps a service orchestrator to determine the concrete business process providing the highest quality of service and protection among all possible design alternatives is presented in [83] In [84, 78], we propose a methodology to design Web service-based business processes together with service level agreements that guarantee a certain quality of execution, with particular emphasis on the security aspects.

In [68], we present the threefold Open Service-Oriented Architecture approach in System, Software Architecture and Practical Implementation levels. We define a context model that represents context information used in the Rural Living Labs involved in the Collaboration@Rural European

project in [79]. The proposed model is based on ontologies, which offers designers capabilities to describe and extract semantic from context information and build reasoning process on top of them.

Furthermore, the author participated in several PhD Symposiums such as IBM PhD Symposium at the 3rd International Conference on Service-Oriented Computing [76] and Doctoral Consortium at the International Conference on Web Engineering [77] where the results of the early phase of the dissertation where presented and discussed.

[4] M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? A Formal Analysis and an Extension of WS-Agreement. Technical Report DIT-05-039, DIT, University of Trento, 2005.

[5] M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? An Analysis and an Extension of WS-Agreement. In B. Benatallah, F. Casati, and P. Traverso, editors, *Proceedings of the Third International Conference on Service Oriented Computing (ICSOC 2005)*, LNCS 3826, pages 424–436, Amsterdam, the Netherlands, December 2005. Springer.

[68] J. Dörflingerd, G. Frankova, A. Lucientes, R. de Louw, M. Navarro, C. Peña, C. Ralli, and T. Robles. Enhancing an Open Service Oriented Architecture with Collaborative Functions for Rural Areas. In K-D. Thoben, K.S. Pawar, and R. Goncalves, editors, *Proceedings of the 14th International Conference on Concurrent Enterprising (ICE 2008)*, pages 1117–1126, Lisbon, Portugal, June 2008. University of Nottingham.

[79] G. Frankova, and A. Chibani, and Y. Amirat and F. Sannicolò. Towards Context Modeling for Cooperative Rural Living Labs. In P. Cunningham and M. Cunningham, editors, *Collaboration and the Knowledge Economy: Issues, Applications, Case Studies*, The eChallenges e-2008 Conference, pages 625–632, Stockholm, Sweden, October 2008. IOS Press.

[76] G. Frankova. Web Service Quality Composition Modelling. In *Proceedings of the IBM PhD Symposium at the Third International Conference*

on Service-Oriented Computing (ICSOC 2005), 2005. IBM Research Report RC23826.

[77] G. Frankova. Service Level Agreements: Web Services and Security. In L. Baresi, P. Fraternali, and G-J. Houben, editors, *Proceedings of the Seventh International Conference on Web Engineering (ICWE 2007)*, LNCS 4607, pages 556–562, Como, Italy, July 2007. Springer. Doctoral Consortium.

[78] G. Frankova, M. Aiello, M. Séguran, F. Gilcher, S. Trabelsi, and J. Dörflingerd. Deriving Business Processes with Service Level Agreements from Early Requirements. 2009. *Manuscript submitted to the Journal of Systems and Software.*

[80] G. Frankova, D. Malfatti, and M. Aiello. Semantics and Extensions of WS-Agreement. *Journal of Software*, 1(1):23–31, July 2006.

[81] G. Frankova, F. Massacci, and M. Sèguran. From Early Requirements Analysis towards Secure Workflows. Technical Report DIT-07-036, DIT, University of Trento, 2007.

[82] G. Frankova, F. Massacci, and M. Sèguran. From Early Requirements Analysis towards Secure Workflows. In S. Etalle and S. Marsh, editors, *Proceedings of the joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM 2008)*, IFIP 238, pages 407–410, Moncton, New Brunswick, Canada, July-August 2007. Springer.

[83] G. Frankova and A. Yautsiukhin. Service and Protection Level Agreements for Business Processes. *Proceedings of the Second European Young Researchers Workshop on Service Oriented Computing (YR-SOC 2007)*, pages 38-43, Leicester, UK, June 2007.

[84] G. Frankova, A. Yautsiukhin, and M. Séguran. From Early Requirements to Business Processes with Service Level Agreements. Technical Report DIT-07-037, University of Trento, 2007.

[91] G.R. Gangadharan, G. Frankova, and V. D'Andrea. Service Li-

cense Life Cycle. In W. McQuay, editor, *Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS 2007)*, pages 150–158, Orlando, FL, USA, May 2007. IEEE Press.

[176] M. Sèguran, C. Hèbert, and G. Frankova. Secure Workflow Development From Early Requirements Analysis. In C. Pahl, S. Clarke, and R. Eshuis, editors, *Proceedings of the Sixth IEEE European Conference on Web Services (ECOWS 2008)*, pages 125–134, Dublin, Ireland, November 2008. IEEE Press.

These papers are available at http://dit.unitn.it/~frankova.

Chapter 2

State of the Art

Service oriented architecture, business process management and requirements engineering are very promising and hot topics. First, we give the main notions that appears in the thesis such as Service-Oriented Computing, Web service, Business process/Workflow and name the related standards as SOAP, UDDI, WSDL, BPEL. Then, we name works on quality of service for Web services dimensions and metrics, models. A definition of service level agreement and a description of the approaches aimed to its specification, negotiation and monitoring are provided. We overview the specifications developed in order to build secure Web services, then we review works on trust issues for Web services. The approaches on design of Web services and Web service-bases business processes conclude the chapter.

2.1 Service-Oriented Computing

Today we are experiencing a major paradigm shift in the way that software applications are designed, architected, delivered and consumed. *Service-Oriented Computing (SOC)* is the computing paradigm that utilizes services as fundamental elements to support the development of rapid, low-cost and easy composition of distributed applications. SOC not only intro-

duces a concept of *services* as as autonomous platform-independent computational elements that can be described, published, discovered, orchestrated and programmed for the purpose of developing massively distributed interoperable applications, but also framework for service publishing, discovery, binding and composition. SOC relies on the *Service-Oriented Architecture (SOA)*, which is a way of reorganizing software applications and infrastructure into a set of interacting services [163]. In [68], we present the threefold Open Service-Oriented Architecture approach in System, Software Architecture and Practical Implementation levels.

The services encapsulate the business functionality and some form of inter-service infrastructure is required to facilitate service interaction and communication. Different forms of this infrastructure are possible because services may be implemented on a single machine, distributed across a set of computers on a local area network, or distributed more widely across several wide area networks. A particularly interesting case in which XML standards are utilized and there is a stack of related technology that go from the messaging up to the coordination of loosely coupled elements. These services are called *Web services* [164].

A Web service is described using a standard XML-based interface description language called WSDL (Web Services Description Language) [46]. The service provider uses a WSDL document in order to specify the operations a Web service provides, as well as the parameters and data types of these operations. The description covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location. The interface hides the implementation details of the service, allowing it to be used independently of the hardware or software platform on which it is implemented and also independently of the programming language in which it is written. This allows and encourages Web services-based applications to be loosely cou-

pled, component-oriented, cross-technology implementations. Then, Web services are published in UDDI (Universal Description, Discovery, and Integration) registry [27] that is used to store and retrieve information on service providers and Web services.

Figure 2.1 shows the relationship between the core elements in SOA [194].

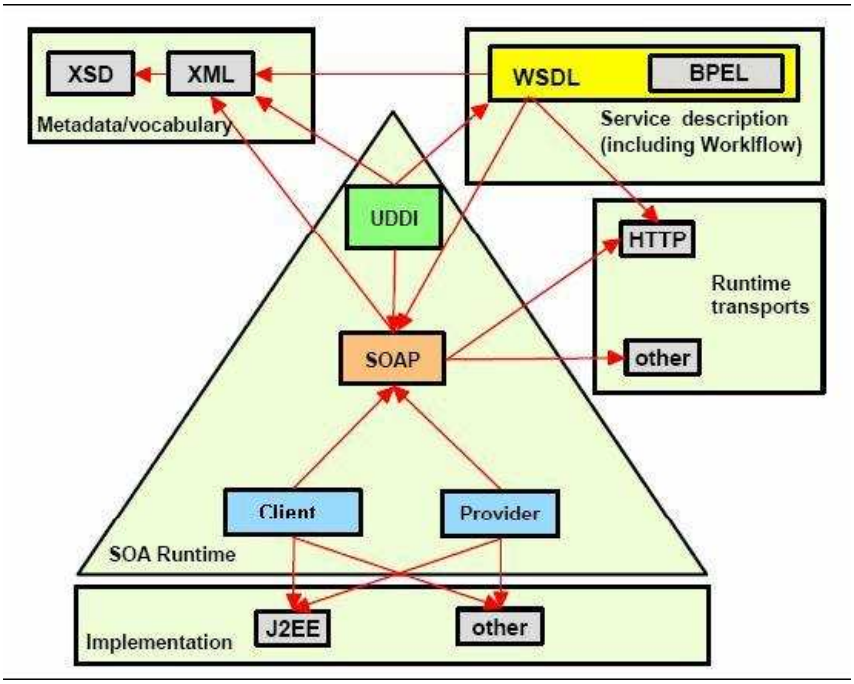

Figure 2.1: Main building blocks in a SOA approach based on Web services.

All elements use XML including XML namespaces and XML schemas. Arrows denote communication among the main building blocks. WSDL is the base for SOAP server deployment and SOAP client generation. SOAP (Simple Object Access Protocol) [102] is a network, transport, programming language and platform neutral protocol that allows a client to call a remote service.

Web services fulfill a specific task or a set of tasks. They can be used alone or with other Web services to realize more complex functionalities typified by Web service-based business processes. A *business process* is

a set of interrelated tasks linked to an activity that carry out meaningful business operation. Business processes vary in the level of granularity, and the details of a business process will vary from enterprise to enterprise. *Workflows* are business processes that are run in an IT environment. Workflow software does not create business processes, but applying workflow to a business process brings the details of the process into focus. Workflow provides orchestration for interactions among component Web services. A Web service that serves as an activity in one workflow can itself consist of a series of sequenced activities or a workflow [193].

Both business processes and workflows are described by the Business Process Execution Language (BPEL) [9]. BPEL is an orchestration language for Web services. It can be seen as a workflow description, where each atomic task is a call to a Web service. BPEL is an XML-based language, supporting common process flow patterns, such as execution of tasks in sequence or in parallel, splits, AND and OR joins, as well as fault handling and process compensation mechanisms. The tasks themselves are seen as blackboxes from the BPEL engine perspective: the way the task is executed by the service is left open to the service host. In [75], architecture of an e-Business application using Web services composition was proposed. The business process and the corresponding workflow for proving that the approach is feasible was developed.

Recently, two complementing specifications for BPEL, respectively called BPEL4People [2] and WS-HumanTask [2], were published in order to further narrow down Web service invocation to so-called people tasks. These specifications are useful for managing already at process design time assignment of tasks to specific groups of people, and the full lifecycle of task claim, task delegation and task completion at a user level. Our work does not specifically rely on these human-related management activities. The concepts introduced by our approach are primarily about handling trust

and permissions from a B2B perspective, where the permissions and delegation introduced in BPEL4People are by nature intra-organizational. It can further be noted that BPEL4People does consider security issues at out-of-scope and thus offer no mean to deal with trust nor authorization management.

2.2 Quality of Service for Web Services

With the term *Quality of Service (QoS)* we refer to the non-functional properties of an individual service, or a composition of services. The term is widely used in the field of networking [53, 174] and in real time issues [48]. Usually it refers to the properties of availability and performance. In the field of Web services, the term has a wider meaning. Any non-functional property which affects the definition and execution of a Web service falls into the category of QoS, most notably, accessibility, integrity, reliability, and security [142, 149, 162, 158].

Various approaches for defining QoS requirements exist. Lee et al. [130] describes QoS requirements for Web services. Ran [169] organizes the aspects of QoS into categories, i.e., runtime, transaction support, configuration management and cost, security. The author argues that each category needs to have a set of quantifiable parameters or measurements. In [85], QoS aspects are qualified by characteristics as direction and value type. A set of measures for reliability and performance are proposed. A taxonomy for quality dimensions can be found in [40]. A classification of quality dimensions with analysis on correlation between the quality attributes of components and those of their composition is presented in [52]. While [200] characterize possible relations between QoS metrics and business metrics. The concept of Quality of Business metrics is introduced in [191]. The work provides an approach that relates Quality of Service, Quality of Ex-

perience and Quality of Business in the Web service environment. Atzeni and Lioy [22] overview security system assessment methods and metrics. The work in [103] presents the effect of security requirements on the functional requirements. Analysis of security requirements of business processes of e-Commerce is presented in [123]. In [137], it is argued that networking issues have need to be taken into account by both Web service providers and consumers.

2.2.1 Quality of Service Models

A number of approaches to QoS models rely on extensions of the Web Service Description Language (WSDL), e.g., [96, 184]. The main idea is simple: provide syntax to define terms which refer to non-functional properties of operations. Given such description, one can then build a framework for the dynamic selection of Web services based on QoS requirements. On the negative side the QoS definition is tied to the individual operation, rather to the service as a whole. Furthermore, there is no run-time support, i.e., once a quality parameter is set, it can not be changed at execution time. In [208, 1, 206, 19], the description of elementary service qualities as a quality vector each component of which is a quality parameter for the service is proposed. The authors propose to compute quality criteria for composite services by using special aggregation functions, e.g., sum, product. Based on the model the aggregation of numerical QoS properties can be easily performed, but the approach does not consider the case of non-numerical parameters. In [133] Lin et al. propose a fuzzy way to express QoS requirements. While as some QoS metrics such as response time and invocation price can be changed at run-time, the approaches dealing with rigidly fixed values is not appropriate. Adding a new data structure to the UDDI model in order to take into account non-functional properties is presented in [169]. As the description of quality of service information is

static, i.e., it is specified for a particular service and can not be changed at run-time: the approach does not allow to cope with the problem of run-time support. As users rate services based on their expectations on the quality of service and the expectations are often different, in [62] the authors propose a quality of service management framework based on users' expectations. A model for expressing the non-function properties both from the service and user perspective is proposed in [87]. The model is compliant with the WS-Policy framework [23]. However the idea is feasible, the work does not support negotiation of QoS between service provider and consumer. With the simple QoS model [41] that includes three dimensions such as time, cost, and reliability, it is possible to describe workflow components from a QoS perspective. The model is predictive as allows computing the quality of service for workflows automatically based on atomic task QoS attributes [41]. In our opinion, the model is very simple and should be extended to accommodate more QoS dimensions. An approach for defining QoS requirements is QML [85]: a language for QoS description using XML. QML contains a refinement mechanism allows reuse and customization of QoS contracts. The work is focused on the usage of QML in the general context of software design, but not Web services in particular. Maximilien and Singh [145] develop an ontology-based framework for dynamic QoS-aware Web services selection. The positive side of the approach is that it takes into account provider's policies and consumer preferences, but the approach does not allow for negotiation. In addition, a semantic web approach, in which services are searched on the basis of the quality of semantically tagged service attributes is presented in [30]. In [8], a quality of service model of composition which considers the information flow is described. Jureta et al. [116] provide a survey on quality models for SOC. The authors analyse similarities between the models proposed in the literature, review them and integrate them into a single quality model. Priority and dependency

information is integrated in the proposed model. The approach is feasible and the integration of the model to UML is needed. The use of the agent-oriented methodology Tropos to model a wide spectrum of quality of Web services properties is proposed in [6]. WS-Policy [23] defines a framework and model for expressing capabilities, requirements, and general characteristics of individual services. The application of the policy-based software paradigm to the automated provisioning architecture is described in [18]. The authors show how the use of policies can enhance utility computing services. In [197], a middleware-based approach for managing dynamically changing QoS requirements of components. Non-functional capabilities are described as policies in GlueQoS language that is an extension of WS-Policy language. The approach supports matching, interpret and mediate QoS requirements of clients and server sites both at deployment and runtime. Although plenty approaches for modelling QoS for Web service exist, in [76] we claim that current models are far from ideal with respect to the identified requirements, and there is a lot of space for further investigation and innovative research.

2.2.2 Service Level Agreement

Quality of service is an important concern in dynamic service composition and selection, given that several service providers can provide similar services with common functionality but different QoS and cost. Modelling and measuring QoS is only one aspect of the management and procurement of services. The other half of the picture is the negotiation of QoS aspects. A negotiation mechanism between service consumers (i.e., an integrator or an end-user) and service providers has to be in place to reach mutually-agreed guarantees and establish agreements on service provisioning which include the non-functional properties of the services. It is also important for service providers to be able to guarantee the promised QoS at runtime [150].

The concept of service level agreement represents expectations and obligations of the partners regarding service characteristics. Though there are many definitions of SLA in the literature, in this work we use the term *service level agreement* as a machine interpretable specification of the value of a set of selected parameters of a service, involving more than one party (two parties in case of SLA for Web services), to assist in automation [151]. SLAs have been used for a while. At the beginning they served as general operating procedures to buy or rent machine time on a mainframe. Nowadays SLAs are widely used in networking and telecommunication and as a result they became more complex and broader in scope. A customer can have several SLAs with different providers and a provider may have its own SLA with other providers, each with a different set of requirements and measurement criteria. In the world of Web services, the relations among providers and consumers become more complex and Web services paradigm has made SLAs more challenging. The SLAs for Web services are used to guarantee not only network performance and uptime availability as they do in networking, but also application performance. It is relevant because each Web service has its own characteristics and network requirement [157]. Further, we describe key factors of involved in service level agreements, namely, SLA specification, negotiation and monitoring [186, 151].

Service Level Agreement Specification

Several languages for specifying SLAs have been proposed, most notably, IBM's Web Service Level Agreement (WSLA) Language [139] focuses on Web service interactions. The goal of WSLA is twofold: at deployment time it helps the interacting parties to configure their resources to meet a predefined SLA; at run time it helps the interacting parties to monitor the performance of each other and to detect and notify violations. However, the monitoring framework does not answer the question "How close a guarantee

is to being violated?"

Web Service Offering Language (WSOL) [185] focuses on Web service interactions. The language is used to formally specify various constraints, management statements, and classes of services for Web services. SLAng [124] is an XML-based language that describes QoS properties to include in SLAs. SLAng does not focus only on Web service interactions, but also to specify SLAs for hosting service provisioning (between container and component providers), communication service provisioning (between container and network service providers), etc. Although SLAng is expressive enough to represent the QoS parameters included in SLA, more work is needed on the definition of the semantics of SLAng. Web Services Agreement [13] defines the interaction between a Web service provider and a consumer, and a protocol for creating an agreement using agreement templates. The specification is described in details in Chapter 4.

The work in [186] names the main problem and suggests solutions for correct SLA specification. Furthermore, it addresses the specification of SLA based on three service management principles: continuity in SLA specification, the SLA context and content, and the principle of specifying the quality of both a service process and a service object. Sahai et al. [172] proposes a specification language that enables definition of precise an flexible SLAs. In [11] the requirements for a precise SLA specification are discussed. The authors argue that the correct definition of QoS parameters corresponds to the establishment of an ontology between a service provider and a consumer. This ontology should provide a definition of terms and the semantics between them. Annotation of service level agreement templates with semantic QoS metrics is proposed in [86] and in [112] the authors illustrate how to specify an agreement with ontology language instead of XML schema. With the help of ontology, the author propose the solution of service selection problem as matching of SLAs [47]. Buscemi and Mon-

tanari [37] present the cc-pi calculus for modelling processes able to specify SLA contracts. The proposed language allows for resource allocation and for joining differen SLA requirements. The notion of contract from a logical perspective is presented in [26]. The authors extended intuitionistic propositional logic with a new connective, that models contractual implication. Jin et al. in [113] focus on information collection and analysis at the creation stage of SLA, on the relation SLA from service provider side-IT infrastructure of the provider and the impact of the SLA the service consumer sign on their productivity. A customer-oriented approach for specifying service contracts is presented in [175]. The author propose the usage of workflow concepts for designing and writing high quality service contracts for IT services. The approach is feasible, while it should be improved to be used for derivation of customer/oriented, but measurable quality parameters. The notion of contracts formalization, a contract construct and related function that bridges the gap between service matching and service mapping are introduced in [16]. In [154] an extension that allows the WS-Agreement specification supporting temporality is proposed. The authors define an appropriate domain-specific language that allows to express many temporal properties. We consider the proposal to be useful for the re-negotiation of an agreement in our work.

Karten in his book "How to establish Service Level Agreements" [120] provides the business point of view on how to be successful in establishing your SLAs and names the factors that accounts for a SLA never reaches completion or proper functioning. A method to convert the contract from text into an electronic equivalent that can be executed and enforced is presented in [152]. The authors propose using finite state machines to describe standardized conventional contracts. Angelov and Grefen [17] presents a reference architecture for the development of e-contracting systems. The architecture introduces standardized view on the systems, facilitate the de-

sign of logical view and allows faster development of e-contracting systems.

Service Level Agreement Negotiation

The negotiating of service agreements has a vital role in the life-cycle of a SLA. Presently, negotiation is mainly a manual process and full or partial automation is needed. Theoretical bases of SLA negotiation are provided by Demirkan et al. [60] where the authors identify negotiation support system requirements. The term *negotiation* is viewed as the interaction among participants, i.e., service provider and service consumer in the context of deriving mutual commitment. i.e., service level agreement. A negotiation description language is introduced in [70]. The language is rather simple as it provides a high-level description of a negotiation between parties in service-oriented context. Hung et al. [110] propose WS-Negotiation language, an XML language that contain three parts: negotiation message to describe the format for messages exchanged, negotiation protocol to describes the mechanism and rules that negotiation parties should follow, and negotiation decision making, which is an internal and private decision process.

Gimpel et al. [94] propose PANDA - Policy-driven Automated Negotiations Decision-making Approach. The approach automates decision-making within negotiation. An automated negotiation framework based on a finite state automata and a set of negotiation protocols is in [132]. In [119] an approach for automated SLA creation through a negotiation from a set of service level objectives is proposed. The approach is feasible while the question on service level objectives obtaining remains open. In our approach the use of requirements engineering methodology solves this problem. A protocol for dynamic SLA negotiation is proposed in [167]. The authors propose a simple extension to the WS-Agreement protocol that facilitates the negotiation process. Hasselmeyer et al. in [106] focus

on outsourcing the function of the provider's negotiator to external negotiation broker. The approach decreases cost of SLA negotiation, while implies loss of control. Therefore, it is needed to state where the negotiation has ended, independent on whether the agreement was reached or no. The modelling of high-level policy specification for negotiation and a middleware broker framework for conduction an automated-based negotiation is presented in [210]. A scheme for negotiation of e-service under uncertainty is proposed in [204]. The idea is that the participant who is negotiating under uncertainly obtain an assistance from other reputable participants who have already negotiated the same issue. A model and a protocol for negotiating SLA over accessing resources in distributed environments are presented in [56].

The critical issue in SLA negotiation is a common understanding of the terms among negotiating parties, i.e., there is an ontology problem of electronic negotiations is raised in [182]. One of the proposed solution is to use SLA templates [171] and annotate the templates with semantic QoS metrics [86]. Yarmolenko and Sakellariou [203] specify a SLA's agreement terms as functions rather than variables, constraint values or ranges. This approach minimize the number of re-negotiations and reduce agreements failures. An approach of SLA matching is presented in [202]. The work syntactically matches SLAs by parsing them into syntax trees. While in [161], the matching of providers and consumers is done by using semantic web technologies that helps helps to achieve more accurate results.

Decision making support in SLA creation and negotiation is presented in [136]. The authors propose using dynamic service profiles that contain historical service execution data and precautionary avoid non-SLA-conformant service behavior. The mechanisms of the COSMA approach [136] for an integrated management of atomic and composite SLAs during the whole life cycle is used.

Cappiello et al. [39] present a negotiation model to support the automatic generation of SLA on-the-fly. The authors developed a model to express Web service quality, provider capabilities, and user requirements that is further employed in the negotiation model to generate SLA. In our approach, we tie business processes with SLAs. We do not focus on SLA negotiation, while we take into account early requirements provided by the end-user, the structure of the business process and security and trust concepts.

The issue of re-negotiation as a second or further negotiation that may change the terms of an existing agreement[1] is raised in [105]. The work describe the protocol for re-negotiation of an agreement that can be used with WS-Agreement. The protocol is based on the principles of contract law [181] to make the new agreement legally compliant. The authors of the work [64] follow the direction proposed by us in [5] proposing the integration of new functionality to the protocol that enable the parties of a WS-Agreement to re-negotiate and modify its terms during the service provision. We consider both the proposals to be useful for the re-negotiation of an agreement in our work.

He et al. [108] propose an agent-based framework that uses the agent's ability of negotiation, interaction, and cooperation to facilitate autonomous SLA management in the context of service composition provision. Negotiating a complex service is discussed in [71]. Such a negotiation deals with uncertainty. The problem is that the whole dynamic composition fails as a result of failure to contract one of individual services. Our approach can not guarantee the service availability before the SLA establishment, while it aims to avoid SLA of the whole composition failure by introducing re-negotiation phase in the SLA life cycle.

The cost of SLA negotiation is discussed in [147]. As negotiating mul-

[1] Oxford English Dictionary, Second Edition, 1989.

tiple QoS criteria is a costly process, the authors propose to consider the advantages and disadvantages of negotiation carefully and execute multi-step negotiation only where its cost are justifiable. One approach to keep negotiation costs low is the supermarket approach or the take-it-or-leave-it approach. Its name correlates with the business model of supermarkets, where customers can only decide whether to take certain product or not. If the customer does not find the brand (i.e. the offer) they likes, another supermarket (i.e. the service provider) may be an option.

An approach that accomplish SLA decomposition and translates service level objectives, specified in SLA to lower-level resource requirements for each system involved in providing the service is presented in [44]. The work is useful to create an efficient design to meet SLA.

A framework for automating of the Web service contract specification and establishment is proposed in [51]. The authors propose a QoS model that define both domain-dependent or domain-independent and negotiable or non-negotiable QoS dimensions. The model is used in the proposed mechanisms for service matchmaking and selection. The matchmaking algorithm for the ranking of functionally equivalent services, which orders services on the basic of their ability to fulfill the consumer requirements, while maintaining the price below the specific budget. The configuration of the negotiable part of SLA exploits the top-ranking services identified in the matchmaking phase. The contract establishment activity produces SLA in the WS-Agreement specification. The framework is developed to be self-heading in reaction to faults on non-functional properties. The authors claim that the most suitable action in this case is the service substitution. While we consider that re-negotiation of SLA saves time and money both of the provide and consumer. Furthermore, there is no requirements gathering phase in the presented approach.

Monitoring of Service Level Agreement

Monitoring an established SLA is essential for a service consumer. Non-functional monitoring is concerned with the statistical QoS metrics collection to evaluate wheatear a provider complies with the QoS level specified in the SLA [151]. Fundamental concepts of non-functional SLA monitoring are presented in [153] which contains a discussions on the separation of the computation and communication infrastructure of the provider, service points of presence and metric collection approaches. The authors propose an architecture for QoS monitoring by third parties to ensure that the results are trusted by both the provider and consumer. A Web Service Level Agreement framework for defining and monitoring SLAs is presented in [122]. The work addresses the definition of a language for SLAs specification, creation, and the implementation of a SLA compliant monitor. Greenwood et al. [99] propose an automated and distributed SLA monitoring engine that considers both provide's and client's side measurement of SLA. The approach deals with the scenario where providers contract with each other to fulfill the customer's request.

In [55], the Agreement-Based Open Grid Service Management (OGSI-A) model is proposed. Its aim is to integrate Grid technologies with Web service mechanisms and to manage dynamically negotiable applications and services, using WS-Agreement [13]. The WS-Agreement is supported by the definition of a managing architecture: CREMONA - An Architecture and Library for Creation and Monitoring of WS-Agreement [138].

A list of correctness requirements the most business contract should satisfy is identified in [180]. The provided correctness requirements are mapped into conventional safety and liveness properties. The authors described contract by means of Finite State Machines and showed how it can be validated using standard model checker such as Spin. Sahai et al. [173]

propose an automated and distributed SLA monitoring engine that monitors a SLA. The SLA should be specified in the proposed in [172] specification language that enables definition of precise an flexible SLAs. The work in [65] focuses on SLAs testing. The authors proposed the use of genetic algorithms to generate inputs and configurations for service-oriented systems that cause SLA violations. Jurca et al. [115] show that independent monitoring can be replaced by a reputation system where monitoring is based on feedback provided by the clients. Rana et al. [170] present a work on SLA penalties and types of violations that can occur during SLA provisioning.

In [155], a mechanism to check the consistency of SLAs and explain WS-Agreement inconsistencies is described. The authors map an agreement to CSPs that enables the use of CSP solvers for consistency check and explain inconsistencies of SLAs. The issue of compliance between WS-Agreement templates and offers is raised in [155] by the same authors. CSP and its solver is used to check and explain compliance of WS-AgreementḢowever, the approach are applied not to the whole WS-Agreement but to a less expressive subset of it.

Pro-active monitoring technique can be applied to minimize incidents of violation detection caused by provider side. The main idea of the technique is deploying monitoring mechanisms by provider to monitor his own resources. In this case rather reacting to violations notified by the notification and violation service the provider prevents them. Pro-active monitoring on electronic contracts is presented in [201]. The work proposes an approach to formalize electronic contracts into a set of representations to enable automatic monitoring. The approach not only supports the detection of actual violations but also detection of imminent contract violations. Although functional monitoring mechanism is developed, non-functional monitoring is out of scope in the work.

The above approaches show that frameworks for QoS definition and management are essential to the success of the Web service technology, but there are a number of shortcomings that still need to be addressed. First, a formal definition of the semantics of a SLA is missing. Second, the frameworks should be more flexible at execution time because actual qualities of services may change over time during execution.

2.2.3 Web Service Security and Trust

We overview the specifications developed in order to build secure Web services, then we review works on trust issues for Web services.

WS-Security [129] specifies enhancements to SOAP messaging that while building secure Web services can be used in order to implement message content integrity and confidentiality. The specification provides a general-purpose mechanism for associating security tokens with message content. No specific type of security token is required, the specification supports multiple security token formats. WS-Security describes how to encode binary security tokens (e.g., X.509 certificates and Kerberos tickets), a framework for XML-based tokens, and how to include opaque encrypted keys. It also includes extensibility mechanisms that can be used to further describe the characteristics of the tokens that are included with a message. WS-Security is flexible and is designed to be used as the basic for securing Web services within a wide variety of security models including PKI, Kerberos, and SSL. Specifically, WS-Security provides support for multiple security tokens, multiple trust domains, multiple signature formats, and multiple encryption technologies. The specification intentionally does not describe explicit fixed security protocol. It provides three main mechanisms: (i) ability to send security tokens as part of a message, (ii) message integrity, and (iii) message confidentiality. To summarize, the focus of WS-Security is to describe a single-message security language that provides for

message security that may assume an established session, security context and/or policy agreement. WS-Security can be seen as a business process that enables application to construct secure SOAP message exchanges.

WS-Security does not address the issues of interoperability between SOAP client and SOAP service. The standard does not specifies how a SOAP client and a SOAP service can agree on the nature and characteristics of the security tokens. WS-Security begins with the assumption that, if one of the parties uses a particular type of security token within the WS-Security header, then the other party will be able to interpret and process the token. As there are multiple viable formats for security tokens (e.g., X.509 certificates and Kerberos tickets), it is unlikely that an arbitrary SOAP endpoint will be expected to understand each of these options. While the guarantee that both partners who wish to use WS-Security to secure their SOAP messages support the security token they will be able to understand and process is needed. We face the problem of heterogeneity of the security environments between which WS-Security must operate. At this point the guarantee that there will be an intersection between the sets of supported security token format of two different SOAP actors who wish to use WS-Security to secure their SOAP messages is needed. Therefore, interoperable application of WS-Security across security domains with different security infrastructures will require either mechanisms by which actors can come to an agreement on the nature of security token they will use in any subsequent SOAP transactions, or mechanisms by which different security tokens can be mapped into others, such that individual SOAP actors can be guaranteed to receive only security tokens that they will be able to understand and process. The following specifications support both scenarios for addressing this interoperability issue:

WS-SecurityPolicy specifies how Web services actors can assert to potential transaction partners their policies with respect to WS-Security

mechanisms, including their capabilities and preferences with respect to security tokens (e.g. a SOAP service can assert "I can process X.509 certificates and SAML assertions but my first choice is SAML") [128].

WS-Trust enables security token interoperability by defining a request/response protocol by which SOAP actors can request of some trusted authority that a particular security token be exchanged for another [127].

Even if the given security token's format is acceptable to a recipient of a WS-Security, interoperability at the syntax level is no guarantee that the recipient will be able to trust the token. **WS-Trust** [127] addresses the issue of trust interoperability issues by defining a simple request/responce for security token exchange. A client sends security token request to a Security Token Service (STS), the request includes the security token that the client is asking to be exchanged (old token). The STS responce contains the exchanged token (new token). In addition to token exchange, the WS-Trust request/response protocol is general enough to support token issuance (the client presents a claim to the STS for the STS to authorize through the issuance of a corresponding security token) and token validation (the client presents a token to the STS and asks that its validity be determined). Issuance and validation can be thought of as special cases of exchange, as both the client claim in the issuance case and the STS validity assertion response in the validation case can be thought of as tokens [140]. WS-Trust supports broker trust relationships and therefore can be used to build delegation and trust chains between partners. A semantic of the main mechanisms of WS-Trust and typical protocols, relying on these mechanisms, are modelled in [29]. The core security properties of the specification are proved and some limitation and potential vulnerabilities are discussed. Designing secure business processes is out of the scope

of this work as it focuses at the lower level, i.e, protocols modelling and verification. WS-Trust can be seen as a business process that enables interoperability between the multiple formats for security tokens (that might be used in a WS-Security protected message) and broker trust relationships.

The **WS-Federation** [134] specification builds on WS-Trust specification to allow different security realms to federate by allowing and brokering trust of identities, attributes, authentication between participating Web services. Here we have several actors. Identity Provider (IP) (which is an extension of STS) is an entity that acts as an authentication service to end requestors an data origin authentication server to service providers. Attribute service is an entity used to obtain authorized information about a principal to allow the sharing of data between authorized entities. Pseudonym service is an entity that allows the principals to have different aliases at differen resource/services or in different realms, and to optionally have pseudonym change per-service or per-login. WS-Federation allows attributes and pseudonyms to be integrated into the token issuance mechanism to provide federated identity mapping mechanisms. WS-Federation can be seen as a business process that enables federation of identity, attribute, authentication, and authorization information.

Trust is a directional relationship between two parties that can be called the trustor and the trustee. Trust is an essential aspect for decision on security since it is related to belief in honesty, trustfulness, competence and reliability [32, 43, 148]. In [97] and [98], Grandison and Sloman consider trust as a quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context. Trust is not symmetric, so this belief by the trustor does not imply any similar belief by the trustee. Distrust is a quantified belief by a trustor that a trustee is incompetent, dishonest, not secure or not dependable

within a specified context. Gambetta [89] emphasizes the subjective level of trust: "trust is the subjective probability by which an individual A, expects than another individual, B, performs a given action on which its welfare depends". In [43], trust is considered from a cognitive point of view: trust is a mental state based on a set of beliefs (depending on the feeling of trust more than the trust itself). There are various reasons for distrusting agents such as unskillfulness, unreliability and abuse. According to the authors, trust implies that having high trust in a person is not sufficient to imply the decision of trust, it could depend on the situation and the evaluation of the risk [72]. In [114], Jøsang introduces the notion of decision and gives the definition of trust as the extent to which a given party is willing to depend on something or somebody in a given situation with a feeling of relative security even though negative consequences are possible.

Usually, there is a level of trust associated with a trust relationship [146]. Trustworthiness is defined as a measure of level of trust that the trusting agent has in the trusted agent. The trust level is a measure of belief in another entity and thus it is a measure of belief in the honesty, competence, security and dependability of this entity (not a measure of the actual competence, honesty, security or dependability of a trustee) [98]. Considering trust level, we can emphasize the following two approaches. Firstly, there might be some degrees in the trust level, i.e., so called, "[0..1] trust level approach" that points the level of trust one entity trusts another one. It means some degrees between absence and presence of trust. In the definition given in [97] and [98], quantification is linked to the notion of trust. Quantification reflects that a trustor can have various degrees of trust (distrust), which could be expressed as a numerical range or as a discrete classification such as low, medium or high in [98] and the work done in SULTAN (Simple Universal Logic-oriented Trust Analysis Notation) has incorporated concepts such as experience, reputation and

trusting propensity. SULTAN proposes an abstract, logic-oriented framework designed to facilitate the specification, analysis and management of trust relationships [97, 98]. One of the disadvantages of this "[0..1] trust level approach" is that it is not clear how to define the exact degree of the trust level. It is not a straightforward task to reason which option to trust to or which alternative to distrust especially in case of the trust level is not high. Secondly, there is the "0/1 trust level approach" that means strictly absence/presence of trust dependencies. In [21], the authors consider three trust levels: Trust, Distrust, and NTrust (i.e., neither trust nor distrust). Trust and Distrust means 1/0 trust level, NTrust is necessary since the requirements specification may not define any trust or distrust relation between two specific actors. In our approach, the trust level is determined from the reasoning on the presence/absence of trust dependencies in the early requirements model. The trust level value denotes the level of trust between the truster and the trustee on the fulfilling the business process. The determined trust level of service providers might be employed when there is a possibility to choose one business process from the several alternatives suggested by different providers.

In the loan origination area, where the aim is to provide the loan to the reliable customers, the Credit Bureau responsible for the credit worthiness check shall be reliable. In this domain, several Credit Bureau can coexist and so the best one shall be selected. The advantage of the "0/1 trust level approach" is to have the possibility of choosing the best alternative, i.e., the right partner to work with, in case of distrust to another one.

2.3 Web Service and Business Process Design

Various approaches aimed to use requirements engineering methodologies (and not only) in the context of Web services and Web service-bases busi-

ness processes design.

Basic principles of Web services and business processes design are presented in [165]. While the work does not distinguish logical business processes and their implementation. While our approach produces executable secure business processes with SLAs.

Distante et al. [66], analyse and compare web applications design methodologies with regards to their support for modelling business processes. Further, a comprehensive design model for integrated business processes in web applications is proposed. The model is based on UWAT+, an extension of the ubiquitous web applications design model called UWA. The proposed model satisfies plenty of requirements, while it does not work with non-functional properties and SLAs.

Lapouchnian et al. [125] propose a requirements-driven approach for business process design. Requirements goal models are used to capture business goals and alternative process configuration. Quality attributes such as customer satisfaction serve as the selection criteria for choosing among business process alternatives induced by the goal models. Executable business processes are generated in semi-automatic way from goal models. The approach does not focus neither on secure business processes nor SLA building for generated business processes.

The Tropos methodology [35] is a requirements engineering methodology that supports all analysis and design activities in the software development process, from application domain analysis down to the system implementation. Lau and Mylopoulos [126] propose a design methodology for Web services adapted from the Tropos methodology. The work is based on the use of goals to determine the space of alternative solutions to satisfy the goals. The key point is that the solutions are represented by Web services. The generated Web services design is expected to accommodate as many of those solutions as possible rendering the design usable by a broader class

of applications. On the negative side, Tropos is not tailored specifically to Web service design. Therefore the proposed methodology does not address the issue of integration neither of Web Service Business Process Language in order to specify actual behavior of participants in a business interaction nor WS-Agreement Language to specify SLAs of the services. In [121], a methodology for business requirements modelling that uses the Tropos framework to capture the strategic goals of the enterprise is described. The proposed methodology enables to produce concrete business processes expressed by BPEL4WS description. The concrete business processes are elicited from the description of business process notions with Tropos concepts extended with formal annotation called Formal Tropos [88]. On the contrary, our work aims not only to obtain business processes from an early requirements analysis, but also to provide them with SLAs. Furthermore, the work involves the Tropos methodology that does not support the notion of trust and delegation dependencies while the Secure Tropos does. The agent-oriented methodology Tropos is used for analysing Web service requirements by Aiello and Giorgini in [7]. In the approach the authors do not model every individual Web service as an agent, but rather model the whole set of interacting services as a multi-agent system where different dependent hard and soft goals coexist. Penserini et al. [166] address the issue of refining the Tropos methodology and tailoring it to the design of Web services. The Tropos design process is extended to support a revised notion of capability that explicitly correlates actor plans with stakeholders needs and environmental constraints. The agent capability is considered as a service. Furthermore, the authors sketch how Tropos design-time models can support service discovery and composition by relating stakeholder goals to sets of services available. Even if, the idea is feasible, the work is in an early stage and there is a need for more precise mapping of agent capability that is considered as a service. Furthermore, there is no secure

business processes design support.

A methodological approach for deriving the software functionality from organizational model is presented in [59]. The authors model an organization by means of BPMN and use the goal/strategy Map approach. The work allows for organization analysis and system goals understanding in a participative way with customers. The approach does not focus at non-functional properties of business processes.

Modelling of Web service structural and behavioral aspects using UML [101] is studies in several works. An approach of mapping UML activity diagrams into BPEL4WS is proposed in [92]. Deubler ar al. [63] introduce aspect-oriented techniques for UML sequence diagrams modelling. The authors propose to specify certain behavior aspects of overlapping Web services (so called crosscutting services). Composite Web services design using UML activity diagrams is proposed in [179]. An important feature of the method is the transformation of WSDL [46] descriptions into UML diagrams. While Marcos et al. [143] describe an extension of UML for representation of WSDL specifications. In [38], UML sequence diagrams are used for representing service-oriented business processes with time constraints. The work focuses on capturing main elements of WS-BPEL and automatic translation of UML diagrams into business process execution language. A set of software pattern primitives for process-driven SOAs development is proposed in [207]. The primitives are specified using a proposed UML2 profile for activity diagrams and the UML Object Constraint Language (OCL) [100].

In additional, Business Process Modeling Notation (BPMN) [196], a notation that is readily understandable by all business users, from the business analysts to the technical developers, and finally, to the business people. The use of User Requirements Notation [10] for business process modelling is proposed by Weiss and Amyot [195]. In [33], a conceptual framework

for designing Web service-based systems is proposed. The authors adopt xBPEM methodology [135] for designing service-oriented systems. The approach includes client-centered analysis, identification of functionalities and collaboration patterns of involved Web services, service discovery and selection. Vanderfeesten et al. [192] introduce cross-connectivity metric that helps validating process models for understandability. A conceptual framework called COSMO, for service modelling is presented in [168]. The framework supports not only service modelling, but also service discovery and composition performed at design and run time. While the work does not consider business process modelling. The research works of Colombo et al. [50] presents a methodological framework that supports the modelling and formal analysis of requirements for service composition through a social and process perspective. In [118], the authors propose a goal driven approach to service elicitation, distribution and orchestration. An architecture for managing business processes life cycle is proposed in [31].None of these methodologies aims to support secure business processes.

We define *secure Web service-based business processes* as security-enhanced Web service-based business processes [159].

Georg et al. [93] propose the use of aspects for designing a secure system. The work illustrates how an aspect-oriented approach to modelling allows to encapsulate the concerns of security, availability of services and timeliness so they can be woven into a secure system design. The weaving strategy identifies security aspects based on the kinds of possible attacks and the mechanisms that allows the detection, prevention, and recovery from such attacks. Haley et al. [104] represent security requirements as crosscutting threat descriptions using aspect-oriented software development crosscutting concepts and problem frames. Security requirements are seen as constraints on functional requirements intended to reduce the scope of vulnerabilities. This allows to analyse secure requirements along with other

constraints when producing specification for the problem. Cheng et al. [45] propose the use of security patterns for modelling and analysing secure systems. The authors describe a collection of security patterns using a template that addresses difficulties inherent to the development of secure-critical systems. An approach to develop secure software with extensive pattern-driven process is presented in [107]. Employing the patterns, it is possible to gain insight into the issue of modelling and analysing security concerns starting from the requirements engineering phase. An extension of the Business Process Modeling Notation to enable a description of authorization constraints is presented in the work of Wolter and Schaad [199]. On the negative site, the approaches do not support the design of software and business processes based on a SOA.

Domingos at al. [67] proposes a methodology that allows deriving workflow access control information from business models. The approach adopts the Eriksson-Penker Business Extension to UML in order to describe business models. The obtained workflow access control information is represented as a set of rules in XML format. Unfortunately, the proposed methodology does not address the issue of workflow development and so usage of current standards for Web services and security. The problem of defining and enforcing access control rules for securing service invocations in the context of business processes is addressed in [189]. A novel security model called EFSOC (Event-driven Framework for SOC) is proposed. While the issue of delegation of authorization is not taken into account.

An approach for secure service composition is presented in [25]. A static approach determines how to compose services while guaranteeing that their execution is always secure, without resorting to any dynamic check. The proposed primitives can enforce local security policies and invoke services that respect given security requirements. The work does not focus on Web services or business process design.

Secure Tropos Framework

Secure Tropos is a formal framework and a methodology for modelling and analysing security requirements [95, 144]. Secure Tropos is an extension of the well established Tropos software engineering methodology [35].

Secure Tropos uses the concepts of actor and goal. *Actor* models an entity that has strategic interests, i.e., *goals* with the system. An actor represent a physical, social or software *agent* as well as its *role*. It might happen that an actor does not have the capabilities to achieve his own objectives by himself. In this case that actor has to delegate the objectives to other actors that leads to their achievement outside the control of the delegator. Secure Tropos supports two types of delegations. *Delegation of execution*, i.e, at-least delegation, means that one actor (called delegator) delegates to another one (called delegatee) the responsibility to execute a service. *Delegation of permission*, i.e, at-most delegation, models the transfer of entitlements from delegator to delegatee. Two types of trust dependencies are supported. *Trust of execution*, i.e, at-least trust, means that one actor (called trusted) trusts that another one (called trustee) will at least fulfill a service. While the meaning of *trust of permission*, i.e, at-most trust, is that an actor trusts that another actor will at most fulfill a service, but will not overstep it. Trust modelling aims at identifying actors trusting other actors for services, and actors which own the services.

From a methodological perspective, Secure Tropos is based on the idea of building a model of the system that is incrementally refined and extended. Specifically, goal analysis consists of refining goals and eliciting new social relationships among actors. They are conducted from the perspective of single actors using *AND/OR decomposition*. In case an actor does not have the capabilities to achieve his own objectives or assigned responsibilities by himself, he has to delegate them to other actors making their achievement

outside his direct control.

Various modelling activities contribute to the acquisition of the early requirements model, namely [95, 144]:

Actor modelling aims at identifying actors and analysing their goals.

Functional dependency modelling aims at identifying actors depending on other actors for obtaining services, and actors which are able to provide services.

Permission delegation modelling aims at identifying actors delegating to other actors the permission on services.

Trust modelling aims at identifying actors trusting other actors for services, and actors which own services.

The above constructs and modelling activities allow to capture the functional, security and trust requirements in a number of diagrams, namely:

Actor diagram describes objectives, entitlements and capabilities of each actor which are also analysed using goal refinement and contribution analysis techniques from the perspective of the actor.

Functional dependency diagram identifies the dependencies among actors, in particular, to which actor has been delegated the execution of which services by which actor.

Authorization diagram identifies the transfers of right among actors, in particular, to which actor has been delegated the permission, on which services and by which actor.

Trust diagram describes the expectations of actors about the behavior and capabilities of other actors in terms of trust of permission and trust of execution.

The examples of the diagrams based on the loan origination case study can be found in Section 3.1.1.

Chapter 3

Secure Workflow Development From Early Requirements

Requirements engineering is being increasingly adopted as a key step in the software development process and therefore new challenges and possibilities emerge. There are many requirements engineering frameworks for modelling and analysing security requirements, such as Secure Tropos [95, 144], UMLsec [117], MisuseCase [178], AntiGoals [190]. Designing of Web services and developing of Web service-based business processes and workflows is one of the most thought challenging issues in requirements engineering. There are several methodologies aiming at Web services, business processes and workflows design [165, 126, 166]. We claim that there is a gap between the requirements engineering methodologies and the actual production of software and business processes based on a SOA. Business processes and security issues are developed separately and often do not follow the same strategy [159]

The question is "How to obtain a secure workflow from the early requirements?". We address the issue of secure workflows modelling based on the analysis of early requirements, namely, Secure Tropos, by presenting a methodology that bridges the gap between early requirements and secure workflows for Web services development. We introduce a specifica-

tion language for secure business processes, which is a dialect of BPEL for the functional parts and which abstracts away low level implementation details from WS-Security and WS-Federation specifications. At the end, the deployment of a Secure BPEL process is described.

3.1 From Early Requirements to Secure Workflow

A secure business process is originated by the early requirements analysis and then is used for the development of an appropriate workflow. The process of deriving a secure workflow from early requirements is presented in Figure 3.1.

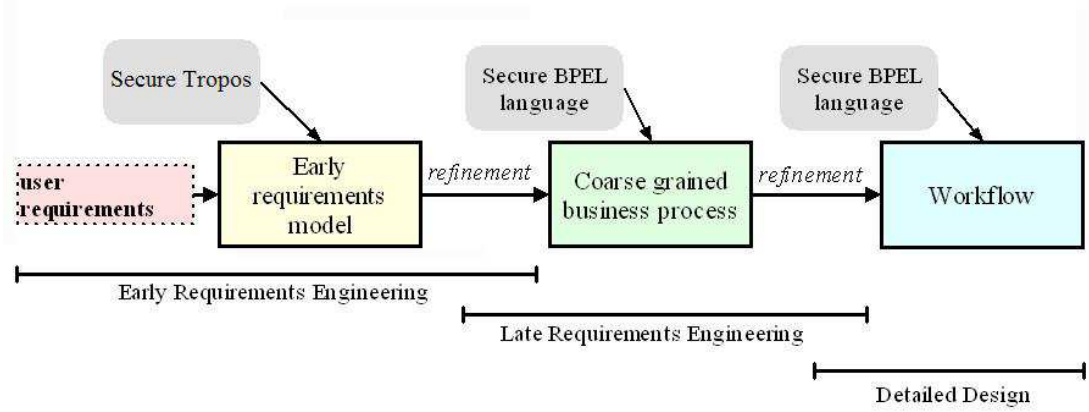

Figure 3.1: Relations among early requirements, business process and workflow levels.

The process includes three phases, namely, (1) early requirements engineering, (2) late requirements engineering and (3) detailed design. Detailed design is just further refinement adding more low level implementation details as a workflow is an implementation of business process.

BPEL offers constructs for orchestrating Web services into repeatable processes. WS-Trust is an extension of WS-Policy enabling the deployment and enforcement of "trust relationships" among partners. However, there is no way to enforce delegation requirements across Web services,

from the workflow execution language. We chose to leverage BPEL as this language appears as the most natural way to orchestrate independent organizations such as the different banking and Credit Bureau organizations considered in the example in Section 1.3. BPEL does not deal with authorization, trust and delegation, where Secure Tropos does. For the purpose of developing secure workflows based on the early requirements analysis, we propose a refinement methodology and a language Secure BPEL that enhances the BPEL language with constructs related to Secure Tropos, allowing the workflow engine to automatically enforce the trust and delegation requirements as introduced in the problem statement. Further we describe the phases of the process of deriving a secure workflow from early requirements in details.

3.1.1 Early Requirements Engineering

Early requirements engineering aims to analyse stakeholder interests, how they might be addressed or compromised by system requirements and understand the organizational context within which the system-to-be will eventually function [205, 35]. During the early requirements analysis phase, the domain actors and their dependencies on other actors for goals to be fulfilled are identifed. For early requirements elicitation, one need to reason about trust relationships and delegation of authority.

We employ the Secure Tropos modelling framework to derive and analyse both functional dependencies and security and trust requirements. The modelling activities presented in Section 2.3 contribute to the acquisition of the early requirements model, namely actor modelling, function dependency modelling, permission delegation modelling and trust modelling. A graphical representation, i.e, diagram, build according to these modelling activities is given respectively through the actor, functional dependency, authorization, and trust diagrams.

The early requirements model acquisition is depicted in Figure 3.2. The process starts from user requirements, goes thought actor, functional dependency, permission delegation and trust modelling and ends with actor, functional dependency, authorization, and trust diagrams, i.e., the requirements model obtaining.

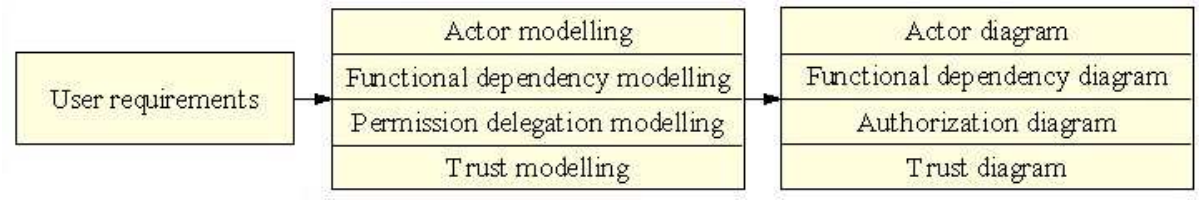

Figure 3.2: Early requirements model acquisition process.

Figure 3.3 and Figure 3.4 show the examples of the diagrams based on the loan origination case study proposed as a running example in Section 1.3. Actor and functional dependency diagram (see Figure 3.3) describes the actors (agents, depicted as circles with straight lines, and roles, depicted as circles with curves); some of the bank manager's goals, depicted as ovals; goal refinement by AND decomposition, depicted with a goal refinement symbol marked with AND; and the delegation of execution dependencies among bank manager, pre-processing and post-processing clerks, depicted with two lines connected by a delegation of execution (De) graphical symbol.

One of the variants of authorization and trust diagram is presented in Figure 3.4. The diagram identifies the actors, that participate, i.e, the BBB bank and bank manager, and involved services, i.e, the launch loan origination process goal, in delegation of permission, trust on permission and trust of execution dependencies, depicted with two lines connected by a delegation of permission (Dp), trust on permission (Tp) and trust of execution (Te) graphical symbols.

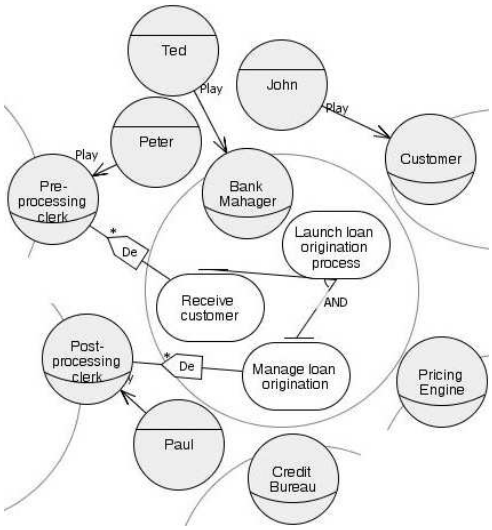

Figure 3.3: Actors and functional dependencies.

3.1.2 Late Requirements Engineering

Late requirements engineering is concerned with a definition of the functional and non-functional requirements of the system-to-be [54, 35]. During the late requirements analysis phase, the system-to-be is introduced within its operational environment. The requirements are to be detailed, modelled and analysed in the presence of non-functional requirements.

In this thesis the proposed refinement methodology aims to obtain an appropriate coarse grained business process and then workflow at the workflow level from early requirements. The obtained in the early requirements engineering phase early requirements model is refined by diagrams as presented in Figure 3.5. The methodology takes the components of the diagrams and derives a secure business process constructs from them. Then, the secure business process is described by the proposed Secure BPEL language.

The relevant components of actor diagram are actors, goals and spawning of dependency relationships among actors. In functional dependency

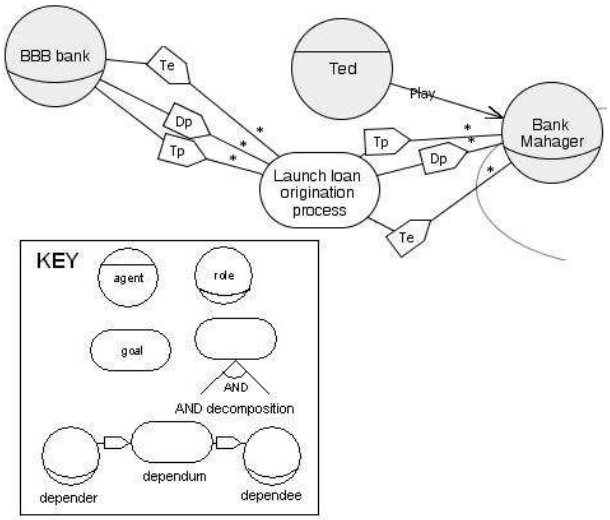

Figure 3.4: Authorization and trust.

diagram, we consider dependencies among actors that delegate or are delegatees of execution of services. The components of authorization diagram are transfers of right among actors that delegate or are delegatees of permission on services. In trust diagram we consider the expectations of actors about the behavior and capabilities of other actors in terms of trust on permission and trust on execution.

Secure BPEL language is an extension of Web Services Business Process Execution Language (WS-BPEL v2.0) [9] that allows for secure business processes specification. The Secure BPEL was firstly introduced in [81, 82] and then described in details together with the deployment of a Secure BPEL process in [176].

The proposed language is an extension of standard business process specification language. Hence, if a business process designer is familiar with WS-BPEL processes, he simply needs to understand the additional constructs introduced by Secure BPEL. We suffix each new or refined construct with the keyword "S" to clearly distinguish them. At the workflow level, the Secure BPEL process will then be refined into a combination of

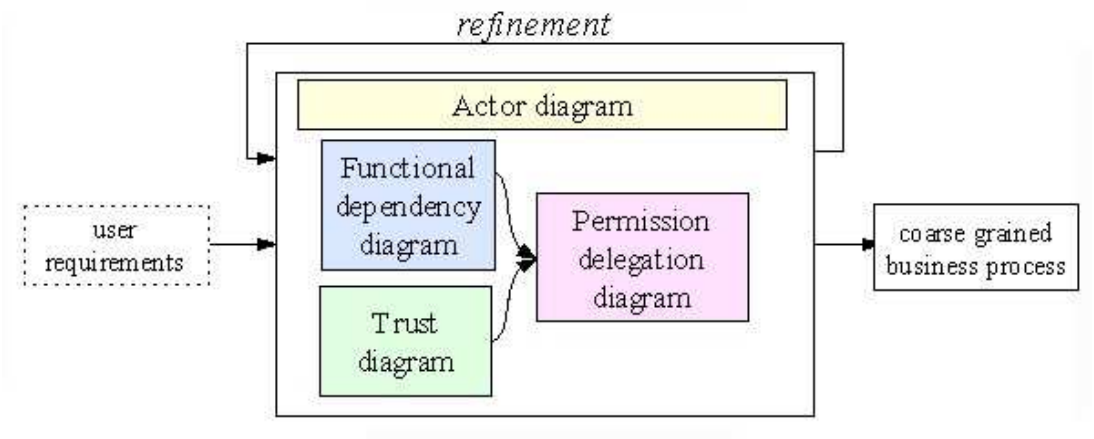

Figure 3.5: Late requirements engineering.

standard BPEL and WS-Security policy for process execution.

Presenting the proposed methodology phases, we provide details of the Secure BPEL constructs and explain the context in which the language is used.

Figure 3.6 presents two steps of actor diagram refinement. In first step, partners are designed based on the actors identified in the early requirements engineering stage. We assume that each actor has a single root goal that can be decomposed by AND/OR goal decomposition. Each AND/OR goal decomposition lead to operationalization phase. The second step considers partner and orchestration specification by the Secure BPEL language. Operationalization is completed with additional information to AND/OR goal decomposition on choice of sequential or parallel operation.

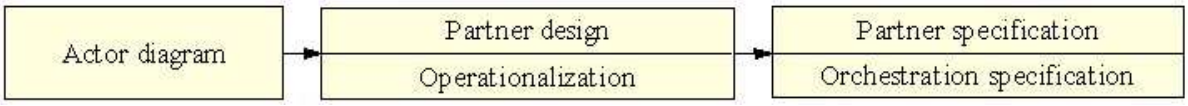

Figure 3.6: Actor diagram refinement.

The other diagrams refinement is done in the similar way. The idea is that in the first step dependencies (for functional dependency and au-

thorization diagrams) or trust (for trust diagram) and choreography are designed and in second step choreography is specified. Here we consider that the level of goals is the level of services.

The table in Figure 3.7 presents the diagrams to Secure BPEL language notions refinement. Considering actor diagram, the notion of actor is refined into partner in Secure BPEL, a root goal is refined into business process while AND/OR goal decomposition with delegation are refined into orchestration. The notions of delegation of execution and delegation of permission presented in dependency and authorization diagrams are refined into choreography of services and authorization respectively. As for trust diagram, trust on execution and permission are refined into choreography of attestation that is further refined into attestation of integrity for the notion of trust on execution and attestation of reporting for trust on permission.

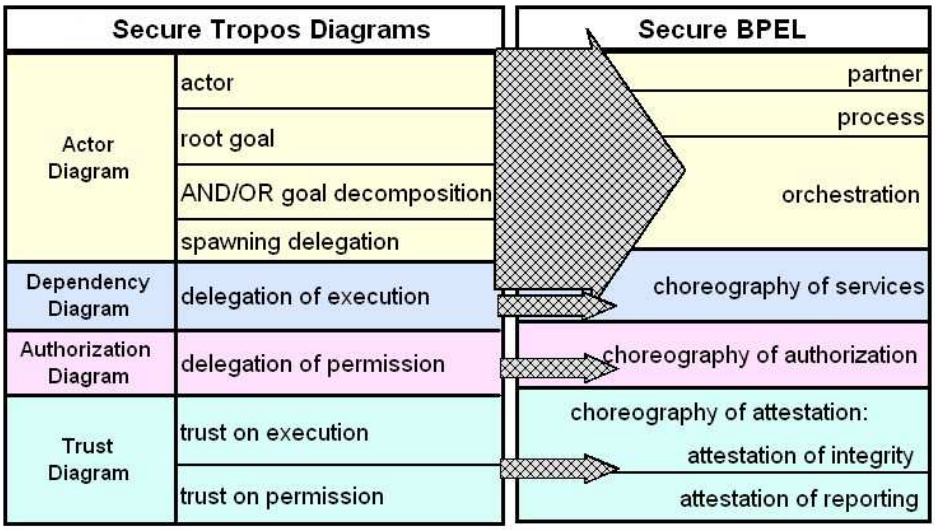

Figure 3.7: Tropos diagrams to Secure BPEL.

Refining Actor Diagram

Actors identification consists in identifying all actors, i.e., agents and roles, involved in a business process and all roles played by all the agents identified. The concept of actors at the business process level is refined as partners and specified in Secure BPEL by the `<partnerS>` construct (see Figure 3.8).

```
<partnersS>
    <partnerS nameS = "agentName">+
      roles played by agent
    </partnerS>
</partnersS>
```

Figure 3.8: Actor identification.

To ease the language specification we provide a slight extension to the WS-BPEL v2.0 standard by retaining the `<partner>` construct from the Business Process Execution Language for Web Services (WS-BPEL v1.1) [12]. While such extension is not necessary for actually writing down the workflow solution (because each partner role is specified on every individual invocation), it is extremely convenient at the requirements level because it offers a compact view of who is doing what. Further, at this stage, we also need to identify which agent has to run which process and hence the addition of the `nameS` attribute.

Each partners interaction at the business process level is specified by the `<partnerLinkS>` construct[1] and specifying all roles played by a partner (see Figure 3.9). The role of the partner itself is indicated by the attribute `myRole` and the role of the companion is indicated by the attribute `partnerRole` within the `<partnerLinkS>` construct. When there is only

[1] As we work at a high level of abstraction, at this point some workflow details are not considered. Most notably, we do not specify partner link types that characterizes the conversational relationship between two partners by defining the roles played by each of the partners in the conversation.

one role, one of these attributes is omitted as appropriate. The partner is identified by the `partnerNameS` attribute. Each `partnerLinkS` is named and this name is used for all service interactions via that `partnerLinkS`.

```
<partnerLinksS>
    <partnerLinkS name="partnerLinkName"
      myRole = "myRoleName"?
      partnerNameS = "agentName"?
      partnerRole = "partnerRoleName"?>+
    </partnerLinkS>
</partnerLinksS>
```

Figure 3.9: Actor description.

Example 1 *According to the first scene of the loan origination case study presented in Section 1.3, in the actor identification step, two agents (specified by the* ***<partnersS>*** *construct) are identified, namely, John and Peter (the* ***nameS*** *attribute of the* ***<partnerS>*** *construct). For the partners interaction at the business process level (the* ***<partnerLinkS>*** *construct), the agents roles are described (the* ***myRole*** */* ***partnerRole*** *attribute within the* ***<partnerLinkS>*** *construct). Partner John has a role customer. John is a partner of Peter whose role is a pre-processing clerk. In such manner it is possible to identify and describe all actors presented in the case study.*

The concept of actor is specified in the Secure Tropos metamodel [183] as an agent can play several roles. In Secure BPEL metamodel [81] the concept of partner and role are specified. One secure business process can be composed of several partners. While to each partner can be associated one or more partner links that specify all roles played by a partner. The role of a partner itself is specified by the `myRole` attribute and the role of the companion is indicated by the `partnerRole` attribute.

Structured Activities

Structured activities is a basis of orchestration specification and consist of a sequential/parallel composition and branching statement. The notion of sequential and parallel composition corresponds to a refinement of the concept of AND goal decomposition. Branching statement is a refinement of the concept of OR goal decomposition.

Sequential composition is specified by the `<sequence>` construct. The construct defines a collection of activities to be performed sequentially, in the lexical order in which they appear within the construct. Parallel composition is specified by the `<flow>` construct. The construct defines one or more activities to be performed concurrently. While branching statement is specified by the `<if>` construct that is used to select exactly one activity for execution from a set of choices.

Example 2 *Following the loan origination case study presented in Section 1.3, all the main activities are sequential. The following activities: customer identification, check rating, calculation of the price and signature of the contract are done in a sequential way. At the business process level, the process defining these activities in the sequential order, is implemented by the* `<sequence>` *construct.*

In the second scene of the case study presented in Section 1.3, the process of checking the credit worthiness is divided in two parallel subprocesses: the external part (provided by Credit Bureau) and the internal one (based on internal scoring). Nevertheless the internal one is stopped when the results coming from the Credit Bureau are negative. At the business process level, the process is implemented by the `<flow>` *construct.*

Refining Functional Dependency Diagram

Dependencies derived from a functional dependency diagram are notably delegation of execution. The refinement process starts with abstract goals and ends up with concrete atomic activities at the business process level, while those activities can be further refined at the workflow level. Here we consider that the level of goals is the level of services. Atomic activities is a basis of choreography specification and consist of the service invocation activities and the response to a service invocation activities.

Considering one particular dependency, invocation of a service by a depender is specified by the `<invoke>` construct (see Figure 3.10).

```
<invoke
    <partnerLink = "partnerLinkName"
      operation = "operationName">
</invoke>
```

Figure 3.10: Service invocation.

Responding to a service invocation by a dependee is specified by the `<pick>` construct (see Figure 3.11). The construct allows to block and wait for a suitable message to arrive, i.e., a message of service invocation. When the message arrives, the associated activity, i.e., service execution, is performed and the pick completes.

```
<pick
    <on message partnerLink = "agentName"
      execute service
      </onMessage>
</pick>
```

Figure 3.11: Response to service invocation.

The concept of delegation of execution at the business process level is

refined as a process that consists of invocation of a goal (service), from one partner, i.e., a depender and other partner's, i.e, dependee, acceptance of the delegation and execution of the goal.

Example 3 *The concept of delegation of execution is seen in some scenes of the loan origination case study presented in Section 1.3. In particular, in the first scene, John as a bank customer delegates the function of processing the loan origination to the bank. Then the bank delegates the identification of the customer to Peter, the pre-processing clerk, and delegates the managing of the loan origination process to Maria, the post-processing clerk. In the second scene, Maria delegates the credit worthiness check, in particular, external rating analysing, to the Credit Bureau. At the business process level the delegation process of credit worthiness check to the Credit Bureau is a follows. At the delegator side, the partner Maria invokes the operation "credit worthiness check" (by the* `<invoke>` *construct) from the partner Credit Bureau. While at the delegatee side, the partner Credit Bureau, the delegatee responds to a service invocation (see the* `<pick>` *construct) accepting the message of service invocation and executes the "credit worthiness check" goal.*

Refining Authorization and Trust Diagrams

Interactions with partners can be more complicated than delegation of execution represented by the atomic activities. There is a set of activities to represent the Secure Tropos concepts of delegation of permission, trust on execution and trust on permission at the business process level. This set includes request/response for authentication token, authorization token, attestation of integrity and attestation of reporting.

The concept of attestation characterizes the process of vouching for the accuracy of information [187]. In this work we use two types of attestation,

i.e, attestation of integrity and attestation of reporting. Attestation of integrity provides proof that an actor can be trusted to report integrity and performed using the set or subset of the credentials associated with the actor. Attestation of reporting is the process of attesting to the contents of integrity reporting.

The `<RequestSecurityServiceS>` construct is used to request a token for the purpose of authentication, authorization, attestation of integrity and attestation of reporting. The syntax for the construct is presented on Figure 3.12.

```
<requestSecurityServiceS>
    <typeS>
     typeS="Authentication|Authorization|
            Attestation-Integrity|
            Attestation-Reporting"
    </typeS>
    <purposeS>
     goalName+
    </purposeS>
    <participantsS>+
     <participantS nameS = "agentName">
     <participantS>
    </participantsS>
    <onBehalfOfS>... </onBehalfOfS>
    <usageS> ... </usageS>
</requestSecurityServiceS>
```

Figure 3.12: Request security service.

The following describes the attributes and elements listed above:

/requestSecurityServiceS/typeS This element describes the type of security service requested, i.e., authentication, authorization, attestation of integrity and attestation of reporting. That is, the type of

the service that will be returned by the `<requestSecurityServiceResponseS>` construct.

/requestSecurityServiceS/purposeS This element specifies the scope for which the security service is desired, i.e., the goal to which the service applies.

/requestSecurityServiceS/participantsS/ This element specifies the participants sharing the security service. This attribute is used by the requestor to clarify the actual parties involved.

/requestSecurityServiceS/participantsS/participantS This element specifies participant (or multiple participants) that play a role in the use of the service or who are allowed to use the service.

/requestSecurityServiceS/onBehalfOfS This element indicates that the requestor is making the request on behalf of another.

/requestSecurityServiceS/usageS This element specifies a policy (as defined in WS-Policy) that indicates desired settings for the requested service such as `<delegatable> true|false </delegatable>`.

The `<RequestSecurityServiceResponseS>` construct is used to return a security service or response to a security service request. It should be noted that any type of parameter specified as input to a service request may be present on response in order to specify the exact parameters used by the issuer. The syntax for this construct is similar to the one presented on Figure 3.12. The only difference is in the additional `<requestedSecurityServiceS>` element that is used to return the requested security service.

Example 4 *As we shown in the example on delegation on execution, the concept of delegation of execution is seen in some scenes of the loan origination case study presented in Section 1.3. This example aims to show the*

concept of delegation of permission by using the first scene of the case study. The bank delegates the identification of the customer to Maria the pre-processing clerk. At the business process level, from the delegator side, the type of security services requested is authorization (specified with the `<typeS>` *element), the purpose is "customer identification" (by the* `<purpose>` *element) and the participant is Peter (by the* `<participant>` *element), see Figure 3.12. From the delegatee side, the* `<requestSecurityServiceResponseS>` *construct is used to respond to the security service request with the purpose (by the* `<purpose>` *element) "customer identification" and the participant is Maria (by the* `<participant>` *element).*

Following the second scene of the case study, the post processing clerk trusts the Credit Bureau to give trustworthy external rating, i.e, trust on permission concept. At the business process level, from the truster side, the type of security service requested is authentication (specified with the `<typeS>` *element) with the goal check external rating (with the* `<purpose>` *element) and participant Credit Bureau (with* `<participant>`*). From the trustee side, the* `<requestSecurityServiceResponseS>` *construct is used to answer to the security service request with the* `<purpose>` *check external rating and the* `<participant>` *post-processing clerk. After this step, from the truster side, the type of security service is attestation of integrity (specified with the* `<typeS>` *element) with the goal check external rating (with the* `<purpose>` *element) and participant Credit Bureau (with* `<participant>`*). From the trustee side, the* `<requestSecurityServiceResponseS>` *is used to answer to the security service request with the* `<purpose>` *check external rating and the* `<participant>` *post-processing. The concept of trust on execution is considered in the second scene of the case study. At the business process level, the process is very similar to the one presented above for trust on*

permission. The only one difference is the type of the security service involved, which is attestation of reporting in the second step.

Secure BPEL metamodel where the concept of activity is specified is presented in [81]. Activity is composed of partner activity and structured activity. Partner activity, in its turn, consists of the `invoke`, `pick`, `request security serviceS`, and `request security service responseS` activities. While structured activity is composed of the `sequence`, `flow`, and `if` activities.

3.2 Deploying a Secure BPEL Process

The example presented in Section 1.3 is about a classical loan origination process, where each group represents a different organization, and where each swimlane represents a different responsible authority. We have seen that Secure BPEL offers a way to enforce the least privilege principle, namely with respect to delegation of permissions, from the design time. At process start, no user is given any right. The rights are being granted according to the state of the Secure BPEL workflow and revoked upon task or process termination or failure. WS-SecurityPolicy and/or XACML offer means to enforce access control. Secure BPEL offers means to delegate authorization according to workflow instances, that is, in a certain context defined by the workflow history. The delegation of permission is implemented by allowing Secure BPEL processes to alter the security policy on the fly (or via a similar mechanism), by offering specific authorizations in the context of the workflow execution - thus restricting the rights of the services to a minimum.

The problems coming with the deployment of cross-organizational processes, such as the one proposed in the example, are being addressed in

[198]. The IST-FP6-R4eGov project[2], which is the project in which the previous reference was written, chose to deploy these distributed processes using BPEL. To secure the execution of the processes, the evaluation of WS-Security and WS-Conversation has been presented by the project. In this thesis work, we propose to further leverage this approach by tailoring the policy files of each involved organization according to the Secure BPEL process.

The late requirement engineering phase is about generating corresponding BPEL files, realizing the process as described in the early requirements model, where the delegatee's process matches the delegator's one in terms of BPEL service invocation/message pick constructs.

These process chunks are then to be deployed at each organization level. It is assumed that each of these organizations run their services under WS-Trust. The deployment itself corresponds to a final phase, where the Secure BPEL file of the considered organization is refined into two separate artifacts. One of them is a standard BPEL file, to be deployed on to the BPEL engine of the organization. The second one is a policy file, to be used as input for the enforcement of the WS-Security protocol.

Let's now illustrate the execution with our example, across organizations as well as inside one organization (see Figure 3.13). The natural sequence of actions is the following: a Customer makes a request to a Clerk organization. The latter will perform internal tasks and will in turn ask the Credit Bureau for information. In this example, there is a possible threat that the Clerk organization asks the Credit Bureau about sensitive information from any customer he seeks, where the customer would have made no request for a loan in the first place.

With the distributed Secure BPEL paradigm, by default the Credit Bu-

[2]R4eGov (Research for eGovernment) is a research project supported by the European Commission. R4eGov helps tackle one of the major challenges facing eGovernment in Europe today - the ever increasing mobility of people and transactions across and within national boundaries, http://www.r4egov.eu.

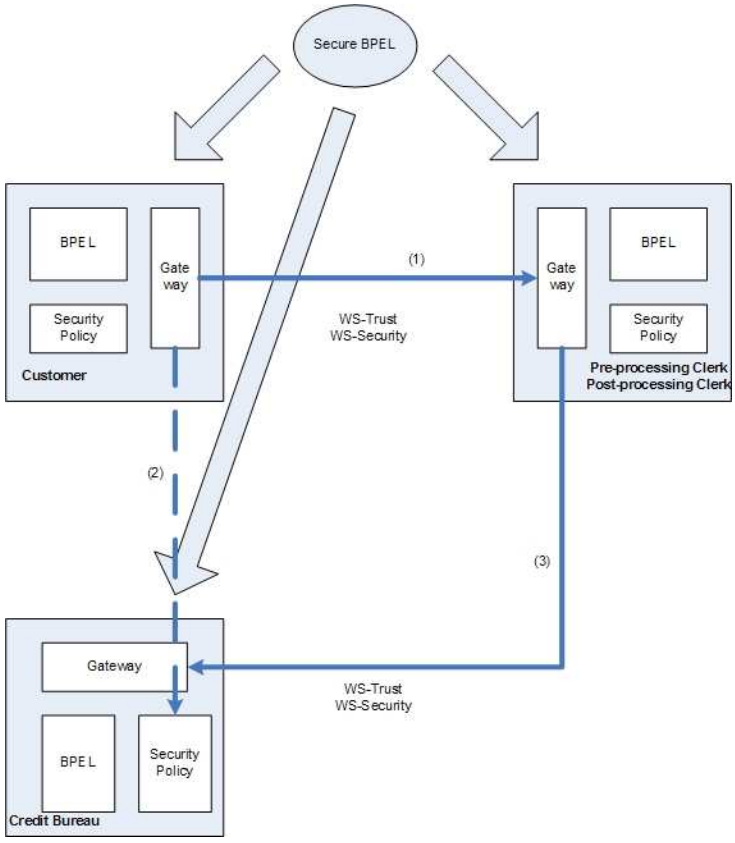

Figure 3.13: Secure BPEL deployment.

reau forbids any request for credit worthiness check. When the Customer instantiates his BPEL process, the corresponding workflow task will invoke the Pre-Processing Clerk (1). At the same time a message will carry over information to the partners of the collaboration that the Customer requests a loan, and that this loan request will be in the end delegated to a specific Post-Processing Clerk (2). According to this information, the Credit Bureau of the collaboration will edit its WS-Security policy in order to accept the request for wealthiness check from the Post-Processing Clerk (3). The details about information spreading (2) is not covered in this work and should be further researched on; it was inspired from the administrative communication channel presented in [198] and is considered at the time of writing as a good candidate for such dissemination.

For what concerns intra-organizational aspects, the same concept can be applied inside the Clerk organization, where separation of duties must be enforced between the Pre-Processing Clerk and the Post-Processing Clerk. As Secure Tropos permits the explicitation of SoD properties[3], the derived Secure BPEL process will then automatically enforce the security property at runtime. The policy aspects in this case are enforced locally, thus complementing the cross-organizational security aspects of the designed process. Actual enforcement of SoD requires the security policy to rely on to an extra component logging and spreading some of the workflow history.

3.3 Concluding Remarks

One of the most thought challenging issues in requirements engineering is that of designing Web services and developing of business processes and workflows for Web services. The research on Web services design is well under way, but the existing design methodologies for Web services do not address the issue of developing secure Web services, secure business processes and secure workflows.

The main contribution of the current chapter is to bridge the gap between early requirements analysis and secure workflows for Web services development. In particular, we have proposed a methodology that allows a designer of a business process to derive the skeleton of the concrete secure business processes based on the early requirements. Furthermore, the secure business processes are refined in order to obtain the appropriate secure workflows that can be described by the proposed specification language for secure requirements called Secure BPEL.

By executing Secure BPEL processes through a collaborative workflow

[3]See Separation of Duties pattern defined in the Serenity Project: http://www.serenity-forum.org/Work-package-1-3.html

runtime architecture, we are able to further restrict authorization of execution down to the least privilege principle at a cross-organizational perspective, as well as to unify inter and intra-organizational security aspects in a single process design. We achieved this result by introducing, via the design of a Secure BPEL process, a context notifications to the request at hand, where the security properties are defined at design time in the formal model of Tropos.

Chapter 4

Semantics and Extensions of WS-Agreement

When having repeated interactions with a service provider, a service consumer might desire guarantees on the delivery of the service. These guarantees involve both functional and non-functional properties of the offered service over a number of invocations. When the guarantee terms are explicitly defined in a document, we talk about a service level agreement.

WS-Agreement is an an industry driven emerging extensible markup based language and protocol for advertising the capabilities of providers, creating agreements based on initial offers, and for monitoring agreement compliance at run-time in the context of Web services. The motivations for the design of WS-Agreement stem out of QoS concerns, especially in the context of load balancing heavy loads on a grid of Web service enabled hosts [74].

Though, WS-Agreement only specifies the XML syntax and the intended meaning of each tag, which naturally leads to posing the question of "What's in an Agreement?" We answer this ques mtion by providing a formal definition of WS-Agreement by resorting to finite state automata, we provide a set of formal rules that tie together agreement terms and the life-cycle of an agreement. From the analysis, some shortcomings of the

protocol become evident. Most notably, the protocol does not contemplate the negotiation of the agreement itself, furthermore, there is no checking of how close a term is to being violated and, even more, breaking one single term of the agreement results in terminating the whole agreement, while a more graceful degradation is desirable. To overcome these shortcomings, we propose an extension of WS-Agreement for which we provide appropriate semantics, that allows (i) early warnings before agreement violation, and (ii) negotiation and possibly re-negotiation of running agreements.

Furthermore, we compare service level agreements and service licenses. Although an agreement is rather different from a license, they both regulate the activities of collaboration services. A basic difference is the fact that an agreement involves at least two parties, while a license is a unilateral statement. Nevertheless, for a license to be enacted, there must be at least a consumer of the service: this is the starting motivation to relate SLA and service licenses. We apply the proposed analysis to service licenses and propose the phases of a service license lice cycle.

4.1 WS-Agreement

In order to be successful, Web service providers have to offer and meet guarantees related to the services they develop. Taking into account that a guarantee depends on actual resource usage, the service consumer must request state-dependent guarantees from the service provider. Additionally, the guarantees on service quality must be monitored and service consumers must be notified in case of failure of meeting the guarantees. An agreement between a service consumer and a service provider specifies the associated guarantees. The agreement can be formally specified using the WS-Agreement Specification [13].

A WS-Agreement is an XML-based document containing descriptions of

the functional and non-functional properties of a service oriented application. It consists of two main components that are the agreement Context and the agreement Terms (see Figure 4.1). The agreement Context includes the description of the parties involved in the agreement process, and various metadata about the agreement. One of the most relevant components is the duration of the agreement, that is, the time interval during which the agreement is valid.

Figure 4.1: WS-Agreement structure.

Functional and non-functional requirements are specified in the Terms section that is divided into Service Description Terms and Guarantee Terms. The first provides information to define the services functionalities that will be delivered under the agreement. An agreement may contain any number of Service Description Terms. An agreement can refer to multiple components of functionalities within one service, and can refer to several services. Guarantee Terms define an assurance on service quality associated with the service described by the Service Description Terms. An agreement may contain zero or more Guarantee Terms. A Guarantee Term consists

of several parts, namely:

/GuaranteeTerm/ServiceScope is the list of service names a guarantee applies to;

/GuaranteeTerm/QualifyingCondition is an optional condition that expresses a precondition under which a guarantee holds;

/GuaranteeTerm/ServiceLevelObjective is a condition that must be met to satisfy the guarantee;

/GuaranteeTerm/BusinessValueList is a list of business value elements associated with a service level objective.

In [57], a definition for guarantee terms in WS-Agreement is specified and a mechanisms for defining guarantees is provided. An agreement creation process starts when an agreement initiator sends an agreement template to the consumer. The structure of the template is the same as that of an agreement, but an agreement template may also contain a Creation Constraint section, i.e., a section with constraints on possible values of terms for creating an agreement. In [15], enabling of customizations of terms and attributes for the agreement creation is proposed. After the consumer completes in the template, they send it to the initiator as an offer. The initiator decides to accept or reject the offer depending on the availability of resource, the service cost, and other requirements monitored by the service provider. The reply of the initiator is a confirmation or a rejection. A draft of the Web services Agreement Negotiation Specification can be found in [14]. WS-AgreementNegotiation is a protocol for negotiation of agreements based on the WS-Agremeent specification.

An agreement life-cycle includes the negotiation, implementation, termination and monitoring of agreement states. Figure 4.2 shows a representation of the life-cycle. When an agreement is implemented, it does

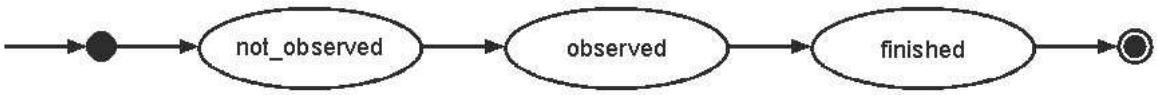

Figure 4.2: The life-cycle of a WS-Agreement.

not imply that it is monitored. It remains in the `not_observed` state until services start their execution. The semantics of the states is as follows:

- `not_observed`: the agreement is created and is in execution, but no service involved in the agreement is running; and
- `observed`: at least one service of the agreement is running;
- `finished`: the agreement terminates either successfully or not.

4.2 What's in an Agreement?

The WS-Agreement specification provides XML syntax and a textual explanation of what the various XML tags mean and how they should be interpreted. Thank to the syntax, it is possible to prepare machine readable agreements, but a formal notion of agreement is missing. In this section, we formalize the notion of agreement by defining its main components.

Definition 1 (Term) *A* term t *is a couple* (s, g) *with* $s \in S$ *and* $g \in G$*, where* S *is a set of* n *services and* G *is a set of* m *guarantees.* $T \subseteq S \times G$ *is the set of the terms* t*.*

In words, a term involves the relationship between a service s and a guarantee g, not simply a specific tag of the agreement structure. If the service s appears in the list of services, which the guarantee g is applied to, it means that the couple (s, g) is a term. The number of terms varies

between 0 and $n \cdot m$, where 0 means that there is no association between services and guarantees, and $n \cdot m$ indicates the case where each guarantee is associated with all services.

Definition 2 (Agreement) *An* agreement A *is a tuple* $\langle S, G, T \rangle$*, where* S *is a set of* n *services,* G *is a set of* m *guarantees, and* T *is the set of the terms* t*.*

In the following analysis, it is more convenient to consider the agreement as a set of *Terms* rather than a set of related services and guarantees. From the definition of WS-Agreement, we say that an agreement can be in one and only one of three states: `not_observed`, `observed` and `finished`.

Definition 3 (External state) *The* external state A_{es} *of an agreement* A *is an element of the set* {`not_observed`, `observed` *or* `finished`}*.*

We call the above state external, as it is the observable one. We also define an internal state of an agreement, which captures the state of the individual terms.

Definition 4 (Internal state) *The* internal state A_{is} *of an agreement* A *is a sequence of terms' states* $ts_1, \ldots, ts_p$ *of maximum size* $n \cdot m$*, where* $ts_i = (ss_j, gs_k)$ *represents the state of* g_k *guarantee with respect to the state of the* s_j *service. Service and guarantee states range over the following sets, respectively:*
$ss_j \in$ {`not_ready`, `ready`, `running`, `finished`}*, and*
$gs_k \in$ {`not_determined`, `fulfilled`, `violated`}*.*

From the definition of *Term*, we see that services and guarantees are related and we can define the internal state of an agreement, but it is necessary to distinguish between terms that have the same service and terms that have the same guarantee.

Proceeding in our goal of answering the question of what is in an agreement, we define the relationship between the internal and external state of an agreement A. First, we note that not all state combinations make sense. For instance, it has no meaning to say that a guarantee is `violated`, when a service is in a `not_ready` state. The only admissible combinations are the following ones:

(1) (`not_ready`, `not_determined`)
(2) (`ready`, `not_determined`)
(3) (`running`, `fulfilled`)
(4) (`running`, `violated`)
(5) (`finished`, `fulfilled`)
(6) (`finished`, `violated`)

In theory, there are 63 possible combinations of states in which terms can be. That is, $\sum_{i=1}^{6} \begin{pmatrix} 6 \\ i \end{pmatrix}$ all terms could be in state (1), or in state (2),... or in state (6); there could be terms in states (1) and (2), (1) and (3), and so on. But again, considering the definition of WS-Agreement in [13], one concludes that not all 63 combinations make sense. Furthermore, it is possible to extract the possible evolutions of these aggregated internal states.

When an agreement is created its external state is `not_observed`, while all services are `not_ready` and all guarantees are `not_determined`, i.e., state (1). In the next stage some services will be `ready` while others will still be `not_ready`, i.e., there will be terms in state (1) and (2). In this case, the external state is also `not_observed`. Proceeding in this analysis, one can conclude that there are 8 situations in which terms can be. We summarize these in the table in Figure 4.3. In the table, we also present the relation between the internal states and the external states, and the set of transitions to go from one set of states to another. The latter transitions

	terms' state	agreement's state	transitions
A	(1)	`not_observed`	B
B	(1)(2)	`not_observed`	C E
C	(1)(2)(3)	`observed`	D E F G
D	(1)(2)(3)(5)	`observed`	F G
E	(1)(2)(4)	`observed`	F H
F	(1)(2)(3)(4)(5)	`observed`	H
G	(5)	`finished`	
H	(1)(2)(3)(4)(5)(6)	`finished`	

Figure 4.3: Transition table for the relation between internal and external states.

are best viewed as an automaton.

Referring to Figure 4.4, at the beginning all the terms are tied to services which are not running (A). At some point, some services will be ready to start (B). Services which are ready will start execution. This may result in an immediate violation of a term (E), or in executions fulfilling the term (C). If the latter is the case, more and more services will execute. This may result in violations, which bring us to states (E) or (F), or in no violation. Some services may successfully terminate execution, case (D). If all services terminate with no violation, we end successfully in state (G). If any service has a violation at any time, we end in state (E) or (F) and from there, unavoidably, in state (H), which is a failure state.

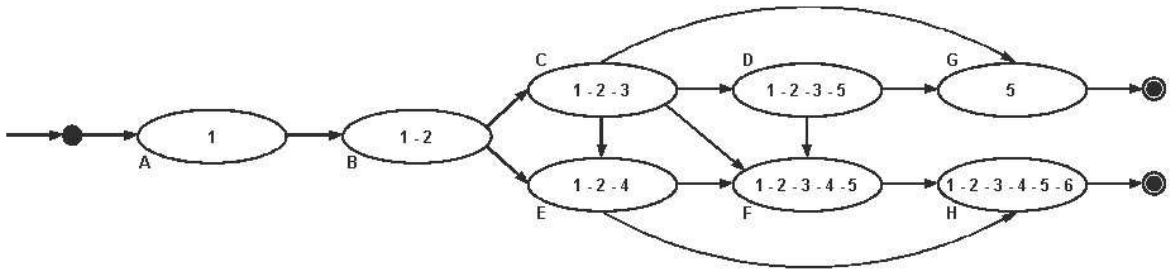

Figure 4.4: Automaton representation of the table in Figure 4.3

4.3 Extension of WS-Agreement

From the semantics and formal analysis presented in Section 4.2, inspecting the automaton provided, we note that if the agreement arrives into the states (E) or (F) there is a non recoverable failure, and consequently an agreement termination. Even if one single term is violated, the whole agreement is terminated. Furthermore, when an agreement is running there is no consideration on *how* the guarantee terms are fulfilled. Our goal is to provide an extension of WS-Agreement and of its semantics in order to make agreements more long-lived, and robust to individual term violations.

We propose to extend WS-Agreement. On the one hand, one can **(i) anticipate violations**; on the other hand, the **(ii) negotiation** of the SLA should be part of its life-cycle. In particular, there is an **(ii.a) initial negotiation** before the execution of the services under SLA, and a **(ii.b) run-time re-negotiation** which occurs in case of a recoverable violation of a term or in case the monitoring system is anticipating a possible violation of a term. **(i)** In WS-Agreement guarantees of a running service are either fulfilled or violated. Nothing is said about how a guarantee is fulfilled. Is the guarantee close or far from being violated? Is there a trend bringing the guarantee close to its violation? We propose to introduce a new state for the agreement in which a warning has been issued due to the fact that one or more guarantees are likely to be violated in the near future. By detecting possible violations, one may intervene by modifying the run-time conditions or might re-negotiate the guarantees which are close to being violated. **(ii)** The negotiating phase occurs in two moments of the life-cycle of the agreement. In the initial phase the service provider and consumer, must agree on what the conditions of the agreement are. The WS-Agreement specification does not focus on parties involved in the

agreement process interactions leading to negotiation of QoS parameters, at most one can use pre-compiled templates. Furthermore, during the execution of the services under agreement, re-negotiation may occur when conditions vary or terms are violated or could be violated in the near future. The WS-Agreement specification does not contemplate changing an agreement at run-time, i.e., re-negotiation. If a guarantee is not fulfilled because of resource overload or faults in assigning available resources to consumers, the agreement must terminate. For maintaining the service and related supplied guarantees, it is necessary to negotiate the QoS again and create another agreement. This approach wastes resources and computational time, and increases network traffic. The goal of negotiation terms applying is to have the chance to modify the agreement rather than respecting the original agreement. Applying the negotiation terms means that the services included in the agreement will be performed according to the new guarantees.

4.3.1 Life-Cycle and Semantics for the Extended Agreement

To obtain the desired extensions, we expand the set of states in which an agreement and a guarantee term can be and thus update the transition system. More precisely, the definition of an agreement does not change with respect to Definition 2, the difference lies in the fact that the set of terms T is now extended with special *negotiation terms*. These terms are defined as in Definition 1, but have a different role, i.e., they specify new conditions that enable modification of guarantees at run-time.

To account for the new type of terms, we need to extend the definition of external and internal state of an agreement. The external states of an extended agreement are enriched by the `negotiated` state, the `checked` state, the `warned` state, the `re-negotiated` state, and the `denied` state. We say that an agreement can be in one of nine states. `not_observed`, `observed`

and `finished` have the same meaning as in WS-Agreement, Figure 4.5. An agreement is in the `negotiated` state while the negotiation process. From the `negotiated` state the agreement can go to the `not_observed` state if the agreement is accepted by all the parties or to come abruptly to an end if it is rejected. An agreement is in the `re-negotiated` state while the re-negotiation process. From the `re-negotiated` state the agreement goes to `observed`. An agreement is in state `checked` when the monitoring system is checking its services and guarantees. From the `checked` state the agreement can go to five different states: to `finished` if the agreement finishes its life-cycle; to `denied` if the agreement is violated and no negotiation terms can be applied, the agreement must terminate; to `warned` if the monitoring system issues at least one warning for at least one term; back to `observed` if the agreement is fulfilled; to `re-negotiated` if the agreement is fulfilled or violated and negotiation terms can be applied.

Definition 5 (Extended External state) *The* extended agreement external state A_{xes} *of an agreement A is an element of the set* {`negotiated`, `not_observed`, `observed`, `warned`, `checked`, `re-negotiated`, `denied` *or* `finished`}.

The transitions between states are illustrated by the automaton in Figure 4.5, which is an extension of the one presented in Figure 4.2. The automaton represents the new evolution of an agreement where a guarantee are negotiated and can be modified during the processing of a service or a warning can be raised. When a guarantee is violated we have two situations: the first presents a recoverable violation which implies the chance to apply negotiation terms and so the agreement is in the `re-negotiated` state, the second presents a non recoverable violation which implies that there is no suitable negotiation term for the current violated guarantee and so the agreement must terminate. Otherwise, if a warning is raised, this

can be ignored or the agreement can go in the `re-negotiated` state. Also, when a guarantee is fulfilled, it is possible to change the current agreement configuration, applying a negotiation term that changes the QoS.

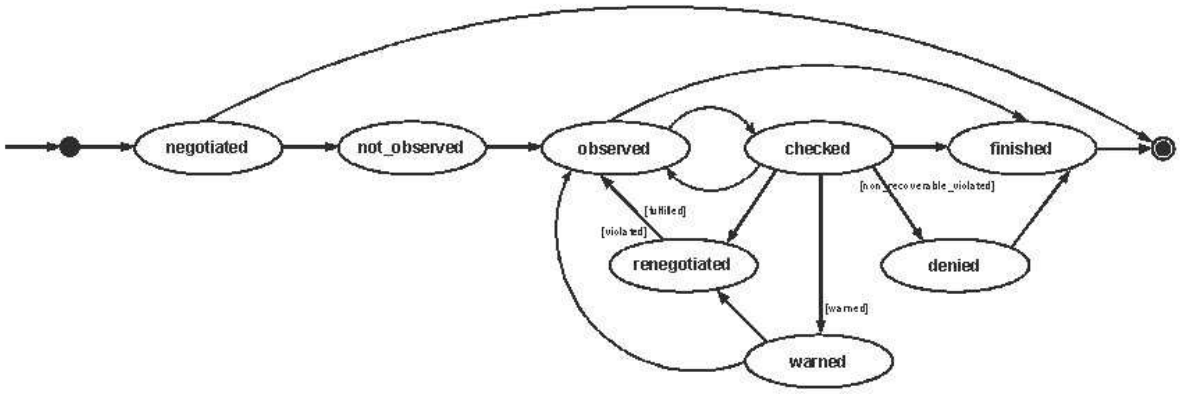

Figure 4.5: The life-cycle of the WS-Agreement extension.

The internal state definition for the extended agreement is similar to the internal state definition stated before, but a new state for the services is added and three for the guarantees. A new state is `stopped` and is needed to define a state of a service where its associated guarantee is unrecoverable violated and the service must terminate or the guarantee can be re-negotiated. It is an intermediate state. A guarantee is `negotiated` while the negotiation or re-negotiation process. A guarantee can also be `warned` if it is close to being violated in a given time instant. Other state for a guarantee is the `non_recoverable_violated` state in which a guarantee is violated and it has no related negotiation term for the current violation.

Definition 6 (Extended Internal state) *The* extended internal state A_{xis} *of an agreement* A *is a sequence of terms' states* $ts_1, \ldots, ts_p$ *of maximum size* $n \cdot m$*, where* $ts_i = (ss_j, gs_k)$ *represents the state of* g_k *guarantee with respect to the* s_j *service. Service and guarantee states range over the*

following sets, respectively:
$ss_j \in$ {`not_ready`, `ready`, `running`, `stopped`, `finished`}, *and*
$gs_k \in$ {`not_determined`, `negotiated`, `fulfilled`, `warned`, `violated`, `non_recoverably_violated`}.

As for Definition 4, one notes that not all the state combinations make sense. The only possible ones are the combinations itemized in Section 4.2 plus the following six:

(7) (`ready`, `negotiated`)
(8) (`stopped`, `negotiated`)
(9) (`stopped`, `fulfilled`)
(10) (`stopped`, `violated`)
(11) (`stopped`, `non_recoverably_violated`)
(12) (`running`, `warned`)

Service is ready and guarantee is negotiated, i.e., state combination (7), while initial negotiation process. The state combinations (8), (9), (10) and (11) determine the states when a service is stopped because a guarantee is violated or is being re-negotiated. In state (9) a guarantee is fulfilled and we try to improve it applying a negotiation term. In (10) and (11) a guarantee is currently violated. In (10) the service is stopped and the guarantee is violated but it is possible to apply a negotiation term and to preserve the agreement again. In (11), instead, the guarantee is irrecoverably violated and the agreement must terminate, there are not any suitable negotiation term. State (12) represents the fact that a warning has been raised for a running service guarantee.

An appropriate XML syntax to implement the proposed extension is provided in [141].

Example 5 *Referring to the loan origination case study introduced in Section 1.3, we can see how the extended version of WS-Agreement behaves.*

We assume that the BBB bank and the Credit Bureau establish an agreement in order to define interactions and the qualities of the service provided. In the agreement they specify some service terms and guarantee terms for credit worthiness check operations. Following the operation and the interaction's model stated in the WS-Agreement specification, consumer and provider negotiate resources and qualities of the services.

For instance, besides the agreement about services and guarantees, with the extension it is possible to add some negotiation terms that give the freedom to change the agreement at runtime.

The main and exclusive service defined in the agreement is the execution of credit worthiness check operation. Associated with this service we specify two variables that are bandwidth and memory, which can be checked on the service provider side by a monitoring system. Depending on this variable, it is simple to identify some service's properties like the number of operation's execution per minute, the number of request per minute and the service cost. We specify the metric of the variable and in the section dedicated to the guarantee statement we assign ranges of values that should be met to fulfill the current agreement.

Let us consider an agreement example adapting the WS-Agreement structure to our example.

```
<wsrp:GetResourcePropertyResponse>
    <wsag:Name>AgreementExample</wsag:Name>
    <wsag:Context/>
    <wsag:Terms>
    <wsag:All>
        ....
        <wsag:All>
            <wsag:ServiceDescriptionTerm wsag:Name="bandWidth"
                    wsag:ServiceName="Operation">
```

```
10              </wsag:ServiceDescriptionTerm>
11              <wsag:ServiceDescriptionTerm wsag:Name="memorySize"
12                      wsag:ServiceName="Operation">
13              </wsag:ServiceDescriptionTerm>
14          </wsag:All>
15
16          <wsag:ServiceProperties wsag:ServiceName="Operation">
17              <wsag:VariableSet>
18                  <wsag:Variable wsag:Name="requestMinute"
19                          wsag:Metric="time:duration">
20                      <wsag:Location>
21                          ...
22                      </wsag:Location>
23                  </wsag:Variable>
24                  <wsag:Variable wsag:Name="numberOfOperationMin"
25                          wsag:Metric="time:duration">
26                      <wsag:Location>
27                          ...
28                      </wsag:Location>
29                  </wsag:Variable>
30                  <wsag:Variable wsag:Name="serviceCost"
31                          wsag:Metric="float">
32                      <wsag:Location>
33                          ...
34                      </wsag:Location>
35                  </wsag:Variable>
36              </wsag:VariableSet>
37          </wsag:ServiceProperties>
38
```

```
39          <wsag:GuaranteeTerm wsag:Name="operationRequestMinute"
40                  Monitored="True" Negotiability="True">
41              ...
42              <wsag:ServiceLevelObjective>
43                  requestMinute IS_LESS_INCLUSIVE 5
44              </wsag:ServiceLevelObjective>
45              ...
46          </wsag:GuaranteeTerm>
47
48          <wsag:GuaranteeTerm wsag:Name="operationMinuteCount"
49                  Monitored="True" Negotiability="True">
50              ...
51              <wsag:ServiceLevelObjective>
52                  numberOfOperationMinute IS_MORE_INCLUSIVE 12
53              </wsag:ServiceLevelObjective>
54              ...
55          </wsag:GuaranteeTerm>
56
57          <wsag:GuaranteeTerm wsag:Name="operationCost"
58                  Monitored="True" Negotiability="True">
59              ...
60              <wsag:ServiceLevelObjective>
61                  serviceCost IS_LESS_INCLUSIVE 1
62              </wsag:ServiceLevelObjective>
63              ...
64          </wsag:GuaranteeTerm>
65
66          <wsag:NegotiationTerm wsag:Name="Neg1"
67                  Counter="2" Monitored="False">
```

```
68              <wsag:GuaranteeScope>
69                  <wsag:GuaranteeName>
70                      operationMinuteCount
71                  </wsag:GuaranteeName>
72                  <wsag:GuaranteeName>
73                      operationCost
74                  </wsag:GuaranteeName>
75                  <wsag:GuaranteeName>
76                      operationRequestMinute
77                  </wsag:GuaranteeName>
78              </wsag:GuaranteeScope>
79              <NegotiationRange>
80                  <wsag:GuaranteeName>
81                      operationRequestMinute
82                  </wsag:GuaranteeName>
83                  <--! requestMinute values-->
84                  <Minimum>4</Minimum>
85                  <Maximun>6</Maximun>
86              </NegotiationRange>
87              <wsag:ServiceLevelObjective>
88                  <ServiceLevelObjectiveAssertion>
89                      numberOfOperationMin IS_MORE_INCLUSIVE 24
90                  </ServiceLevelObjectiveAssertion>
91                  <ServiceLevelObjectiveAssertion>
92                      serviceCost IS_LESS_INCLUSIVE 2
93                  </ServiceLevelObjectiveAssertion>
94              </wsag:ServiceLevelObjective>
95              <wsag:BusinessValueList>
96                  ....
```

```
97              </wsag:BusinessValueList>
98          </wsag:NegotiationTerm>
99      </wsag:All>
100     </wsag:Terms>
101 <wsrp:GetResourcePropertyResponse>
```

Service consumer and service provider start their interactions taking into account the established agreement described above. In this scenario it is possible that the monitoring system at provider side notices that the consumer sends more requests per minute than the number stated in the agreement, exceeding the maximum value, 4 (defined in the guarantee at line 43). For instance, the provider can not fulfill all the requests from the consumer as previously agreed. Thanks to the proposed extension, it is possible to re-negotiate the current guarantee. In the NegotiationTerms (lines 84 to 107), there is a term referring to the current guarantee that gives the freedom to increase the number of requests per minute up to 24, if service cost is increased of 2 USD. Applying this negotiation term, defined and agreed on by both service consumer and provider at agreement creation's time, the consumer will pay more, but can ask more executions per minute: in this case an increase of performance means an increase of service cost. Furthermore, if a monitoring system that interacts with the agreement and service architecture anticipates violation, consumer and provider re-negotiate the agreement in advance.

In this simple execution on the running example, we see that using the extension it is possible to maintain the current agreement, mediating guarantees that are likely to be violated, currently violated, and guarantee that are widely fulfilled. Instead, using the original version the agreement must terminate as soon as a guarantee is violated.

4.3.2 Framework

The proposed extension to WS-Agreement must be handled by an appropriate framework that allows for monitoring and provides run-time re-negotiation.

On the one hand, there must be rules specifying when and how to raise a warning for any given guarantee. These rules should be easy to compute to avoid overloading of the monitoring system and be fast to provide warnings. In addition they should provide good performance in detecting as many violations as possible generating the minimum number of false positives. A forecasting method which enjoys this characteristics is the linear least squares method [34]. The method of linear least squares requires a straight line to be fitted to a set of data points such that the sum of the squares of the vertical deviations from the points to the line is minimized. By analysing such a parameter of the line as a slope ratio, it is possible to predict a change over time.

On the other hand, to allow for re-negotiation of guarantee terms at run-time the parties involved in the agreement need to be able to decide whether a re-negotiation has been agreed upon. Before execution it must be possible to specify negotiation terms. This can be done by using appropriate templates in the spirit of the original work in [138].

4.4 Anticipate Violations Strategy

We have conducted experimentation to show the feasibility of the warning strategy. We used synthetic data. We generated a sequence of 1100 elements considered as a service guarantee for a single operation over a continuous time interval (for instance the cost of a service which should be below the value 10). The data set and the results of the experiments are available in [4]. The points were generated by a function that returns a

random number greater or equal to 6.00 and less or equal to 14.00, evenly distributed. We split the data set into two subsets. The first part of the data set was used to decide the size of the time window and of the threshold values to be used for prediction. The rest of the data was used for evaluating the system.

To evaluate the method we consider the following performance measures: *Precision* is the ratio of the number of true warnings (i.e., warnings thrown to notify violation points) to the number of total warnings (i.e., true warnings and false warnings). *Recall* is the ratio of the number of warned violations (i.e., violation points for which a warning is issued) to the number of total violation points. Total violation points include warned violations and missed violations.

The table in Figure 4.6 summarizes the results of the experimentation. The number of true and false warnings is shown in the first column. The difference in the number of total warnings and violations is due to the fact that more than one warning in the same time window may refer to the same violation. The number of warned and missed violations is reported in the second column of the table. The total sum of warnings and violations is in the "Total" row. The last two rows present the precision and recall of the method.

	Warnings		**Violations**	
	True	***False***	***Warned***	***Missed***
	303	11	156	13
Total	314		169	
Precision	**96.50%**			
Recall			**92.31%**	

Figure 4.6: Experimental results.

The results of experimentation on the first 100 points of the data set is

shown in Figure 4.7. In the figure, two types of warnings, true and false, are marked by diamonds and crosses, respectively. A warning is thrown if the cost and tangent of the cost curve are higher then the threshold (8 for cost and 0.1 for the tangent differences). Squares represent warned violation points, while circles indicate missed violation points.

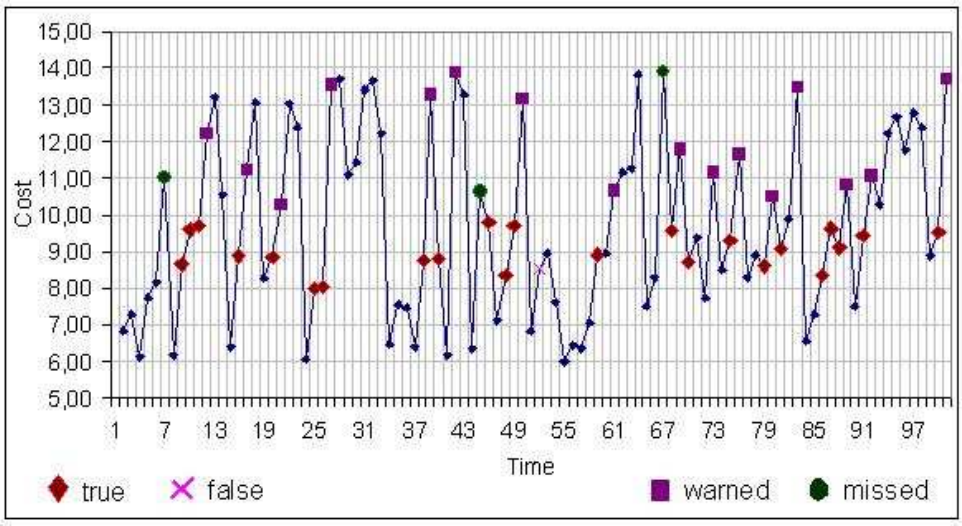

Figure 4.7: Experimental results for 100 points.

The method shows good performance when the increase in cost is smooth (points 8, 9, and 10), a case that normally takes place during Web services execution. If the change in values is abrupt then the method fails to generate warnings, e.g., points 43 (cost is 6.36) and 44 (cost is 10.63). It is difficult to find a violation point if the point is in the very beginning of the process, within or just after the first time window (point 7). The latter cases should be considered exceptional, in fact those occur only 13 times in the whole experiment.

In the experimentation using the method, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings. Using bigger time windows does not improve performances, see [4] for evidence of this fact.

Incidentally, we are not claiming that the proposal based on linear least square is the best approach to provide early warning, in fact, it may turn out that the method depends on the context in which the agreements are established and monitored. Here we are concerned with the extension of the protocol which contemplates the possibility of having early warnings, the way in which these are actually issued will be designed separately for any specific application scenario.

4.5 Application of the Approach: Service License Life Cycle

The concept of software licensing has emerged when the production and sale of individual software came into the market. While licensing was already present in the software world, the move to mass market software has introduced shrink wrap licenses, the terms of which can only be read and accepted by the consumer after using the product. With the advent of the Internet-based marketing and distribution strategies, click wrap licensing (similar to shrink wrap licensing) continues as one of the universal practices. The transformation from software as a product to software-as-a-service [28] is the reflection of the transition of the distribution of software. As SOC extends the concept of software-as-a-service to include the delivery of complex business processes as a service, there arises a requirement for developing service licensing strategies.

Similar to software licensing, service licensing is extremely important for distribution of services. Software serves as a stand-alone application. In contrast, the rationale behind services is making network accessible operations available anywhere and anytime. While software is designed with particular use in mind, services are designed to facilitate potential reuse. The design of services supports loose coupling, wherein a service acquires

knowledge of another services, still remaining independent. Software is designed to incorporate a set of specific functions and usually is not allowed to be integrated with other softwares. Further, software could be restricted by the organizational boundaries and could not communicate with other softwares crossing the boundaries. The fundamental to service orientation is to design services to encourage composition. Thus, the distinguishing characteristics and nature of services prevent services directly to adopt the licensing models of software.

A service is represented by an interface part defining the functionality visible to the external world as a means to access the functionality and an implementation part realizing the interface. Service interfaces, typically described by WSDL [46], together with bindings are publicly available. Several services might be created using the same interface, varying in their performance. However, creating a new service by modifying an existing service interface depends on licensing clauses of the existing service.

In case of the interface reproduction with modifications, several scenarios arise as follows:

- The interface of a service could be modified by changing the name of some operations.
- The interface of a service could be modified by some changes in the service parameters or by some pre-processing and/or post-processing of the service.

By distributing the services as executable, the provider does not allow to modify the service operations. In contrast, it is also possible that a service provider could allow the service realization to be modified. Thus, a service provider allows the creation of another service, by modifying the interface as well as the realization. If the interface and the realization of a service are allowed to be copied, an independent service could be created

by mirroring the source code of realization and interface.

The extensiveness regarding the access and usage of a service arises a spectrum of variable clauses of licensing. A service license intends to describe the following objectives [58]:

- Describing the information regarding the service being licensed and other related information such as an unique identification code, the details of the service provider, and so on.
- Defining the extent to which the service could be used, accessed, and value added (by composition [69] and/or by derivation [58]), on the basis that any use outside the scope of license would constitute an infringement.
- Explaining payment and charging terms.
- Specifying delivery terms (regarding quality of service and performance), acceptance terms, warranties, and limiting the liability of providers in case of failures.
- Declaring the rights over future versions and over evolved services.

Being a way to manage the rights between service consumers and service providers, licenses design collaborative business strategies and enable a broader usage of services.

4.5.1 Service Level Agreement Versus Service License

SLA is a container for holding technical data relating to the operation of services that implies the objectives with regard to a service consumer [156]. Further, a SLA is a document that describes the minimum performance criteria a provider promises to meet while delivering a service. Typically

SLA sets out the remedial action and any penalties that take effect if performance falls below the promised standard.

Licensing [49] includes all transactions between the licensor and the licensee in which the licensor agrees to grant the licensee the right to use some specific contents of information for a specific tenure under predefined terms and contracts. A service license primarily focuses on the usage and provisioning terms of services. Being the mechanism of technology transfer, service licensing is the method of getting financial benefits for the providers. Optionally, a service license can include the SLA terms. Thus, a service license is broader than the scope of SLA, protecting the rights of service providers and service consumers.

An agreement is negotiated between the service provider and the service consumer. In case of SLA, there are two parties, a service provider and a service consumer. They agree on a SLA that covers a service (or a group of services). In case of a service license, there is a service provider that plays the main role of the licensor. There could be many service consumers (the licensees) binded by the service license. However, a license seems as if the licenses were not even involved in the transactions between the licensor and the licensees[1].

The agreement is terminated when either of the party terminates or violates the agreement. If one of the partners violates the agreement, the agreement might be re-negotiated (in case of recoverable violation). Any modifications to the clauses of a service license result in the creation of a new license and in some cases, could lead to the termination of the existing license.

If a license is modified, it leads to the creation of a new version of the license. A new invocation of a service might use the modified version of the

[1]With respect to the software licensing transactions there exists even the class of negotiated licenses [131], here, by license, we refer to the non-negotiated transactions between the consumers and the providers.

license. However, the unmodified version of the license, if it is implemented and executed by a service, will remain active and will not be overridden by the new version.

4.5.2 Service License Life Cycle

Based on the identified differences between a SLA and a service license, we propose a license life cycle inspired from the existing SLA life cycle.

A SLA establishment involves two parties namely, the service provider and the service consumer. A SLA life cycle starts from the templates provision by the agreement initiator. The templates are filled by the other party and negotiated between the provider and the consumer. The failure in reaching an agreement between the parties might result in the termination without having an agreement. In case of a service license, though a service provider and a consumer are involved, the license is often non-negotiated. The licensee is bound to agree the terms of the license.

The licensing terms are defined by the provider and typically described in the ODRL/L(S) [90] language. A service license is implemented when it is attached to a service interface.

Every version of the license, before its implementation, could be modified by the provider by changing the license terms that leads to the next version creation. However, a SLA can only be re-negotiated between the parties resulting in the re-execution of the revised agreement.

The monitoring of a SLA followed by execution is a run-time activity. The violations detected during the monitoring would raise warnings and might call for re-negotiation. In case of a service license, monitoring is also a post-run-time activity as the license governs the service during execution and also the usage of the result caused by the execution of the service. The violations found in the phases of validation during execution or monitoring of a service license would cause several possible actions to manage

violations including the interruption of the service usage.

A service license life cycle (as shown in Figure 4.8) includes the phases of creation, modification, implementation, execution/validation, monitoring, warning, litigation, termination, and withdrawal.

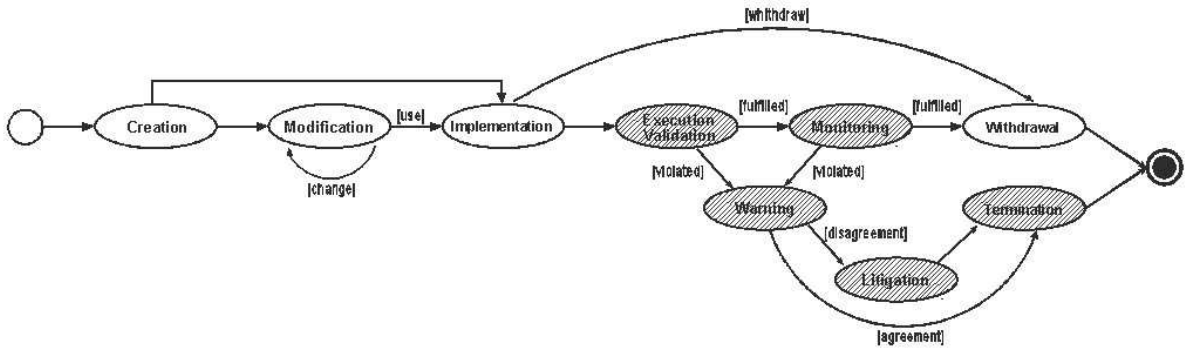

Figure 4.8: Service license life cycle.

The service provider defines a license in the `creation` phase. The licensing clauses are modified by the provider in the `modification` phase. The `implementation` phase refers to attaching the license with the given service. The license becomes enforceable in the `execution/validation` phase, associated with a particular invocation of a service. The usage of an instance of a license associated with a particular service invocation is monitored by the provider in the `monitoring` phase. The `warning` phase presents the warnings caused by the violations of the licensing clauses. The `withdrawal` phase denotes the end of the given version of the license from being used. The `litigation` phase deals with the dispute resolutions and decides the span of the license. The `termination` refers to the end of the scope of the given instance of a license.

The life cycle of a service license obviously illustrates the two distinct but interwined aspects of a license. The phases of `creation`, `modification` and `implementation` are associated with a version of a service license. A

version of a service license could end by the withdrawal, leading to the `withdrawal` phase. The phases associated to the version of a license are specified by the hollowed rounded rectangles. We represent the phases of an instance of a service license associated to a particular invocation of a service by the shaded rounded rectangles. The phases associated with an instance of a service license can repeat with several invocations of a service, each time with a particular instance of the single version of the given service license. The versions and instances of a service license are analogous to the concept of objects and instances in object oriented programming.

The creation of a license by the service provider refers to the definition of the scope of rights and the other related licensing clauses. A service license could be created by the service provider before the existence of a service. In other words, the provider could even determine the usage rights of their forthcoming service. The clauses regarding the usage of a service license could be decided from scratch. This can be referred as the creation of a license from scratch (as shown in Figure 4.9 (a)). Alternatively, a license can be created from an existing license by modifying the licensing clauses.

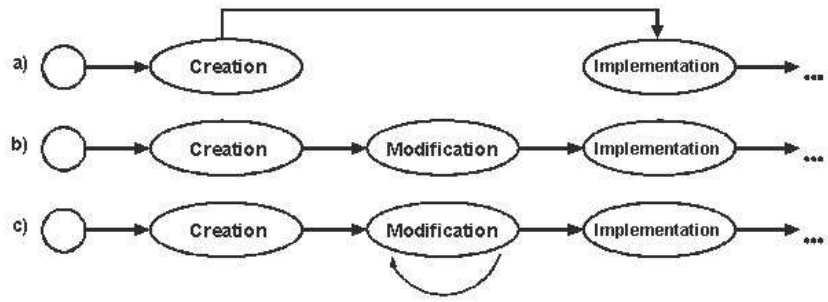

Figure 4.9: Service license versioning by modification

The created license could be modified by revising certain licensing clauses. As the license is not negotiated and there is no involvement of a consumer

at this stage, a service provider could perform the modification arbitrarily at their own discretion. The modifications of a service license causes the license versioning. A created license could be modified and a new version of the license is generated (see Figure 4.9 (b)). This license can be applicable to a service and follow the phases of the license life cycle. This version of the license could even be modified and applied to any service as a newer version of the license. Several versions of licenses arise from the modifications and follow the life cycle individually as a next version of the license (as shown in Figure 4.9 (c)).

Let L_b be the modified license from the license L_a. Then a service provider could implement the licensing as follows: the implemented service (having L_a implemented) could continue with L_a itself while the forthcoming service invocations could have L_b implemented. Here, L_b does not override L_a for the service in execution. It could be even possible for the provider to withdraw a version, say L_a, and could bind the services to the newer version of the license.

A service license implementation follows the creation or modification of the license. The `implementation` phase of a service license refers to the existence of the license, but not referring to any enforcement of the license over the usage of the service.

A service license can be withdrawn at this stage. The `withdrawal` of the license could implicate the possible removal of the particular version of the license from the service, offering the provision of the service even without the license.

The license associated with the service automatically becomes enforceable when the service is provisioned. This is referred as the `execution/validation` phase of a service license. The execution phase of a service license differs from the state of SLA life cycle by embedding the validation of licensing clauses. As the consumer uses the licensed service,

the consumer is bound to agree the licensing clauses.

The violations of the licensing clauses during the phase of `execution/validation` would lead to the `warning` phase. The violations could be caused either by the consumer or by the provider.

The disputes arisen in the case of the `warning` phase would be taken care in the `litigation` phase. As the process of litigation involves judicial matters, we skip this phase as beyond the scope of this work.

On the fulfillment of the licensing clauses, the license progresses to the state of `monitoring` which either could denote the satisfaction of the usage terms or could lead to `warning` phase as described above if the terms are violated. The detection of violations during the monitoring phase makes the license to move towards the `warning` phase. The instance of the license terminates as the service terminates when there is the progression from the state of `warning` on agreeing the terms.

The termination of an instance of a service license does not imply the total termination of the license as SLA termination. As there could exist several instances of a license by several invocations of a service, the termination refers only to the detachment of the instance of the license from the particular invocation of the service.

In [91], we illustrate a collaborative scenario of the service license life cycle by a case study provided by the courtesy of Dnepropetrovsk Hydrometeorology Regional Center, Ukraine.

4.6 Concluding Remarks

WS-Agreement is an industry based protocol for the establishment of service level agreements among loosely coupled service providers and requesters. If on the one hand, WS-Agreement is being adopted widely, on the other hand, it lacks a precise definition of the meaning of its constructs.

In this chapter, we presented a formal definition of an WS-Agreement by resorting to finite state automata. Furthermore, by providing a set of formal rules that tie together agreement terms and the life-cycle of the agreement, we identified some shortcomings of the protocol. That is, the protocol does not support explicitly the negotiation of the agreement, there is no monitoring of how close a term is to being violated at execution time, and, the breaking of one single term of a running agreement results in termination while a more graceful degradation is desirable. To overcome these shortcomings, we proposed an extension of WS-Agreement, for which we provided appropriate semantics. The extension considers initial negotiation of an agreement, it considers the possibility of issuing warnings before a possible term violation, and eventually re-negotiation of a running agreement. Furthermore, we have analysed SLAs and service licenses with the goal of providing differences and similarities between the two concepts. Based on our investigations, we have proposed a life cycle of service license.

Chapter 5

Deriving Business Processes with Service Level Agreements from Early Requirements

When designing a Web service-based business process employing loosely-coupled services, one is not only interested in guaranteeing a certain flow of work, but also in how the work will be performed. This involves the consideration of non-functional properties which go from execution time, costs, up to trust and security. Ideally, the requester of a service to have guarantees over the behavior of the services involved in the process. These guarantees are the object of SLAs.

In this chapter, we propose a methodology to design Web service-based business processes together with service level agreements that guarantee a certain quality of execution, with particular emphasis on the security aspects. Starting from an early requirements analysis modelled in the Secure Tropos formalism, we provide a set of user-guided transformations and reasoning tools the final output of which is a set of processes in the form of Secure BPEL together with a set of SLAs to be signed by participating services.

5.1 BP&SLA Methodology

Judging what is the appropriate SLA to sign after having defined the business objectives [36] is far from being a straightforward task. With the Business Processes with Service Level Agreements (BP&SLA) methodology, we provide means to go from a high-level analysis of the business requirements all the way to the definition of the processes to be executed and the SLAs to be signed in order to guarantee certain quality of service. The methodology consists of four main phases which are, referring to Figure 5.1, (1) early requirements engineering, (2) business process hypergraph derivation, (3) hierarchy of business processes derivation, and (4) constraint reasoning for service level agreements derivation.

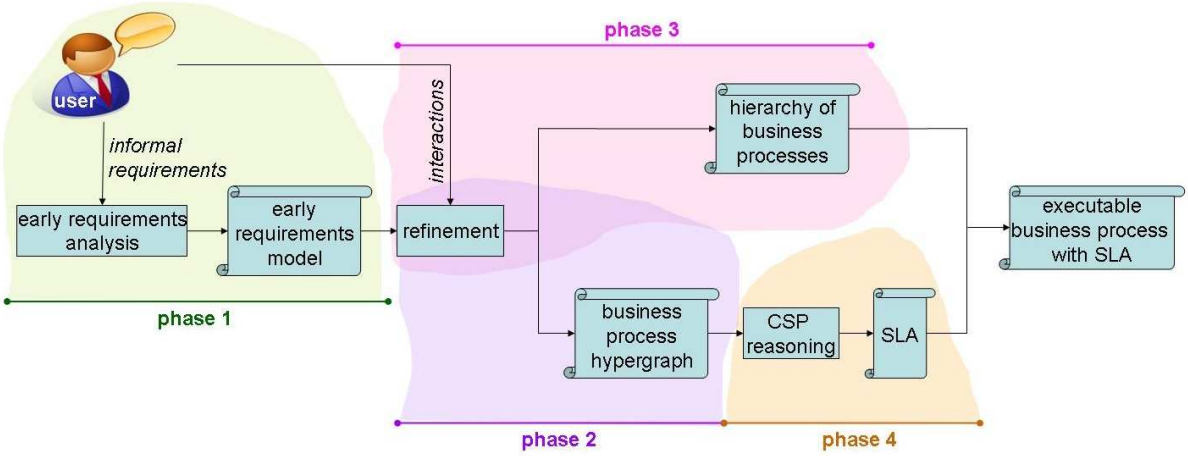

Figure 5.1: The BP&SLA Methodology.

During the first phase, the end-user or domain expert provides informal requirements that form the seed for developing formal processes. These early requirements are formalized following the Secure Tropos methodology, an extension of the well established Tropos software engineering methodology [35]. The output of this phase is an early requirement model. The model is far from being an executable entity, but rather it is a con-

ceptual description of the actors involved in the business, their goals and their trust and security relations. To transform the model into something executable, in the second and third phase, one navigates automatically the model and asks user intervention every time that an unambiguous choice is necessary. The results of the refinement of the early requirements are an intermediate model necessary to perform the reasoning on qualities of services, the business process hypergraph, and a hierarchy of business processes ready for execution, Phases 2 and 3, respectively. The business process hypergraph then is further analysed to build a constraint problem which represents the relationships among the various elements of the processes regarding quality of service and security properties of the processes. By reasoning with these constraints it is possible to derive the appropriate SLAs to be signed in order to guarantee a certain QoS when executing the process, Phase 4. The final output of the methodology is a hierarchy of business processes ready for execution together with SLAs fulfilling a specific QoS. Let us consider next each of these phases individually. We do not only present the phases of the methodology, but also look at how the application of the proposed methodology leads to a set of executable business processes and SLAs. We consider the loan origination case study proposed as a running example in Section 1.3.

5.1.1 Phase 1. Early Requirements Engineering

Early requirements engineering aims at analysing the organizational context within which a system will eventually operate. During an early requirements analysis the domain actors and their dependencies on other actors for goals to be fulfilled are identified. For early requirements model elicitation in the context of security, one needs to reason about trust relationships and delegation of authority.

We employ the Secure Tropos modelling framework [95, 144] to derive

and analyse both functional dependencies and security and trust requirements.

For the acquisition of the early requirements model we employ the modelling activities described in 2.3. Actor modelling is used to identify the principal stakeholders and their objectives. It might happen that an actor does not have the capabilities to achieve his own objectives by himself. In this case that actor has to delegate the objectives to other actors that leads to their achievement outside the control of the delegator. Secure Tropos supports two types of delegations. *Delegation of execution*, i.e, at-least delegation, means that one actor delegates to another one the responsibility to execute a service. *Delegation of permission*, i.e, at-most delegation, models the transfer of entitlements from an actor to another. We use functional dependency modelling to identify actors depending on other actors for obtaining services, and actors which are able to provide services. Permission delegation modelling is used to identifying actors delegating to other actors the permission on services. Secure Tropos supports two types of trust dependencies. *Trust of execution*, i.e, at-least trust, means that one actor trusts that another one will at least fulfill a service. While the meaning of *trust of permission*, i.e, at-most trust, is that an actor trusts that another actor will at most fulfill a service, but will not overstep it. Trust modelling aims at identifying actors trusting other actors for services, and actors which own the services.

Example 6 *The early requirement model for the loan origination case study described in Section 1.3 is depicted in Figure 5.2.*

The model presents the principal entities involved, (1) actors depicted as circles and (2) interests, i.e., goals, presented as ovals. The `Bank` *actor has the goal to* `launch loan origination process`*. The goal is delegated to the* `Bank manager` *actor. The delegation of execution is depicted with two lines connected by a delegation of execution (De) graphical symbol.*

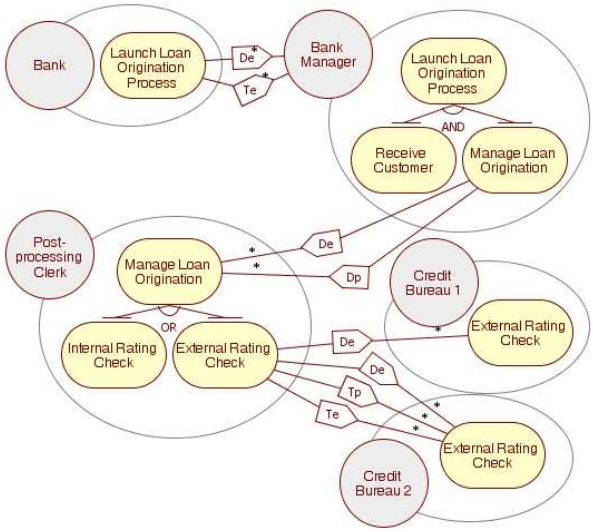

Figure 5.2: Early requirements model.

The **Bank** *actor trust the* **Bank manager** *actor on execution of the goal. The trust on execution is depicted with two lines connected by a trust on execution (Te) graphical symbol. In order to fulfill the goal the* **Bank manager** *actor refine it by an AND decomposition, depicted with a goal refinement symbol marked with AND, into goals to* **receive a customer** *and to* **manage loan origination**. *The* **Bank manager** *actor delegates the last goal to the* **Post-processing Clerk** *actor. Here not only at-least delegation of execution, but also at-most delegation of permission is used. The delegation of permission is depicted with two lines connected by a delegation of permission (Dp) graphical symbol. The* **Post-processing Clerk** *actor refines the* **manage loan origination** *goal into the* **internal rating check** *and* **external rating check** *goals. The goal is refined by an OR decomposition, depicted with a goal refinement symbol marked with OR. The* **external rating check** *goal is delegated to the* **Credit Bureau 1** *and* **Credit Bureau 2** *actors. While the* **Post-processing Clerk** *trusts both on delegation and on permission to the* **Credit Bureau 2** *actor on processing of external credit check, there is no trust relation between the ac-*

tor and the ***Credit Bureau 1*** *actor. The trust on permission is depicted with two lines connected by a trust on permission (Tp) graphical symbol.*

Performance-based Trust Model

A KPI based trustworthiness model takes into account the business objectives described previously, and assigning automatically trust level values (0 or 1). This Performance-based trust model was elaborated in the context of IST-FP7-IP-TAS3 project[1] [61].

The traditional trustworthiness models deployed in famous online shops such as Amazon or eBay are relying on a subjective rating system in which users estimate the "quality" of the transaction over a numerical scale. Knowing that nobody is able to formalize and explain the difference between two successive values like a transaction rewarded at 9/10 and another one 10/10, it is not possible to estimate the correctness and the objectivity of the trust and reputation value.

In this work, we use a less subjective trust model taking into account the performance of each business partner according to their business objectives or to a business agreement like SLA. For example, if a business partner does not satisfy a target in the SLA, he will be penalized. Each trustee entity chooses the business objectives that must be satisfied by the partners to trust. These objectives must be measurable like a set of performance indicators, e.g., price, time, packaging, payments conditions, QoS. After each interaction between two business partners, the trustee gets these quantifiable values and compares it to the objectives in order to obtain trust indicator values. These indicator values are then aggregated and

[1]TAS3 (Trusted Architecture for Securely Shared Services) is a research project funded by the European Union. The TAS3 is an integrated project that aims to have a European-wide impact on services based upon personal information, which is typically generated over a human lifetime and therefore is collected & stored at distributed locations and used in a multitude of business processes, http://www.tas3.eu/project.

normalized in order to obtain a unified trust level value.

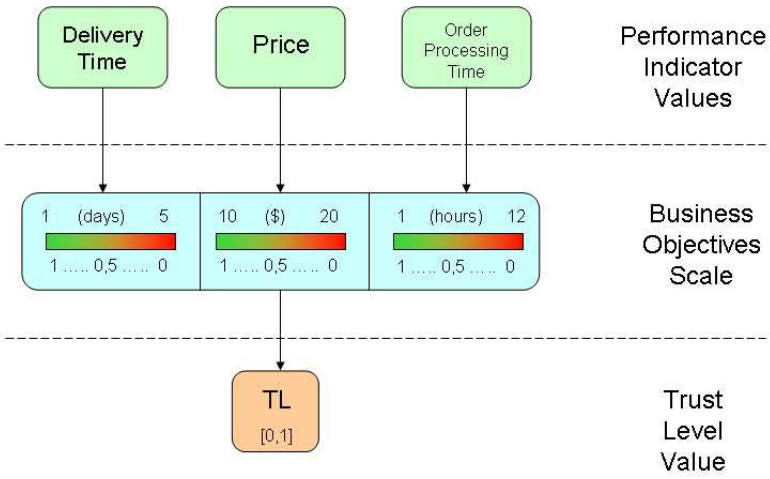

Figure 5.3: Performance-based trust model.

The trust model shown in Figure 5.3 is composed of three complementary layers:

Performance Indicator Values are collected and calculated after each interaction, then compared to the business objective scale.

Business Objectives Scale are fixed by the trustee according to the performance indicators related to their business objectives. An interval of values (min and max) must be chosen for every performance indicator in order to normalize the measured value with a [0,1] scale. The [0,1] normalization rule is written as follow:

$$\begin{cases} 1 & \text{if } K_i > K_{max} \\ \frac{K_i - K_{min}}{K_{max} - K_{min}} & \\ 0 & \text{if } K_i < K_{min} \end{cases}$$

Where K_i is the measured performance indicator value, K_{min} and K_{max} are the maximum and minimum values declared in the business objectives scale.

Trust Level Value is the aggregation of all the normalized performance indicators plus eventually some external values like the recommendation from other trusted entities.

In the loan origination case study, the Credit Bureau response time can be chosen by the BBB bank as a performance indicator. According to the bank's business objectives the delivery delay must be comprised between K_{min}=1 day and K_{max}=5 days. Using this scale we normalize the delivery time values in order to be fitted to a [0,1] scale. For example if the delivery time is K_i=3 days, the trust value will be 0.5.

To summarize, the performance based trust model offers the possibility to quantify the trustworthiness values according to business objectives and SLAs and permit to any business process component to determine which business partner is more trustable according to an objective estimation. Usually in traditional recommendation systems, the trustee relies on a binary recommendation value. In the performance-based trust model the trustee can evaluate the weight of a recommendation by accessing to the business objective scale of the recommender.

5.1.2 Phase 2. Business Process Hypergraph Derivation

The second phase of the BP&SLA methodology is devoted to creating an intermediate structure to reason about the business processes and their qualities. This intermediate structure is an hypergraph, which we define as follows.

Definition 7 A **Business Process hypergraph** (BP hypergraph) $\mathcal{B}$ is a pair $\langle B, H \rangle$ where B is a set of business processes and H is a set of BP

hyperarcs. A *hyperarc* is an ordered pair $\langle N, t\rangle$ from an arbitrary nonempty set $N \subseteq B$ (source set) to a single node $t \in N$ (target node). Each hyperarc is associated with a vector of aggregation functions $\varphi = [\varphi_1\langle N, t\rangle, ..., \varphi_n\langle N, t\rangle]$ which calculate value of a target node taking as arguments source nodes, with the structural activity associated, for a particular QoS parameter .

The BP hypergraph is obtained by navigating the early requirement model and refining it eventually resorting to user interaction. This is performed algorithmically according to the procedure presented in Figure 5.4.

The algorithm takes the early requirements model SI*, the actor with its goal and the vector of QoS parameters as an input. Each node of the BP hypergraph is a business process that corresponds to a goal in the early requirements model. As we consider the goals to be operational. Each hyperarc in the BP hyperarc corresponds to the goal refinement or delegation dependency in the early requirements model.

In the algorithm, we use the `addHyperArc (sourceNode, targetNode)` function to add one hyperarc in the business process hypergraph from a single source node to the target node. While the `addHyperArcForAll (sourceSetOfNodes, targetNode, aggregationFunction)` function adds one hyperarc in the business process hypergraph from a source set of nodes, i.e., `nodei[...]` to the target node. Where `aggregationFunction` is a vector of aggregation functions $\varphi = [\varphi_1\langle N, t\rangle, ..., \varphi_n\langle N, t\rangle]$ assigned to the business process hyperarc. The aggregation functions design takes into account the structural activity associated to the corresponding business processes, i.e, the source set of nodes, and the QoS parameter. Each aggregation function calculates the value of a target node taking as arguments source nodes (with the structural activity associated) for a particular QoS parameter.

```
BPHC (SI*, actor, goal, QoS)
begin
   if goal is not a leaf goal
      currentNode = node (goal)
      for each children in AND
         nodei = BPHC (SI*, actor, childGoal, QoS)
         interactWithUser (sequence | parallel)
         if sequence
            addHyperArcForAll (nodei[...], currentNode, sequence)
         if parallel
            addHyperArcForAll (nodei[...], currentNode, flow)
      end for
      for each children in OR
         interactWithUser (non deterministic choice | design choice)
         if non deterministic choice
            nodei = BPHC (SI*, actor, childGoal, QoS)
            addHyperArcForAll (nodei[...], currentNode, switch)
         end if
         if design choice
            nodei = BPHC (SI*, actor, childGoal, QoS)
            addHyperArc (nodei, currentNode)
         end if
      end for
      for each delegated child
            nodei = BPHC (SI*, actor, childGoal, QoS)
            addHyperArc (nodei, currentNode)
      end for
      if trust dependency
         trustLevel(currentNode) = 1
         for each children
            trustLevel(childNode) = 1
      else
         trustLevel(currentNode) = 0
         for each children
            trustLevel(childNode) = 0
   return currentNode
   end if
   if goal is a leaf goal
      return node (goal)
   end if
end
```

Figure 5.4: Business process hypergraph construction.

The concept of AND goal decomposition is refined as sequential or parallel business process composition in the BP hypergraph. Sequential business process composition corresponds to the sequence flow structural activity and the aggregation function for sequential aggregation of QoS parameters is applied. The parallel flow structural activity is used in case of parallel business process composition and the aggregation function for parallel aggregation of QoS parameters is applied. The concept of OR goal decomposition in the early requirements model is refined as branching statement in the BP hypergraph. If the structural activity is non-deterministic choice, the aggregation function for choice aggregation of QoS parameters is applied. In case of the design choice structural activity, the nodes corresponding to the business processes are connected by different hyperarc with the target node. The design choice structural activity appears in case of presence of different alternatives for the same business process, e.g., the same business process might be delegated to different partners that have different SLA offers.

The refinement of the concept of AND/OR goal decomposition from the early requirements model can not be completely automated, but only supported as it happens in model-driven architectures. For instance, in case of AND goal decomposition, the system can provide assistance in refining the decomposition into sequence flow or parallel flow structural activity. OR goal decomposition might be refined into non-deterministic or design choice structural activity. While determining the proper structural activity is the domain dependence task that involves the user interactions. In the business process hypergraph construction algorithm we use the `interactWithUser (option1 | option2 ... | optionk)` function to support the interaction with the users with the aim to decide which structural activity to apply to for a particular goal decomposition. The users determine the proper structural activity based on the proposed options where the only

one option has to be selected.

Each node in the BP hypergraph is assigned with a vector of QoS parameters and a Trust Level value (TL). The values of the QoS parameters correspond to the QoS that can be achieved by the BP. The trust level value denotes the level of trust between the truster and the trustee on the fulfilling of the business process (here we employ only at-least trust). In [83], we propose a methodology that identifies the concrete business process providing the highest quality of service and protection among all possible design alternatives. The idea is to take into account the level of trust of service providers and adjusts the expected quality value correspondingly. In spite of the fact that the approach to use the notion of trust as weighting factor is promising, the authors do not clarify how the trust values are decided. Instead in our approach the trust level is determined from the reasoning on the presence/absence of trust dependencies in the early requirements model. We also take into account the performance indicator values we introduced in Phase 1. Then, when the business process with SLA is in place, we apply the proposed performance-based trust model in order to determine the partner to work with when there is a possibility to choose one business process from the several alternatives suggested by different providers.

The problem of finding SLAs for business processes is then a problem of reasoning on the business process hypergraph.

Example 7 *The hypergraph corresponding to the case depicted in Figure 5.2 is shown in Figure 5.5. Each goal of the early requirement model is associated with a node of the hypergraph. Each node of the hypergraph is a business process.*

The nodes `Receive Customer` *and* `Manage Loan Origination` *are connected by one hyperarc with the top node*
`Launch Loan Origination Process`*, that means that the business pro-*

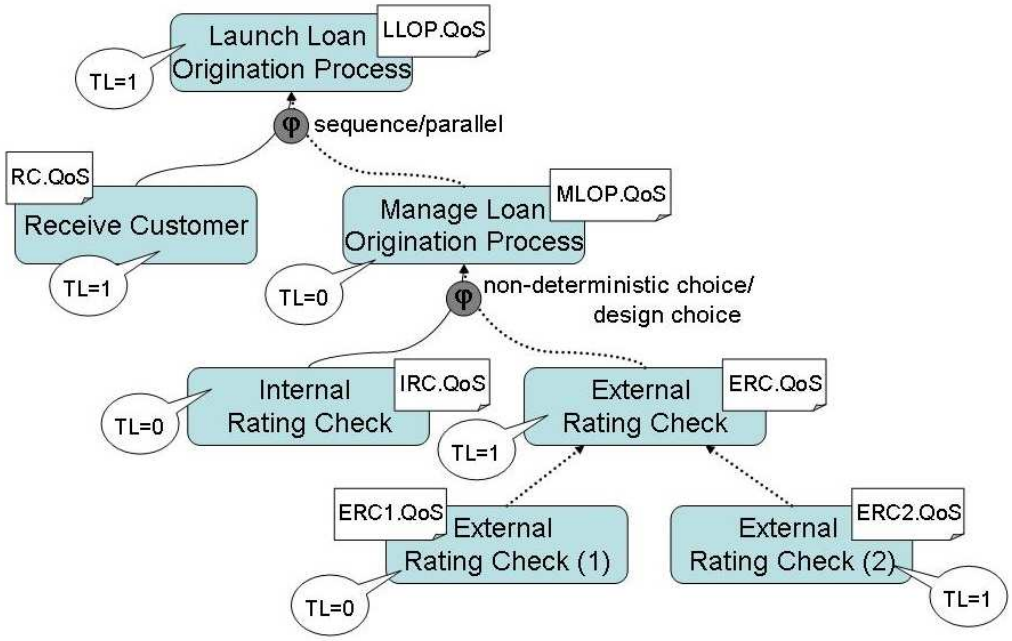

Figure 5.5: Business process hypergraph.

cesses **Receive Customer** *and* **Manage Loan Origination** *contribute to satisfaction of the global goal* **Launch Loan Origination Process**. *The dashed hyperarc leads from the delegated (here we employ only at-least delegation) business process* **Manage Loan Origination** *to the target one. The nodes in the business process hypergraph are assigned with vectors of QoS parameters and trust level values. The trust level is determined from the reasoning on the presence/absence of at-least trust dependencies in the early requirement model presented in Figure 5.2. A vector of aggregation functions* $\varphi = [\varphi_1\langle N, t\rangle, ..., \varphi_n\langle N, t\rangle]$ *is assigned to the hyperarc. The aggregation function takes into account the structural activity associated to the* **Receive Customer** *and* **Manage Loan Origination** *business processes and the QoS parameter. The notion of sequential and parallel composition corresponds to a refinement of the concept of AND goal decomposition. If the structural activity is sequence flow, the aggregation function for sequential aggregation of QoS parameters is applied. If the structural*

activity is parallel flow, the aggregation function for parallel aggregation of QoS parameters is used.

The nodes `Internal Rating Check` *and* `External rating Check` *are connected by one hyperarc with the target node* `Manage Loan Origination`*. The business process* `External Rating Check` *is delegated and is expressed by the dashed hyperarc. A vector of aggregation functions* φ *is assigned to the hyperarc. The aggregation function takes into account the structural activity associated to the* `Internal Rating Check` *and* `External Rating Check` *business processes and the QoS parameter. Branching statement is a refinement of the concept of OR goal decomposition. If the structural activity is non-deterministic choice, the aggregation function for choice aggregation of QoS parameters is applied. If the structural activity is design choice, the nodes* `Internal Rating Check` *and* `External rating Check` *are connected by different hyperarc with the target node* `Manage Loan Origination`*.*

The nodes `External Rating Check (1)` *and* `External rating Check (2)` *are connected by two hyperarc with the target node* `External Rating Check`*. Both the business process* `External Rating Check (1)` *and* `External rating Check (2)` *are delegated that is expressed by the dashed hyperarcs.*

5.1.3 Phase 3. Hierarchy of Business Processes Derivation

The third phase of the BP&SLA methodology is dedicated to hierarchy of BPs construction. We build the hierarchy of BPs with the aim to use it for obtaining a set of executable secure BPs. These are created following the Secure BPEL specifications [81, 82, 176]. Secure BPEL is a dialect of WS-BPEL for the functional parts and abstracts away low level implementation details from WS-Security and WS-Federation specifications. Secure BPEL allows us to describe delegation (both delegation of execution and

delegation of permission) and trust (both trust on execution and trust on permission) relations among all the partners that execute sub-BPs in the context of the global BP. Refer to Section 3.1.2 for the Secure BPEL language specification and examples. In the hierarchy of BPs, each delegated business process is labelled with a SLA derived in Phase 4. The hierarchy of BPs, as well as the BP hypergraph, is derived by refining the early requirements model. As we build the hierarchy to obtain executable BPs with SLAs, we must clearly determine (1) the BPs, (2) which partner proceeds which BP, and (3) delegation and trust dependencies among the involved partners.

For space reason, we do not report the whole algorithm for the hierarchy of BPs construction here, but rather refer to [84] for details. The main idea is that analogously to the BP hypergraph construction, we consider the level of goals in the early requirements model to be the level of BPs in the hierarchy of BPs. Furthermore, the BP(s) proceeded by one actor are grouped and marked with the actor. We introduce the notion of actors to render the hierarchy of BPs ready to be executable. In fact, each partner has to know which business process to proceed.

In this work, we adopt only the Secure Tropos delegation of execution dependencies, but not the delegation of permission ones, to label with SLAs only the BPs that are delegated to be executed. We consider the fact that one needs to sign a SLA with the partner only in case of transfer of responsibilities to the partner, i.e., the business process is delegated to the partner and the partner processes it. While if there is only a fact of transfer of entitlements, i.e., the business process is delegated to the partner and the partner has permissions to processes the BP, but do not actually does it, there are no reasons for a SLA signing. Further, we employ both at-least and at-most trust and delegation notions to implement the relations between the actors in the hierarchical structure of BPs.

Example 8 *The hierarchy of business processes corresponding to the early requirement model for the loan origination process is shown in Figure 5.6.*

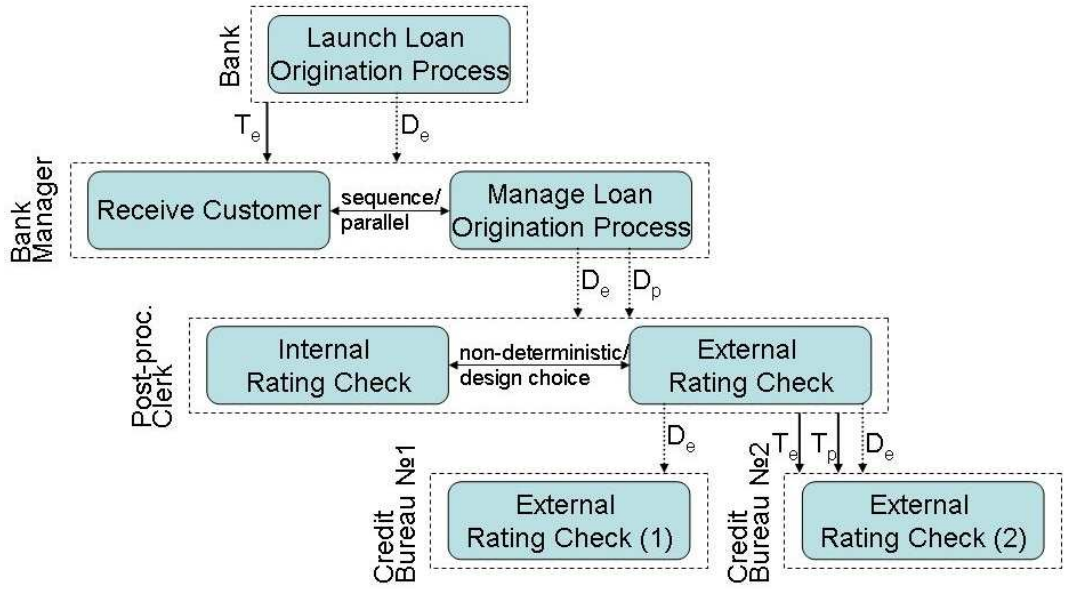

Figure 5.6: Hierarchy of business processes.

Each goal is associated with a business process, represented by a rounded-corner rectangle in the hierarchy. Dashed rectangles are used in order to represent the actors that proceed the business processes. In our case these actors are the `Bank`*, the* `Bank Manager`*, the* `Post-processing Clerk`*, the* `Credit Bureau 1`*, and the* `Credit Bureau 2`*.*

The dependencies among actors, i.e., delegation and trust, are represented as dashed and solid lines correspondingly. The `Bank` *actor delegates the* `Launch Loan Origination Process` *business process to the* `Bank Manager` *actor. The delegation of execution dependency is depicted by dashed line marked with the delegation of execution (De) symbol. The delegation of execution line connects the delegated business process, i.e., the* `Launch Loan Origination Process` *business process with the delegatee, the* `Bank Manager` *actor. The* `Bank` *actor trust the* `Bank Manager` *actor to fulfill the* `Launch Loan Origination Process` *business process. The*

trust on execution dependency is depicted by line marked with the trust on execution (Te) symbol. The trust on execution line connects the trusted business process, i.e., the **Launch Loan Origination Process** *business process, with the trustee, the* **Bank Manager** *actor.*

The relation among business processes proceeded by the **Bank Manager** *actor is defined by the structural activity associated to the* **Receive Customer** *and* **Manage Loan Origination** *business processes. The notion of sequential and parallel composition corresponds to a refinement of the concept of AND goal decomposition. If the structural activity is sequence flow, the sequence relation is applied. If the structural activity is parallel flow, the relation is the parallel one.*

The relation among business processes proceeded by the **Post-proceeding Clerk** *actor is defined by the structural activity associated to the* **Internal Rating Check** *and* **External Rating Check** *business processes. Branching statement is a refinement of the concept of OR goal decomposition. If the structural activity is non-deterministic choice, the non-deterministic choice relation is applied. If the structural activity is design choice, the design choice relation is applied. The* **Bank Manager** *actor delegates the* **Manage Loan Origination** *business process to the* **Post-processing Clerk** *actor. The delegation of execution and delegation of permission lines connects the delegated business process, i.e., the* **Manage Loan Origination** *business process with the delegatee, the* **Post-processing Clerk** *actor. The delegation of permission dependency is depicted by dashed line marked with the delegation of permission (Dp) symbol. There are no trust dependencies between the* **Bank Manager** *and the* **Post-processing Clerk** *actors on the* **Manage Loan Origination** *business process.*

The **External Rating Check** *business process is delegated to the* **Credit Bureau 1** *and the* **Credit Bureau 2** *actors. The delegation of execution*

lines connect the delegated business process, i.e., the ***External Rating Check*** *business process with the delegatee, the* ***Credit Bureau 1*** *and the* ***Credit Bureau 2*** *actor. There are no trust dependencies between the* ***Bank Manager*** *and the* ***Credit Bureau 1*** *actors on the* ***External Rating Check*** *business process. While trust on execution and trust on permission lines connects the trusted business process, i.e., the* ***External Rating Check*** *business process, with the trustee, the* ***Credit Bureau 2*** *actor. The trust on permission dependency is depicted by line marked with the trust on permission (Tp) symbol.*

5.1.4 Phase 4. Constraint Reasoning for SLAs Derivation

In the last phase of the BP&SLA methodology SLAs for BPs are derived by reasoning on the BP hypergraph. The reasoning technique we employ in this work is constraint programming. The key idea is to state the relationships among the qualities of processes and their activities as a set of constraints.

Formally, the Constraint Satisfaction Problem (CSP) is defined as follows [188]:

- a set of variables $\{x_1,.., x_n\}$,
- for each variable x_i a finite set D_i (its domain) of possible values,
- a set of constraints, i.e., relations or expressions, restricting the values that the variables can simultaneously take.

A solution to CSP is an assignment to the set of variables such that all its constraints are satisfied. One may want to find an optimal solution, if some objective function is given over CSP variables [188].

We build a constraint systems by recursively navigating the business process hierarchy and hypergraphs. The algorithm is presented in Figure 5.7.

```
CSPEC (BPH, node, CSP, QoSDomain)
begin
   if node is not a leaf node
      addToCSP (Var node ∈ QoSDomain)
      if decomposition = AND
         for all nodes
            expr = expression (nodes, flow/sequence)
         end for
         addToCSP (Var node = expr)
      end if
      if decomposition = OR
         for all nodes
            if non deterministic choice
               expr = exression (nodes, switch)
            end if
            if design choice
               expr = expression (node, mult xi)
               where xi = 0 or 1 and sum(xi) = 1
            end if
         end for
         addToCSP (Var node = expr)
      end if
   end if
   for every node
      CSPEC (BPH, node, CSP, QoSDomain)
   end for
   if node is a leaf node
      addToCSP (Var node ∈ QoSDomain)
   end if
end
```

Figure 5.7: Constraint system building.

The algorithm takes the BP hypergraph, the node to start with, and the problem domain as an input, and it builds a constraint expression for every level of the hypergraph. Intuitively, the expression represents the quality of service for that level. For each level a new fresh variable is added and its range is restricted to the domain of the quality of service. Depending on what kind of children are available for that level different kind of expressions are built. If the children are connected with AND, the expression is built as an aggregation of the variables representing the children nodes. In the case of choice, there are different expressions for each child and an additional expression represents the fact that only one child will contribute to the execution (the sum of x_i).

Once the constraint expressions are built, the algorithm proceeds recursively on all children. If the node is a leaf node, then one simply adds a variable for that node and a constraint on the domain of the variable.

When the constraint system is in place, one can perform constraint propagation to find the solution space for acceptable qualities of services. If then one desires to have SLAs to attach to the BPs, it is simply a matter of performing a labelling of the solution space and obtaining satisfying values for the qualities of services. We remark that such a solution might not exists. In this case, the result of the methodology will be a set of processes, but with no quality guarantees.

Here we show the generation of SLAs based on given quality of service requirements for the execution of the business process using the loan origination process presented in Section 1.3. The example is based on the real data coming from an actual case study. In order to obtain the quality constraint expressions, we need to be given the domain over which the quality of service range, e.g., integers for costs or real numbers for response time. In the case of the proposed methodology, the QoS Domain is a vector of QoS with corresponding possible values for the parameters. The example

of the QoS Domain we consider is the following vector: [Execution Time (ET) ∈ N, Availability (Av) ∈ N, Time to Recover after an attack (TR) ∈ N].

Several examples of the aggregation functions are presented in [111]. Here we present aggregation functions for such QoS parameters as maximal execution time (Max ET), availability (Av) and maximal time to recover after an attack (Max TR) for sequential, parallel and choice structural activities are the following:

Activity	**Max ET**	**Av**	**Max TR**
sequence	$\varphi = \sum_{p=1}^{k} p_i$	$\varphi = \Pi_{p=1}^{k} p_i$	$\varphi = \sum_{p=1}^{k} p_i$
parallel	$\varphi = \sum_{p=1}^{k} p_i$	$\varphi = \Pi_{p=1}^{k} p_i$	$\varphi = \sum_{p=1}^{k} p_i$
choice	$\varphi = \max(p_1, ..., p_k)$	$\varphi = \min(p_1, ..., p_k)$	$\varphi = \max(p_1, ..., p_k)$

Example 9 *Next we present the quality constraint expressions obtained for the maximal execution time parameter navigating the BP hypergraph from Figure 5.5 following the algorithm.*

Maximal Execution Time (ET)

`LLO=LLO.ET+sum(RC.ET,MLO)`

`MLO=MLO.ET+max(IRC.ET,ERC)`

`ERC=ERC.ET+ERC1.ET` $\cdot x_1$ `+ERC2.ET` $\cdot x_2$ *when* $x_i \in 0, 1$ ***and*** $\sum x_i = 1$.

Availability (Av)

`LLO=LLO.Av` $\cdot \Pi$ `(RC.Av,MLO)`

`MLO=MLO.Av` $\cdot$ `min(IRC.Av,ERC)`

`ERC=ERC.Av` $\cdot$ `(ERC1.Av` $\cdot x_1$ `+ERC2.Av` $\cdot x_2$`)` *when* $x_i \in 0, 1$ ***and*** $\sum x_i = 1$.

Maximal Time to Recover after an attack (TR)

`LLO=LLO.TR+sum(RC.TR,MLO)`

`MLO=MLO.+max(IRC.TR,ERC)`

`ERC=ERC.TR+ERC1.TR` $\cdot x_1$ `+ERC2.TR` $\cdot x_2$ *when* $x_i \in 0, 1$ ***and*** $\sum x_i = 1$.

when `LLO` *stands for Launch Loan Origination,* `RC` *to Receive Customer,* `MLO` *to Manage Loan Origination,* `IRC` *to Internal Rating Check,* `ERC` *to*

External Rating Check, **ERC1** *and* **ERC2** *to External Rating Check(1) and External Rating Check(2) business processes.*

The constraint propagation for maximal execution time QoS property for the super-process in order to achieve execution time less then 35 seconds is performed and we get the following satisfying values for the qualities of services: **ERC1.ET**=*2 s,* **ERC2.ET**=*4 s,* **ERC.ET**=*10 s,* **IRC.ET**=*1 s,* **MLO.ET**=*5 s,* **RC.ET**=*7 s, and* **LLO.ET**=*8 s.*

The SLAs for the delegated business processes are the following.

SLA(LLO)=LLO=LLO.ET+sum(RC.ET,MLO)=8+7+19=34s
SLA(MLO)=MLO.ET+max(IRC.ET,ERC)=19s
SLA(ERC)=ERC.ET+ERC1.ET $\cdot x_1$**+ERC2.ET** $\cdot x_2$**=14s**
when $x_i \in 0, 1$ **and** $\sum x_i = 1$
when **MLO=MLO.ET+max(IRC.ET,ERC)=5+max(1,14)=19s**
ERC=ERC.ET+ERC1.ET $\cdot x_1$**+ERC2.ET** $\cdot x_2$**=10+2** $\cdot x_1$**+4** $\cdot x_2$**=10+4=14s**
when $x_i \in 0, 1$ **and** $\sum x_i = 1$.

Note that while choosing the business process among two alternatives External Rating Check(1) and External Rating Check(2) we rely on the trust levels of the providers Credit Bureau 1 and Credit Bureau 2 correspondingly. As the trust level of Credit Bureau 1 is 0 and the one of Credit Bureau 2 is 1, we choose the last option.

Finally, the obtained SLAs are described using the extended WS-Agreement specification. Furthermore, the agreements might be monitored with the option to anticipate violations and re-negotiated runtime. Refer to Chapter 4 for the extended WS-Agreement specification and example.

5.2 Constraint Reasoning for SLAs Derivation

As an illustration, the constraint algorithm for SLAs (in Appendix B) was implemented in the constraint solving environment ECLiPSe [1]. It is using the IC Hybrid Domain Solver [2]to solve the constraint problem, although only the finite domain capabilities.

The basic algorithm is a typical tree walking algorithm that can be applied to any subtree of a BPH. Constraints are applied from the leafs up, to make the algorithm fail early if a low-level constraint cannot be satisfied. All constraints over multiple child nodes are expressed as a user-defined predicate.

The core algorithm is implemented in two small predicates described below.

```
cspec(BPH, QoSDomain) :-
  qos(BPH, QoSValue),
  %constrain the QoS-Variable present in the current node
  %to the solution domain
  QoSValue #:: QoSDomain,
  %apply the cspec_algorithm on all children
  children(BPH, Children),
  cspec_children(Children, QoSDomain),
  %apply the function phi given for the current node
  apply_fun(BPH).

cspec_children([],_). cspec_children([Head|Tail],QoSDomain):-
  cspec(Head, QoSDomain),
  cspec_children(Tail, QoSDomain).
```

[1] http://www.eclipse-clp.org/

[2] http://eclipse-clp.org/doc/libman/libman016.html

The algorithm expects the BPH to be supplied in Prolog term form. A node in Prolog term form looks like this:

```
node(
  name:mpop,
  natural_name:"Manage product order process",
  aggregate_function:sum_qos,
  trust_level:1,
  cost:20,
  qos:MPOP_ET,
  children:[]
)
```

Note the unbound `qos` variable that will later be constrained. `cost` represents the generic cost variable depending on the scenario. In our case, `cost` represents the execution time of the node. `aggregate_function` is the predicate that is used by `apply_fun` to generate a constraint out of the nodes children.

The implementation walks the BPH in postorder fashion - the predicate `cspec_children` is applied before `apply_fun`. This means that walking the tree is of linear complexity. Under the reasonable assumption that the aggregation function is of linear complexity as well, the same is true for the application of `cspec`.

The prototype mainly exposes two predicates, one for convenient and one for programming use:

`bph:solve_label_and_print/2` builds the constraint system for a given BPH and a Quality of Service domain. It then prints all constrained variables. After that, a set of possible values for those variables is computed (labelling) and printed as well.

`bph:solve/4` only solves the problem of building the constraint system but does not label. It provides the user with the constrained BPH data structure as well as the list of constraint variables present in the BPH as a flatlist. This gives the ability to further modify the tree and to inspect the reduced constraint system.

The easiest way to interact with the system is through the `tkeclipse` interface that is provided with ECLiPSe shown in Figure 5.8. `tkeclipse` allows to directly inspect the constraint system computed and provide a good way of handling and compiling ECLiPSe modules. It requires some knowledge about Prolog but in turn provides a good way to follow the execution of the algorithm.

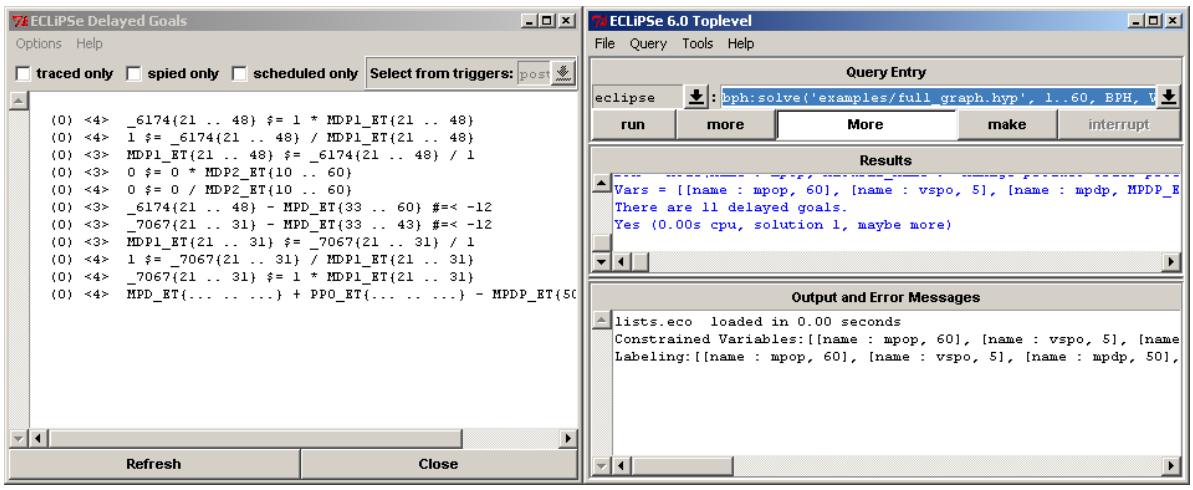

Figure 5.8: tkeclipse: Main Window and Outstanding Constraints.

Our prototype is well separated in 4 modules: data structures, aggregation functions, graph loading and the algorithm implementation. It is properly documented and easily to integrate in other work.

5.3 Concluding Remarks

The SLAs an enterprise has with its service providers must support its business goals insofar as possible. Establishing a service level agreement that favors the business objectives requires significant commitment of resources from the enterprise side, therefore any automation and support that can be obtained for this task is greatly beneficial for the enterprise.

This chapter proposes the BP&SLA methodology for designing Web service-based BPs with related SLAs. The proposal fills the gap that exists between the informally specified early business requirements the user provides and the executable BP. The idea is to enrich business processes with SLAs which are favorable for the enterprise in order to achieve its business objectives with specific QoS. As the activities about assignment of responsibilities on business processes need to be carefully considered from the security point of view, the proposed methodology focuses on security and trust aspects. The framework supports the Secure BPEL language that allows for secure BPs specification.

To show the potential impact of the approach, we illustrate the functioning of the methodology on an e-business case study inspired by an actual research project use case. In [78], we considered the Collaborative Procurement&Logistics use case in the Sekhukhune Rural Living Lab that is a working scenario of the IST-FP7-IP-C@R research project[2].

[2]C@R (Collaboration and Rural) is a R&D project funded by the European Union. The project aims to enable people in remote and rural Europe to fully participate in the knowledge society as citizens and as professionals, http://www.c-rural.eu.

Chapter 6

Conclusion and Perspective

A strong link between enterprise business processes and the company incomes is important. Any support in assisting business process analysts in deriving secure business processes from early requirements analysis is highly required. Enterprises aim to align the service level agreements as much as possible with the business goals. This allows for better planning and reducing costs, facilitating delivery of new kind of services, and convincing management to try new services and applications. While, development an appropriate service level agreement supporting business goals of an enterprise is not a trivial task, it requires significant commitment of resources from the enterprise side. Therefore, any automation that can be obtained for this task is greatly beneficial.

In this thesis we focused on the problem of engineering secure Web service-based business processes with service level agreements from early requirements.

We addressed the problem of secure Web service-based business processes modelling based on the analysis of early requirements. We presented a refinement methodology that allows for obtaining an executable Web service-based secure workflow from early requirements modelled in the Secure Tropos formalism. We introduced a specification language Se-

cure BPEL for secure business processes that allows the workflow engine to automatically enforce trust and delegation requirements. We filled the gap between the requirements engineering methodologies and the actual generation of business processes based on Service-Oriented Architectures with particular emphasis on the security aspects.

We addressed the issue of formalization of the notion of an agreement and proposed a formal representation for the internal and external states of an agreement. We presented a formal analysis of WS-Agreement by resorting to finite state automata and providing a set of formal rules that tie together terms and the life-cycle of an agreement. Such formalization allowed us to discover that an agreement could be made more long-lived and robust with respect to forecoming violations. We presented the details of the proposed extension in formal terms and evaluated the approach through simulation experiments. In the experimentation, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings.

Finally, we proposed a methodology to design Web service-based business processes together with service level agreements that guarantee a certain quality of execution, with particular emphasis on the security aspects. Starting from an early requirements analysis modelled in the Secure Tropos formalism, we provided a set of user-guided transformations and reasoning tools. The final output obtained was a set of processes in the form of Secure BPEL together with a set of service level agreements in WS-Agreement to be signed by participating services. The constraint algorithm for SLAs was implemented in the constraint solving environment ECLiPSe. We used the IC Hybrid Domain Solver to solve the constraint problem. To showed the feasibility of the approach, we evaluated the performance of the methodology on an e-business banking scenario, more specifically, from a typical loan origination process, inspired by an actual research project use case.

The work proposed in this dissertation suggests several directions for future investigation.

SLA Monitoring for Secure Business Process

The proposed framework provides monitoring based on the client's goal described in service level agreement and focuses on how the guarantee is fulfilled. The approach predicts and notifies terms violations. However, the framework extension for SLA monitoring of secure business process is an open issue. The extended framework should not only answer to the question "Is the guarantee close or far from being violated?", but it should also discover the components of the composition that are responsible for the violation. Another issue for future work is the auditing of service level agreement for secure Web service-based business processes.

"Lack of Permission" Problem in Secure Business Process

We proposed the Secure BPEL language as a specification language for secure business process that allows the workflow engine to automatically enforce trust and delegation requirements. While proceeding from early requirements analysis towards secure workflows, we faced the so called "lack of permission" problem. The "lack of permission" situation appears when there is a chain of delegation of execution with no corresponding chain of delegation of permission. Each delegator of execution delegates on execution of a goal to the corresponding delegatee. The delegatee plays the role of delegator of execution and delegates on execution of the goal to other delegatee, etc. When there is no corresponding chain of delegation of permission, the root delegator of execution delegates permission of the goal only to the leaf actor that actually executes the goal. At this point, all the other nodes face the "lack of permission" problem: the actor has delegation of execution, but no permission for doing it. The same holds for trust. In the Secure BPEL language both delegation and trust are modelled by invocations. The delegation of execution concept is modelled

as invocation of an operation by one partner from the other partner. The concepts of delegation of permission/execution are modelled as different types of security services invocation. In order to address the "lack of permission" problem, special types of invocations should be introduced. The new invocation should allow the data to be protected, i.e., allows message confidentiality and integrity. An issue for future investigation is the integration in the language the details of the low level secure requirements of messages integrity and confidentiality.

Multi-Requirement Analysis for SLA Engineering

In the proposed methodology we introduce an intermediate structure business process hypergraph for reasoning about the business processes and their qualities. To calculate the value of a target node taking as arguments source nodes with the structural activity associated for each QoS parameter, we use aggregation functions. The problem of finding service level agreements for business processes is then a problem of reasoning on the business process hypergraph. Further research will be devoted on multi-requirement analysis, when the focus is on several QoS parameters at the same time. This will require the identification of a decision-making function that selects the preferred set of attributes. Definition and validation of the aggregation functions for more QoS requirements will be a challenge.

Bibliography

[1] R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint Driven Web Service Composition in METEOR-S. In *Proceedings of the 2004 IEEE International Conference on Services Computing*, 2004.

[2] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Mller, G. Pfau, K. Plsser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, and M. Zeller. WS-BPEL Extension for People (BPEL4People) 1.0, June 2007.

[3] G. Ahn and R. Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and System Security*, 3(4):2007–226, 2000.

[4] M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? A Formal Analysis and an Extension of WS-Agreement. Technical Report DIT-05-039, DIT, University of Trento, 2005.

[5] M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? An Analysis and an Extension of WS-Agreement. In *Proceedings of the Third International Conference on Service Oriented Computing*, 2005.

[6] M. Aiello and P. Giorgini. Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Quali-

ties of Services. *CEPIS Upgrade - The European journal of the informatics professional*, 5(4):20–26, 2004.

[7] M. Aiello and P. Giorgini. Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities of Services. *CEPIS Upgrade - The European Journal of the Informatics Professional*, 5(4):20–26, 2004.

[8] M. Aiello, F. Rosenberg, C. Platzer, A. Ciabattoni, and S. Dustdar. QoS Composition at the Level of Part Names. In *Proceedings of the Third International Workshop on Web Services and Formal Methods*, 2006.

[9] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guizar, N. Kartha, K. Liu, R. Khalaf, D. Konig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web Services Business Process Execution Language 2.0, April 2007.

[10] D. Amyot. Introduction to the User Requirements Notation: Learning by Example. *Computer Networks*, 43(3):285–301, 2003.

[11] S. Anderson, A. Grau, and C. Hughes. Specification and Satisfaction of SLAs in Service Oriented Architectures. In *Proceedings of the Fifth Annual DIRC Research Conference*, 2005.

[12] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services 1.1, May 2003.

[13] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement), March 2007.

[14] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Negotiation Specification (WS-AgreementNegotiation), 2009. Draft.

[15] A. Andrieux, A. Dan, K. Keahey, H. Ludwig, and J. Rofrano. Negotiability Constraints in WS-Agreement. Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, 2004.

[16] V. Andrikopoulos, S. Benbernou, and M.P. Papazoglou. Evolving Services from a Contractual Perspective. In *Proceedings of the 21st International Conference on Advanced Information Systems*, 2009.

[17] S. Angelov and P. Grefen. An E-contracting Reference Architecture. *The Journal of Systems and Software*, 81(11):1816–1844, 2008.

[18] K. Appleby, S.B. Calo, J.R. Giles, and K.-W. Lee. Policy-based Automated Provisioning. *IBM Systems Journal*, 43(1):121–135, 2004.

[19] D. Ardagna and B. Pernici. Dynamic web service composition with qos constraints. *International Journal of Business Process Integration and Management*, 1(4):233–243, 2006.

[20] A. Arsanjani, B. Hailpern, J. Martin, and P. Tarr. Web services: Promises and compromises. *Queue*, 1(1):48–58, 2003.

[21] Y. Asnar, P. Giorgini, F. Massacci, and N. Zannone. From Trust to Dependability through Risk Analysis. In *Proceedings of the the*

Second International Conference on Availability, Reliability and Security, 2007.

[22] A. Atzeni and A. Lioy. Why to Adopt a Security Metric? A Brief Survey. In *Proceedings of the First Workshop on Quality of Protection*, 2005.

[23] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Malhotra, A. Nadalin, N. Nagaratnam, H. Prafullchandra M. Nottingham, C. von Riegen, J Schlimmer, C. Sharp, and J. Shewchuk. Web Services Policy Framework (WS-Policy) 1.2, April 2006.

[24] J. Bart. SLA Savvy: Five Secrets for Making Sure you Get the Most from Your Service-Level Agreements. *Network World*, 1999.

[25] M. Bartoletti, P. Degano, and G.L. Ferrari. Security Issues in Service Composition. In *Proceedings of the 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems*, 2006.

[26] M. Bartoletti and R. Zunino. A Logic for Contracts. Technical Report DISI-09-034, DISI, University of Trento, September 2009.

[27] T. Bellwood, S. Capell, L. Clement, J. Colgrave, Dovey M.J., D. Feygin, A. Hately, R. Kochman, P. Macias, M. Novotny, C. Paolucci, M. von Riegen, T. Rogers, K. Sycara, P. Wenzel, and Z Wu. Uddi version 3.0.2, October 2004.

[28] K. Bennett, P. Layzel, D. Budgen, P. Brereton, L. Macaulay, and M. Munro. Service-Based Software: The Future for Flexible Software. In *Proceedings of the Seventh Asia-Pacific Software Engineering Conference*, 2000.

[29] K. Bhargavan, R. Corin, C. Fournet, and A.D. Gordon. Secure Sessions for Web Services. *ACM Transactions on Information and System Security*, 10(2), 2007.

[30] A.S. Bilgin and M.P. Singh. A DAML-Based Repository for QoS-Aware Semantic Web Service Selection. In *Proceedings of the Second International Conference on Web Services*, 2004.

[31] M. Bitsaki, O. Danylevych, W.-J. Heuvel, G. Koutras, F. Leymann, M. Mancioppi, C. Nikolaou, and M. Papazoglou. An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network. In *Proceedings of the First European Conference on Towards a Service-Based Internet*, 2008.

[32] K. Blomqvist and P. Ståhle. Building Organizational Trust. In *Proceedings of the 16th Annual Conference on Industrial Marketing and Purchasing*, 2000.

[33] M.A. Bochicchio, V. D'Andrea, N. Kokash, and F. Longo. Conceptual Modelling of Service-Oriented Systems. In *Proceedings of the International Conference on Web Engineering Workshops*, 2007.

[34] R.K. Bock and W. Krischer. *The Data Analysis: Briefbook*. Springer Verlag, 1998.

[35] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[36] M.J. Buco, R.N. Chang, L.Z. Luan, C. Ward, J.L. Wolf, and P.S. Yu. Utility Computing SLA Management Based upon Business Objectives. *IBM Systems Journal*, 43(1):159–178, 2004.

[37] M.G. Buscemi and U. Montanari. CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements. In *Proceedings of the 16th European Symposium on Programming*, 2007.

[38] M.E. Cambronero, G. Diaz, J.J. Pardo, and V. Valero. Using UML Diagrams to Model Real-Time Web Services. In *Proceedings of the Second International Conference on Internet and Web Applications and Services*, 2007.

[39] C. Cappiello, M. Comuzzi, and P. Plebani. On Automated Generation of Web Service Level Agreements. In *Proceedings of the 19th International Conference on Advanced Information Systems Engineering*, 2007.

[40] P. Cappiello, C. amd Missier, B. Pernici, P. Plebani, and C. Batini. QoS in Multichannel IS: The MAIS Approach. In *Proceedings of the International Workshop on Web Quality in conjunction with the Fourth International Conference on Web Engineering*, 2004.

[41] J Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of Service for Workflows and Web Service Processes. *Journal of Web Semantics*, 1(3):281–308, 2004.

[42] F. Casati, M-C. Shan, and D. Georgakopoulos. E-Services - Guest editorial. *The VLDB Journal*, 10(1), 2001.

[43] C. Castelfranchi and R. Falcone. Principles of Trust for MAS: Cognitive Anatomy, Social Importance and Quantification. In *Proceedings of the Third International Conference on Multiagent Systems*, 1998.

[44] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai. Translating Service Level Objectives to Lower Level Policies for Multi-tier Services. *Journal of Cluster Computing*, 11(3):299–311, 2008.

[45] B.H.C. Cheng, S. Konrad, L.A. Campbell, and R. Wassermann. Using Security Patterns to Model and Analyze Security Requirements. In *Proceedings of the IEEE Workshop on Requirements for High Assurance Systems*, 2003.

[46] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana. Web Services Description Language (WSDL) 1.1, June 2007.

[47] N. Chudasma and S. Chaudhary. Service Composition Using Service Selection with WS-Agreement. In *Proceedigs on the Second Bangalore Annual Compute Conference*, 2009.

[48] D.D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of the 1992 ACM SIGCOMM*, 1992.

[49] W. Classen. Fundamentals of Software Licensing. *IDEA: The Journal of Law and Technology*, 37(1), 1996.

[50] E. Colombo, J. Mylopoulos, and P Spoletini. Modeling and Analyzing Context-aware Composition of Services. In *Proceedings of the Third International Conference on Service-Oriented Computing*, 2005.

[51] M. Comuzzi and B. Pernici. A Framework for the QoS-Based Web Service Contracting. *ACM Transaction on the Web*, 3(3), 2009.

[52] I. Crnkovic, M. Larsson, and O. Preiss. Concerning Predictability in Dependable Component-Based Systems: Classification of Quality Attributes. In *Architecting Dependable Systems III, LNCS 3549*. Springer Verlag, 2005.

[53] R.L. Cruz. Quality of Service Guarantees in Virtual Circuit Switched Networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, 1995.

[54] L. M. Cysneiros and E. Yu. Requirements Engineering for Large-Scale Multi-Agent Systems. In *Software Engineering for Large-Scale Multi-Agent Systems Research Issues and Practical Applications, LNCS 2603*. Springer Verlag, 2003.

[55] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based Grid Service Management (OGSI-Agreement). Technical report, Global Grid Forum, GRAAP-WG Author Contribution, 2003.

[56] K. Czajkowski, I.T. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *Processings of the Eighth Workshop on Job Scheduling Strategies for Parallel Processing*, 2002.

[57] A. Dan, K. Keahey, H. Ludwig, and J. Rofrano. Guarantee Terms in WS-Agreement. Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, 2004.

[58] V. D'Andrea and G.R. Gangadharan. Licensing Services: The Rising. In *Proceedings of the International Conference on Internet and Web Applications and Services*, 2006.

[59] J.L. de la Vara, J. Sánchez, and Ó. Pastor. Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering*, 2008.

[60] H. Demirkan, M. Goul, and D.S. Soper. Service Level Agreement Negotiation: A Theory-based Exploratory Study as a Starting Point for Identifying Negotiation Support System Requirements. In *Proceed-*

ings of the 38th Hawaii International Conference on System Sciences, 2005.

[61] J. den Hartog, C. Hutter, and S. Trabelsi. Combined Trust Management Architecture. In *Proceedings of the International Conference on Mobility, Individualisation, Socialisation and Connectivity*, 2010.

[62] V. Deora, J. Shao, W.A. Gray, and N.J. Fiddian. Expectation-Based Quality of Service Assessment. *International Journal on Digital Libraries*, 6(2):260–269, 2006.

[63] M. Deubler, M. Meisinger, S. Rittmann, and I. Kruger. Modeling Crosscutting Services with UML Sequence Diagrams. In *Proceedings of the ACM/IEEE Eighth International Conference on Model Driven Engineering Languages and Systems*, 2005.

[64] G. Di Modica, O. Tomarchio, and L. Vita. Dynamic SLAs Management in Service Oriented Environments. *Journal of Systems and Software*, 82(5):759–771, 2009.

[65] M. Di Penta, G. Canfora, G. Esposito, V. Mazza, and M. Bruno. Search-based Testing of Service Level Agreements. In *Proceedings of the Ninth Conference on Genetic and Evolutionary Computation*, 2007.

[66] D. Distante, G. Rossi, G. Canfora, and S. Tilley. A Comprehensive Design Model for Integrating Business Processes in Web Applications. *International Journal of Web Engineering and Technology*, 3(1):43–72, 2007.

[67] D. Domingos, A. Rito Silva, and P. Veiga. Workflow Access Control from a Business Perspective. In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, 2004.

[68] J. Dörflingerd, G. Frankova, A. Lucientes, R. de Louw, M. Navarro, C. Peña, C. Ralli, and T. Robles. Enhancing an Open Service Oriented Architecture with Collaborative Functions for Rural Areas. In *Proceedings of the 14th International Conference on Concurrent Enterprising*, 2008.

[69] S. Dustdar and W. Schreiner. A Survey on Web Services Composition. *International Journal of Web and Grid Services*, 1(1):1–30, 2005.

[70] A. Elfatatry and P. Layzell. A Negotiation Description Language. *Software - Practice and Experience*, 35(4):323–343, 2005.

[71] A. Elfatatry and P. Layzell. Negotiating in Service-Oriented Environments. *Communication of the ACM*, 47(8):103–108, 2005.

[72] R. Falcone and C. Castelfranchi. Social Trust: A Cognitive Approach. In *Trust and Deception in Virtual Societies*. Kluwer Academic Publishers, 2001.

[73] International Organization for Standardization. Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality requirements, 2007. ISO/IEC 25030.

[74] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. Grid Services for Distributed System Integration. *IEEE Computer*, 35(6, pages =), 2002.

[75] G. Frankova. An Application in e-Business Using Web Services". Master's thesis, Computer Science Department, Applied Mathematics Faculty, Dnepropetrovsk National University, 2004.

[76] G. Frankova. Web Service Quality Composition Modelling. In *Proceedings of the IBM PhD Symposium at the Third International Con-*

ference on Service-Oriented Computing, 2005. IBM Research Report RC23826.

[77] G. Frankova. Service Level Agreements: Web Services and Security. In *Proceedings of the Seventh International Conference on Web Engineering*, 2007. Doctoral Consortium.

[78] G. Frankova, M. Aiello, M. Séguran, F. Gilcher, S. Trabelsi, and J. Dörflingerd. Deriving Business Processes with Service Level Agreements from Early Requirements. 2009. Manuscript submitted to the Journal of Systems and Software.

[79] G. Frankova, A. Chibani, Y. Amirat, and F. Sannicolò. Towards Context Modeling for Cooperative Rural Living Labs. In *Collaboration and the Knowledge Economy: Issues, Applications, Case Studies*, The eChallenges e-2008 Conference. IOS Press, 2008.

[80] G. Frankova, D. Malfatti, and M. Aiello. Semantics and Extensions of WS-Agreement. *Journal of Software*, 1(1):23–31, 2006.

[81] G. Frankova, F. Massacci, and M. Séguran. From Early Requirements Analysis towards Secure Workflows. Technical Report DIT-07-036, DIT, University of Trento, 2007.

[82] G. Frankova, F. Massacci, and M. Séguran. From Early Requirements Analysis towards Secure Workflows. In *Proceedings of the joint iTrust and PST Conferences on Privacy, Trust Management and Security*, 2007.

[83] G. Frankova and A. Yautsiukhin. Service and Protection Level Agreements for Business Processes. European Young Researchers Workshop on Service Oriented Computing, 2007.

[84] G. Frankova, A. Yautsiukhin, and M. Séguran. From Early Requirements to Business Processes with Service Level Agreements. Technical Report DIT-07-037, University of Trento, 2007.

[85] S. Frølund and J. Koistinen. Quality-of-Service Specification in Distributed Object Systems. *Distributed Systems Engineering*, 5(4):179–202, 1998.

[86] H.M. Frutos, I. Kotsiopoulos, L.M.V. Gonzalez, and R.D. Merino. Enhancing Service Selection by Semantic QoS. In *Proceedings of the Sixth European Semantic Web Conference on The Semantic Web: Research and Applications*, 2009.

[87] M.G. Fugini, P. Plebani, and F. Ramoni. A User Driven Policy Selection Model. In *Proceedings of the Fourth International Conference on Service Oriented Computing*, 2006.

[88] A. Fuxman, L. Liu, j. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso. Specifying and Analyzing Early Requirements in Tropos. *Requirements Engineering*, 9(2):132–150, May 2004.

[89] D. Gambetta. Can We Trust Trust? In *Trust: Making and Breaking Cooperative Relations*. Blackwell Publishers, 1990.

[90] G.R. Gangadharan and V. D'Andrea. Licensing Services: Formal Analysis and Implementation. In *Proceedings of the Forth International Conference on Service Oriented Computing*, 2006.

[91] G.R. Gangadharan, G. Frankova, and V. D'Andrea. Service License Life Cycle. In *Proceedings of the International Symposium on Collaborative Technologies and Systems*, 2007.

[92] T. Gardner. UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In *Proceedings of the Second European Workshop on Object Orientation and Web Services*, 2004.

[93] G. Georg, I. Ray, and R. France. Using Aspects to Design a Secure System. In *Proceedings of the Eight IEEE International Conference on Engineering of Complex Computer Systems*, 2002.

[94] H. Gimpel, H. Ludwig, A. Dan, and B. Kearney. PANDA: Specifying Policies for Automated Negotiations of Service Contracts. In *Proceedings of the First International Conference on Service Oriented Computing*, 2003.

[95] 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.

[96] D. Gouscos, M. Kalikakis, and P. Georgiadis. An Approach to Modeling Web Service QoS and Provision Price. In *Proceedings of the First Web Services Quality Workshop at the Fourth International Conference on Web Information Systems Engineering*, 2003.

[97] T. Grandison and M. Sloman. Specifying and Analysing Trust for Internet Applications. In *Proceedings of the Second IFIP Conference on Towards The Knowledge Society: E-Commerce, E-Business, E-Government*, 2002.

[98] T. Grandison and M. Sloman. Trust Management Tools for Internet Applications. In *Proceedings of the First International Conference on Trust Management*, 2003.

[99] D. Greenwood, G. Vitaglione, L. Keller, and M. Calisti. Service Level Agreement Management with Adaptive Coordination. In *Proceedigs on the Second International conference on Networking and Services*, 2006.

[100] Open Management Group. Object Constraint Language (OCL), Version 2.0, May 2006.

[101] Open Management Group. Unified Modeling LanguageTM (UML), Version 2.2, February 2009.

[102] M. Gudgin, M. Hadley, N. Mendelsohn, H.F. Moreau, J. Nielsen, A. Karmarkar, and Y. Lafon. SOAP Version 1.2, April 2007.

[103] C.B. Haley, R. Laney, J.D. Moffett, and B. Nuseibeh. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008.

[104] C.B. Haley, R.C. Laney, and B. Nuseibeh. Deriving Security Requirements from Crosscutting Threat Descriptions. In *Proceedings of the Third International Conference on Aspect-Oriented Software Development*, 2004.

[105] P. Hasselmeyer, B. Koller, M. Parkin, and P. Wieder. An SLA Renegotiation Protocol. In *Proceeding of the Second Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop*, 2008.

[106] P. Hasselmeyer, C. Qu, L. Schubert, B. Koller, and P. Wieder. Towards Autonomous Brokered SLA Negotiation. In *Exploiting the Knowledge Economy - Issues, Applications, Case Studies*, volume 3. IOS Press, 2006.

[107] D. Hatebur, M. Heisel, and H. Schmidt. Analysis and Component-based Realization of Security Requirements. In *Proceedings of the Third International Conference on Availability, Reliability and Security*, 2008.

[108] Q. He, J. Yan, R. Kowalczyk, H. Jin, and Y. Yang. Lifetime Service Level Agreement Management with Autonomous Agents for Services Provision. *Information Sciences*, 179(15):2591–2605, 2009.

[109] How to Series. SLA: Getting It Right - An Enterprise's Business Objectives Should Form the Fundamental Basis of an SLA. Voice&Data, March 2005.

[110] P.C.K. Hung, H. Li, and J. Jeng. WS-Negotiation: An Overview of Research Issues. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[111] M.C. Jaeger, G. Rojec-Goldmann, and G. Mühl. QoS Aggregation in Web Service Compositions. In *Proceedings of the Seventh IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005.

[112] H. Jin and H. Wu. Semantic-enabled Specification for Web Services Agreement. *International Journal of Web Services Practices*, 1(1–2):12–20, 2005.

[113] L. Jin, V. Machiraju, and A. Sahai. Analysis on Service Level Agreement of Web Services. Technical Report HPL-2002-180, Hewlett-Packard Laboratories Palo Alto, Software Technology Laboratory, June 2002.

[114] A. Jøsang. Trust and Reputation Systems. In *Foundations of Security Analysis and Design. Tutorial Lectures, LNCS 4677*. Kluwer Academic Publishers, 2007.

[115] R. Jurca, W. Binder, and B. Faltings. Reliable QoS Monitoring Based on Client Feedback. In *Proceedings of the 16th International World Wide Web Conference*, 2007.

[116] I.J. Jureta, C. Herssens, and S. Faulkner. A Comprehensive Quality Model for Service-Oriented Systems. *Software Quality Journal*, 17(1):65–98, 2009.

[117] J. Jürjens. *Secure Systems Development with UML*. Springer Verlag, 2004.

[118] R.S. Kaabi, C. Souveyet, and C. Rolland. Eliciting Service Composition in a Goal Driven Manner. In *Proceedings of the Second International Conference on Service Oriented Computing*, 2004.

[119] H. Kaminski and M. Perry. SLA Automated Negotiation Manager for Computing Services. In *Proceedings of the The Eighth IEEE International Conference on E-Commerce Technology and The Third IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, 2006.

[120] N. Karten. *How to establish Service Level Agreements*. Karten Assosiates, 1998.

[121] R. Kazhamiakin, M. Pistore, and M. Roveri. A Framework for Integrating Business Processes and Business Requirements. In *Proceeding of the the Eighth International IEEE Enterprise Distributed Object Computing Conference*, 2004.

[122] A. Keller and H. Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management, Special Issue on E-Business Management*, 11(1):57–81, 2003.

[123] K Knorr and S. Röhrig. Security Requirements of E-Business Processes. In *Proceedings of the IFIP Conference on Towards The E-Society: E-Commerce, E-Business, E-Government*, 2001.

[124] D.D. Lamanna, J. Skene, and W. Emmerich. SLAng: A Language for Defining Service Level Agreements. In *Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, 2003.

[125] A. Lapouchnian, Y. Yu, and J. Mylopoulos. Requirements-Driven Design and Configuration Management of Business Processes. In *Proceeding of the Fifth International Conference on Business Process Management*, 2007.

[126] D. Lau and J. Mylopoulos. Designing Web Services with Tropos. In *Proceedings of the Second IEEE International Conference on Web Services*, 2004.

[127] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist. Web Services Trust Language (WS-Trust) 1.3, March 2007.

[128] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist. WS-SecurityPolicy 1.2, July 2007.

[129] K. Lawrence, C. Kaler, A. Nadalin, R. Monzillo, and P. Hallam-Baker. Web Services Security: SOAP Message Security 1.1 (WS-Security), February 2006.

[130] K. Lee, J. Jeon, W. Lee, S. Jeong, and S. Park. QoS for Web Services: Requirements and Possible Approaches. W3C Working Group, November 2003.

[131] M. Lemley, P. Menell, R. Merges, and P. Samuelson. *Software and Internet Law*. Aspen Publishers, 2006.

[132] H. Li, S.Y.W. Su, and H. Lam. On Automated e-Business Negotiations: Goal, Policy, Strategy, and Plans of Decision and Action. *Journal of Organizational Computing and Electronic Commerce*, 13(1):1–29, 2006.

[133] M. Lin, J. Xie, H. Guo, and H. Wang. Solving QoS-Driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction. In *Proceedings of the Seventh IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005.

[134] H. Lockhart, S. Andersen, J. Bohren, Y. Sverdlov, M. Hondo, H. Maruyama, N. Nadalin, A. Nagaratnam, T. Boubez, K.S. Morrison, C. Kaler, A. Nanda, D. Schmidt, D. Walters, H. Wilson, L. Burch, D. Earl, S. Baja, and H. Prafullchandra. Web Services Federation Language (WS-Federation) 1.1, December 2006.

[135] A. Longo. *"Conceptual Modelling of Business Process in Web Application Design"*. PhD thesis, University of Lecce, Innovation Engineering Department, 2004.

[136] A. Ludwig and M. Kowalkiewicz. Supporting Service Level Agreement Creation with Past Service Behavior Data. In *Proceedings of the First Workshop on Service Discovery and Selection in SOA Ecosystems*, 2009.

[137] H. Ludwig. Web Services QoS: External SLAs and Internal Policies or How do we Deliver what we Promise? In *Proceedings of the First Web Services Quality Workshop at the Fourth International Conference on Web Information Systems Engineering*, 2003.

[138] H. Ludwig, A. Dan, and R. Kearney. CREMONA: an Architecture and Library for Creation and Monitoring of WS-Agreements. In *Proceedings of the Second International Conference on Service-Oriented Computing*, 2004.

[139] H. Ludwig, A. Keller, A. Dan, R.P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. Version 1.0. IBM Corporation, January 2003.

[140] P. Madsen. WS-Trust: Interoperable Security for Web Services. June 2003.

[141] D. Malfatti. A Framework for the Monitoring of the QoS by extending WS-Agreement. Master's thesis, Corso di Laurea in Informatica, Università degli Studi di Trento, 2005. In Italian.

[142] A. Mani and A. Nagarajan. Understanding Quality of Service for Web Services. IBM DeveloperWorks Technical Paper, 2002.

[143] E. Marcos, V. de Castro, and B. Vela. Representing Web Services with UML: A Case Study. In *Proceedings of the First International Conference on Service-Oriented Computing*, 2003.

[144] F. Massacci, J. Mylopoulos, and N. Zannone. An Ontology for Secure Socio-Technical Systems. In *Handbook of Ontologies for Business Interaction*. IDEA, 2007.

[145] E.M. Maximilien and M.P. Singh. A Framework and Ontology for Dynamic Web Services Selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[146] F.L. Mayer. A Brief Comparison of Two Different Environmental Guidelines for Determining "Levels of Trust". In *Proceedings of the Sixth Annual Computer Security Applications Conference*, 1990.

[147] P. McKee, S.J. Taylor, M. Surridge, R. Lowe, and C. Ragusa. Strategies for the Service Market Place. In *Proceedings of the Fourth International Workshop on Grid Economics and Business Models*, 2007.

[148] D.H. McKnight and N.L. Chervany. The Meanings of Trust. Technical Report WP 96-04, University of Minnesota, MIS Research Center, 1996.

[149] D.A. Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[150] D.A. Menascé, H. Ruan, and H. Gomaa. QoS Management in Service-Oriented Architectures. *Performance Evaluation Journal*, 64(7–8):646–663, 2007.

[151] C. Molina-Jimenez, J. Pruyne, and A. van Moorsel. The Role of Agreements in IT Management Software. In *Architecting Dependable Systems III, LNCS 3549*. Springer Verlag, 2005.

[152] C. Molina-Jimenez, S. Shrivastava, E. Solaiman, and J. Warne. Runtime Monitoring and Enforcement of Electronic Contracts. *Electronic Commerce Research and Applications*, 3(2):108–125, 2004.

[153] C. Molina-Jimenez, S.K. Shrivastava, J. Crowcroft, and P. Gevros. On the Monitoring of Contractual Service Level Agreements. In

Proceedings of the First IEEE International Workshop on Electronic Contracting, 2004.

[154] C. Müller, O. Mart'ın-D'ıaz, A. Ruiz-Cortés, M. Resinas, and P. Fernández. Improving Temporal-Awareness of WS-Agreement. In *Proceedings of the Fifth International Conference on Service-Oriented Computing*, 2007.

[155] C. Müller, A. Ruiz-Cortés, and M. Resinas. An Initial Approach to Explaining SLA Inconsistencies. In *Proceedings of the Sixth International Conference on Service-Oriented Computing*, 2008.

[156] N. Muller. Managing Service Level Agreements. *International Journal of Network Management*, 9:155–166, 1999.

[157] J. Myerson. Use SLAs in a Web Services Context. IBM DeveloperWorks Technical Paper, 2002. Part 1.

[158] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):488–497, 1992.

[159] T. Neubauer, M. Klemen, and S. x. Biffl. Secure Business Process Management: A Roadmap. In *Proceedings of the First International Conference on Availability, Reliability and Security*, 2006.

[160] B. Nuseibeh, C.B. Haley, and C. Foster. Securing the Skies: In Requirements We Trust. *Computer*, 42(9):64–72, 2009.

[161] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic WS-Agreement Partner Selection. In *Proceedings of the 15th International World Wide Web Conference*, 2006.

[162] J. O'Sullivan, D. Edmond, and A.H.M. ter Hofstede. What's in a Service? Towards Accurate Description of Non-Function Service Properties. *Distributed and Parallel Databases*, 12(2–3):117–133, 2002.

[163] M. Papazoglou, V. D'Andrea, D. Plexousakis, P. Grefen, J. Yang, M. Mecella, and P. Plebani. SOC: Service-Oriented Computing Manifesto, 2003.

[164] M. Papazoglou and D. Georgakopoulos. Service-Oriented Computing. *Communications of the ACM*, 46(10):25–28, 2003.

[165] M.P. Papazoglou and J. Yang. Design Methodology for Web Services and Business Processes. In *Proceedings of the International Workshop on Technologies for E-Services*, 2002.

[166] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. From Stakeholder Needs to Service Requirements. In *Proceeding of International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements*, 2006.

[167] A. Pichot, O. Waldrich, W. Ziegler, and P. Wieder. Towards Dynamic Service Level Agreement Negotiation:An Approach Based on WS-Agreement. In *Proceeding of the Fouth International Conference on Web Information Systems and Technologies*, 2008.

[168] D.A.C. Quartela, M.W.A. Steen, S. Pokraev, and M. van Sinderena. A Conceptual Framework for Service Modelling. In *Proceedings of the Tenth IEEE International Enterprise Distributed Object Computing Conference*, 2006.

[169] S. Ran. Model for Web Services Discovery with QoS. *SIGEcom Exchanges*, 4(1):1–10, 2004.

[170] O. Rana, M. Warnier, T.B. Quillinan, and F. Brazier. Monitoring and Reputation Mechanisms for Service Level Agreements. In *Proceedings of the Fifth International Workshop on Grid Economics and Business Models*, 2008.

[171] D.M. Reeves, M.P. Wellman, and B.N. Grosof. Automated Negotiation from Declarative Contract Descriptions. In *Computational Intelligence*, volume 18, pages 482–500, 2002.

[172] A. Sahai, A. Durante, and V. Machiraju. Towards Automated SLA Management for Web Services. Technical Report HPL-2001-310, Hewlett-Packard Laboratories Palo Alto, Software Technology Laboratory, July 2001.

[173] A. Sahai, V. Machiraju, M. Sayal, A. van Moorsel, F. Casati, and L.J. Jin. Automated SLA Monitoring for Web Services. In *Proceedings of the 13th International Workshop on Distributed Systems: Operations and Management*, 2002.

[174] K. Salamatian and S. Fdida. Measurement Based Modeling of Quality of Service in the Internet: A Methodological Approach. In *Proceedings of the Thyrrhenian International Workshop on Digital Communications: Evolutionary Trends of the Internet*, 2001.

[175] H. Schmidt. Service Contracts Based on Workflow Modeling. In *Proceedings of the 11th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Services Management in Intelligent Networks*, 2000.

[176] M. Séguran, C. Hébert, and G. Frankova. Secure Workflow Development From Early Requirements Analysis. In *Proceedings of the Sixth IEEE European Conference on Web Services*, 2008.

[177] R. Simon and M. Zurko. Separation of Duty in Role-Based Environments. In *Proceedings of Computer Security Foundations Workshop*, 1997.

[178] G. Sindre and A.L. Opdahl. Eliciting Security Requirements with Misuse Cases. *Requirements Engineering*, 10(1):34–44, 2005.

[179] D. Skogan, R. Grønmo, and I. Solheim. Web Service Composition in UML. In *Proceedings of the Eighth International IEEE Enterprise Distributed Object Computing Conference*, 2004.

[180] E. Solaiman, C. Molina-Jimenez, and Shrivastava. S. Model Checking Correctness Properties of Electronic Contracts. In *Proceedings of the First International Conference on Service Oriented Computing*, 2003.

[181] R. Stone. *Contract Law*. Cavendish publishing, 2005.

[182] M. Ströbel. *Engineering Electronic Negotiations*. Kluwer Academic Publishers, 2002.

[183] A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini. The Tropos Metamodel and its Use. *Informatica*, 29:401–408, 2005.

[184] M. Tian, A. Gramm, H. Ritter, and J. Schiller. Efficient Selection and Monitoring of QoS-aware Web Services with the WS-QoS Framework. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2004.

[185] V. Tosic. WSOL Version 1.2. Technical Report SCE-04-11, Department of Systems and Computer Engineering, Carleton University, July 2004.

[186] J.J.M. Trienekens, J.J. Bouman, and M. van der Zwan. Specification of Service Level Agreements: Problems, Principles and Practices. *Software Quality Journal*, 12(1):43–57, 2004.

[187] Trusted Computing Group. TCG Specification Architecture Overview Revision 1.2, April 2003.

[188] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1995.

[189] W.-J. van den Heuvel, K. Leune, and M.P. Papazoglou. EFSOC: A Layered Framework for Developing Secure Interactions between Web-Services. *Distributed and Parallel Databases*, 18(2):115–145, 2005.

[190] A. van Lamsweerde, S. Brohez, R. De Landtsheer, and D. Janssens. From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In *Proceedings of Workshop on Requirements for High Assurance Systems*, 2003.

[191] A. van Moorsel. Metrics for the Internet Age: Quality of Experience and Quality of Business. In *Proceedings of the Fifth Performability Workshop*, 2001.

[192] I.T.P. Vanderfeesten, H.A. Reijers, J. Mendling, W.M.P. van der Aalst, and J. Cardoso. On a Quest for Good Process Models: The Cross-Connectivity Metric. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering*, 2008.

[193] M. Virdell. Business Processes and Workflow in the Web Services World. 2006.

[194] U. Wahli, G.G. Ochoa, S. Cocasse, and M. Muetschard. *WebSphere Version 5.1 Application Developer 5.1.1 Web Services Handbook.* IBM, February 2004.

[195] M. Weiss and D. Amyot. Business Process Modeling with URN. *International Journal of E-Business Research*, 1(3):63–90, 2005.

[196] S.A. White. Business Process Modeling Notation (BPMN), Version 1.1, January 2008.

[197] E. Wohlstadter, S. Tai, T. Mikalsen, I. Rouvellou, and P. Devanbu. GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. In *Proceedings of the 26th International Conference on Software Engineering*, 2004.

[198] C. Wolter, H. Plate, and C. Ség. Collaborative Workflow Management for eGovernment. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, 2007.

[199] C. Wolter and A. Schaad. Modeling of Task-Based Authorization in BPMN. In *Proceedings of the Fifth International Conference on Business Process Management*, 2007.

[200] K. Wolter and A. van Moorsel. The Relationship between Quality of Service and Business Metrics: Monitoring, Notification and Optimization. Technical Report HPL-2001-96, Hewlett-Packard Laboratories Palo Alto, Software Technology Laboratory, April 2001.

[201] L. Xu and M.A. Jeusfeld. Pro-active Monitoring of Electronic Contracts. In *Proceedings of the 16th Belgium-Netherlands Artificial Intelligence Conference*, 2004.

[202] W. Yang, H. Ludwig, and A. Dan. Compatibility Analysis of WSLA Service Level Objectives. Technical Report RC22800 (W0305-082), IBM Research Division, 2003.

[203] V. Yarmolenko and R. Sakellariou. Towards Increased Expressiveness in Service Level Agreements. *Concurrency and Computation: Practice and Experience*, 19(14):1975–1990, 2007.

[204] G. Yee and L. Korba. Bilateral E-services Negotiation Under Uncertainty. In *Proceedings of the Symposium on Applications and the Internet*, 2003.

[205] E. Yu. Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, 1977.

[206] T. Yu and K.J. Lin. Service Selection Algorithms for Web Services with End-to-end QoS Constraints. *Journal of Information Systems and e-Business Management*, 3(2):103–126, 2005.

[207] U. Zdun, C. Hentrich, and S. Dustdar. Modeling Process-Driven and Service-Oriented Architectures Using Patterns and Pattern Primitives. *ACM Transactions on the Web*, 1(3), 2007.

[208] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, 2004.

[209] O. Zimmermann, N. Schlimm, G. Waller, and M. Pestel. Analysis and Design Techniques for Service-Oriented Development and Integration. In *Proceedings of GI Jahrestagung (2)*, 2005.

[210] F. Zulkernine, P. Martin, C. Craddock, and K. Wilson. A Policy-Based Middleware for Web Services SLA Negotiation. In *Proceedings of the Seventh IEEE International Conference on Web Services*, 2009.