University *of* Leicester

Management Concerns in Service-Driven Applications

Thesis submitted for the degree of

Doctor of Philosophy

At the University of Leicester

By

Ahmed Musfer Alghamdi

Department of Computer Science

University of Leicester

July 2008

AUTHOR'S DECLARATION

I herby declare that this submission is my own work and that it is the result of my work done mainly during the period of registration. To the best of my knowledge, it contains no material previously published or written by another person nor material which to as substantial extent has been accepted for the award of ant other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Parts of this submission appeared in the following conjoint publications, to each of which I have made substantial contributions:

- Al-Ghamdi, A. and Fiadeiro, J.L.: Architectural Handling of Management Concerns in Service-Driven Business Processes. MSVVEIS 2006: pages 111–120, 2006.
- Al-Ghamdi, A., Fiadeiro, J.L. and Paschke, A.: RBSLA based Implementation for Architectural Management Laws. 4th International IEEE Conference on Innovations in Information Technology (Innovations'07), Dubai, United Arab Emirates: pages 556–560,

ABSTRACT

With the abundance of functionality-similar Web-Services, the offered or agreed-on qualities are becoming decisive factors in attracting private as well as corporate customers to a given service, among all others. Nevertheless, the state-of-art in handling qualities, in this emerging service paradigm, remains largely bound to the aspects of technology and their standards (e.g. time-response, availability, throughputs). However, current approaches still ignore capital domain-based business qualities and management concerns (e.g. customer profiles, business deadlines).

The main objective of this thesis is to leverage the handling of quality and management issues in service-driven business applications toward the intuitive business level supported by a precise and flexible conceptualisation. Thus, instead of addressing qualities using just rigid IT-SLA (service-level agreements) as followed by Web Services technology and standards, we propose to cope with more abstract and domain-dependent and adaptive qualities in an intuitive, yet conceptual, manner. The approach is centred on evolving business rules and policies for management, with a clean separation of functionalities as specific rules. At the conceptual level, we propose specialised architectural connectors called management laws that we also separate from coordination laws for functionality issues. We further propose a smooth and compliant mapping of the conceptualisation toward service technology, using existing rule-based standards.

AKNOWLEDGEMENT

My first and foremost thanks go to ALLAH and then I Offer my sincerest gratitude to my supervisors Prof. Josė Fiadeiro and Dr. Stephan Reiff-Marganiec for their support throughout this thesis with their patience and knowledge whilst allowing me the room to work in my own way. They reviewed the drafts of this thesis carefully and provided lots of important suggestions for improvement; I greatly appreciate their excellent supervision with time and efforts.

Many thanks to the Department of Computer Science staff working in different positions for their support during my Ph.D. programme. The Department has been extremely helpful in financing me to attend conferences which have been an important and exciting element of my research. Thanks to all the personnel of the department for creating an inspiring and comfortable research atmosphere. All my friends and colleagues have made the past few years enjoyable.

Final thanks go to my parents, my brothers Mohammad, Ali, Abdullah, Abdulaziz, Makhzoom, and Saeed, my sisters Salhehh, Fatemah, Hamdah, my wife Hoda, my sons Abdulaziz, Mohammad, Saud (I haven't seen him till now) and My liltle princes daughter Rand for their understanding and encouragement which helped me through many tough times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

Chapter 1

Introduction

1.1. General Motivation

Geared by fierce competition, and increasingly globalizing markets, companies are forced to cooperate, while offering just-in-time solutions as required by their demanding (private / public, individual / corporate) customers. These facts are boosted by advancements in the IT world through the use of internet; thus, changes in technology have been urging organizations to shift from their traditional integrated vision with centralized control toward more loosely-coupled Web-based networked cross-organizational applications.

To respond to this challenging shift in organizational realities at the technological level, the Internet has been promoted from just a glue of unstructured and static information to a sophisticated enabler of online complex behavioural and process-centric services, referred to as Web Services. These are characterized as autonomous Internet-based business-driven applications, exposing well-described interfaces aimed to be described, published, requested and composed by any customer or user [KKL+02]. The universality of Web Services is reflected through the uniform adoption of XML-based standardized languages for describing, publishing, and composing them (e.g., SOAP, WSDL, BPEL, etc.; see next chapter). Service-oriented architecture (SOA), as the new Service Oriented Computing (SOC) paradigm, is based on subscribing, invoking, binding, and composing different Web Services to fulfil the application problem at hand.

As this Web-Services technology is maturing, more and more (private as well as corporate) cross-organizations are embracing it at a rapid pace to automate their businesses as composite services. This has already resulted in an abundance of functionality-similar Web-Services covering most potential business areas (e.g. tourisms, health, banking, etc.). One may just look at the exponential number of online services offering flight tickets or accommodation all over the world.

Nevertheless, with the keen competition among such (cross-)organizations geared by market globalization and volatility, it is becoming essential to go beyond the limited capabilities of functionality-based Web-Services developed using current standards. Among the severe shortcomings, which are under intensive exploration by both academia and industry and to which this thesis aims to contribute, the following are emphasized:

Management concerns on top of functionalities: One main advantage of Web-Services, among traditional "offline" software, is that service requestors can freely, quickly and easily switch from one service to another. Towards attracting and convincing more customers to opt for a given service, from the early days of this emerging service technology, the quality of services [BK05, OYP03] have been put at the centre of focus, as an important selection factor between functionality-similar services. In the state-of-the-art, as will be detailed in Chapter Three, such quality-of-service (QoS) has been expressed in terms of service-level agreements (SLA) from their service providers (SPs). The SLA is a "customer-provider contract" specifying how performance, throughput, time-response, and availability, among others, are to be measured [DHP02]. Despite the great benefits of such technology-driven QoS guarantees in advancing the state-of-the-art, service providers are still facing difficulties when it comes to the understandability, transparency, abstraction and adaptability of such qualities [TA03]. This thesis proposes thus to contribute in leveraging the level of handling management concerns to the business level at a first stage, so that the desired features outlined above can be achieved. Indeed, we argue and demonstrate that quality of service and management concerns in general can be tackled at two different levels of abstraction: (1) the business level, at which management concerns should be understood, elaborated, communicated, and adapted by separating them from any other concerns, including the basic functionalities and interactions concerns; and (2) the technological level, at which qualities are expressed in terms of corresponding IT features such as time-response, availability, throughput, etc.

Adaptability and separation of concerns: Although boosting functionality-similar services with QoS and management concerns represents a crucial improvement factor towards selecting best services, it remains far from satisfactory on its own. Indeed, with high-volatility and harsh competitiveness in the market, high-flexibility and dynamism are required on both services' functionalities and management concerns. Unfortunately, current standards such as WSDL and BPEL are well-known for their rigidity. We argue and demonstrate that handling adaptability at the business level represents an essential requirement towards achieving highly flexible and adaptable services. In addition, separation of concerns such as functionalities and management greatly improves flexibility, understandability and transparency.

Adequate Conceptualization prior to service deployment: It is without doubt that this emerging service computation paradigm is still technology-centric. That is, the focus is more on how to use Web-Services standards to implement a given business application, and not on how to first understand, and reason about service requirements without referring to any standards. Indeed, the exclusive adoption of standards and the focus just on deployment are hindering the development services that are correct-by-construction and highly flexible. Several ongoing research programmes are aiming to overcome this deficiency, and to which we are aiming to contribute in this thesis while handling management concerns at the conceptual level.

This thesis concentrates therefore on the challenges of exploring qualities and management concerns at the business and conceptual levels where, despite the high beneficial potentials of ongoing investigations, little has been attempted so far. We refrain from presenting yet another framework for monitoring, reporting, and controlling performance quality of Web services, as often presented in the literature [CCD+03, ZBN+04, DDK+04]. Instead, we are concerned with business-level elicitation, conceptual modelling, and intrinsic adaptation of management concerns, without disadvantaging, mixing, or fixing other functionality and interaction concerns. Moreover, to demonstrate the practicability of the approach, we are mapping it to the technological level to benefit from both worlds: business and technology.

The main principles and advantages of the approach we are putting forward, include:

1. The tackling of management concerns at the business activity level, thereby taming the complexity of business processes, while promoting flexibility;
2. The adoption of an event-driven rule-centric approach at the business level, so that all (cross-)organizational stakeholders can be involved in eliciting, describing, and evolving management qualities;
3. The modelling and validation of both management and interaction concerns through business rules encoded as architectural connectors that we refer to as management and coordination laws;
4. Last but not least, to present the practicability of the approach using service technology, we map this business-conceptual approach to Web technology using related recent advances (RBSLA).

The remaining sections of this chapter are as follows: Firstly, we motivate the general research scope of this thesis within the service paradigm, and then motivate the challenges we have been focussing on. Secondly, we go into detail about the main research question tackled in this thesis and its ramifications. In the third part, we enumerate the main original contributions of this thesis. Finally, this chapter is wrapped up by highlighting the content of the remaining chapters.

1.2. Research scope, objectives, and questions

After a period of literature research about all facets of Web services and their enabling service-oriented architecture, we came up with the following observations as the driving forces toward the formulation of the main research question. The first observation stipulates that qualities are important within this "online" service paradigm. Indeed, qualities represent important factors for attracting customers to providers and vice versa, within the immense internet world. Building on that first observation, we then moved focus on the state of the art about existing approaches addressing qualities in Web services. The analysis of this study could be summarized as follow:

1. Due to the compositional (and thus partnership) nature of Web Services, the qualities are defined between different partners (e.g., customer and providers). Consequently, we discovered that Service Level Agreements are the dominating framework, as quality-driven contracts between requestors and the service provider(s).
2. Moving from one approach to another, we found that the addressed quality criteria and features vary. Nevertheless, except for some strong deviations, such as mixing/merging qualities with security issues, the tackled qualities range over availability, time-response, reputation, and throughput.
3. As we already pointed out, almost all existing approaches address the above qualities from an implementation technological perspective. More specifically, concrete (XML-driven) programming techniques and languages are being proposed for quantifying, measuring, monitoring, controlling, and reporting on selected qualities with respect to concrete running Web services.
4. Since we also were attracted by topical software engineering issues such as adaptability, separation of concerns, and behaviour-driveness, we noticed that none of such good principles ever has been mentioned in the explored literature.

From these observations on the state of the art in handling QoS in Web services, we decided to investigate QoS with respect to the whole life cycle of service-driven business applications. Since implementations using Web services were shown to be unsatisfactory on their own, we proposed to focus first on qualities at early phases of business requirements and precise conceptualization. Consequently, instead of speaking directly about Web service quality-driven solutions, we were attracted by qualities and management concerns at all development stages, with emphasis on business and conceptual levels.

1.2.1. Research objectives

The global objective of the investigated research consists of leveraging the handling of management concerns in service-driven business applications to the business level and the conceptual level without losing the link to service technology as the deployment phase. Thus, we aim to complement the explored SLA-driven approaches to qualities at the infrastructural level with application-level qualities. Indeed, as we just pointed out, the conceptualisation of SLAs at the business level has received little attention beyond the use of rigid notations, such as the Unified Modeling Language (UML) through activity diagrams [SLE04]. Yet, addressing SLAs and management concerns in general at the business and conceptual levels, in complement to the IT level, can provide significant benefits, including the following, which are the main objectives of this thesis:

1. The involvement of all stakeholders in formulating, describing, and adapting qualities, without referring to any specific concrete implementation;
2. The ability to put aside any other concerns (e.g., functionalities and interactions, security) and focus just on the qualities (perhaps with devoted teams when required). This essential ability towards separation of concerns remains difficult if not impossible to realize at the implementation level;
3. The capacity to decide on the level of granularity in handling such management concerns. Indeed, whereas most existing approaches consider the business processes as a whole, as we will present in this thesis, the business level facilitates a flexible fine-grained handling at the (business) activity level;
4. Instead of coding qualities as rigid (boolean programmable) constraints, we are proposing to handle any qualities-driven concerns as highly flexible specific event-driven (management-centric) business rules. The same is applied for any other functionalities: we first explicitly and separately capture using appropriate (functionality-centric) business rules pattern.

5. The ability to validate any formulated qualities, including the detection of unwanted or inconsistent integrations / compositions.
6. The complete freedom to choose and adapt the best available service technology once a satisfactory and reliable conceptualisation is achieved; when necessary, several implementation alternatives can be combined, a fact that is difficult to realize at the coding level on its own.

1.2.2. Research question(s)

The main research question we have been exploring can be formulated as follows:

"How can the handling of management concerns in service-driven applications be leveraged smoothly, from its dominating Web technology trend toward more intuitive as well as precise business levels at which separation of concerns, business rule centricity, and adaptability become the driving forces?"

In order to address this question effectively, we are required to further decompose/detail it into the followings sub-questions.

1. How can we cope best with management concerns as specific (cross-) organizational knowledge, in such ways that it remains aligned with the business goals of the targeted service-driven applications? In other words, which business mechanisms are most suitable to express management knowledge while promoting adaptability and separation of concerns, among other desired properties?
2. At which granularity level should we address such management issues in service-driven applications, so that separation of concerns and adaptability could be promoted further? In more detail, is the holistic business process coarse granularity suitable or is the more fine-grained activity level better?
3. How could the interaction and manangement concerns be handled in coherent, yet explicit and separate ways?

4. Which conceptual setting allows for a smooth moving from the business-level description while promoting the main features such as separation of concerns, flexibility, and adaptability?
5. Once a business-conceptual handling of management concerns (besides functionality concerns) is obtained, how can we exploit Web technology and Web service standards to result in a compliant implementation?
6. How can a stepwise (business, conceptual, and implementation levels) approach to management concerns be supported through clear methodology and validated using case studies?

1.3. Thesis Contributions and Methodology

Considering the above detailed research sub-questions as a roadmap for the whole investigation, the first contribution concerned tackling the first four questions, which should bring answers to the following: (1) The appropriate business artefacts in which adaptability and separation of concerns are targeted; (2) the granularity level for tackling management concerns, and the conceptual primitives that preserve the business-level features and enhance them with rigor and reasoning qualities. In the following, we motivate and report on the forwarded solutions to these research concerns as driving forces for the rest of our investigations.

Event-driven Business rules centricity: To tackle adaptability, evolution, and separation of concerns, in complex agile and knowledge-intensive service applications, we found that business rules [KL04, WL04, KL05] represent the most suitable business components. Indeed, business rules, which express policies and constraints on how to do business, are ubiquitous in any cross-organizations. They are understandable and independent from any specific process usage, and thus evolving and promoting separation of concerns. They further represent main assets to keep any cross-organization complying steadily with national and international markets and government laws. On the other side, and more importantly, they represent the main strategic competitive force within any cross-organization, since they require innovative and attractive, yet simple, business ideas in terms of laws, policies, and rules. Besides, and among other potentials, they fit well with the event-driven publish / subscribe SOA architecture as they reflect ECA behaviour: "on the occurrence of events do actions under some constraints (ECA rules)." Along all business-conceptual and implementation levels, we therefore adapt an ECA-based development approach, with specific emphasis on management concerns.

Activities as working granularity-level: By analysing different sources of rigidity and mixing of concerns while tackling quality service-driven applications, we discovered that, besides the lack of dealing with (ECA-driven) business rules, most existing approaches follow a holistic business process vision. To overcome this limitation, we are proposing to adopt a fine-grained approach, with business activities as the building-block for handling management concerns. As will be detailed in the thesis, we present that in contrast to the process-based approaches. With an activity-centric approach we achieve more flexibility and transparency not only at the activity level but also at the business process as a whole.

Architectural techniques with ECA-driven connectors: In order to close the gap between the business level and the more precise conceptual level while enhancing ECA rule at the activity level, as above, we built our approach on top of the architectural techniques presented in [Fia02, AF01b, AF02]. In this sense, we proposed new forms of architectural connectors that are governed by ECA rules and focus on qualities at the activity level that we refer to as management laws.

As answers to those specific research questions, we summarize the achieved contributions:

Characterisation of management features at the business level: Whereas at the implementation or Web technology level the term quality is distinguished (e.g. response time, availability, throughput, reputation, etc.), we were confronted with the definitions of qualities at the business level. In this sense, we proposed a set of characteristics for management concerns at the business level that include: deadlines to meet, partners (e.g., customer, provider), and preferences.

Description of such management characteristics in terms of appropriate ECA-driven rules at the activity level: To express management concerns, besides coordination ones, at the business level, we put forward a generic pattern of ECA-driven rules by recapitulating the explored business characteristics for qualities in service-driven applications.

Proposition of management laws with mapping steps from the above management-based ECA rules: To keep the conceptual level aligned with the above business level, we thus put forward new primitives reflected in terms of particular architectural connectors that we refer to as management laws. That is, besides coordination laws that focus on coordination concerns, we derive management laws from management-based ECA rules.

Integration of management and coordination laws around business activities: With an aim to describe and conceptualize any service-driven business process from any business activities, we propose methods to bring together in a flexible and suitable way both concerns around any activity.

Proceeding from management laws to RBSLA-based deployment: To demonstrate the relevance and practicability of this business-conceptual approach, we also implement steps to derive corresponding rule-centric Web technology deployment. Moreover, we implemented a tool to support this translation and execute management laws using current implementation of RBSLA, a variant of the (XML-based) RuleML language (used extensively in semantic Web).

Undertaking of case study as validation of the approach: We considered a case study variant of an E-shopping application dealing with PC selling. All management-based characteristics, ECA rules, and management laws are motivated and explained through this case study. Besides that, in the fourth chapter, all activities comprising this case study are detailed from both management and coordination perspectives.

Implementation of tools supports the laws – RBSLA mapping: Since, RBSLA is associated with advanced language and software environment; we will focus on how to automate the proposed translation steps from management laws to RBSLA. An implementation procedure consisting of; an editing, translating and executing management laws using the RBSLA language is proposed.

Finally, we propose a stepwise methodology that we detailed in the fourth chapter. It includes the following steps:

1. Given the global goals and intuitive business processes of the service-driven application at hand, we first propose to understand all involved activities better.
2. We propose to describe the business rules coping with functionalities and coordination for each considered business activity.
3. We describe the management concerns in terms of ECA business rules at the level of each activity.
4. From the coordination-based ECA-driven business rules and for each activity, we derive the conceptual level in terms of corresponding coordination laws (i.e., functionality-based ECA-driven architectural connectors).
5. From the forwarded management-based ECA-driven business rules and for each activity, we derive the conceptual level in terms of corresponding management laws (i.e., management-based ECA-driven architectural connectors).
6. We integrate both coordination and management concerns.
7. We construct the complete business process by putting these two activities in partial order (i.e., high flexibility in constructing any business process from the modelled activities).
8. We translate this business conceptual approach toward RBSLA. This translation is followed by the execution of modelled service-driven application on current tools supporting RBSLA.

1.4. Organization of the Thesis

The next chapter will present an overview of the general ingredients involved in this thesis as well as provide concepts and elements required for building the proposed approach. In detail, first Web services and the service-oriented architecture are summarised. Secondly, Business rules then are presented in general and with respect to their specific benefits to Web services by surveying existing proposals. Finally, architectural techniques are introduced in general and coordination techniques in particular.

The third chapter delves into the main topic of the thesis that concerns the quality management in service-driven applications. In this respect, first we survey related work and existing approaches to the handling of management issues in Web services. Particular emphasis is put on service-level agreements and their deployment, using service technology. We survey and classify existing proposals around established criteria to compare their potentials and weaknesses, and clarify the main advantages of the new approach and its complementation of existing implementation-oriented efforts.

The fourth chapter presents the contribution of this thesis toward handling management (and interaction) concerns in service-driven applications in flexible and suitable ways. The chapter presents the potential of addressing management concerns at the business level and its fine-grained activity level, and explains the use of architectural techniques to enhance flexibility and evolution. The chapter then provides details of the different conceptual primitives put forward, their motivations, syntax and intended semantics, and coherent adoption as management laws. This conceptual framework to management concerns is applied to a case study dealing with PC selling application.

The fifth chapter bridges the gap between the proposed conceptual framework of management concerns, namely management laws, and their concrete and beneficial deployment, using current Web technology. It examines different XML rule-based languages and their adequacy in capturing management laws with respect to preserving important properties, such as flexibility, interaction centricity, and separation of concerns. RBSLA was selected as a suitable Web language for the deployment of management laws. The following sections describe the translation of management laws into RBSLA and then assess the results through a case study. Finally, the chapter describes the general deployment architecture based on Web services.

The sixth and last chapter recapitulates the goals, achievements, and contributions of the thesis, and outlines future conceptual and practical extensions of this work.

CHAPTER TWO

Chapter 2

Background and Preliminary Concepts

As we emphasized in the previous introductory chapter, the main purpose of this chapter concerns the leveraging of the handling of management and functionalities concerns in service-oriented business applications towards more business and conceptual levels, with adaptability and separation of concerns at the fine-grained activity level at the core. We further suggested that for such leveraging event-driven business rules and architectural techniques will be playing important roles. Besides such conceptualization, we also pointed out that this thesis will look at the practicability side, namely the benefits from service technology and available standards.

With the aim of providing all required ingredients for a self-contained thesis, we are first summarizing all the required knowledge about (Web-) service technology and its main standards. Secondly, since business rules will be at the core of the envisioned conceptual model for management concerns, we overview the main definitions and elements related to business rules in general as well as most of the emerging attempts to benefit from business rules to bring more flexibility and behaviour to current Web-Services. Thirdly, as we are working from an architectural conceptualization, we present the main concepts about architectural techniques in general, and the so-called coordination laws developed in our group in particular.

2.1. Web-Services: Concepts and Standards

Although it is still a premature to find a completely agreed-on working definition for Web Services, there are at least some common characteristics of this emerging new IT technology. Firstly, Web services (WS) are network-addressable software units (e.g., components, modules, programs); that is, they are developed to be used on the Internet. Secondly, and in contrast to other Internet-based applications (e.g., web sites), WS are accessible using well-defined and explicit interfaces.

Thirdly, such interfaces should be exposed, invoked, and composed easily. Fourthly, there is a rich package of interoperable technologies and standards (e.g., SOAP, UDDI, BPEL, WSDL), well adapted to exploit the potentials of the Web.

An interesting first definition suggested in [CGS01] regards Web services as application-oriented services using specific Web standards while serving application-to-application business processes. A more elaborated definition was forwarded by Sun as one of the significant providers of this technology: any Web Service corresponds "services offered through the web, where typically any business application sends a request to a service at a given URL using the SOAP protocol over HTTP. The service receives the request, processes it, and returns a response" [Wus02]. An often-cited example of a Web Service is that of a stock quote service, in which the request asks for the current price of a specified stock, and the response gives the stock price. Another interesting definition coming from IBM as one of the first companies building Web Services, says: "Web Services provide an application integration technology that can be successfully used over the Internet" [Web08].

It is important to reemphasise that Web Services have been triggered and boosted by the growing challenges in organisational information systems. Indeed, with market globalisation and fast advances and confluence of computation and communication (over the Internet), organisations are coming under huge pressure to interconnect their know-how in a decentralised and process-aware manner. So, with the limitations of platform-dependent middlewares such CORBA or DCOM [Dou03], Web Services have emerged as platform-independent and composition-driven to boost the aimed decentralisation and loose-coupled cooperation / integration in a universal setting [OYP03, TAA+01, CGS01].

Before surveying the main standards underlying Web Services technology, we should recall that service-orientation as new paradigm for software development over the Internet is governed by the so-called "triangular" service-oriented architecture. As depicted in Figure 2-1, SOA is based on three main principles: Publish-Find-Bind. That is, following the SOA architecture, any software used has to be published (not necessarily over the Internet), where subscribers can invoke it and finally bind it to others to build complex composite services.

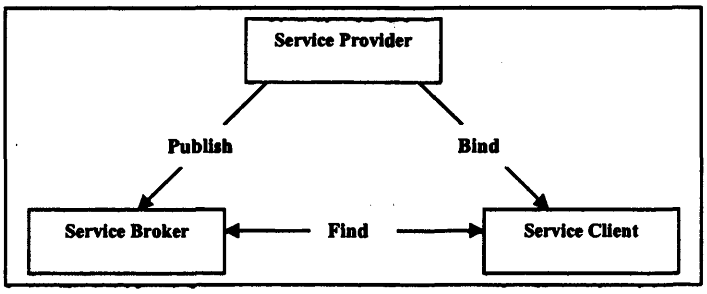


Figure 2-1. The SOA architecture illustration

2.2. Web Services Standards

The characteristics common to Web standards are: (1) adopt XML as universal language; (2) linked with the Web; (3) promote the exchange of messages in an event-driven manner; (4) support heterogeneous protocols for communicating with each other without being dependent upon the implementation of the underlying system. These standards are being developed by many leading IT organisations, including IBM, Microsoft, ARIBA, and many others, and are being submitted to the World Wide Web Consortium (W3C). The most important standards for Web Services can be described as follows:

- **SOAP:** Simple Object Access Protocol. It is a common messaging protocol that is used over HTTP and other Internet protocols for communication between applications [TSP+04]. SOAP is simple, extensible, and platform-independent; that is, designed for sending XML-based messages over the Internet.
- **WSDL**: Web Services Description Language. XML-based language for describing the programmatic interfaces of Web Services and how to access them [Web01]. WSDL is an XML document that is used to locate Web Services.
- **UDDI**: Universal Description, Discovery, and Integration. A business registry standard for indexing Web Services so their WSDL

descriptions can be located by development tools and applications [Oas07]. UDDI communicates through SOAP and acts as a directory for storing information about Web Services.

- **BPEL**: Business-process Execution Language. BPEL builds on all the above basic standards and aims at composing services into complex and thus realistic services. Indeed, it is very hard to find a given basic service that satisfies any meaningful customer request so that the composition belongs to the essence of service orientation [WCL+05]. BPEL is like workflows and business processes; it permits partially ordered different business activities to build service-oriented business processes (e.g., using sequence, choice, parallel, switch, and other operators) [MN05].

W3C says of the Web Service interface WSDL: "A WSDL file contains descriptions of one or more interfaces and binding information for one or more services. A service is actually a collection of ports. A port is the combination of a portType, which describes the interface of the port, and a binding, which describes the mechanics of invoking the port" [Web01].

The descriptions of services discovered by Web Services clients should be published to a service registry, as Figure 2-2 illustrates. This service registry usually is defined by the UDDI project, UDDI.org. The interaction between the Web Services and requesting application, which is an important feature of Web Services, can be established once a service has been discovered, and a binding established based on information in the registry [UDD07].

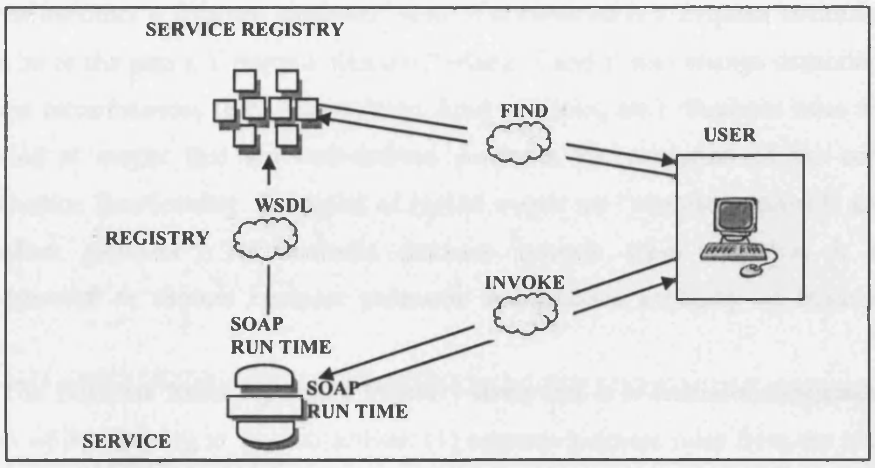


Figure 2-2. Web services as efficient and practical instantiation of SOA

The invocation of a service involves sending and receiving an XML message to the service. These XML interactions are governed by an open standard, SOAP, which W3C describes in this way: "SOAP defines a message header that describes the message and indicates which operation in the interface of the service is being invoked. The header is an envelope that contains an XML message body in which the parameters are passed. SOAP supports both a remote procedure call and a general XML document passing paradigm. SOAP messages must be carried on a communications layer, which most often is the Hyper Text Transport Protocol (HTTP)" [Soa01].

2.3. Business Rules and their role for adaptable Web Services

The Business Rules Group defines a business rule as "a statement that defines or constrains some aspect of a business." It is intended to assert a business structure or to control the behaviour of the business [BRG00] and reflect the way of doing business in general.

A significant characteristic of business rules is that they tend to change whenever the business policies they embody change, which is more often than the core application functionality does [Ars01]. In an e-shopping application, typical business rules include: "If a customer has purchased more than X books, then he or she becomes a frequent customer" and "If a customer is a frequent customer, then he or she gets a Y percent discount," where X and Y may change depending on the circumstances (e.g., season, time, kind of books, etc.). Business rules are applied at events that are well-defined points in the execution of the core application functionality. Examples of typical events are "request a product" and "confirm payment". As business domains become more complex, it is fundamental to capture business processes and policies explicitly as business rules.

The Business Rules Approach [Von01] states that it is crucial to implement them while adhering to four objectives: (1) separate business rules from the core application, (2) trace business rules to business policies and decisions, (3) externalise business rules for a business audience, and (4) position business rules for change. Nevertheless, despite all efforts, and due to diversity of business rules, this explicit separation still is hard to achieve, for instance for object-oriented mechanisms.

Business Rules have been classified in the literature [Wag02, BRG00, TW01] into four different main types: integrity rules, derivation rules, reaction rules, and deontic assignments. Integrity rules specify an assertion that must be satisfied in all stages of the system. They built a core part of databases and information systems in general. A derivation rule is a statement of knowledge derived from other knowledge by a mathematical calculation or an inference. Reaction rules, or event condition action (ECA) rules, are the most frequent ones and include the others. For this, general ECA rules are the most adopted and investigated form of business rules. They specify the invocation of action in response to an event, and under different constraints and/or circumstances. The actions are performed only when the specified constraints apply [RD05a, RD05b, RND05].

Business rules, as key factor for regulating intra-organisational activities as well as fitting and relating to the external environments (e.g., market, institutions), thus are important elements in any (cross-) organisational information system. Further, business rules are present in any knowledge-intensive applications, such as expert and intelligent systems. Illustrations of potential rule-intensive domains include e-commerce, financial industry, television and radio broadcasting, health and hospital management, and rental businesses. Business rules capture different knowledge, ranging over policies, preferences, decisions, advice, and recommendations [Dat00].

2.3.1. Business Rules and Web-Services

As Web Services applications are going beyond simple applications and investing domains, such as e-commerce, e-health and e-government, among others, it becomes obvious that pure process-centricity through BPEL no longer is sufficient. Indeed, WSDL and BPEL standards for describing and composing services are static and manual, whereas the above potential applications for Web Services are knowledge-intensive and very volatile.

To leverage Web Services standards toward these challenging rule-intensive and volatile service-oriented business applications, several attempts are being made to bring business rules to such standards. In the following sketch are some of the most referenced ones in the recent Web Services literature.

The first proposal was forwarded by Papazoglou et al. [OYP03,Pap03]. In this approach, starting from a very general specification, the composition is scheduled, constructed, and finally executed with the assistance of business rules classified judiciously in a repository. Besides such basic elements as events, conditions, and messages, this classification includes rules dealing with activity flows, the data required for their composition, and the constraints to be respected. The direct construction and subsequent execution of the composition from the business rules is performed in terms of XML-like descriptions, without any prior precise modelling.

Another approach proposed recently in [CM04] consists of explicitly separating business rules from the flow of business processes (i.e., splitting BPEL into business flow and business rules). This allows business rules to evolve/be specified independent of the (reduced) BPEL descriptions. This vision fits well with the ideas in this study and can be exploited for further concretisation or implementation of this approach. Other approaches that are close to those in this work include [QDS04, ZLB04].

Another innovative proposal is the one forwarded by S. Dustdar et al. in [RND06]. In this work, business rules are considered and externally exposed as Web Services described using extensions to reactive RuleML [Rul05], instead of the passive and static WSDL. In this sense, rules can be discovered and composed like any services while being (internally) processed using logic-based engines such as Prolog or Jess. The rules thus are considered as independent agreements to be invoked over the Web as services. The approach is automated with a supporting tool called ViDRE [RND06]. Nonetheless, the approach does not tackle the conceptualisation level nor does it cope with the dynamic composition of the (WS) rules to specific business process activities.

2.4. Architecture techniques and coordination

Software architecture is a high-level software design dealing with the structure and organisation of large software systems [MPX04]. It describes the components of the system and how those components interact at a high level [BHH00]. After identifying system components, each component is assigned responsibilities that client components interact with through "contracted" interfaces. The interactions between components are called connectors. Component interconnections define control mechanisms and support all interactions between various components needed to accomplish system behaviour. The configuration of components and connectors offers a structural and a behavioural view of the system [SN00].

The software architecture discipline aims to reduce complexity through abstraction and separation of concerns [SG96]. A good decomposition of a complex system satisfies the theory of loose coupling between reasonably independent components that can be undertaken separately. In software engineering, the concept of software architecture is getting more attention, with research work concentrating on architectural styles, better known as patterns, architecture description languages, and formal methods, among others.

To represent software designs at the architecture level, architecture description languages (ADLs) were developed by either academic or industrial groups. There are six types of ADL element forms that are a sufficient vocabulary to express any software architecture [SG96]. They are listed as components, connectors, ports, roles, representation, and binding. Service-oriented architecture (SOA) is being promoted in the industry as the next evolutionary step in software architecture to help organisations meet their complex challenges [CHT03]. SOA existed before Web services with respect to supporting the software system.

Web services can be used to implement a service-oriented architecture [ACK+04, WCL+05]. However, Web services do not necessarily translate to SOA, and not all SOA is based on Web services. The relationship between the two technology directions is important and they are mutually influential: Web services momentum will bring SOA to mainstream users, and the best-practice architecture of SOA will help make Web services initiatives successful [GR05]. The ability to access share services efficiently is a critical step toward the full deployment of the new on-line economic, political, and social era. Adopting SOA requires the development of techniques (such as services description, discovery, querying, composition, monitoring, security, and privacy) to address various challenging issues [TAA+01].

The advantages of SOA as part of an enterprise can be summarised as follows [Dou03]:

1. Permits the IT group to be more responsive to the organisation needs. Implementing a system solution based upon service orientation helps organisations plan ahead for change rather than respond reactively.
2. The ability to use more packaged software helped reduce the development and maintenance costs. SOA leverages existing IT investments so that the overall organisational goals are met and reduces the cost to manage and maintain them.
3. The adoption of Web services as an integration technique helped to minimise the costs of integration systems. The service orientation

approach enables stakeholders to create dynamic collaborative applications that meet the organisational goals.

4. The possibility for smaller organisations to share in Electronic Data Interchange (EDI) independent of communication and software technologies.

2.4.1. Coordination

Software architecture modelling techniques have been proposed to support interaction-centric approaches that promote interconnections to architectural connectors by separating the code that, in traditional approaches, is included in the components for handling the way they interact with the rest of the system.

The architectural approach forwarded in [AF01] uses and extends software architecture techniques in what has become known as the three Cs approach (Computation, Coordination, and Configuration). These layers can be describe as follows: semantic primitives that address the "business architecture," i.e. the means that need to be provided for modelling business entities (Computation); the business rules that specify how the entities can interact (Coordination); and the business contexts through which specific rules can be superposed, at run-time, to specific entities (Configuration). The CCC can be classified as a coordination-based approach [GC92] that gains essential ideas from software architecture in order to separate (externalise) interactions from computations, and superimposition known from parallel program design [Kat93] to support compositional evolution.

Semantic modelling primitives have been put forward in [AF02, AFG+02] that rely on architectural connectors to separate the coordination of interactions between business entities from computations that entities perform to ensure required services. More precisely, so-called coordination laws and contracts externalise as first-class entities any intra- or cross-organisational interactions between business components. This clean separation permits changes to business rules to be performed at the level that is required without affecting other aspects.

The way coordination aspects can be captured as architectural connectors has been reported in several publications, for example [AFG+02, AF01a, Fia02,AF01b, GKW+02,AF02]. From a conceptual modelling point of view, they are captured in semantic primitives called Coordination Laws. Coordination laws comprise the connector concept while coordination interfaces depict the roles of connector types that must be instantiated with components when a law is to be triggered on them [AG97].

The coordination rules of a law, as shown in Figure 2-3, identify under *when* a trigger (such as an event published by one of the partners); under *do*, they specify the reaction that is executed if the trigger is accepted. The reaction consists of a set of operations that are executed atomically as a transaction. The operations either are local to the law or made available by the partners.

Coordination laws can be instantiated by binding the coordination interfaces to concrete run-time components. The instantiation creates a connector – co-ordination contract – in the architectural sense that executes the coordination rules of the law by invoking the services of the components.

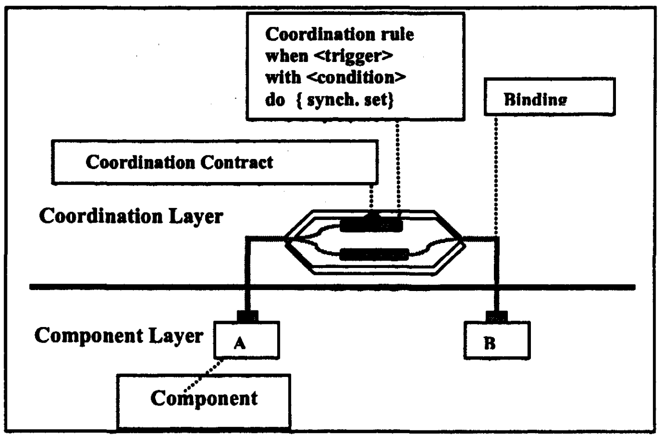


Figure 2-3. The CCC layers

2.5. Chapter Summary

In this chapter we presented a background to the Web service area and how the principles of software engineering can assist Web service engineers to build Web service applications that promote correctness, knowledge-intensivity, and adaptability. The survey of existing presentations of business rules and their application benefits in Web Services shows the need for more explorations, particularly at the conceptual side.

Finally, we gave an introduction to software architectural modeling techniques with a focus on coordination techniques, since they will play an important role in the approach we will develop over the next chapters.

In Chapter 3, we describe existing solutions to handle management issues in Web Services.

CHAPTER THREE

Chapter 3

Management in Web Services—State-of- art and Classification

As we have suggested in the introduction, this thesis aims at leveraging the handling of management (as well as functionality) concerns in service-driven applications towards early business-conceptual-levels, in such a manner that adaptability and separation of concerns are supported.

Unfortunately, due to the scarcity of research explorations at these early and essential service requirement levels, in this chapter we report on existing approaches handling quality-of-service, with respect to Web-Services deployment, mainly geared by Service-Level-Agreements (SLA) description. This chapter thus explores in-depth the intensive state-of-the-art literature research on Web technology-centric quality-of-service.

With respect to the targeted objectives of this thesis, among the envisioned benefits we aim to achieve by undertaking this extensive exploration, we should emphasize at least the following:

- We wanted at first to be close to the topic of quality-of-service in Web-Services, in as much of an exhaustive manner as possible. Indeed, most of the exposed approaches are dispatched in different publications and thus to the best of our knowledge, there are no exhaustive surveys on QoS in Web-Services. Furthermore, we claim that without understanding what is "going on" at the deployment-level, one can never appreciate the added-value when tackling management concerns at the business and conceptual levels.
- Although we were convinced of the benefits of leveraging management concerns at the domain and conceptual levels, at the beginning we were puzzled at the core thesis's research question on how to achieve such leveraging. Towards that core question, we have first concentrated on how QoS are tackled at the deployment-level, while keeping in mind how

to benefit from this low-level work and how to abstract it as much as possible.

- As mentioned in the introduction, this thesis aims also at exploring any smooth translation of any proposed conceptualization for management concerns towards Web-Service technology. In other words, by surveying existing proposals for QoS in Web-Services, our objectives further include the quest for the existing proposal to experience such envisioned translation.
- As will be detailed in Chapter Five, we have opted for the RBSLA language [Pas07] as a suitable implementation target for the presented conceptual model of this thesis. That is to say, without this focused survey on proposals for QoS in Web-Services, at least we could never be able to address such translation that is for sure not the only possible one

The remaining sections of this chapter are organised as follows. In the next section, we present a general overview of service-level agreements in Web Services, their underlying principles, and management. In the second section, we classify and enumerate different criteria being (partially) adopted to define and manage qualities in Web Services. In the third section, we go through different existing proposals for handling qualities in Web Services. We close this chapter by emphasising the role of business-level qualities, that is, qualities related to the business requirements that thus are independent of any specific Web Services standards, languages, or deployment techniques. Unfortunately these kinds of qualities have been disadvantaged if not neglected, and this thesis aims at contributing to fill this serious gap

3.1. Service Level Agreements and Management concepts

First, it is worth mentioning that SLA has been introduced as concept to cope with qualities in information systems. Indeed, with the development of client / server enterprise information systems, organisation stakeholders have found themselves increasingly dependent on these automated systems. At the same time, because of the dependence of business on the external environment, the users wanted certain guarantees on the quality of offered functionalities, which later included non-functional qualities such as availability, reliability, and response time. As these expectations grew in complexity and scope, an explicit agreement on such qualities was necessary: this is the so-called Service-Level Agreement or SLA for short.

According to Judith Myerson, SLA is defined as "... a formal contract between a service provider and a client guaranteeing quantifiable performance at defined levels" [JM02]. Thus, failing to meet agreed SLAs standards possibly will have serious financial impact on a provider. SLAs between a service provider and its customers will guarantee customers that they can get the service they sought and will force the service provider to deliver its service promises. Therefore, service providers need to have a deep understanding of what they promise to deliver and what they actually are capable of delivering. In a highly competitive business environment, SLAs offer a way to differentiate among similar service providers [LVA02]. SLAs thus are capital while handling qualities in Web Services [CSD+03, ZBN+04].

A more detailed definition of SLA was provided by Li-jie Jin et al. [LVA02] in terms of different characteristics that have to be covered by any two parties. Besides the involved parties and their roles (e.g., customer, provider), these SLA characteristics include:

- Purpose – describes the reasons behind the creation of the SLA.
- Validity period – defines the period of time that the SLA will cover. This is delimited by start time and end time of the term.
- Scope – defines the services covered in the agreement.
- Restrictions – defines the necessary steps to be taken in order for the requested service levels to be provided.
- Service-level objectives – the levels of service that both the users and the service providers agree on, and usually include a set of service level indicators, like availability, performance, and reliability. Each aspect of the service level, such as availability, will have a target level to achieve.

- Penalties – spells out what happens in case the service provider under-performs and is unable to meet the objectives in the SLA. If the agreement is with an external service provider, the option of terminating the contract in event of unacceptable service levels should be built in.
- Optional services – provides for any services that normally are not required by the user, but might be required as an exception.
- Exclusions – specifies what is not covered in the SLA.
- Administration – describes the processes created in the SLA to meet and measure its objectives and defines organisational responsibility for overseeing each of those processes.

"SLA providers need thus to design their SLAs only after understanding their capabilities. On the other hand, if there is too much leeway in the specification of SLAs, a Web Service may not be able to fully capitalise on its capabilities. Thus, it is important to design SLAs that are able to balance between risk and benefit of all parties. This balance should be based on a good understanding of the impact of various service levels on business processes in both the service provider and the customer" [LVA02].

Since there is no fully agreed-on final definition for SLA, we continue to comment on most significant forwarded definitions. In this respect, Sun Microsystems Inc. [Wus02] defines the Service Level Agreement (SLA) as a type of contract that sets the expectations between the consumer and service provider. Another definition that seems more flexible is provided by Pratt [Pra03]; it defines a service level agreement "as a statement of various service level options from which one will be selected by the customer or client specifying timing, frequency, cost, etc., to match the business need."

An SLA defines therefore the relationship between the two parties, and can be considered as the cornerstone of how the service provider sets and maintains commitments to the service consumer. The purpose of the SLA is to make clear the provider's promises and how these promises will be delivered. It is also intended that it sets out who measures the delivery, in what way and what should happen if there is a shortfall in the provision.

In graphical terms, we can regards an SLA as a contract relating different service partners (e.g., customer, provider, third party). Figure 3-1 below illustrates this concept of Service Level Agreements as a contract.

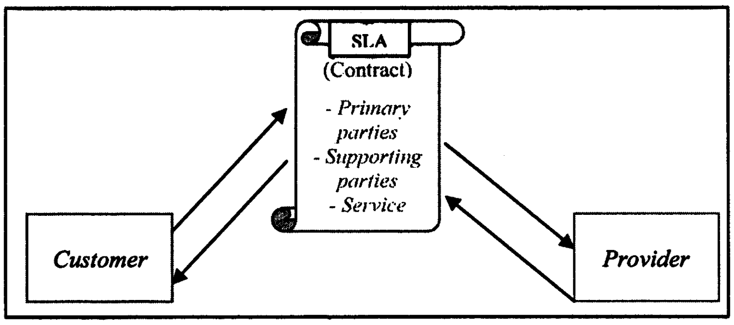


Figure 3 -1. SLA as a contract between customer and provider

In order for the SLA to be complete, there must be a description of the agreed service to be provided to the customer along with a breakdown of the agreed planned workload for the forthcoming period and any information concerning its uptake (e.g., in relation to the spread of work or seasonal variation; quality to which the service is provided, including monitoring and audit arrangements). Other standards or agreed protocols for use of the service, such as the availability being on condition of consultant's signature or second referral also are standard inclusions. Furthermore, a rundown of the costs of the service provides a comparative index of resource utilisation [DFG98].

The concept of SLA has been used in many areas of Web Services including: e-commerce, offline businesses, computer services, construction management [Hil93], library services [Ash94], and health or hospital services [TGR+04]. To conclude this introduction to SLA, we emphasise again that QoS in Web Services is driven by two demands: (1) Clients that seek a good service performance, e.g., low waiting time, high reliability, and availability to successfully use services whenever the need arises [TGR+04]; and (2) providers that seek to stay competitive by proposing the best qualities to attract such clients and customers.

3.1.1. Different categories of SLAs

In the literature, we can distinguish at least three (complementary) classes of IT-SLAs [Pio02]. These are performance, reactive, and proactive SLAs. In the following, we summarise their main characteristics.

Performance Service Level Agreements are those that address continuing services. These SLAs set the quality of these services based on objective measurement and a baseline of values that establish acceptable service levels. One issue that must be dealt with in any performance SLA is control. Typical Performance Service Level Agreements for Web services contain uptime and performance of services, connectivity, and satisfied delivery. Performance SLAs and metrics help define and measure the performance of Web Service systems to ensure they meet performance requirements.

Reactive Service level agreements are based on the provider's reaction to proceedings, and the main measurement is time. Typical reactive SLAs for Web services address response time, resolution of failures recoveries, and response to security threats. The fundamental issue for all reactive SLAs is the categorisation of events, and which provider is accountable for the resolution of problems. Events can be categorised according to priority and severity. Severity is a possession of the problem, while priority is a possession of the solution. The event severity also can be categorised into critical (performance unacceptable or service not available), urgent (service is working normally but a redundant component or supporting feature has failed), and routine (service is available and performance sufficient but there could be other problems).

Proactive SLAs describe services that are intended to prevent problems before they occur. Typical Proactive SLAs for Web services include constant system monitoring, backup audits, installation of patches and upgrades, DNS changes, performance analysis, and tuning and capacity analysis and planning.

3.1.2. Service Level Management

Simply defined, Service Level Management is the process of managing (composed) services so that they can fulfill SLA requirements. Resources that should be managed include personnel, infrastructure, applications, and budget [Git03]. In order to be able to cope with the ever-increasing technological and infrastructural advances in service-oriented computing, as well as very demanding customers, service-level management has evolved from a limited to a broad service portfolio.

A more motivated SLM definition is given in [MS04], where the authors stated: "The web has become a major vehicle for transforming business processes, but ineffective management of web-based services can result in high costs and user dissatisfaction. Service Level Management is therefore a competitive weapon in the web marketplace, providing the tools needed to improve performance and reliability of Web Services while simultaneously controlling costs.". Indeed, today's services are becoming sophisticated, and a successful process of SLM should pull different information from multiple resources and services, including inventory, fault management, performance management, and customer care, such are the diverse activities required of it [W301, Kar04].

The International Engineering Consortium [IEC04] defines SLM in direct relationship with SLA. It is regarded as the set of people and systems that allow the organisation to ensure that SLAs are being met and that the necessary resources are being provided efficiently. That is, SLAs represent the main ingredients for SLM, in addition to the human and environmental factors.

According to Microsoft, "Service Level Management aims to align and manage IT services through a process of definition, agreement, operation measurement, and review. The scope of Service Level Management includes defining the IT services for the organisation and establishing service level agreements for them. Fulfilling SLAs is assured by using underpinning contracts (UCs) and operating level agreements (OLAs) for internal or external delivery of the services. Introducing Service Level Management into a business will not give an immediate improvement in the levels of service delivered. It is a long-term commitment. Initially, the service is likely to change very little; but over time, it will improve as targets are met and then exceeded" [YTS+08].

However, this tight SLA–SLM relationship so far has not been explored sufficiently. In fact, whereas the process of IT–SLA is defined better through models and frameworks, there still is a lack of well-established approaches regarding its management. Basically, it also can be said that the failure to provide SLA can be traced to poor SLM.

3.2. Quality Criteria in SLA: Studies and Classification

After presenting the essentials about SLA and SLM concepts, we judged it important to go into more detail about different quality criteria when defining SLA. In this manner, when we address existing approaches to SLA in Web Services, we can have clear ideas which criteria are (not) supported by which approach. Moreover, the detailed description of such existing criteria for qualities in Web Services will support our investigation of more abstract business-level criteria, the main focus of this thesis.

Indeed, after intensive exploration of the state of the art on Web Services and their qualities, we came to the conclusion that there exist two main categories of approaches in handling management in service-oriented computing: IT SLAs, and business SLAs. Business SLAs deal with business management at a high application level, and they aim to cope with the description of service-oriented management at the business level. In contrast, IT SLAs deal only with system-related qualities of services. Because business SLAs practically still are absent in existing approaches, this section will focus on the criteria for defining IT SLAs. First, we distinguish two main categories of IT–SLA criteria: (1) Those defined as the qualities of services to be presented in a given SLA; and (2) those inherent to a particular execution of given Web Services as a business process.

3.2.1 Qualities criteria related to IT-SLAs

The qualities criteria we could recognise from different service level agreements, approaches, and related literature on Web Services could be summarised in the following:

- **Availability:** This defines whether the Web service is present or ready for immediate use. It is represented by a probability value that reflects the probability of the service being available at a particular point of time. For instance, a service could be unavailable due to a failure on its provider system [MN02].
- **Accessibility:** The quality aspect of a service that represents the degree to which it is capable of responding to a service request. There could be situations when a Web service is available but not accessible. Accessibility is related strongly with system scalability. For example, a system is said to be scalable if it is capable of providing access to large number of users [MN02].
- **Accuracy:** This defines the error rate produced by the service [Ran03].
- **Payment Rate:** Rate at which the service/transactions are charged [SDM02].
- **Throughput:** This metric represents the actual number of user requests that are handled by the system [Ran03]. The response time of a Web service is related to its throughput.
- **Integrity:** Integrity is the quality aspect associated with how the Web service maintains the correctness of the interaction in respect to the source [MN02].
- **Response Time:** This is the most important QoS metric from a user's perspective. Response time measures the time interval between sending a request to execute a service and the time that the response has been received by the user [SDM02].
- **Latency:** The time taken between the services request arriving, and the request being serviced. The throughput of a system is also related to its latency [Ran03].
- **Performance**: This measure the quality aspect associated with a Web service. It is measured in terms of throughput, latency, and possibly other metrics like accuracy. Higher throughput and lower

latency values represent good performance of a Web service [MN02].

- **Reliability:** This also measures the quality aspect of a Web service, and represents the degree of being capable of maintaining the service and service quality. The number of failures represents a measure of reliability of a Web service. Reliability is defined as the probability that a request is responded to correctly within the maximum expected time frame [MN02].
- **Regulatory compliance:** A measure of conformance with some pre-defined (and agreed on) rules, law, standards, or established SLA [MN02, Ran03].
- **Security:** Security is the quality aspect of the Web service of providing confidentiality and non-repudiation by authenticating the parties involved, encrypting messages, and providing access control [MN02].

3.2.2 Qualities criteria related to the execution of Web-Services

Another classification of SLAs can be based on the properties of the condition that is assessed over process execution data to find whether the SLA has been violated. Particularly, SLAs can be classified according to the nature of the SLAs section as follows [CCD+03]:

- **Duration**: This defines the time interval between two steps of the process workflow. It may impose constraints on the process execution such that the time between the "receive order" and then subsequent "confirm shipment" does not exceed a certain threshold. Human interaction can be a determining factor for the length of a distributed business-to-business transaction. In some cases, the workflow might have to wait for human input, i.e., approval by the administrator. This might take far less than a minute or extend to several days, depending on the speed of the user. The long duration could lead to undesirable extensive locking of resources.

- **Data**: This clause is related to a condition on the service composition variables. These variables are used to store intermediate values that relate to the state or history of the process. For example, it may necessitate that the minimum order quantity is more than N items. The number of items is assumed to be able to be determined from service composition variables.
- **Path**: This SLA clause defines the execution path of a specific process. For instance, some SLA may impose decoupling of the shipment path in the flow corresponding to arranging delivery for premium customers different from that for standard customers.
- **Count**: The clause defines the frequency of activating a specific resource. For example, an SLA can be related to a condition stating that an order should be confirmed two times before shipment. From an implementation perspective, this corresponds to stating that the step confirm should be executed at least twice.
- **Resource**: This clause is related to a condition requiring that a given step of the flow is to be executed by a resource with specific properties. For example, an SLA could require that strategy team members first review projects submitted by certain employees.

3.3. State of the art on SLAs for Web-Service

For the handling of IT-SLAs, we also have distinguished two main categories: (1) Approaches putting forward new infrastructure for specifying QoS issues associated with Web Services; and (2) approaches extending existing standards such as UDDI to cope with qualities through brokers. WSLA, WSOL, and SLAng are of the first type; UX and UDDIe belong to the second.

3.3.1 Web Service Level Agreement (WSLA)

A WSLA document defines assertions of a service provider to perform a service according to agreed guarantees for IT-level and business process-level service parameters, such as response time and throughput. It further specifies the measures to be taken in case of deviation and failure to meet the asserted service guarantees, for example, a notification of the service customer. The assertions of the service provider are based on a detailed definition of the service parameters, including how basic metrics are to be measured in systems and how they are aggregated into composite metrics. In addition, a WSLA expresses which party monitors the service, third parties that contribute to the measurement of metrics, supervision of guarantees, or even the management of deviations of service guarantees. Interactions among the parties supervising the WSLA also are defined [Lud02]. The WSLA language is based on XML; it is defined as an XML schema.

WSLA can be used both by service provider and service customer to configure their respective systems to provide and supervise their service. This configuration step includes the creation and parameterisation of relevant services implementing the system, as well as the WSLA for supervising such services. Parts of the WSLA (or derived information) could be passed on to third parties that support the WSLA's supervision. After the configuration step, the WSLA can be enacted by supervising services.

An important aspect of WSLA is its capability to deal with specifics of particular domains and technologies. The language is extensible to include specific types of operation descriptions, measurement directive types for specific systems, special functions to compose aggregate metrics, and predicates to evaluate specific metrics. The extension mechanism makes use of the ability to create derived types using XML schema. By design, the core of the WSLA language is very compact. To be of immediate use, the WSLA language encompasses a set of standard extensions to define complete agreements that relate to Web Services and includes guarantees for response time, throughput and other common metrics.

Web Service Level Agreement:

- Parties
  - signatory partes
  - supporting parties
- Service description
  - covered service (WSDL)
  - monitored SLA parameters
  - functions
- Obligations
  - Validity Period
  - SLOs
  - action guarantees

Figure 3-2. Web Service Level Agreement Entities [TGR+04].

As shown in Figure 3-2, a WSLA description is given as an XML schema, and is divided into three main sections [Lud02]:

- **Parties section**: A signatory and supporting part, on which all signing party sponsors and supporting ones should appear.
- **Service description**: It contains all information on the service's characteristics and the parameters to be observed. Resource Metrics are derived directly from the managed resources and are to be specified in the Measurement Directive part. In the SLA Parameters part, the retrieved metrics are related to a specific customer who supplies the value and to whom it will be reported.
- **Obligations section**: It describes guarantees and constraints imposed on the SLA parameters in the form of SLOs (offers), i.e., high/low watermarks for associated SLA parameters that are promised to be met for a certain period of time. In the case of SLA violations, appropriate compensating activities are defined in Action Guarantees.

3.3.2 The Web Service Offering Language (WSOL)

WSOL allows the formal and unambiguous specification of prices, monetary penalties, management responsibilities, and third parties, especially accounting parties. The main targets of the WSOL project are creation of service offerings, definition of QoS constraints, management statements, reusability, and a mechanism called service offering dynamic relationship (SODR) that allows for a switching between services [TPP+03].

WSOL Service Offering:

- important/ include of external specification
- subscription
- price
- penalty
- management responsibility
- constrains
    - domain (service-/port-/operation-name)
    - condition (as Boolean expression)
    - metrics (defined in external ontology)
    - rules for metrics aggregation
    - management entity for measurement
- statement
    - domain (service-/port-/operation-name)
    - <any definition>
- constraint group/ instantiated CGT
    - <any item listed above>

Figure 3-3: WSOL Service Specification [TPP+03].

WSOL has a very low overhead as it defines classes of service instead of individually managed SLAs. Another important benefit of WSOL is that it supports the reusability of specifications, through the concept of constraint groups and constraint group templates. Constraint groups and constraint group templates include formerly defined elements and the import of elements defined in other WSOL files. Classes of service are a mechanism for the description and differentiation of a Web Service and QoS associated with that Web Service. A service offering also can be seen as a contract or SLA, and consists of several self-explained components as listed in Figure 3-3 [TPP+03].

3.3.3 Service Level Agreement language SLAng

SLAng's main aims are support for interorganisational service provisioning, including storage, network, middleware, and applications, as well as the specification of non-functional parameters at service level in order to enable QoS description and negotiation [LSE03].

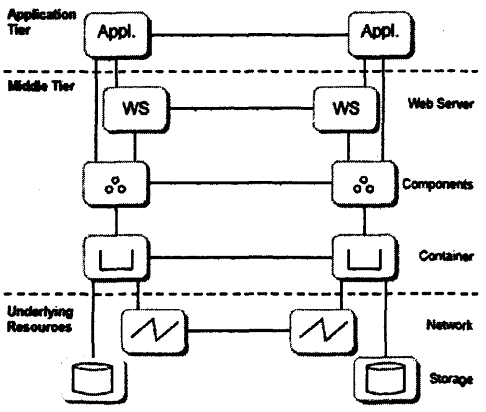


Figure 3-4: Service Provision model [LSE03].

Figure 3-4 reflects the service provisioning model of SLAng [LSE03]. ``The three-tier architecture consists of the application tier, the middle tier, and underlying resources. Applications of the first tier consume the underlying components or Web Services abstracted by the middle tier. The containers located in the middle tier support QoS negotiation, establishment, and monitoring, while the components abstract the resources in the underlying resource tier. The network and storage facilities are grouped to the underlying resources `` [TGR+04].

Nevertheless, SLAng allows only for static SLAs, in the sense that it does not support dynamic lookup of new services and update of non-functional service properties at runtime.

3.3.4 RBSLA: Focused Overview with Illustration

Although it is not specifically aimed to address qualities in Web-Services, the Rule-based Service Level Agreement (RBSLA) project[1] [Pas07] is general and

[1] http://ibis.in.tum.de/projects/rbsla/

open enough to be easily adopted by Web technology as we will demonstrate in chapter five. RBSLA is particularly devoted to the development of adequate knowledge representation concepts for the formalization and serialization of Service Level Agreements and IT service policies. It develops a rule-based Knowledge Representation (KR) framework [PDK05,Pas07] to describe contracts in a formal way, execute them in standard generic rule engines such as Prova[2], and manage and interchange them in an XML-based mark-up language, the RBSLA language [Pas07,Pas05,PKB07].

The RBSLA language is implemented as an extension to RuleML [Rul06] (see also our overview in chapter five) in order to address interoperability with other rule languages and tool support. It adds additional modelling power and expressiveness to RuleML to implement higher-level policies and SLAs declaratively (in contrast to the above SLA proposals). Among others, RBSLA main features and characteristics include the followings [Pas07]:

- Global ECA-style reaction rules and event messaging reaction rules for active monitoring and complex event processing, and workflow-like communication patterns.
- A Web-based module concept, which allows bundling rule sets to modules with a unique module id, meta data labels (e.g., authoring information) and qualifications (e.g. priorities between modules), dynamic imports of modules from Web URIs, and scopes (constructive views over selected parts of the knowledge base).
- Test-driven extensions for verification, validation, and integrity testing leading to self-validating rule bases.
- Order-sorted polymorphic type systems compatible with XML-based (XSD), relational (SQL data types), Semantic Web ontology languages RDF/RDFS and OWL, and object-oriented class hierarchies (e.g. Java).
- Seamless integration of dynamic object-oriented API invocations via expressive procedural attachments (e.g. Java method invocations or Web service calls).

[2] http://www.prova.ws/

- External data access by, e.g. SQL, XQuery, OWL2Prova RDF triple queries, SPARQL.
- Deontic norms for normative reasoning.
- Defeasible rules and rule priorities between rules and modules (rule sets).
- Rich libraries and built-ins for, e.g. math, date, time, string, interval, list functions.

3.3.5 A UDDI eXtension (UX)

UX is an architecture providing QoS-aware and cross-organisational support for UDDI. The first objective of UX is to rate services by reputation in order to permit service requestors to discover services with high-quality. The second objective is to share the ratings among UX servers – the proposed extension of UDDI – in different domains [CLS+03].

Reputation is measured through QoS feedback made by service customers. The proposed UX server uses the clients' reports containing response time, terminating state, and cost to generate summaries in order to predict the services' future performance. A lookup interface in UX allows the discovery and distribution of the QoS summaries among different UX servers [TGR+04]. UX applies a protocol called Cooperating Server Graph (CSG) [BPT98] that dynamically updates the links between cooperating servers over a WAN according to different events happening, either to some servers, or to the underlying WAN. Figure 3-5 depicts the basic architecture of UX.

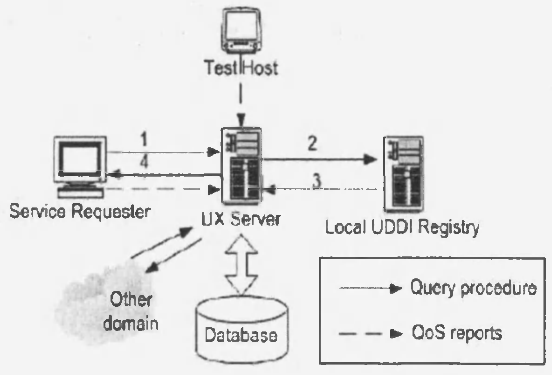


Figure 3-5. UX architecture [CLS+03].

UX extends UDDI with the ability to predict services' future performance based on reputation, which is measured through QoS feedback made by service customers. The users' experience is shared in a local and inter-domain way. Since reputation is influenced mainly by user perception and, furthermore, can be manipulated easily, a research group from Monash University, Australia, extended the approach to reputation with a QoS attribute termed "verity." Verity is used as an indicator of trustworthiness for the quality driven selection and composition of services [KKL03].

3.3.6. UDDIe

It extends UDDI's functionalities within UDDI. Service providers can associate their services with QoS properties such as bandwidth, and memory requirements, which are encoded in the service interface. They can make their services available by means of leasing. UDDIe presented a concept allowing the definition of three leasing types: finite, infinite, and future lease. Moreover, UDDIe supports qualifier-based search by introducing qualifiers such as EQUAL-to, LESS-than, and GREATER-than [TGR+04].

UDDIe is implemented in the context of the G-QoSM framework for grid service discovery. The client applications send their requests to the QoS broker of the G-QoSM system. The broker is not part of UDDIe. It processes the requests and forwards them to the UDDIe registry. After receiving a list of services that implement the particular service type, the broker does the final selection of the most appropriate service for the client by applying a weighted average concept. Figure 3-6 shows a code fragment of a client request with QoS requirements [ARW+03].

```
<?xml     version="1.0"     encoding="UTF-8"?>
<wsdl :definitions
xmlns :wsdl=http://schemas.xmlsoap.org/wsdl/">
...
<targetNamespace="http:// MyService-Interface">
<wsdl :messagename="printNameResponse">
<wsdl :message>
...
<QoS>
<service_cost> 5 </service_cost>
<network_bandwidth> 256K </network_bandwidth>
<memory> 48MB </memory>
...
</QoS>
</wsdl :definitions>
```

Figure 3-6. Client request with QoS requirements [SRA+03].

3.4. Business SLAs for management in service-driven applications

In contrast to IT-SLAs, business SLAs refer to agreements on how a specific service is delivered, and to the semantics of the service rather than to system or application level metrics [CCD+03]. For example, a business SLA could state that orders of 200 PCs should be received within 45 days of the order, or within 20 days from the payment. This clearly has nothing to do with machine-dependent qualities. Business SLAs and IT SLAs are linked to each other. For instance, a malfunctioning of the system or part of the system will result in performance bottlenecks in terms of response time in addition to the total service delivery time, thus business SLAs.

In comparison to ordinary business SLAs (as in information systems), Web service business SLAs are more complex and have specific characteristics that must be conceptualised precisely. These characteristics include:

- the role of customers as decisive partners;
- the involvement, in most cases, of more than one service, as Web services are composition-driven applications;
- the possibility of (dynamically) including new services or removing participating services from a given agreement, because Web services are loosely coupled inter-enterprise applications.

As we presented before, most existing approaches to qualities are restricted to the handling of SLAs at the infrastructural level provided by chosen IT platforms.

The conceptualisation of SLAs at the business level has received little attention beyond the use of notations, such as the Unified Modeling Language (UML), through activity diagrams [SLE04]. Yet, addressing SLAs at the business domain level in addition to the IT level can provide significant benefits, such as:

- formulation of quality at the business domain level, independent of the choice of IT platform;
- participation of a wider community of partners in the organisation, or across different organisations, concerned with quality (e.g., users, managers, designers, marketers), and not just service designers and programmers; and
- The ability to change or enhance management rules without interfering with the IT level, and vice versa.

In the next chapter, we present how to cope with business SLAs at the conceptual level while keeping and enhancing all these properties. The mutual complementarity with IT-SLAs will be tackled in the fifth chapter.

Chapter Summary

A survey of existing approaches to the handling of qualities in Web-services was the focus of this chapter. This survey allowed us to distinguish between two kinds of management issues: IT SLAs and business SLAs. Since Business SLAs have been scarce in literature, we restricted ourselves to the state of the art around IT-SLAs. Before that, we presented as exhaustive as possible a set of criteria for qualities. We also distinguished two main classes of criteria: business SLAs and IT SLAs.

CHAPTER FOUR

Chapter 4

Architectural Modelling of Management Concerns in Service-driven Applications

To be effective and meet organisational goals, service-driven applications require clear specification of the management concerns that establish business level agreements among the parties involved in given business processes. In this chapter, we show how such concerns can be modelled explicitly and separately from other concerns through a set of new semantic primitives that we call management laws. These primitives support a methodological approach that consists of extracting management concerns from business rules and representing them explicitly as connectors in the conceptual architecture of the application.

Service-driven business processes are among the topical innovations through which organisations hope to cope with ever-changing business requirements. This new trend represents how typical product-oriented business models are characterised, among other things, by much more flexible, loosely-coupled interactions that sustain high levels of agility and adaptability to change. From a technological point of view, the response to the challenges raised by such models relies on the pervasiveness of Internet technologies, namely on Web services. These are platform-independent components with explicit interfaces tailored to the Web that can be used in combination with others to form large-scale, evolvable business applications. Because the ultimate objective of the service-driven economy is to match customer requests with optimal services, either elementary or composed from simpler ones, it becomes clear why the notion of quality of service plays a central role in this new paradigm [ACD+04]. It is not surprising that the notion of Service Level Agreement has been placed at the heart of such Web technologies [CSD+03, BCT+03]. Notions of quality addressed by these efforts include availability, accessibility, accuracy, throughput, and reputation, among others.

The remaining sections of this chapter are organized as follows: The next section brings more motivation and methodological support to the proposed conceptual framework for handling management concerns in service-driven business applications. In particular, the approach strengths are emphasized in connection with the widely explored infrastructural service-level agreements (IT-SLA). Moreover, the progressive methodological steps supporting the approach are highlighted. The second section presents the case study that runs through this thesis, which deals with a realistic variant of Online PC Selling by focusing on its different business activities. The third section details how business rules govern any business activity. The fourth section presents an overview of how basic functionality/interaction concerns are conceptualized by means of interaction-focused business rules supported by the coordination techniques that have been explored in [Fia02, AF01b, AF02]. In the fifth section, the main elements underlying the new concept of business service-level agreements (BSLA) in terms of high-level business-driven qualities first are proposed and then illustrated through activities of the running case study. Then, the method of eliciting BSLA as management-oriented ECA-driven business rules is presented. Finally, these are conceptualized as transient architectural connectors; that is, management laws. In the sixth section, it is shown how to bring together both concerns at the activity level, using the case study. The seventh section illustrates how flexible and multi-concern business processes can be derived from the above architectural concepts of different activities. Finally, section eight emphasizes the scalability and practicability of the approach, by carrying out the whole PC selling case study using the stepwise architectural approach. This chapter concludes by outlining the achieved work.

4.1. Business Handling of Management: Approach Milestones and Steps

The ultimate objective of service orientation is to present the best service to its customers and providers, because the quality of services belongs at the heart of this paradigm. The situation is more acute when it is known that at present there are plenty of functionality-similar Web-Services (e.g., a plethora of similar published WSDL interfaces) over the Internet (e.g., UDDI registry); one may think just about the numbers of airline services.

As reported in chapter two, following the dominating WS technological trend, the handling of qualities while composing (Web) services has been focusing on the infrastructural level. Several proposals have been coined for codifying, enforcing, measuring, and monitoring low-level qualities such as time-response, availability, throughput, etc. In particular, these implementation-centric qualities have been proposed as technical agreements or contracts between service partners in so-called IT-SLA. SLA-tailored language, which is XML-based language, also has been proposed [LSE03, GR04, Pas07], as reported in the related work of chapter three.

Addressing such low-level qualities and inherent SLA agreements while composing Web services has been shown to be very important to increasing efficiency and competitiveness while attracting technology-proficient partners (e.g., customers, providers). Specifically, at least the following key issues may be highlighted, which are by essence of a conceptual nature and thus could not be (by any innovative technological means) tackled correctly at the technological level on its own:

- **High-level potential qualities and agreements**: When stakeholders (e.g., customers and providers) from different organizations decide to establish a service-driven cross-organizational alliance, they negotiate first on high-level business goals. These then are expressed through conceptual primitives, criteria, and rules [KL05]. Beside basic functionality issues, the formulation of business-level qualities and inherent agreements also must be tackled. In this respect, alliances such as service composition are preconditioned by the reputation, trust, and reliability of the partners (e.g., private/public customers and providers). How such high qualities are to be (semi-) automated (or not at all) should not be an issue here (e.g., a friend simply may recommend a provider, or a more complex mining process could suggest best customers). Agreements on how deadlines (not time-response!) should be respected while performing a specific activity also aim at improving qualities.

- **Intrinsic flexibility and adaptability**: Speaking directly about low-level qualities using a particular implementation language leaves no room to address any form of adaptability and flexibility, except through artificial codification using complex and inflexible clues. In contrast, when high qualities such as those emphasized earlier are described conceptually, all the intrinsic potentials of reformulating, composing, prioritizing, refining, adapting, and evolving them without risk or (high) cost are kept. Also, afterward, the most suitable operational mechanizations can be decided (e.g., algorithms, techniques), based on Web technology deployment technologies.
- **Explicit separation of concerns**: In most proposals focusing on quality of services in WS [CD01, Hil93], basic functionalities and other advanced ones (e.g., security, social concerns) are disadvantaged, if not ignored. This is because separation of concerns is a conceptual feature, and advanced technologies such as componentization, aspect orientation, etc., would be needed to tackle it. It will be attempted to address any concern in a clean, separate, and intuitive manner at the business level. This thesis achieves that aim for functionalities (i.e., interaction) and management concerns. Business rules and transient architectural connectors represent our business and concepts for this purpose.
- **Activity- versus process-based perception**: Due to the fact that most technology-based composition standards (e.g., BPEL, WSCI, and WS-CDL) focus on the process level, they lack scalability and flexibility [CDK+02, RD05a]. To tackle the composition complexity and at the same time offer adaptability, the activities are considered as candidates for describing the composition of business

processes. Besides absorbing that complexity and rigidity, by addressing functionality and management concerns at that fine-grained activity level, the exploitation of resources (e.g., entities, services) will be optimized further. Indeed, by focusing on activities, respective resources are requested dynamically while performing any specific activity (within a business process). Once they proceed to the next activity, previous resources will be released systematically and only the required new ones will be requested.

- **Reliability and certification**: Although this thesis does not explicitly investigate the rigorous operational formalization of the concern-based architectural concept, it is worth emphasizing that such validation is straightforward. Indeed, different underpinnings currently are available for architectural modelling, for example, category theory and graph transformation-based ones [FM97,HC07].

4.1.1. A Stepwise Approach for Conceptualizing Management Concerns

The approach being put forward respects the required software engineering concepts while eliciting and modelling (and validating) complex service-driven applications. Although the main focus of this thesis is on management concerns (as business-level service agreements), the arguments as explained at section 4.1 present how important it is to consider functionality concerns; otherwise, the management problems will be solved but other, more severe, problems about the basic features of the system will be created. In other words, it is argued that any serious attempt at tackling management concerns at the business level should be accompanied by a stepwise methodology for covering the development of flexible and reliable multi-concern service-driven applications.

To explain and to reflect the above advanced concepts in modelling multi-concern service-driven business applications, a stepwise methodology is proposed, depicted graphically in Figure 4-1.

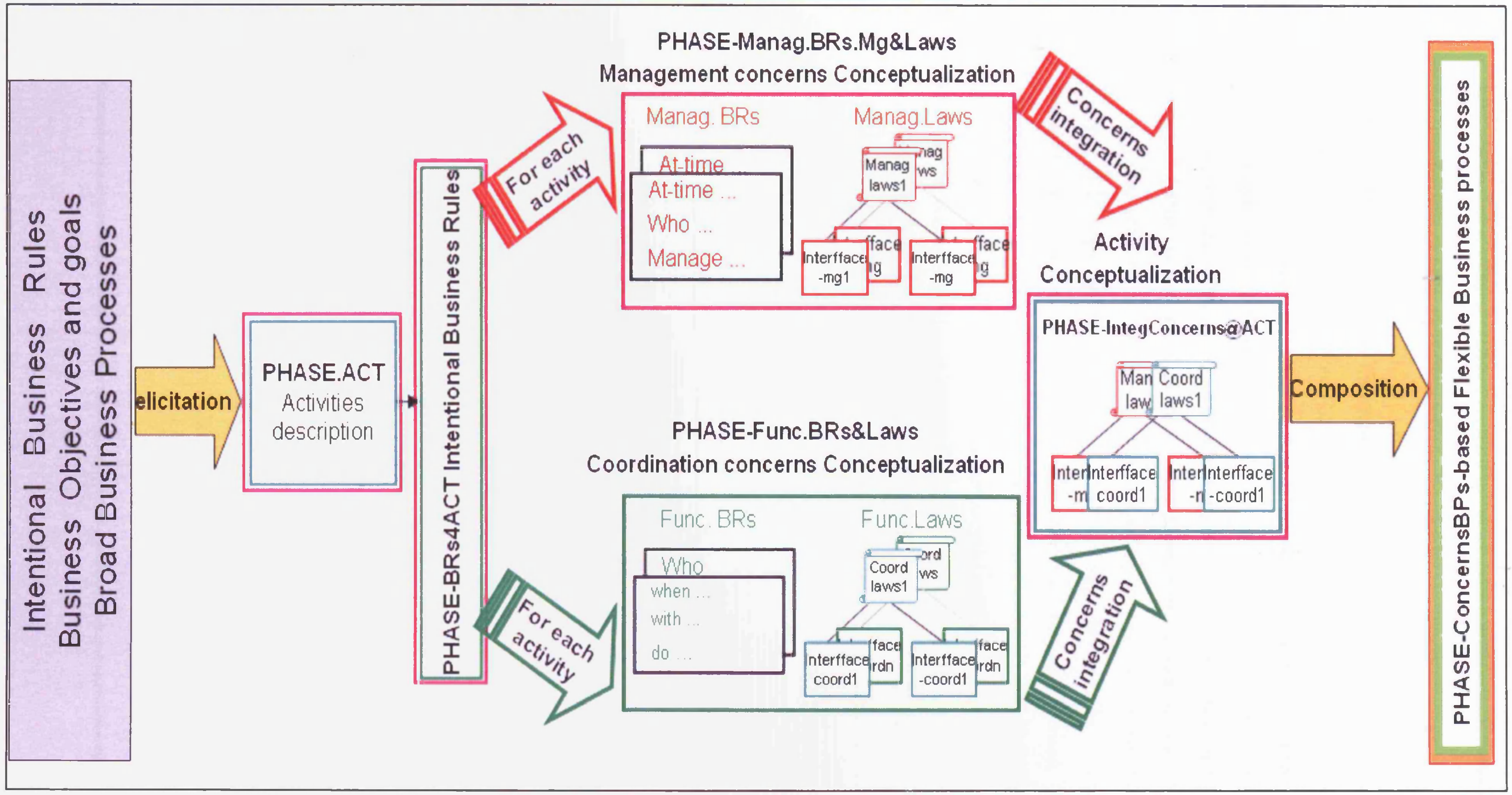


Figure 4-1. A stepwise Business-Conceptual Approach for Interaction-Management centred Service-driven business applications.

Phase-ACT—Activities description: At this early requirements stage, determining service orientation (i.e., planning for a deployment using service technology and its standards), any service-driven business application is perceived as interconnected business entities (mostly to be regarded subsequently as (Web) services). On the other hand, by pursuing an activity-based approach, it is proposed further at this initial phase to build more informal descriptions of each activity that may be a candidate for participating in any envisioned business process. It is important to emphasize that at this stage any specific or detailed composition of such activities into service-oriented business processes is not considered. The following are mentioned as immediate benefits. First, optimal use of resources is obtained; Secondly, the complexity is understood better, since business processes are difficult to handle as whole units. Thirdly, flexibility and evolution are promoted, since different activities' partial orderings (as tailored business processes) may be decided depending on the customers/providers involved, their profiles, and their respective qualities.

Phase-BRs4ACT—Intentional Business Rules for activities: Business rules should be formulated at this early stage to reflect in a flexible and broad manner what is to be done and with what means (e.g., resources, services) in any activity. These rules should be intuitive, intentional, and focused on different concerns, as they are supposed to be elicited and described by different people and are intended to reflect general broad business goals. Natural language thus is sufficient to report on these generic and broad business rules with respect to the identified activities.

Phase-Func.BRs&Laws—Activity-based functionality ECA rules and Coordination laws: This is the main phase of the approach. Indeed, at this stage, it is necessary to focus on functionality/interaction concerns. Nevertheless, instead of fixing it, we propose to adapt such functionalities to the requesters and the potentially provided services. For that purpose, a two-step process is proposed. Firstly, the intentional general-purpose business rules will be refined by focusing on those dealing with functionalities. This is achieved by fitting these functionality-specific (ECA-based) business rules to the corresponding identified activities/tasks (within the respective business processes). Secondly, these activity- and event-driven functional specific, operational business rules are levered to a conceptual architectural level. Since cross-organizational interactions and cooperation are the driving forces in the service paradigm, it is proposed to tailor ECA-driven architectural connectors for this architectural approach. More precisely, the approach will capitalize on coordination technology and its coordination laws (as architectural connectors), proposed and pushed forward (theoretically and experimentally) in recent years [Fia02, AF01b, AF02].

Phase-Manag.ECAs&Laws—Activity-based Identification of Management-specific business rules and Conceptual management laws : In the same spirit as proposed to capture functionality-specific (ECA-driven) business rules at the activity level, this phase proposes to concentrate on management-specific business rules. In contrast to the minimal functionalities defining an activity, management concerns aim to give each activity high-level qualities, so that discovery and composition of the involved services become more competent and more realistic. As will be explained in detail later, at this business level, the main characteristics are to formulate business rules and to focus on management concerns. Firstly, it is necessary to express conceptual time-based constraints, which are relevant and essential in any (service-level) agreement, such as deadlines and triggering events timing (e.g., invocation/response timing). Once again, it should be pointed out that deployment-based, time-based constraints such as time response and throughput are not handled at this level. Secondly, the level of partnership between the involved partners (e.g., customers, providers) is considered important. Nevertheless, we will not discuss how to quantify such relationships. More precisely, such partnerships are expressed in terms of the history of previous collaborations. On the basis of such business-level management characteristics, a pattern of management-specific business rules will be proposed in section 4.5.

To achieve more efficient handling of such extracted management-specific business rules, it is proposed to move them toward a customized architectural-driven approach. That is, we propose customized transient architectural connectors that fit the proposed management-specific business rules. Such management-driven architectural connectors are referred to as management laws.

Phase-IntegConcerns@ACT—Architectural Integration of functionality and Management Laws: At this stage, each business activity already should have been described and validated, using business rules and corresponding coordination/management laws. The purpose of this phase is to bring together both concerns, so that the complete meaning of each activity is captured. A systematic technique is proposed for integrating the different components comprising the coordination and management laws, while defining the expected meaning of any activity. The result of this integration is a multi-concern architectural connector that deals with both concerns.

Phase-ConcernsBPs—Flexible and Multi-concern business processes Modelling: The proposed final step is the modelling of business process out of the conceptualized business activities. That is, in contrast to most existing (Web) service approaches/technologies, which start right away with the process modelling, it is argued and explained in this thesis that aiming to explore important requirements (e.g., flexibility/evolution, separation of concerns, taming of the complexity) can be achieved only through postponing the process modelling.

4.2. Phase-ACT: Illustration with a PC Selling Case study

This first phase starts by identifying different (concrete and conceptual) business entities. Since a service-oriented technology is targeted, most business entities later will be automated as (individual or composite) services. Nevertheless, at this business stage, the term Web services is avoided, since in the next chapter the right deployment strategy using Web-service technology is tackled. Secondly, in order to follow an activity-based methodology, all business activities reflecting the service-driven business application at hand will be identified.

Due to the fact that concepts such as business entities and activities are well known, no further explanation is required, but it can be found in textbooks about object-orientation, business processes, and workflows. Instead, this phase is illustrated through the case study that will be carried through in this thesis, namely a case of a service-driven PC Selling business application. This section is divided into two sub-sections. Firstly, a general presentation of the case study is given by emphasizing the different business entities/services involved. Secondly, the different business activities involved in this application will be documented.

4.2.1. PC Selling Case Study: General Description

As depicted in Figure 4-2, this application is composed of several interacting business entities (all to be deployed as services):

- *Customer services* (CS): This provides PC Selling with a front end that handles interactions with the customers (i.e., end-users). That is, CS allows customers to post their requests, buy PCs, pay their dues, and cancel/accept offers.
- *Provider services* (PS): This represents the milestone business entity in this application. It provides the customer services with tailored offers satisfying their demands. It controls the delivering of PCs to customers and also plays a key role in payment procedures (e.g., discounts, refunds, penalties, etc.).
- *Shipment services* (SS): This service intervenes when there is a need for shipping the goods to the customer (i.e., when there is no provider branch in the customer's area or city). In such a case, the provider enters into interaction with this service to select the appropriate shipment method, depending on different criteria.
- *Banking services* (BSs): As shown in figure 4-2, this service also is crucial as it interacts with all other services to accomplish any required payment. For instance, the bank has to interact with the shipment service for paying the shipment cost from the provider. Besides the payment-related tasks, this service also is needed as insurance for the customer to proceed with the buying process.

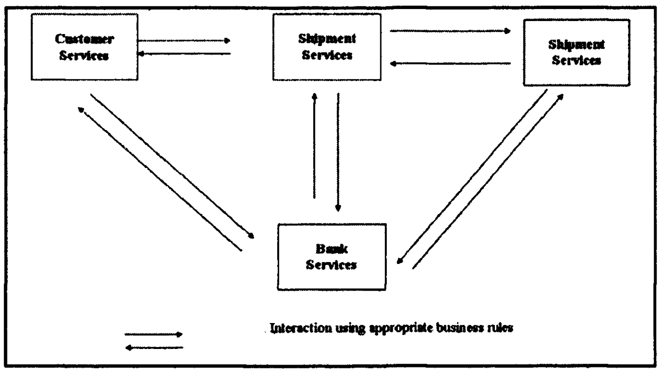


Figure 4-2. The general view of PC Selling application interaction.

4.2.2. PC Selling Case Study: Different Business Activities

In a typical operational business scenario, the provider service (PS) receives requests to buy PCs from the customer service (CS) and checks the availability, customer insurance, etc. At the same time, the customer imposes certain requirements, such as delivery time, specific providers, specific brand, etc. When different requirements from both the provider and the customer are agreed, an offer is made to the customer. Such an offer contains, in particular, the duration of the validity of the offer, the delivery time, the payment mode, etc. Delivery, payment, shipment, and possible cancelling then will follow in the same conversational manner. In some detail, the main activities involved in this service-oriented business application can be summarized as follows:

- **Request**: This activity is the first step in this process-driven application and corresponds to the customer's request for a number of PCs.
- **Offer**: In reply to the customer's request, this activity consists of an offer from the provider, including some necessary details such as the price.
- **Cancellation**: This optional business activity is performed when the customer wants just a part of the requested quantity and more precisely accepts part of the offer. Such a cancellation is possible

only before delivering the initial request; otherwise, penalties are applied.

- **Delivery**: This activity begins when the customer accepts the proposed offer. It notifies the provider to start delivering the PCs agreed on.
- **Shipment**: This activity concerns a case in which the customer's place of residence requires a shipment (i.e., no provider branch is available nearby).
- **Payment**: This activity deals with the execution of all the different aspects of paying, refunding, penalties, etc.

4.3. Phase-BRs4ACT: The Request Activity as an Example

As stated in chapter two, business rules play a key role in reshaping interorganizational behavioural existence and its competence. Indeed, business rules encompass all sorts of regulations, policies, and laws governing the internal functioning of an organization, as well as its interaction with the external world (e.g., customers, organizations, market laws). On the other side, as they are **evolving** inherently and changing rapidly, business rules represent the main **competence factor** within any organization and between collaborating ones.

Service orientation is interaction-driven by essence; that is, in rare cases one service or one organization can satisfy realistic requests, and this case study is not an exception. From this fact, those inter-service business rules that cross organizations are of due importance, and thereby present the behavioural contract or agreement between partners (e.g., frequently at least customer/requester and provider). Secondly, since service orientation is event-driven (e.g., publish/subscribe and invoke/reply/interact), the study will focus on event-driven business rules, and more specifically those following the ECA (event-conditions-actions) paradigm; that is, on the occurrence of some triggering events and under specific conditions and constraints to perform corresponding necessary actions.

To explain an activity-centric procedure, the focus is more on describing such cross-service, event-driven business rules with respect to each identified activity.

That is, any global business rule coping with the whole business process is postponed accordingly until the last phase. At this early stage, business rules will be described only at an informal, general level and, when possible, addressing any concerns (specifically functionality and quality ones). This, of course, is due to the fact that such general-purpose rules will be detailed and refined when addressing each concern separately.

After the general form and assumption about business rules, the following section will be focusing on explaining this phase through one of the activities of the case study. i.e., the request activity.

Request: The customer asks for a specific number of particular PCs. The customer has the right to opt for a preferred provider; this may depend on the history of similar operations or recommendations/ads from friends/spots. The customer also specifies the deadlines for responding, estimated prices, etc. The provider is entitled to accept or reject the request, depending on his/her workload, on that client's trustworthiness, location, etc. When all requirements are fulfilled, the provider makes this request a pending one by assigning it a reference, and notifies the customer. When the conditions are not met, the request is cancelled by the provider and the customer is notified.

It is worth mentioning that this informal business rule is event-driven, that is, triggered only when the customer asks for PCs. This business rule deals with different entities, such as customer, provider, PC Selling service, and so on. This business rule mentions both functionality and quality concerns, yet without exploring their detailed descriptions to be addressed at the level of each concern (i.e., next steps).

4.4. Phase-BRsCoordLaws—Modelling Coordination Concerns

Toward a clean separation of concerns and a high-level of adaptability, we propose at this stage to focus more on the functionalities in the involved activities; these already should have been identified and described in the previous phase. More precisely, a two-step of functionality issues is proposed. First, with respect to each identified activity, it is proposed to extract and refine all issues dealing with functionality concerns from the previous general-purpose intentional (cross-service) business rules, in terms of functionality-focused operational ECA business rules. Secondly, it is proposed to move these functionality business rules to the architectural level, using architectural connectors and specifically the so-called coordination laws. Through the case study and its request activity, more light will be shed on these two steps.

4.4.1. Functionality-focused ECA Business Rules: Description and illustration

The purpose of this sub-section is to go into detail about the coordination-driven functionality concerns in terms of ECA business rules, while ignoring completely any management issues, as they are the subject of the next section. More precisely, for each business activity, it is proposed to extract first the functionality concerns from the corresponding (general and intentional) business rules, and to refine them in a more structured and operational ECA-driven manner. Secondly, with the aim of bringing these ECA-driven rules closer to the (architectural) service paradigm, they will be modelled as architectural connectors, i.e., coordination laws [Fia02].

More precisely, the straightforward generic pattern proposed to describe functionality-focused ECA-driven rules can be sketched as follows:

COORD-RULE	<Business activity name>
PARTNERS	<Involved business entities/services as resources for the rule>
EVENT	<Different (composite) events triggering the respective activity>
CONSTRAINTS	<Different conditions to hold for performing the rule>
ACTIONS	<Actions to perform by events triggering and constraints holding>

Table 4-1. Functionality-focused ECA-driven rule.

It is important to emphasize in this ECA-driven pattern the presence of the <PARTNERS> clause. Indeed, as illustrated in the case study and in general, activities in service-driven applications involve in most cases at least two partners; in other words, the composition remains the essence of the service-orientation paradigm. For that reason and also to capture in the next steps the (service) interfaces required from such partners, it is proposed to make this clause mandatory in any concern-based (i.e., functionality but also management-focused) ECA-driven business rule. Due to this interaction, these rules are referred to, in an exchangeable manner, either as functionality or coordination rules. It is noted that all other clauses are self-explained and require no further clarification at this level.

4.4.1.1. Application to the request activity

From the functionality/coordination perspective, the provider has only to check the availability of the quantity and type requested to decide whether the request will be accepted or rejected. More precisely, the extracted ECA-driven business rule from the already described general-purpose intentional one could be detailed as follows:

COORD-RULE:	Request activity,
PARTNERS:	Customer requesting PCs and Providers of PCs,
EVENT:	The customer asks for a specific number of particular types of PC.
CONSTRAINTS:	After checking the availability and the types, the provider is entitled to accept or reject the request.
ACTIONS:	By accepting the request the provider puts it as a pending one by assigning it a reference, and notifies the customer in consequence.
EXCEPTIONS:	By rejecting the request, the provider cancels the request and notifies the customer.

Table 4-2. The extracted ECA-driven business rule.

It is worth mentioning that besides this basic rule for requests, more "profiled" yet functionality-focused interactive ECA-driven ones can be proposed. For instance, the provider may suggest more flexible and attractive constraints for "privileged" customers, such as discounts and credits, and so on. In other words, these rules are quite adaptable and evolve depending on the organization's policies, market laws, competitiveness, etc.

4.4.2. From Functionality ECA Business Rules to Coordination Laws

This phase aims to bridge the gap between the business level and the more conceptual level, where architectural decisions are to be taken to validate the described rules and facilitate/automate the move toward the service technology. In this respect, instead of opting for any functional/logic-based (e.g., Prolog [UM00,Zho06]) formalism, a behaviour-intensive architectural approach is adopted. As already highlighted, architectural techniques with their transient connectors permit a heightening of the level of rigor and ECA-driven business rules. Moreover, it is crucial to reemphasize the suitability of architectural techniques when targeting service technology as the ultimate deployment phase, as will be presented in the next chapter. More precisely, first, the required roles for architectural connectors to express any behavioural interaction (as glue) represent no more than (a selected part of) service interfaces, which are more likely to be from service components (in service-driven business applications). In other words, architectural techniques with their connectors already bridge the gap between the business/conceptual level and service deployment. Secondly, architectural connectors enhance composition, as their behavioural glues are built mostly from more than one interface/role. Thirdly, as shown in [MK96], architectural approaches support dynamic evolution as required for service agility and reconfigurability.

Intuitively speaking, given a functionality-focused interactive ECA-driven business rule governing an activity, a smooth refinement-based moving toward a corresponding architectural connector is proposed through the following steps.

1. Depending on the rule ingredients (e.g., events, attributes, and messages required from different partners) each involved partner-name is transformed into a connector role (see chapter 2, section 2.4) by defining all involved messages, events and/or properties.
2. Still depending on the rule core, specifically the constraint part, additional messages, attributes, constants, and invariants could be defined as part of the connector glue.

3. Finally, the rule itself is captured, still following the ECA-driven paradigm as connector glue by following the coordination law pattern [Fia02, AF01a, AFG+02, AF01b]. That is, the mechanisms that are required for regulating the relationships, functioning, and cooperation of services are externalized from business rules in terms of semantic primitives called coordination laws (see chapter 2, section 2.5). These describe composition mechanisms in terms of event-condition-action (ECA) rules that can be superimposed dynamically on stable core business entities and services. Superimposition [Kat93] is non-intrusive on the code that implements the services. Therefore, business architectures can be evolved dynamically, as volatile business rules change or new cross-organizational links come into force, while ensuring compliance with core business invariants.
4. The semantic modelling primitives being put forward for this phase rely on architectural connectors to separate the coordination of interactions between business service interfaces from their hidden computations. More precisely, so-called coordination laws and contracts externalize as first-class entities any intra- or cross-organizational interactions between business services using their interfaces in the form of roles. This clean separation permits changes to business rules to be performed at the level that is required without affecting other aspects.

4.4.2.1. From the Request Rule to the Corresponding Coordination Law

As described above, the first step consists of producing precise interfaces from the involved partners, in our case the Customer and the Provider. Starting with the Customer partner, the event part of the rule indicates that the request should be triggered by the customer. To be precise, such a request event should include the requested PC item (identity/name) and the requested quantity. Going to the actions part of that rule, it further is found that the customer has to accept or reject any request. These two actions thus are declared as services. Moreover, as shown below in this Customer Interface, to identify the customer, it is included as a type, and any required data has to be imported, such as Item, RequestRef. Finally, a meaningful name is assigned to the interface, composed of the name of the partner, the activity, and the symbol CI referring here to Coordination Interface.

```
coordination interface CustReqCI
partner type CUSTOMER
import type Item,RequestRef
events request(i:Item,Qt:nat)
services cancelled(),accepted(r:RequestRef)
end
```

Table 4-3. Required coordination Customer's interface for the Request activity.

Similarly, the coordination interface for the provider partner, as described below *ProReqCI*, involves the following services that a provider needs to interact with the customer: (1) make a request pending – *makePending(i,Qt)*; (2) the kinds of PCs offered by the provider – *typePrv(i)*; and (3) the quantity of PCs currently available for a given kind – *availQt(i)*.

```
coordination interface ProvReqCI
partner type PROVIDER
import type Item, Request
services
            makePending(i:Item,Qt:nat,Rq:Request)
            typePrv(i:Item):bool
            availQt(i:Item):nat
end
```

Table 4-4. Required coordination provider's interface for the Request activity.

Finally, there is the coordination law itself. First, it uses the two interfaces declared above, by declaring instances of them. With respect to service orientation, these instances may correspond to concrete services. Secondly, it reflects precisely the three parts of the above-described request business rule following the ECA-based paradigm. Syntactically, the connector law adopts the notation: **when** events with [or **if**] conditions do [or **then**] actions [**else** actions].

```
coordination law RequestCL
partners: Ct:CustReqCI; Pr:ProvReqCI
import type Item, Request
attribute   ReqNo:Request
  rule: Request in coordination
       when Ct.request(i,Qt) do
       if Pr.availQt(i)≥ Qt and
          Pr.typePrv(i)
          Pr.makePending(i,Qt)and
          Ct.accepted()
       else Ct.cancelled()
end
```

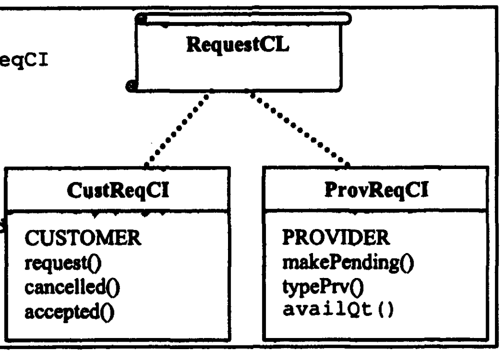


Table 4-5. Coordination law for the Request activity.

Referring again to the request activity and its identified business rule, the corresponding coordination law, *RequestCL*, as depicted, can be highlighted as follows: First, for each of the two involved interfaces, an instance variable is declared, namely **Ct** for the customer and **Pr** to refer to the provider. These interface instance variables will prefix any associated message or attribute from the respective interface. It should be noted that, depending on concrete situations, several instances also may be declared (e.g., when two providers are required, for instance, two instance variables are declared for the provider interface). Secondly, the cross-service event-driven ECA business rule now is conceived precisely. That is, after the clause when, the triggering event *Ct.request(I, Qt)* (from the customer Ct) is specified by including the requested item (I) and the quantity (Qt). At the constraint level (the if clause), the provider then checks for availability of type and quantity; if available, it assigns the request a number, notifies the customer, and makes it pending (the then part). When such constraints are not met, the provider notifies the customer that the request cannot go through, that is, the request simply is cancelled.

4.5. Management concerns: From Rules to Management laws

The previous section concentrated on the functionality concerns and presented how they can be elicited and formalized suitably and flexibly, using established interaction-centric ECA-driven business rules and coordination techniques at the business activity level. Nevertheless, it already has been emphasized that in order to get composite services, with stress on the quality, best possible managed business activities, the management issues have to be taken into account equally. Moreover, inspired by such functionality/coordination concerns we propose an approach on how such management concerns are required to be elicited and conceptualized at the activity level. More precisely, we suggest first to capture management concerns using management-based ECA-driven business rules. Secondly, such management-focused business rules are used to form new behaviour-intensive architectural connectors reflecting the management characteristics of such focused business rules.

To address management concerns in service-driven business applications, it is necessary to know that functionality/coordination concerns already have been conceptualized for any involved business activity in the business application at hand. In this thesis the main characteristics and features found to be essential when addressing management concerns at the business level are investigated. Finally, architectural approach is proposed, centred on management-customized behavioural connectors, referred to as management laws. How to move from the business rules to management laws is addressed.

4.5.1. Management-focused Business Rules: Characteristics and Pattern

Referring to the literature, e.g., [TPP+03,CCD+03, Kay02], quality-of-services usually are restricted to implementation-related metrics such as performance, invocation, response time, and availability, among others. These generally are part of any low-level implementation-driven service level agreement (IT-SLA) between the requester and web service providers. In contrast,[3] this work will focus first on qualities that may be agreed on at the business level; that is, without referring to any kind of implementation. In other words, above all business management concerns that arise between customers and providers at a more abstract, domain levels are addressed. As direct potentials of handling business

1. In reality, both conceptual- and implementation-centric qualities are achieved complementarily, and the next chapter will present ways to move the business-conceptual approach to the deployment level, where both classes of qualities can coexist and complement each other.

qualities, the following again may be singled out: (1) flexibility and evolution, since no technological constraints are imposed; (2) the involvement of all business stakeholders in eliciting and manipulating the corresponding rules; and (3) the separation of qualities from other concerns and from the targeting business processes and applications.

In the face of these advantages for business-level handling of management concerns, there is the challenging and decisive question: What should be the characteristics of management issues/qualities at that business-level? Indeed, whereas at the concrete implementation level there is a wide consensus as to which characteristics may improve performance and qualities (e.g., time-response, throughput, reliability, etc.), little information is found about what really characterizes business-level qualities, particularly for service-driven business applications. Toward this objective, we followed two main strands. On one hand, all approaches were considered that focused on technology-driven handling of qualities and management in Web services applications and that worked on abstract them to the business level, by decoupling them from implementation details. Several case studies also were carried out (e.g., travel agency, auctions, and of course, the PC Selling) directly at the domain level, by extracting and understanding their requirements from management perspectives.

The effort of these two complementary explorations revealed the three main characteristics as potential elements for management issues when addressed at the business-level. Before reporting on these proposed characteristics, the following facts should be emphasized. First, it was essential to maintain the identified characteristics as generic and as minimal as possible. In other words, no specific or concrete solution on how to tackle the proposed characteristics was suggested. Furthermore, it is not the intension to attempt a complete handling of all management issues; instead, it is argued that by putting forward a global abstraction that may be enriched/refined/specialized depending on the specific nature of the domain problems at hand, the required level of abstraction and the consequent deployment may be achieved.

These discovered management-focused characteristics and features, related to service-driven business applications, were categorized into three main classes. The first category is the one dealing with time constraints (e.g., deadlines, events timing, etc.). The second category concerns the importance of the profiles of involved partners (e.g., partner preferences, trust-level, etc). Finally, the third category is related to the current records of cooperation between any involved partners (e.g., history of agreements, activities, etc.). It is important to observe that all these characteristics indeed are interaction-driven, thus fitting with the service-orientation paradigm and composition-centricity.

- **Timing issues:** These represent essential elements for reflecting (business-level) timing constraints between partners (e.g., customers and providers). It is proposed to be distinguished explicitly and to avoid confusing business-level requirement timing issues with those related to system performance and timing. The latter have been considered extensively in different approaches [KL02, CCD+03, LSE03, LKD03], and consequently do not address the higher business level agreements.

  Deadlines to be observed, particulars of specific periods (weekend, holidays, Christmas, when discounts, special offers, etc., are at stake), periods for events invocation/replying and similar timing constraints represent the driving elements when tackling time-dependent quality requirements at that business level. In the case study, for instance, when the customer requests goods/PCs, (s)he has the right to set a specific time limit for a reply, as well as a specific period limit for accepting delivery of the goods. The provider in turn may set timing constraints, such as the duration of the validity of any offer, or the time for shipment in relation to the delivery time agreed upon with the customer. Furthermore, requests arriving during the week are handled slightly differently from those sent during the weekend, depending on the policy of the provider.

- **Partner preferences:** It is found that such profile-related properties of service partners (e.g., customer and provider) are deemed essential when looking for a better output/quality result in a given business activity associated with a service-based business process. That is, to satisfy a customer best, his/her preferences have to be taken into account carefully and addressed. In the case study, when requesting PCs, the customer may choose some PC providers over others depending on a number of factors such as: (1) how well-known the provider is; (2) what kind of PCs may be offered, and so on. The preferences of the provider also have to be considered in accepting/rejecting the customer's request. In summary, the preferences of the partners play a crucial role in terms of confidence, trust, and reputation toward better quality and management. Nevertheless, it is emphasized again that there is no focus on how such preferences are quantified; it just is assumed that they are to be set by the customer/provider (interfaces).
- **History of states of activities:** In addition to the two categories above, it was discovered that acceptance/rejection or adaptation of a given business activity is influenced by the previous history of similar transactions made by those partners with respect to such activity (e.g., several/few or no rejection/acceptance). Specifically in deciding on a long-time and persistent and thus costly business activity, the engagement of (one of) the partners (particularly the provider) depends heavily on how bad/good were the previous similar experiences. These previous experiences with respect to a given business activity are referred to as the state history of such activity, insisting again that the aim here is not to inspect operations at the system level, but to set appropriate formulae and primitives at the business level, so that appropriate management strategies can be built, depending on assumed, accepted, rejected, adapted, or even skipped business activities. With respect to the case study, for instance, the formulae that

are of most interest include how many (e.g., low, high, average) requests or offers have been accepted, rejected, or ignored; delivery or payment delays, etc. In this way, the behaviour of different activities can be adapted and/or improved, thus avoiding undesirable results such as a critical decrease in requests/offers or bottlenecks in the service. Finally, it is emphasized that although the management of the history of the states of activities may contribute to the quantification of the degree of trust of both the customer and the provider, both are kept separated since such details pertain to the deployment/measurement level.

Having identified these characteristics for coping with management concerns at the business activity level, the next logical step consists of supporting the stakeholders (e.g., designers, analysts, managers, and potential users) in eliciting and describing management-focused business rules on the basis of these characteristics, though still proposing to respect consistently the ECA paradigm for expressing these quality-specific business rules. In this manner, the stakeholders are helped to focus exclusively on the management issues and thereby avoid or at least minimize overlapping and cross-cutting between (interaction and management) concerns. The aim also is to provide the application designers with primitives for facilitating the description of management concerns in terms of specialized business rules, and afterwards specialized management-driven connectors (laws), and finally management-driven implementation-centric rules (e.g., RBSLA, see next chapter).

More concretely, three corresponding primitives are proposed; they are distributed over the three ECA clauses, namely the events, constraints, and actions. These primitives first will be summarized; then the whole corresponding ECA pattern for management concerns will be given; and finally they will be illustrated with the request activity of the PC Selling case study.

4.5.1.1. The ECA clauses for management concerns

The event clause: Since the event triggering an activity essentially is independent of any concern, it is proposed to keep the event clause as defined for the interaction concerns. However, besides the description of the event itself, the timing constraints will be included. This is achieved by adding a sub-clause to the event clause that is explicitly referred to as **at-time**. At-time allows the expression of any constraints related to deadlines, specific periods, and so on, as was explained for the timing issues characteristic.

The constraint clause: To emphasize that the constraints for the management concerns should be dealing with the preferences and the history of activities as detailed above, a new constraint clause has been added as management law primitive, which is denoted by the **who** clause. This primitive represents conditions clause constraints in the new form of business rules, and allows all kinds of constraints related to the preferences and the history of activities to be expressed. That is, for simplicity under the umbrella of **who**, both the second and third categories of management characteristics are merged. Finally, to stress that management concerns are being dealt with exclusively, it is proposed to adopt the primitive **merge** at the actions clause. To recapitulate, the ECA pattern put forward for expressing management-focused business rules takes the following form.

MANAG-RULE	<Respective chosen activity-name >
PARTNERS	<Involved business entities / services as resources for the rule>
EVENT	<Different (composite) events triggering the respective activity>
WHO	<Composed of all constraints involving preferences, trust and history>
AT-TIME	<Includes all constraints related to the timing issues >
MANAGE	<Management-related actions to perform when constraints at the who and at-time clauses hold>
[EXCEPTIONS]	<This optional actions part concerns the case when the constraints are not met>

Table 4-6. Expressing management-focused business rule.

Please note that since inter-services and interorganizational agreements for management are being promoted, as was done for coordination concerns, the partners clause still is kept as an essential element in this rule pattern.

4.5.1.2. The request activity as illustration for the management ECAs

As was explained, in addition to the functionality concerns, to seek more quality, each business activity needs to be boosted by management concerns. These management concerns generally are orthogonal to the interactions that business relationships impose. In terms of our case study, when a customer requests a batch of PCs, he/she first may prefer specific providers over others, possibly depending on previous experience with them or simply because their "reputable" products are preferred. Secondly, to obtain a quicker response to his/her request, the customer may set a desired response time to be respected by the provider. The provider has to consider its response time limit, which has to be smaller than the requested reply time. The day and the time at which the request is "posted" by the customer also may play an important role. For instance, dealing with any request during the weekend or outside normal working hours is more expensive. Following the proposed template, the extracted management-centric business rule may be expressed as follows:

MANAG-RULE:	The request activity.
EVENT:	The customer asks for a specific number of particular types of PC.
PARTNERS:	Customers and PCs providers.
WHO:	Customer opts for specific (on-line) providers and in turn the providers accept or reject that customer on the basis of history operations.
AT-TIME:	Specific response time from the customer to be respected by the provider.
MANAGE:	When both at-time and who constraints are met the request is considered as successful.
EXCEPTIONS:	If the constraints do not hold, the management concerns of the request are considered as having failed and alternatives should considered.

Table 4-7. The extracted management-centric business rule

4.5.2 Management Laws: Concepts and Illustration

In terms of a services-oriented architecture, the way that composite services need to be constructed not only shall obey a composition "logic" but also a management logic derived from management concerns. The different timing constraints, preferences, and degrees of trust between the involved partners, as well as the history of the relationship, represent important features that need to be taken into account at the level of management logic.

Following the same spirit undertaken for coordination concerns while moving from the business world to the service-engineering world, yet preserving all the strengths of the management-centric business rules, a variant of transient architectural connectors is put forward, referred to as management laws. More precisely, inspired by the primitives of coordination laws and more importantly by the specifics of management-centric business rules, this new concept of management laws is composed of: (1) management interfaces through which one can identify all the features (operations and events) that partners should expose to engage in a given business activity from a management perspective; and (2) management-centric behavioural glue, which allows accurate yet intuitive mirroring of the intended corresponding management business rule. In other words, it is required for such glue that all the rule clauses—and more specifically those associated with the management characteristics (e.g., **at-time** and **who**)—be reflected. Besides the rule itself, in order to strengthen the rigor, data types also are associated with every management interface to identify the class of instances, referred to as the partner type.

With these guidelines for management laws and coordination concerns, a stepwise methodology is proposed in the remainder of this section for moving management-focused business rules toward management laws. More, precisely, the interfaces first are extracted from the involved partners and then the behaviour of the expected management law is constructed. Afterward, the management laws are illustrated with the request activity.

4.5.2.1. Derivation of the management law interfaces from the partner names

As developed for moving from interaction-centric business rules toward conceptual coordination laws, the involved service partners in a given management-driven business rule should be detailed as interfaces for defining the associated management architectural connector; i.e., be a part in management law. More precisely, all informally explicit and/or implicit ingredients such as events, operations, and other attributes (e.g., variables, data types) required for expressing the management rule at hand (e.g., events, at-time, who and manage clauses) are to be defined precisely at the level of each corresponding partner interface. In this respect, for instance, precise yet meaningful names must be assigned to events, operations, and their parameter types defined.

4.5.2.2. Derivation of the management law glue from the management rule

Once all required interfaces have been defined precisely, the next step consists of capturing the informal event-driven management-centric business rule in equivalent yet architectural connector glue, referred to as management law glue. To make this move smooth, it is proposed to keep all its clauses as defined at the business rule level; that is, the primitive when it is proposed for capturing the event. The primitive **at-time** is kept for capturing the associated timing constraints and the primitive **who** for formally capturing constraints related to preferences, trusts, and history of state operations. Finally, to specify the management-centric actions to undertake, the primitive **manage** is proposed as invoked in the management business rule. The resulting general pattern of the management laws being put forward can be expressed as follows.

```
management law < law-name>
partners <variables typed with management interfaces>
types <optional clause to import any needed datatype>
rule <rule-name >
when <trigger>
 who <conditions on partner preferences and operation
        histories>
 at-time <conditions on time issues>
 manage <set of operations>
 else   <set of operations>
end
```

Table 4-8. The general pattern of management law.

Thus, under the clauses **when, who,** and **at-time**, the event triggering the concerned business activity, the related management constraints and conditions expressing partners preferences, and the degree of trust using the history of previous operations, if necessary, are specified. Under the clause **manage**, all (management) actions to be undertaken when the three above clauses hold are specified. Finally, when the triggering event happens but some constraints under **who** and/or **at-time** do not hold, the actions identified under *else* are performed.

4.5.2.3. Management law: Illustration using the request activity

By considering this management-centric business rule governing the request activity from the quality perspective, the first step consists of deriving the two interfaces for the customer and provider partners. The second step allows the capture of the ECA rule itself as management law. The details of the derivation of these three components (i.e., the two interfaces and the law) are given below.

A. **The customer management law interface:** The analysis of the corresponding request management business rule results in the following required customer interface.

```
management interface MaCuReMI
partner type CUSTOMER
type import Item, Request, Time
events request(i:Item,Qt:nat)
services
        TmRsp(): Time
        RqHrs(): Time
        RqDay(): Day
        PreferListPrv():(ProviderNames)
        HistPrv(): List(Operations)
end
```

Table 4-9. Customer Management interface for the Request activity.

More precisely, this management interface first identifies the triggering event, namely the request that must come from the customer side. Looking at formality and also for consistency with the coordination side, it is proposed to denote this event as *request(i,Qt)*; that is, on the management side, the request still involves two required parameters: the identity of the asked item and its quantity. The *datatypes* of both parameters are assumed to be defined elsewhere and imported here for use. Since in the clauses **at-time** and **who** of the request management rule, the customer was required informally to impose a time response and preference list of providers, which may depend on the history of operations, it is proposed to formalize them as attributes with shortened yet meaningful names and associated sorts. In this respect, *TmRsp()* refers to time-response and *PreferListPrv()* captures the list of preferred providers for each customer. Additionally, since the day and the hour of the arrival of a given request is important, these two attributes are captured precisely, using respectively *RqHrs()* and *RqDay()*. Finally, because the history of different operations also may also be relevant to the customer, this is considered through the operation *HistPrv()*.

B. The provider management law interface: Similarly, looking at the request management business rule straightforwardly results in the following provider interface.

```
management interface MaPrReMI
partner type PROVIDER
import type Item, Request
services
            NamePrv():Name
            TmRepLm():Time
            WrkDays():List(Days)
            WrkHrs():interval(Time)
            ManageRequest()
end
```

Table 4-10. Provider Management interface for the Request activity.

As the management business rule dictates, a number of operations are required from the provider: name, denoted by *NamePrv()*; current availability in terms of time for reply, denoted by *TmRepLm()*; information for the customer about the possible time for delivering the goods, denoted by *DeliverTm()*; and of the working days and hours, denoted by *WrkDays()* and *WrkHrs()*, respectively. Finally, there should be an operation for telling the provider that the management for the activity can proceed – *ManageRequest()*.

C. **The Management law for the request activity:** Building on these interfaces while being bound by the general pattern of management laws, the exploration of different clauses of that business rule results in table 4-11, which includes the management law for the request activity.

```
management law RequestML
partners MgCt:MaCuReMI; MgPr:MaPrReMI
rule     Request in Management
when     MgCt.Request(i,Qt) do
who      MgPr.NamePrv()in MgCt.PreferListPrv()
at-time
         MgCt.TmRsp()>= MgPr.TmRepLm()and
         MgCt.OrdHrs()in MgPr.WrkHrs()and
         MgCt.RqDay() in MgPr.WrkDays()
manage   MgPr.ManageRequest()
else     MgCt.Add(HistPrv,Cancel)
end
```

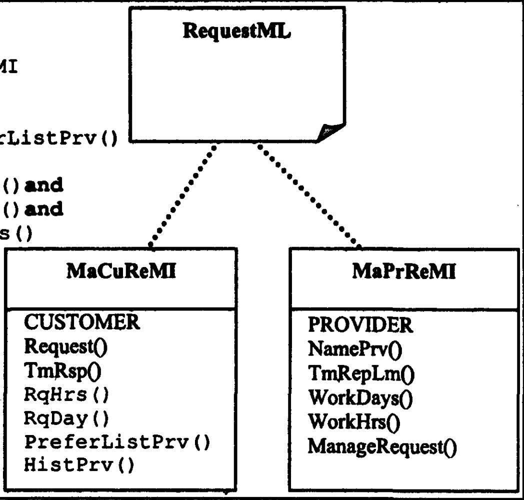


Table 4-11. Management law for the Request activity.

As for the coordination law, the management process is triggered by a request from the customer as specified by the when clause of the business rule. The condition part, as already mentioned, is split into two parts:

- **The time part,** under the clause at-time, specifies any conditions reporting on timing issues. The request activity allows a check that the time for reply requested by the customer is not beyond the ability of the provider. Such information is declared as an attribute of the law. The day and hour of the customer request also should come within the normal working days and hours of the provider, as stated in this management law.

- **The preference and history part** is specified under the clause who. As the name suggests, this conditional part concerns any condition regarding the preferences of the customer or the provider, as well as giving a history of operations in the business activity concerned. For the request activity, it is necessary only to check that the name of the provider is among the providers in the customer-preferred list.

The management actions to be performed are declared under the clause *manage*. In the case of the request activity, the only action that has to be performed consists of informing the customer and provider "partners" that the request activity has fulfilled all the requirements from the management perspective.

Finally, to describe the case in which there has been a failure to fulfil all the requirements (whether this is to do with timing or preference or history), the clause *else* specifies the actions that need to be undertaken. In the case study context, it adds the cancellation to the customer's history.

4.6. Integration of Concerns: Interactions Meet Management

In the previous sections, a stepwise approach associated with a set of business and conceptual primitives was put forward for handling two crucial concerns while developing service-oriented business processes: the coordination mechanisms that are necessary to compose business entities and services with respect to business rules governing given business activity, and the mechanisms that are responsible for handling management issues for the same business activities. Among the already emphasized advantages of this separation of concerns, it must be emphasized that each dimension can be described/modelled/reasoned on and evolved independently, where changes in one dimension can be carried out without interfering with decisions made with respect to the other.

Being able to address these concerns separately at early stages does not mean that they still should stay independent during the later development life cycle, particularly during the detailed analysis, deployment, execution, and maintenance levels. More concretely, during these phases, the way a business activity is required to be performed within a business process emerges from the coordination and management laws that apply jointly to that activity. The next paragraph illustrates this mandatory integration between both concerns for each business activity and applies it to the requested activity.

4.6.1. Integration of Concerns at the Request Activity

To consider further the necessity of integrating both concerns around a given business activity, the request activity is considered again. At runtime, the way a request is processed is not captured by independent coordination and management partners: rather, both coordination and management interfaces are instantiated by the same business entities. That is, the request is executed by a (run-time) customer service that is an instance of both the coordination interface *CustReqCI* and the management interface *MaCuReMI* associated, respectively, with the request coordination and management laws. In other words, both coordination and management interfaces are instantiated by the same run-time services or components; instantiating coordination and management laws means binding the coordination and management interfaces to services that are running on the current system configuration.

As depicted in the table 4-12, in the case of the request activity, two services, *CS* and *PS,* will be running, corresponding to the customer and the provider, respectively, where the event triggering the requested business activity is *request(i,Qt)* in both the coordination and management interfaces instantiated by the customer service. The joint execution of the corresponding coordination and management rules reflects the conjunction of the conditions and the parallel composition of the actions. That is, all conditions from both laws have to be brought together, and at the same time all actions under do and manage have to be brought together.

Thus, the request activity is performed according to both coordination and management rules, which share the same trigger – *CS.request(i,Qt)* – where *CS* is the run-time customer service. This means that all parameters are gathered so that all the information related to both coordination and management is available. The condition of the reaction to be performed is the conjunction of the condition of the coordination rule with the two conditional parts of the management rule – the at-time and who clauses. The operations under the manage clause of the management rule are combined with those of the then clause of the coordination rule. Finally, the operations under the after clause of the management rule are combined with those of the else clause of the coordination rule. As a result, when all coordination and management conditions are met, the request becomes pending and the provider service is notified to proceed. In the negative case in which some conditions are not met, the request is cancelled on the provider service and added to the history of the consumer service.

```
when CS.Request(i,Qt) do
if PS.AvailQt(i)≥Qt
   and PS.TypePrv(i)
   and CS.TmRsp()>= PS.TmRepLm()
   and CS.OrdHrs(inPS.WrkHrs()
   and CS.RqDay()inPS.WrkDays()
   and PS.NamePrv()in
       CS.PreferListPrv.list()
   then
       PS.PendingRequest(i,Qt)
    and PS.ManageRequest()
else   PS.CancelRequest(i,Qt)
    and CS.Add(HistPrv,Cancel())
```

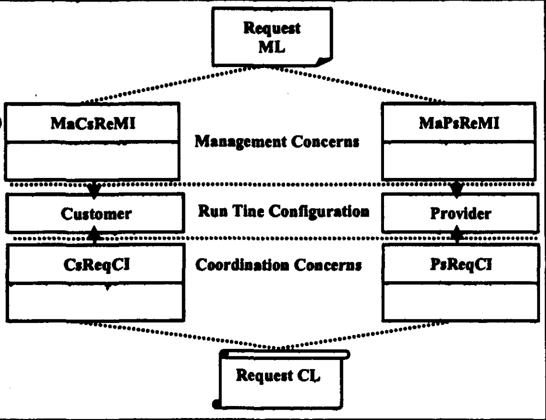


Table 4-12. Integration of concern for the Request activity.

4.7. Activity-centric Flexible Business Processes — PC Selling Illustration

After integrating both concerns around their respective activities, the next step consists of building business processes from such activities. That is, in contrast to most existing proposals for service composition, which start with the holistic business process modelling, it is judged and presented that "postponing" the process modelling at this stage brings several strengths. First, by focusing on activities and their concerns, the challenging problems such as flexibility, adaptability, and separation of concerns are being tackled. Secondly, by postponing the modelling of business processes at this level, the service mobility at the activity level becomes inherently supported since, depending on running requester/provider and their current context, the most appropriate workflow can be selected. This issue nevertheless is outside the scope of this thesis, and thus could be one of the potential research directions for future investigations.

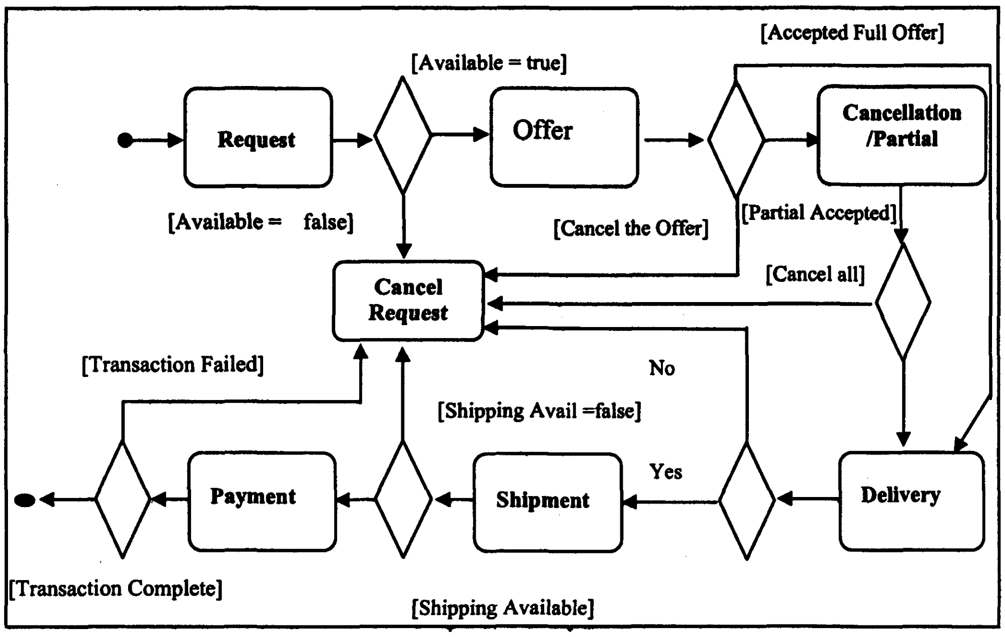


Figure 4-3. The conversational business processes.

A projection of these advantages in the case study can be highlighted in the following possible candidate business processes. As depicted in Figure 4-3, once the request is processed and after accepting a part or the whole of the offer from the customer's side, and before any delivery activity, the provider has to check whether a shipment is required (i.e., there are no provider branches in the customer's area). In this case, requirements from the shipment service should be met, such as: respecting the delivery time agreed with customer (i.e., the shipment time should be within the delivery time), shipment type, the normal cost and extra cost, etc. Other quite possible and privileged activities that flow (i.e., tailored business process) from some providers consist of first paying (the goods) and then shipping as an independent activity from paying. More additional variants consist of allowing the cancellation after only partial delivery, for instance. Through these alternatives, the intrinsic flexibility of the proposed activity-centric approach to service composition is emphasized.

Following the depicted process alternative, the provider service now can proceed to the delivery of the agreed and signed offer. The requirements that must be met for this business activity include, for instance, that the received items and quantity should reconcile with those in the agreed offer, the receipt time should not exceed the delivery time, and so on. Penalties have to be imposed on the provider in cases when such requirements are not respected. The final activity to conclude this conversational process is the payment. Thus, all postponed payments, penalties, refunds, etc., from all business partners (i.e., the provider, the customer and the shipment) have to be made.

4.8. The Approach-at-Work—Application to All PC Selling Activities

This section aims to present the "business-conceptual" approach by considering all the involved business activities as depicted in Figure 4-3 (without any partial ordering).

In the following, for each of the identified PC Selling business activities, the details therefore are delivered in the same spirit as was done for the request activity; that is, respecting the stepwise methodology of the approach: (1) general-purpose multi-concern intentional business rule description of the activity at hand; (2) the coordination-driven business rule and its associated coordination law; (3) the management-driven business rule and its corresponding management law; and (4) the integration of both concerns around the considered activity. Nevertheless, to gain some space, yet without losing expressivity, the details of the concern-based business rules (i.e., coordination and management) will be skipped and reported at the integration of the associated laws. Furthermore, since the integration of concerns is mostly straightforward, it will be left as a trivial training exercise for the reader.

4.8.1. Offer Activity: Business Rules, Coordination, and Management Laws

The general-purpose multi-concern business rule governing the description of the offer activity could be detailed informally as follows.

The offer intentional rule: The provider is entitled to accept or reject the pending request but must notify the customer of the decision. When the provider accepts the pending request, an offer is proposed under the following conditions: with respect to the reply and delivery time and the customer having a budget sufficient to pay the amount. When the customer intends to pay cash, the provider encourages the customer with specific discounts. When the above conditions are fulfilled, the customer puts this offer into a pending state. When the conditions are violated, the provider rejects the offer and records that the operation has failed.

4.8.1.1. The coordination concerns in the offer activity

As described in the above rule, the coordination focuses in this activity on the reply and delivery time, and also on the customer's budget. This means that the customer has only to check that the budget is sufficient to cover the offered price and the corresponding type to decide whether to accept or reject the offer. The part of the business rule that then is of concern may be detailed as follows:

1. The provider is entitled to accept or reject the pending request but must notify the customer of the decision.
2. When the provider accepts the pending request, an offer is proposed under the following conditions: the reply and delivery time, and that the customer has a sufficient budget to pay the amount.
3. When the above conditions are fulfilled, the customer puts this offer into a pending state.
4. When the conditions are violated, the provider rejects the offer.

A formalization of this business rule in terms of coordination laws, as shown in figure 4-4, requires the following interfaces and the respective coordination:

The customer *CustOfferCI.* provides the following operations: (1) sufficient budget *Budget()*; (2) the operation for offer pending – *makePendingOffer(i, Qt)*; and (3) the operation for cancelling an offer – *cancelled()*.

The required interface from the provider *ProvOfferCI* includes first the event triggering the offer, that is, *Offer()*. Furthermore, the following operations are required: (1) the quantity price – *PricePrv()*; and (2) a pending request being accepted – *AcceptedPendingRequest(i, Qt)*.

```
coordination interface CustOfferCI
partner type CUSTOMER
services
  Budget():Money
  MakePendingOffer(i:Item,Qt:nat)
  Cancelled()
end
```

```
coordination interface ProvOfferCI
partner type PROVIDER
services
   PricePrv():money
   AcceptedPendingRequest
             (i:Item,Qt:nat)
events
   Offer()
end
```

```
coordination law OfferCL
partners  Ct: CustOfferCI;
          Pr: ProvOfferCI
attribute ReqNo:Request
rule: Offer in coordination
when  Pr.Offer()and Pr.AcceptedPendingRequest(i,Qt)do
     if   Ct.Budget()>= Pr.PricePrv()then
          Ct.MakePendingOffer(i,Qt)
     else Ct.Cancelled()
end
```

Figure 4-4. Required interfaces and coordination law for the Offer activity.

By publishing the offer made by the provider service, the customer discovers it through the first when clause. The customer then checks for the acceptance pending request and the offer from the provider and checks that there is a budget sufficient for this transaction using the if clause, in which case the reference number is assigned to this offer and it is put in a pending state using the do clause. Otherwise, it is cancelled.

4.8.1.2. The management concerns in the offer activity

The extracted management-focused business rule for the offer activity, from the above general-purpose business rule, may be expressed as follows:

1. The provider is entitled to accept or reject the pending request, but must notify the customer of the decision.
2. When the provider accepts the pending request, an offer is proposed under two conditions in respect to the reply and the delivery time.

3. When the customer intends to pay cash, the provider encourages the customer by offering specific discounts.
4. When the above conditions are fulfilled, the customer puts this offer into a managing state.
5. When the conditions are violated, the provider records this as a failed operation.

The moving of this management business rule for the offer activity toward a corresponding management law is depicted in Figure 4-5 and can be summarized as follows:

The customer management interface *MaCuOfferMI* delivers four operations: (1) the time that the customer set for replying to the offer *reply2Offer()*; (2) the delivery time *deliveryRqst()*; (3) the payment method *choosepay(type)*; and (4) that the customer is managing the offer *manageOffer(i,Qt)*.

The provider management interface *MaPrOfferMI* publishes above all the intention to offer a specific quantity of PCs by accepting the pending offer and then by making an offer to the customer – *Offer()*. Also, from the provider's side, the following operations are required: the offer validity time – *ValidyOffer()*; the delivery time – *DeliveryTm()*; the method of payment – *PayMethodPrice()*; the acceptance of pending request – *acceptedPendingRequest()*; and finally, the history of different operations that also are relevant to the customer – *HistCst()*.

The management law *OfferML* that regulates the qualities between customer and provider during the offer activity first ensures that the pending request is accepted at the arrival of the offer event. Then, it checks the timing constraints, including the delivery and the reply imposed timings. Then, the method of payment is set and checked; in such cases, the offer is considered as being managed; otherwise it has to be cancelled.

```
management interface MaCtOfferMI
partner type  CUSTOMER
services
    Reply2Offer(): Time
    DeliveryRqst(): Time
    Choosepay():[Credit, Cash]
    ManageOffer(i:Item,Qt:nat)
end
```

```
management interface MaPrOfferMI
partner type PROVIDER
services
    ValidyOffer():Time
    DeliveryTm(): Time
    PayMetdPrice()
    AcceptedPendingReque(i:Item,
    Qt:nat)
    HistCst():List(operations)
events
    Offer(i:Item, Qt:nat)
end
```

```
management law OfferML
partners        MgCt : MaCtOfMI
                MgPr : MaPrOfMI
rule: Offer in Management
when MgPr.Offer(i,Qt)and  MgPr.AcceptedPendingRequest(i,Qt) do
at-time MgCt.DeliveryRqst()<= MgPr.DeliveryTm
  and   MgCt.Reply2Offer()<= MgPr.ValidyOffer()
  and   MgCt.Choosepay()= MgPr.PayMetdPrice()
manage  MgCt.ManageOffer()
else    MgPr.HistCst()
end
```

Figure 4-5. Required interfaces and management law for the Offer activity.

4.8.2. Delivery Activity: Business Rules, Coordination and Management Laws

The Deliver intentional rule: When the customer accepts the pending offer, delivery of the agreed PCs from the provider has to be asked for. For this purpose, the customer must arrange for a bank deposit not less than the amount the provider has fixed. On the provider's part, there has to be a bank ensuring the customer of the good conduct of the provider (that is, the provider must have a good bank history). On the customer's side, the customer insists that the requested delivery time is respected. Also, depending on the degree of trust of the customer, adequate discounts may be granted by the provider. Finally, when the customer is far from any of the provider's branches or main location, an arrangement for shipment must be planned.

4.8.2.1. The interaction concerns in the delivery activity

In this activity, when the customer accepts the offer and asks for delivery of the PCs, the customer must have a bank account and pay a deposit of not less than the amount the provider requests. The extracted coordination-focused business rule that is of concern here may be detailed as follows:

1. When the customer accepts the pending offer, the customer has to ask for delivery of the agreed PCs from the provider.
2. For this purpose, the customer must arrange for a bank deposit of not less than the provider has requested.
3. When these conditions are met, the customer puts this offer into a pending state.
4. When the conditions are violated, the customer rejects the offer.

This rule may be presented in terms of coordination law as follows:

The customer interface for this activity *CustDeliverCI* triggers the event for the delivery through *Ask4Deliver(i,Qt)*. In this event, the item types for delivery (i.e., the PC types) and the respective quantity are explicitly defined. Also the acceptance of the pending offer is requested as an operation – *AcceptedPendingOffer(i,Qt)*.

The interface from the provider, *ProvDeliverCI*, delivers in its turn the following operations: (1) the deposit must be paid by the customer – *DepositPrv()*; and (2) there must be an acceptance or refusal to deliver the required quantity of PCs, denoted by *Accept2Deliver (i,Qt)* and *Cancel()*.

The third coordination interface is from the bank, denoted by *BankDeliverCI*. That is, the customer should have a bank account and the deposit for the PCs should be reserved. These two operations are denoted by *AccountCst()* and *DepositCt()*.

From the delivery coordination interfaces above, the corresponding coordination law presents no difficulty and is more or less self-explanatory as it reflects the business rule described for the coordination concerns of the deliver activity.

```
coordination interface
CustDeliverCI
partner type CUSTOMER
services
    AcceptedPendingOffer
           (i: Item,Qt:nat)
events
    Ask4Deliver(i,Qt)
end
```

```
coordination interface
ProvDeliverCI
partner type PROVIDER
services
  DepositPrv()
  Accept2Deliver(i:Item,Qt:nat)
  Cancelled ()
end
```

```
coordination interface BankDeliverCI
partner type BANK
services
   AccountCst()
   DepositCt()
end
```

```
coordination law DeliverCL
partners Ct: CustDeliverCI; Pr: ProvDeliverCI;Bk: BankDeliverCI
attribute ReqNo: Request
rule : Deliver in coordination
when   Ct.Ask4Deliver(i,Qt) and Ct.AcceptdPendingOffer(i,Qt) do
   if BK.DepositCst() >= Pr.DepositPrv() and
      BK.Ct.AccountCst() then pr.Accept2deliver(i,Qt)
   else   Pr.Cancelled ( )
end
```

Figure 4-6. Required interfaces and coordination law for the Deliver activity.

4.8.2.2. The management concerns in the delivery activity

The extracted management-focused business rule can be expressed as follows:

1. When the customer accepts the pending offer, the customer has to ask the provider to deliver the agreed PCs.
2. For that purpose, the customer must arrange for bank insurance of not less than the price of the PCs.
3. On the provider's side, there must be a bank that will guarantee the good conduct of the customer (good bank history).
4. The customer must ensure that the requested delivery time will be respected. Also, depending on the degree of trust of the customer, adequate discounts may be granted by the provider.
5. Finally, when the conditions are violated, the provider records this as a failed operation.

The corresponding management law as depicted in figure 4-7, requires three interfaces, exactly as the coordination law; a fact that facilitates the integration of concerns and shows how coherent are both concerns, though conceived independently.

```
management interface MaCuDeMI
partner type CUSTOMER
services
   CstName(): name
   Ask2ReceiveTime():Time
   HistPrv():List(operations)
   AcceptedPendingOffer(i,Qt)
events Ask4Deliver(i:Item,
             Qt:nat,Plc:address)
end
```

```
management interface MaPrDeMI
partner type PROVIDER
services
   DeliveryTm():Time
   ListPrefCst:List[Cst,Percent]
   Deliver(i:Item, Qt:nat)
   ManageDeliver()
end
```

```
management interface MaBaDeMI
partner type BANK
services
   HistoryCstAcnt():[Good,bad]
   BankEnsuranceCst():Boolean
   AccounCst()
end
```

```
management law Deliver ML
partners   MgCt : MaCuDeMI; MgPr : MaPrDeMI; MgBk : MaBaDeMI
rule Management Deliver
when MgCt.Ask4Deliver(i,Qt,plc)and MgCt.AcceptedPendingOffer(i,Qt) do
who  MgCt.CstName()in MgPr.ListPrefCst()
at-time MgCt.Ask2ReceiveTime()>= MgPr.DeliveryTm()and
     MgBk.HistoryCstAcnt()= Good and MgBk.BankEnsuranceCst()= True
manage
     MgPr.ManageDeliver()
else MgCt.HistPrv()
end
```

Figure 4-7. Required interfaces and management law for the Deliver activity.

The Customer management interface *MaCuDeMI* identifies the event – *Ask4Deliver(i:Item,Qt:nat,Plc:address)* – that will be used as a trigger. Besides that event, four operations are required: (1) the customer name – *CstName()*; (2) the time by which customer asks to receive the goods; (3) the list of previous history for the provider – *HistPrv():List(operations)*; and (4) the accepting of pending offer – *AcceptedPendingOffer(i,Qt)*.

The Provider management interface *MaPrDeMI* publishes three operations: (1) the delivery time – *DeliveryTm();* (2) the list of preferred customers – *ListPrefCst:List[Cst, Percent]*; and (3) the customer is managing the delivery – *ManageDeliver(i,Qt)*.

Finally, the bank management interface *MaBaDeMI* exposes three operations: (1) the situation of the customer bank account – *HistoryCstAcnt():[good,bad]*; (2) the customer bank insurance – *BankEnsuranceCst()*; and the customer bank account – *AccounCst()*.

The management law that regulates the management behaviour described informally at the business rule is more or less self-explanatory, as given in Figure 4.7.

4.8.3. The Cancellation Activity: Rules, Coordination and Management Laws

The Cancellation intentional business rule: Before any delivery of the goods is made, the customer has the right to ask for cancellation of a part or all of the ordered PCs. Nevertheless, if the cancellation is more than a minimum amount of PCs stipulated by the provider, a penalty will be paid by the customer. In such a case, the provider has to refund the customer the appropriate amount and deliver the rest of the PCs, if any.

4.8.3.1. The Interaction concerns in the Cancellation activity

Sometimes the customer wants to cancel some or all of the quantity that already has been requested and both sides have agreed to do this. This section explains this activity starting from the request for cancellation from the customer to the acceptance or refusal to do so by the provider. The coordination part of the above business rule that is of concern here may be detailed as follows:

1. Before any delivery of goods, the customer has the right to ask for cancellation of a part or all of the ordered PCs.
2. Nevertheless, the cancellation should be less than a minimum number of PCs set by the provider when the customer accepted the offer and decided later to cancel some of his/her entire request.

3. In such a case, the provider has to refund to the customer the appropriate amount and deliver the rest of the PCs, if any, when this cancellation came after the acceptance request.

A formalization of this business rule in terms of the corresponding coordination law is given as depicted in Figure 4-8.

```
coordination interface CustCancelCI
partner type CUSTOMER
services
  AcceptedOffer (i,Qt,)
  AccountCst()
events
  Ask4Cancel(i:Item,Qt-Cl: nat )
end
```

```
coordination interface ProvCancelCI
partner type PROVIDER
services
 MaxCancelPercent(): Percent
 AccountPrv()
 CancelPrice():Money
 Accept2Cancel(i:Item,Qt-Cl:nat)
end
```

```
coordination interface BankCancelCI
partner type BANK
services
   AccountCst()
   AccountPrv()
   ToRefund:(AccountPrv,AccountCst,Money)
end
```

```
coordination law CancellationCL
partners Ct:CustCancelCI ; Pr:provCancelCI ; Bk.BankCancelCI
rule   cancellation in coordination
when   Ct.Ask4Cacel(i,Qt-Cl)and Ct.AcceptedOffer(i,Qt)do
if     Ct.Qt-Cl < Ct.AcceptOffer (Qt) then Pr.Accept2Cancel(i,Qt) and
       Bk.ToRefund(AccountPrv,AccountCst, Pr.CancelPrice())
end
```

Figure 4-8. Required interfaces and coordination law for the Cancel activity.

The coordination interface from the customer *CustCancelCI* first allows the triggering of the cancellation event *Ask4Cancel(i, Qt-Cl).* Besides that event, two operations are required: the accepted offer *AcceptedOffer(),* and bank account *AccountCst().* These are denoted by *AcceptedOffer (i,Qt,)* and *AccountCst().*

The coordination interface from the provider ProvCancelCI should publish the following operations: (1) the maximum percentage that has been allowed – *MaxCancelPercent()*; (2) the provider's bank account details that have been given – *AccountPrv()*; (3) the cancellation price for the quantity that has been offered –

CancelPrice(); and (4) the cancellation has been accepted – *Accept2Cancel(i:Item,Qt-Cl:nat)*.

From the bank, the interface BankCancelCI is required with the following operations: (1) details of the customer's and provider's accounts – *AccountCst()*, *AccountPrv()*; and (2) the amount that the provider will refund to the customer regarding the quantity that has been cancelled – *ToRefund:(AccountPrv, AccountCst,Money)*

From the above cancellation coordination interfaces, the corresponding coordination law CancellationCL reflects exactly the intended behaviour reflected in the business rule.

4.8.3.2. The management concerns in the Cancellation activity

The management-focused business rule for this cancelling activity has to be extracted and takes the following form:

1. Before any of the goods are delivered, the customer has the right to ask for cancellation of a part or all of the order placed for PCs.
2. Nevertheless, the cancellation must not go below a minimum amount of PCs set by the provider: otherwise, a penalty will be paid by the customer.
3. In such a case, the provider has to refund to the customer the appropriate amount and deliver the rest, if any.
4. When the conditions are violated, the provider records this failed operation.

The management law depicted in figure 4-9, requires three interfaces: one for the customer, the others for the provider and the bank.

The customer management interface *MaCuCaMI* first identifies the triggering event – *Ask4Cancel(i,Qt-Cl)*. Then, two further operations are published: (1) the cancellation time – *CancelTime()*; and (2) the customer account – *AccountCust()*.

The provider management interface *MaPrCaMI* publishes five operations: (1) the delivery time – *DeliveryTm()*; (2) the maximum cancellation percentage – *MaxCancelPercent()*; (3) the provider bank account – *AccountPrv()*; (4) the list of previous history for the customer – *HistCust():List(operations)*; and (5) the managing of the cancellation – *ManageCancel()*.

```
management interface MaCuCaMI
partner type CUSTOMER
services
   CancelTime():Time
   AccountCst()
   AcceptedOffer(i,Qt)
events
   Ask4Cancel(i,Qt-Cl)
end
```

```
management interface MaPrCaMI

partner type PROVIDER
services
   DeliveryTm(): Time
   MaxCancelPercent():Percent
   AccountPrv()
   HistCst:List(operatios)
   ManageCancel()
end
```

```
management interface MaBaCaMI
partner type BANK
services
   AccountCst()
   AccountPrv()
   ToTransfer(AccountCst, AccountPrv)
end
```

```
management law CancellationML
partners    MgCt : MaCuCaMI ; MgPr : MaPrCaMI ; MgBk : MaBaCaMI
attribute
rule     Cancellation in Management
when     MgCt.Ask4Cancel(i,Qt-Cl)and Mgct.AcceptedOffer(i,Qt)
at-time  MgCt.CancelTime()<= MgPr.DeliveryTm()and
         MgCt.Qt-Cl()<= MgPr.MaxCancelPercent()
manage   MgPr.ManageCancellation() and
         MgBk.ToTransfer(AccountCst, AccountPrv)
else     MgPr.HistCst()
end
```

Figure 4-9. Required interfaces and management law for the Cancel activity.

The Bank management interface *MaBaCaMI* exposes three operations: (1) the customer bank account – *AccountCst()*; (2) the provider bank account *AccountPrv()*; and (3) the amount that will be transferred *–ToRefund(AccounCst, AccountPrv)*.

Finally, the management law on the arrival of a cancelling event checks first whether the offer already has been accepted. Then, it checks if the cancel period has not already passed, as well as the tolerated percent of PCs to be cancelled. In this case, the cancel is considered as being managed.

4.8.4. Shipment Activity: Rules, Coordination and Management Laws

The Shipment intentional business rule: The shipment provider has the right to accept or reject the request for shipment from the provider. When the shipment request is accepted, the provider has to agree to pay the agreed amount for such a shipment. The duration of the shipment should be less than the delivery time agreed between the customer and the provider. Depending on the type of shipment (special or normal), the distance between the provider and the customer, and the weight of the shipped PCs, different formulae for either refund or extra payment can be agreed on.

4.8.4.1. The interaction concerns in the Shipment activity

Each delivery activity needs shipment because the items purchased need to be delivered to the customer, although sometimes in special cases the provider may deliver the order, when the customer is based in the same area. The coordination-driven business rule may be described as follows:

1. First, the provider asks the shipment provider for shipment with details of the item type, the quantity, and the location.
2. The shipment provider has the right to accept or reject the shipment request from the provider.
3. When the shipment request has been accepted, the provider has to agree to pay the agreed amount for such a shipment.

The precise coordination law of this activity from that informal business rule is depicted in fugure 4-10.

The coordination interface from the customer, *CustShipCI,* exposes the address, including the street, post code, city and telephone number. The coordination interface from the provider *ProvShipCI* allows the triggering of the shipment event *Ask4Ship(i,Qt,Loc).* Two further operations also are required from the provider side: the address and bank account – *addressProv()* and *accountPrv().*

The coordination interface from the shipment provider *ShipPrCI* gives the ability to decide whether to accept or refuse the shipment through the operation *ShipAccept().* After that, the acceptance to make the shipment for the specific types and quantity of items to the specific place is denoted by *AcceptShip(i,Qt,Loc)*; the price for the shipment is denoted by *ShipPrice()*.

A coordination interface from the bank *BankShipCI* provides the shipment provider's account number – *AccountShPr()* – and the provider's account number – *AccountPrv()* – from their respective banks. Also, the price to be paid for this transaction from the provider's account into the shipment provider's account – *ToPay()*.

Finally, the shipment coordination rule permits reasoning on these interface operations to capture the intended functional logic of the shipment, as described informally in the corresponding business rule.

```
coordination interface CustShipCI
partner type CUSTOMER
services
  AddressCst(): address
end
```

```
coordination interface ProvShipCI
partner type PROVIDER
services
   AddressProv():Address
   AccountPrv()
events
   ask4Ship(i:Item,Qt:nat,
                      Loc:address)
end
```

```
coordination interface  ShipPrCI
partner type Shipment Provider
services
   ShipPrice(): Money
   ShipAccept(): Boolean
   AcceptShip(i:Item,Qt:nat,Loc)
   Cancelled()
end
```

```
coordination interface BankShipCI
partner type BANK
services
  AccountShPr()
  AccountPrv()
  ToPay():(AccountPrv,AccountShPr,
                          Money)
end
```

```
coordination law ShipmentCL
partners
     Ct:CustShipCI ; Pr:provShipCI ; Bk.bankShipCI ; ShPr.shipPrCI
rule shipment in coordination
when Pr.Ask4Ship(i, Qt, Loc) do
 if  ShPr.ShipAccept() = true then ShPr.AcceptShip(i,Qt,Loc)and
     Bk.ToPay(AccountPrv, AccountShPr, ShPr.ShipPrice())
else
     ShPr.Cancelled()
end
```

Figure 4-10. Required interfaces and coordination law for the Shipment activity.

4.8.4.2. The management concerns in the shipment activity

The management-driven business rule for the shipment activity can be described informally as follows:

1. The shipment provider has the right to accept or reject the shipment requested by the provider.
2. When the shipment request is accepted, the provider has to agree to pay the agreed amount for such a shipment.
3. The shipment's duration should be less than the total delivery time agreed between the customer and the provider.
4. Depending on the type of shipment (special or normal), the distance between the provider and the customer, and the weight of the shipped PCs, different formulae for either refund or extra payment must be agreed on.

The management law itself is derived progressively as depicted in figure 4-11.

The management interface from the customer *MaCuShMI* for this business activity should expose at least one operation to specify the address needed to deliver the goods – *AddressCst().*

The management interface from the provider *MaPrShMI* first identifies the triggering event for the management law, that is the request for shipment – *Ask4Ship(i:Item,Qt:nat).* Further required operations are: (1) the delivery time – *DeliveryTm();* (2) the provider bank account – *AccountPrv();* (3) the provider address – *AddressPrv()*; and (4) the required shipment type – *Requiresh type()*; the shipment price – *Shpprice()*; and provider history – *HistShPr().*

The management interface from the bank denoted by *MaBaShMI* exhibits at least three operations: (1) the shipment provider bank account *-AccountShPr()*; (2) the provider bank account – *AccountPrv()*; and (3) the amount to be refunded – *ToRefund(Account, Account).*

The management interface to be exposed by the shipment provider is denoted by *MaShPrShMI* and offers the following operations: (1) the time for shipment – *ShipTime()*; (2) the shipment type – *ShipType();* (3) the shipment price –

ShipPrice(); (4) the shipment provider bank account amount – *AccountShPr()*; and (5) the managing of the shipment – *ManageShipment()*.

```
management interface MaCuShMI
partner type CUSTOMER
services
   AddressCst():Address
end
```

```
management interface MaPrShMI
partner type PROVIDER
services
    DeliveryTm() : Time
    AccountPrv(): Money
    AddressPrv(): Address
    RequireShtype()
    ShpPrice(): Money
    HistShPr(): List(operatios)
events
    Ask4Ship(i:Item,Qt:nat)
end
```

```
management interface MaBaShMI
partner type BANK
services
   AccountShPr()
   AccountPrv()
   ToRefund(AccountPrv,
            AccountShPr)
end
```

```
management interface MaShPrMI
partner type ShipmentPr
services
   ShipTime():Time
   ShipType()
   ShipPrice():Money
   AccountShPr()
   ManageShipment()
end
```

```
management law ShipmentML
partners MgCt:MaCuShMI ; MgPr:MaPrShMI ; MgShPr:MaShPrShMI
rule Shipment in management
when     MgPr.Ask4Ship(I,Qt,Loc,ShipType) do
at-time  MgPrSh.ShipTime()<=MgPr.DeliveryTm() and
         MgShpr.ShipType()= MgPr.RequireShtype()and
         MgPr.ShpPrice()= MgPrSh.ShipPrice()
manage   MgPrSh.ManageShipment()
else     MgPr.HistShPr()
end
```

Figure 4-11. Required interfaces and management law for the Shipment activity.

The management law *ShipmentML* on the arrival of the shipment event, checks the timing constraints between the requirement of the customer and the ability of the provider. It also checks for the compatibility of the types of the to-be-shipped goods (PCs) between the requester and the provider. Finally, the price given by the provider for shipment should be equal to what is asked by the shipment provider. In such cases, the shipment is declared to be managed.

4.8.5. Payment Activity: Rules, Coordination and Management Laws

The Payment intentional business rule: This conversational business process ends with the definitive payment process. First, the customer has to pay the provider the agreed total price for the delivered PCs. Depending on the output of each of the above activities in terms of refunding amounts, penalties, and so on, either on the side of the customer or the provider, such refunds have to be completed at this stage by bank transfer. This last operation is subject to the condition that the PCs are received in time (i.e., received at a time earlier than the delivery time).

4.8.5.1. The coordination concerns in the payment activity

At the end of any business transaction, and for the PCs Selling case study in particular, the payment activity comes as an important stage in the business process. In this activity, all promises and penalties from both sides (the customer and the provider) should be completed, depending on what was agreed.

This conversational business process ends with the definitive payment process.

1. First, the customer has to pay the provider the agreed total price of the delivered PCs.
2. When the provider has agreed to deliver the PCs and the customer has received the agreed quantity and item type at the agreed price, the customer has to pay the amount due by bank transfer.

This business rule is expressed formally in terms of the following coordination law.

The coordination interface from the customer *CustPayCI* for the payment activity should expose at least: the bank account number of the customer – *AccountCst()*; and details of the items received – *Received(Item-Rc,Qt-Rc,Price-Rc)*. The coordination interface from the provider *ProvPayCI,* besides triggering payment through the *Ask4Payment()* event, exposes the provider's bank account details as an operation – *AccountPrv()*.

The coordination interface from the bank *BankpayCI* exhibits the following operations: (1) the customer's and provider's account numbers must be provided –

AccountCst(), AccountPrv(); and (2) the agreed amount must be transferred from the customer's bank account to the provider's bank account – *ToTransfer(): (AccountCst, AccountPrv,Money)*.

The payment rule for the coordination concerns *PaymentCL* ensures that all requested items indeed are received by the customer with the agreed quantity and price. In such a case, the customer is required to proceed to the transfer of the total amount.

```
coordination interface CustPayCI
partner type CUSTOMER
services
  Received(i-Rc,Qt-Rc,Price-Rc)
  AccountCust()
end
```

```
coordination interface ProvPayCI
partner type PROVIDER
services
   AccountPrv()
events
   Ask4payment()
end
```

```
coordination interface BankPayCI
partner type BANK
services
   AccountCst()
   AccountPrv()
   ToTransfer:(AccountCst,
                AccountPrv,Money)
end
```

```
coordination law PaymentCL
partners    Ct: CustPayCI;   Pr: ProvPayCI ; Bk: BankPayCI
rule   payment in coordination
when   Pr.Ask4Payment()do
if     Ct.Reiceived(i-Rc,Qt-Rc,Price-Rc) and Ct.Item-Rc =i and
       Ct.Qt-Rc = Qt and Ct.Price-Rc= Price
then   Bk.Totransfer(AccountCst, AccountPrv,Price())
end
```

Figure 4-12. Required interfaces and coordination law for the Payment activity.

4.8.5.2. The management concerns in the payment activity

The management-focused business rule for the payment activity can be formulated intuitively as follows:

This conversational business process ends with the definitive payment process.

1. First, the customer has to pay the provider the agreed total price for the delivered PCs.
2. Depending on the output of each of the above activities in terms of refunding amounts, penalties, and so on, either on the side of the

customer or the provider, such refunds have to be completed at this stage through bank transfer.

3. This last operation is subject to the condition that the PCs are received in time (i.e., the received time is less than the delivery time).
4. Finally, when the conditions are violated, the customer records this as a failed operation.

The corresponding management law as depicted in figure 4-13 involves three interfaces: one for the customer, the others for the provider and the bank.

```
Management interface MaCuPaM
Partner type CUSTOMER
Services
   ReceivedTime():Time
   AccountCst()
   HistPrv():List(operations)
end
```

```
Management interface MaPrPaM
Partner type PROVIDER
Services
    DeliveryTm(): Time
    AccountPrv()
events
    Ask4payment()
end
```

```
Management interface MaBaPaMI
Partner type BANK
services
   AccountCst()
   AccountPrv()
   ManagePayment()
end
```

```
Management law paymentML
Partners    MgCt : MaCuPaMI ;  MgPr : MaPrPaMI;MgBk : MaBaPaMI
Attribute   PenaltyPercent : Percent
Rule: management Payment
when     MgPr.Ask4Payment()do
at-time MgCt.ReceivedTime()<=MgPr.DeliveryTm()
manage  MgBK.ManagePayment()
else
        MgCt.HistPrv()
end
```

Figure 4-13. Required interfaces and management law for the Payment activity.

The customer management interface is denoted by MaCuPaMI. Three operations are needed: (1) the goods received time – *ReceivedTime()*; (2) the customer bank account – *AccountCust()*; and (3) the previous history for the current provider – *HistPrv()*.

The provider management interface identifies an event – *Ask4Payment()*,which will be used as a trigger in a law. This management interface is denoted by MaPrPaMI. Two operations are denoted as follows: (1) the delivery time – *DeliveryTm()*; and (2) the provider bank account – *AccountPrv()*.

The bank management interface is denoted by *MaBaPaMI*. Three operations are denoted as follows: (1) the customer bank account – *CustomerCst()*; (2) the provider bank account – *AccountPrv()*; and (3) the managing of payment – *ManagePayment()*.

4.9. Chapter Summary

To conclude this chapter, it again is emphasized that business processes require the cooperation of several organizations; this presents challenging problems such as adaptive coordination and interaction, suitable handling of management issues. Service-oriented Architectures (SOAs), as infrastructures, together with their Web services, are proposed in [RW02, RD05a, RD05b, RND05] as useful platforms for deploying such modern business processes. Nevertheless, with the growing scale and complexity of service-driven applications and business processes, need rigorous approaches at an early stage when considering business requirements.

We proposed a stepwise approach based on business rules, architectural techniques, and a separation of concerns. In this approach, interaction and management concerns are elicited separately as ECA business rules and then modelled, validated, and evolved as architectural connectors following an event-driven methodology. To reflect the semantics of business activities, both concerns then are integrated in a smooth and clear way. These ideas and concepts are illustrated using a case study dealing with the selling of PCs.

The next chapter will address enhancing the practicability of this architectural approach. In particular, it will investigate how both coordination and management laws should profit best from Web technology, without losing their main strengths and features.

CHAPTER FIVE

Chapter 5

Web Service-based Management laws

In the previous chapter, we put forward a stepwise approach for specifying and evolving interaction and management concerns in service-driven business oriented architectures. At the business level, the approach is governed through event-driven and business rules. At the conceptual level the approach moves toward service orientation through architectural connectors, i.e. coordination and management laws. These architectural laws already move the process one step closer to service orientation. Indeed, on one hand, with their explicit interfaces composed of messages/events and properties/attributes, both management and coordination laws capture the concept of service interfaces. On the other hand, the ECA rules governing such laws express the orchestrated exchanges of messages while composing such services (interfaces).

This chapter aims to present the proposed approach and support its compliance with service-oriented techniques: event-driven, communication through interfaces, message-based composition, activity- and process-centricity. More precisely, this chapter concentrates on "conceptually compliant" moves of both management and coordination laws toward Web Service technology. Conceptually compliant means the preservation of strengths and advantages of this approach, including explicit separation of concerns, activity-centricity, flexibility, and being ECA driven.

Indeed, whereas the fine-grained activity level is the suitable conceptual level, WS standards focus on the holistic business process level. Moreover, the aim is to push for persistent ECA-driven behaviour; WS standards allow only message exchanges with basic conditions on variables. Finally, this work is looking for dynamic and evolving management and coordination concerns through transient connectors and WS standards such as WSDL and BPEL.

To overcome the above limitations of Web Services standards, related fields were explored, such as the semantic web [MPM+04, BHL01] as well as ongoing innovative emerging approaches at the research level handling missing issues such as the integration of Web-Services and business rules [OYP03, Pap03, CM04]. These explorations have allowed the preservation of the potentials of the deployment approach using WS-based deployment. The main ingredients for the deployment of this approach consist of: (1) exploiting the potential of available XML-based reactive RuleML languages, and mainly their management-oriented variant called RBSLA; and (2) benefiting from recent ideas of bringing such RuleML-based languages to Web services.

The rest of this chapter is organized as follows. In the next section motivation is brought to the proposed deployment approach and its translation steps. In the second section, all concepts about the required ingredients such as RuleML and RBSLA are presented. The third section, with the support of one of the business activities of the PC Selling case study, details how the interfaces for management/coordination laws are translated to WSDL. In the fourth main section, details are given of how different concepts and ingredients of management law are translated into RBSLA mechanisms. Since RBSLA inherently includes all RuleML mechanisms, we will just hint at the straightforward mapping of coordination laws towards RuleML as a direct result of the proposed translation of management laws. In the fifth section, the software prototype implemented for editing management law and automating the translation is presented. The chapter concludes with a summary of its main contributions.

5.1. From Conceptual MLaws toward WS Technology: Motivation and Translating Milestones

In this process we have aimed to preserve the following main capabilities and advantages of our architectural approach to management concerns:

Explicit separation of concerns: In the proposed architectural approach, not all concerns are put together in a way that follows the semantics of business activity. Instead, this integration of concerns is postponed and thus a Web-based implementation is derived first for each concern separately. This allows the performance of additional concrete testing. After such separate testing of each concern, the integration of coordination and management concerns around their corresponding activity semantics then takes place.

ECA-driven pattern: As was emphasized in the previous section, the event-driven character of the proposed architectural approach represents the main building stone for enhancing *adaptability*, and invocation as event-based (business) rules. Although most Web standards, such as BPEL, WSDL, or WSCI, are not explicitly event driven, this gap has been investigated recently by the Web Semantics community, leading to Reaction RuleML[1] [PKB07] as a sub-language of the RuleML language family [Bol06]. These recent advances around RuleML are respected, while translating the approach to the Web-based implementation. To preserve one main essence of the management laws, it was necessary to investigate the state of the art and literature about the Semantic Web, since they annotate services with richer semantics. It was found that the XML-based RuleML language and its extensions toward *Reaction RuleML* fulfil most requirements on preserving the ECA setting of activity-centric *coordination*/management laws. From the coordination perspective, RuleML definitely caters to all involved ECA clauses, namely, the events, conditions, and actions to perform.

Web Service-based deployment of rule-based management laws: To emphasize the service orientation of this Web-based deployment of management laws effectively, each management law (also coordination law) was considered as a Web service, with inherent interfaces such as WSDL required protocol descriptions and the rules themselves as business-logic (called management-logic) protocols that both providers and requesters must adhere to. In this way, everything becomes highly transparent and network-centric, whereby the management laws are agreed or disagreed on conversationally. These RBSLA-based management laws as Web services could be owned, preferably by the

[1] http://ibis.in.tum.de/research/ReactionRuleML/

provider that handles management concerns between services (providers and requesters).

Activity-centric compliance: Finally, in contrast to BPEL and WSCI, which are limited to elementary coordination activities such as invoke, send, and receive (the so-called composite activities are just basic ordering activities), the complex application-related business activities as were achieved at the architectural level must be described explicitly. For that purpose, with the support of different variants of RuleML languages, the approach supports separating implementation of activities (by bringing all their concerns) and then projects their causal/timing ordering.

As depicted in Figure 5-1, putting into force all the above necessary ingredients and deployment decisions results in a progressive and conceptually-compliant supporting approach for deploying both management and coordination laws. More precisely, as illustrated in the figure, at least five mandatory steps are to be performed.

Step 1 – From Coordination interface to corresponding WSDL: As hinted at above, the fact that both management and coordination laws are explicitly associated with interfaces, this translation is more or less straightforward, as will be illustrated later.

Step 2 – From Management interface to corresponding WSDL: This translation is like the one above and presents no difficulties. It is illustrated later through a business activity from the case study.

Step 3 – From Coordination rule to corresponding RuleML: At this stage it is proposed to translate the ECA conceptual coordination rule into a similar but programmable rule using the RuleML language.

Step 4 – From Management rule to corresponding RBSLA: To keep all peculiarities of management rules, particularly the at-time and the who characteristics, it is proposed to translate ECA conceptual management rules into the RBSLA language. The proposed translating ideas are detailed in the main section.

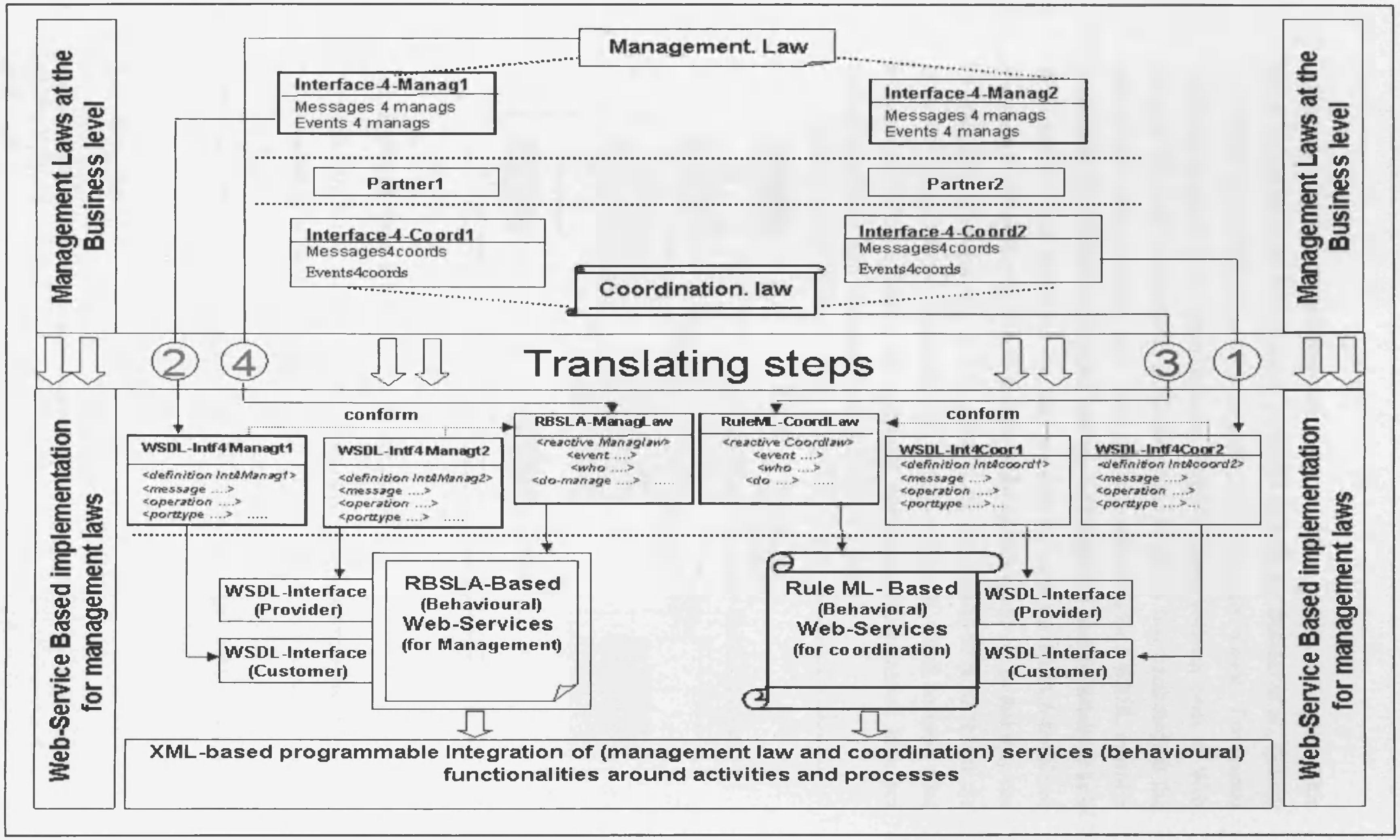

Management Laws at the Business level
Management. Law
Interface-4-Manag1
Messages 4 manags
Events 4 manags
Interface-4-Manag2
Messages 4 manags
Events 4 manags
Partner1
Partner2
Interface-4-Coord1
Messages4coords
Events4coords
Interface-4-Coord2
Messages4coords
Events4coords
Coordination. law
Translating steps
2
4
3
1
conform
RBSLA-ManagLaw
<reactive Managlaw> <event ....> <who ....> <do-manage ....> .....
RuleML-CoordLaw
<reactive Coordlaw> <event ....> <who ....> <do ....> .....
WSDL-Intf4Managt1
<definition Int4Managt1> <message ....> <operation ....> <porttype ....> .....
WSDL-Intf4Managt2
<definition Int4Managt2> <message ....> <operation ....> <porttype ....> .....
WSDL-Int4Coor1
<definition Int4coord1> <message ....> <operation ....> <porttype ....>...
WSDL-Intf4Coor2
<definition Int4coord2> <message ....> <operation ....> <porttype ....>...
WSDL-Interface (Provider)
WSDL-Interface (Customer)
RBSLA-Based (Behavioural) Web-Services (for Management)
Rule ML-Based (Behavioral) Web-Services (for coordination)
WSDL-Interface (Provider)
WSDL-Interface (Customer)
XML-based programmable Integration of (management law and coordination) services (behavioural) functionalities around activities and processes
Web-Service Based implementation for management laws

Figure 5-1. The General Approach Architecture for Deploying MLaws using WS Technology.

Step 5 – Coordination/Management laws as rule-centric Web Services: This phase was inspired by the approach proposed recently by Dustdar et al., namely the ViDRE [RND06] architecture as graphically illustrated below. The crucial contribution consists of conceptualizing RuleML-based business rules as Web Services that can be described, published, and composed (i.e. combined) at the wish of the customer in this case. Each rule is associated with a WSDL interface (containing the involved messages) and the rule itself can also be accessed as a Web service. This capitalizes on this proposal to lever the RBSLA-translated management/coordination laws to be conceived as Web Services, so that they can be integrated subsequently, as was done at the conceptual level to reflect the semantics of each business activity. In contrast to RuleML-based business rules on which ViDRE is based, the proposed management/coordination laws are already associated with (management/interfaces) interfaces.

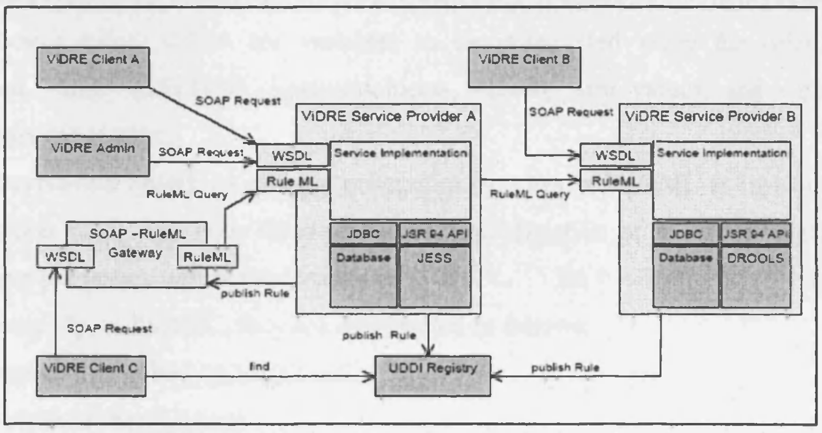


Figure 5-2. The ViDRE Approach Architecture as given in [RND06].

5.2. RuleML and RBSLA: An Overview

In this section, before delving into the detail of the proposed deployment approach for management / coordination laws, we first present a brief overview of the RuleML language elements. Then, since we already summarized in chapter three most general-purpose concepts around RBSLA as a language for SLAs, we devote the second sub-section to the study of most specific RBSLA concepts that will be playing important role in the translation of management laws.

5.2.1. RuleML: Overview and Illustration

With the increasing need for describing and reasoning the semantics of Web-based information and services explicitly, and the limitations of standards such as WSDL, BPEL, and others in representing knowledge-intensive rules, much effort has been invested recently in the so-called semantics Web, aimed at developing rule-based standards. RuleML belongs to the XML-based standard for expressing rules on the Web [RD05, TWB03]. The fundamental ingredients of the Derivation RuleML, part of the RuleML language family, can be summarized as follows [Bol06, Rul06]:

[Predicate]: is an n-ary relation introduced via an <Atom> element, which has the following form: Atom(Rel, (Ind|Var|Data|Expr)*): where <Rel> is the relation (predicate) name, <Var> are variables to be instantiated when the rules are applied, <Ind> individual constants/objects, <Data> data values, and <Expr> complex expression.

[Derivation rules]: as in logic programming, rules in RuleML as instance of derivation rules, e.g. horn clauses, that is, a conjunction of (possibly negated) premise predicates and a conclusion: H ← B1 ^... ^ Bn ^ ~Bn+1 ^... ^ ~ Bm . Syntactically, in RuleML, they are represented as follows:

Implies ((head,body)|(body,head))

body(And) head(Atom)

And(Atom+) .

[Facts]: are stated as atoms considered always being true. Their syntax is Atom(Rel,(Ind|Var|Data|Expr)*).

5.2.2. RBSLA: Focused Overview with Illustration

The main concepts and principles of RBSLA that are of interest to the management laws translation proposal can be summarized as follows:

Explicit Temporal Qualification of ECA Reaction Rules: RBSLA extends the ECA paradigm stemming from the active database domain with additional parts such as a pre-condition part, which is used to specify, e.g. the *triggering time* of the event detection or event monitoring, or a post-condition part that specifies the conditions (e.g. integrity constraints) after the action execution. More precisely, the general logical form of global ECA rules in RBSLA consists of the following: **eca (Time, Event, Condition, Action, Postcondition, Else)**, where the clauses of Event, Condition, Action, Postcondition and Else correspond closely to the primitives in management or coordination laws. It should be mentioned here that, besides ECA-driven and messaging reaction rules, RBSLA also allows production rules (condition-action rules), derivation rules of the form IF-THEN-ELSE and mixed forms between reasoning derivation rules and active reaction rules, such as serial derivation rules (e.g. serial Horn rules).

Deontic Normative Relations : As (service-level) agreements mean not just describing rules and actions but also **enforcing** them, RBSLA is associated with a set of deontic primitives to express and enforce normative relationships such as obligations, permissions, and prohibitions between contract partners. The general construct is **norm(S,O,A)**, where S plays the role of the subject (requester), O the object (provider), and A the corresponding action to be performed. Normative statements in RBSLA are expressed as modality functions and relations by the attribute per = "*modal,*" i.e. a function or relation with "*modal*" use interpreted as modality. **Example:** If obliging a customer to pay a penalty to a provider, this is expressed in RBSLA as shown in table 5-1.

Normative modalities in RBSLA typically are embedded in a temporal KR event/action logic (Event Calculus axioms) where the norms are managed as changeable states, which are initiated and terminated by happened events/actions time, using two additional constructs denoted by **Initiates** and **Terminates.** This approach helps to overcome well known paradoxes of standard deontic logic and allows temporal state tracking of the valid contractual rights and obligation at a time.

```
<Expr>
<Fun per="modal">oblige</Fund>
      <Var>Customer</Var>
      <Var>Provider</Var>
      <Expr><Fun>pay</Fun><Var>Penalty</Var></Expr>
</Expr>
```

Table 5-1. Example oblige a customer to pay penalty.

Procedural attachments: First, recalling that SLA and policy expressions deal in essence with aggregated and correlated information, such as number/frequency/percent of requests/selling of a given item/product, frequency of availability of product offered on a web site, etc. This data is system-based; that is, it relates to information collected on the IT infrastructure level. Typically, these kinds of data measurement and data processing are done in highly optimized and specialized systems, such as system and network management tools and data bases/data warehouses. To access these systems at run-time and use the low-level raw system information or the aggregated high-level business data dynamically, RBSLA supports expressive procedural attachments to call external APIs seamlessly and use their efficient procedural functionalities and query capabilities.

```
<!-- Bind the constructed Java Calendar object with the
actual system time to the variable Date =
java.io.Calendar.getInstance() -->
<Equal>
  <Var>Date</Var>
  <Expr>
    <!-- class -->
    <oid><Ind uri="java://java.util.Calendar"/></oid>
    <!-- constructor  -->
    <Fun per="effect">getInstance</Fun>
  </Expr>
</Equal>
```

Table 5-2. Example of procedural attachment.

In management law, the *who* primitive is required for expressing implementation-related knowledge such as the history of operations or *the preference of* provider/customer, which basically depend on the history of the running components' configuration services. Therefore, these issues are not addressed on this level of abstraction.

Moreover, typical SLA domain specific vocabularies and elements are well represented as metrics in RBSLA; therefore, it fits with the SLA-based requirements in our management law.

Finally, it should be emphasized that invariants also can be expressed in RBSLA, using the so-called fluents (changeable states represented by an Event Calculus axiomatization) and query primitives such as holdAt and valueAt. These are useful in the presence of invariants in the management of coordination laws.

5.3. Management/Coordination Interfaces into WSDL

Proposed by W3C, the Web Service Description Language (WSDL) is an XML-based standard for describing Web services interfaces. WSDL descriptions are composed of interfaces and implementation definitions [CGM+04, HHL04]. Interfaces are abstract and reusable descriptions that can be referenced by multiple implementations.

As described in Chapter Two, a WSDL document defines services as collections of communication endpoints capable of exchanging messages. Elements used in the definition of services include: type as a container for data type definitions; message as a description of data being communicated; operation as the definition of an action; set of operations as port type protocol; and data format specification for port type to allow the reuse of abstract definitions, port, and services. The most important WSDL element is the <portType>; it describes a Web service through the operations that can be performed and the messages that are involved.

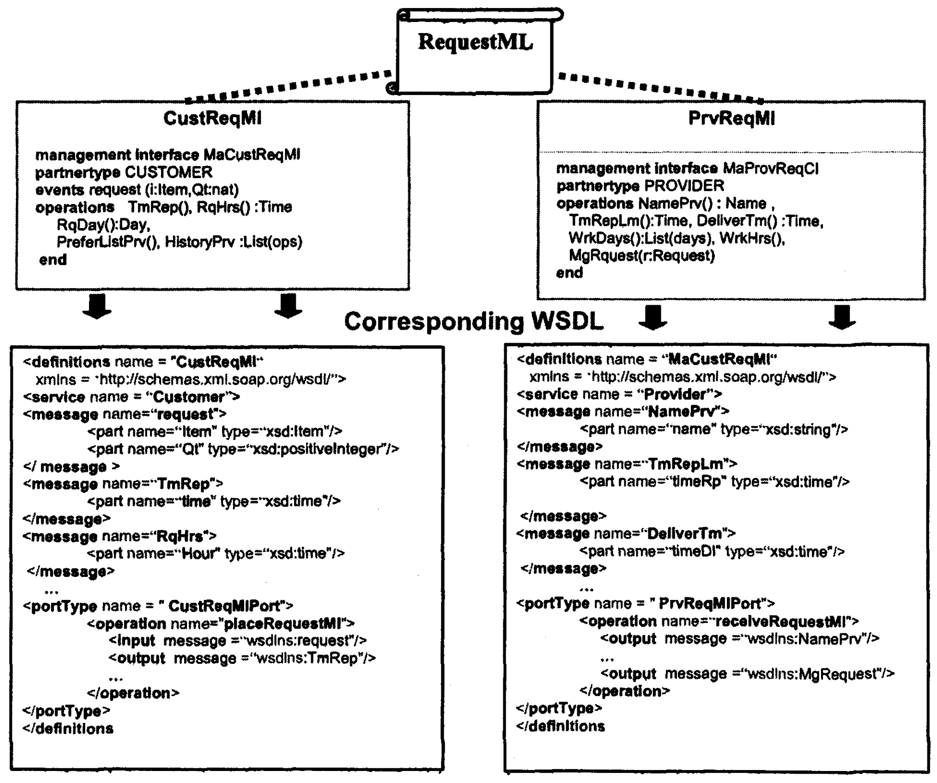
The translation of management and coordination interfaces into WSDL programs could be seen as a simple exercise. Indeed, both management and coordination interfaces manipulate exactly the same elements, such as interface names and (input/output) messages. Table 5-3 depicts this trivial translation. As an illustration, the translation of the request management law interfaces is presented below. Events are considered as inputs from the law point of view. The operation can be output only or input/output, depending on the existence of the return type of the service.

Management (Coordination) Interface		WSDL interface
Interface name	➔	<definitions name>
Partnertype	➔	<service name>
Event	➔	<Input message>
Services	➔	<input / output message>

Table 5-3. Translation of management law to WSDL.

5.3.1. Management in WSDL: Illustration

Applied on the request activity of the PC Selling case study, the corresponding two interfaces thus are translated into corresponding WSDL as follows.



```
management interface MaCustReqMl
partnertype CUSTOMER
events request (i:Item,Qt:nat)
operations   TmRep(), RqHrs() :Time
    RqDay():Day,
    PreferListPrv(), HistoryPrv :List(ops)
 end
```

```
management interface MaProvReqCl
partnertype PROVIDER
operations NamePrv() : Name ,
  TmRepLm():Time, DeliverTm() :Time,
  WrkDays():List(days), WrkHrs(),
  MgRquest(r:Request)
end
```

```
<definitions name = "CustReqMl"
  xmlns = "http://schemas.xml.soap.org/wsdl/">
<service name = "Customer">
<message name="request">
          <part name="Item" type="xsd:Item"/>
          <part name="Qt" type="xsd:positiveInteger"/>
</ message >
<message name="TmRep">
          <part name="time" type="xsd:time"/>
</message>
<message name="RqHrs">
          <part name="Hour" type="xsd:time"/>
 </message>
    ...
<portType name = " CustReqMlPort">
          <operation name="placeRequestMl">
              <input  message ="wsdlns:request"/>
              <output  message ="wsdlns:TmRep"/>
              ...
          </operation>
</portType>
</definitions
```

```
<definitions name = "MaCustReqMl"
  xmlns = "http://schemas.xml.soap.org/wsdl/">
<service name = "Provider">
<message name="NamePrv">
          <part name="name" type="xsd:string"/>
</message>
<message name="TmRepLm">
          <part name="timeRp" type="xsd:time"/>

 </message>
<message name="DeliverTm">
          <part name="timeDl" type="xsd:time"/>
 </message>
          ...
<portType name = " PrvReqMlPort">
          <operation name="receiveRequestMl">
              <output  message ="wsdlns:NamePrv"/>
              ...
              <output  message ="wsdlns:MgRequest"/>
          </operation>
</portType>
</definitions
```

Figure 5-3. WSDL interface for the Request activity.

5.4. From Management Laws to RBSLA-driven Web Services

This section presents the details of the corresponding RBSLA concepts/primitives for each clause in management law.

Recalling these management clauses, the corresponding RBSLA translations are presented. Management laws are in fact composed of the following clauses: (1) the **management interfaces;** it already has been shown how they can be codified using the WSDL Web Service Interface standard; (2) the **involved partners,** such as the provider and the requester, who have to enter into a contract partnership; (3) the **management rule,** which itself is composed of the following:

- the triggering event with all its parameters;
- the timing constraint, which has conditions reporting on timing issues;
- the who clause, which has conditional part concerns regarding the preferences of the customer or the provider, as well as giving a history of operations in the business activity concerned; and finally the actions to perform.

A management law contains one or more management rules, which are of ECA structure. Each management rule identifies, under the "when" clause, a trigger to which the contract will react; the trigger can be just an event observed directly by one of the partners or can be a more complex condition built from one or more events. Management has extra clauses like "who" and "at-time", which address management concerns. The former are related to partner preferences and operation histories, and the latter provide conditions on time issues. The reaction to occurrences of the trigger is identified under the "manage" clause as a set of operations. If any conditions pertain other than the "when" clause condition, then the manage clause is not performed and the optional clause "else" will be fired (see Chapter 4, Section 4.5.1.1).

The main objective of this chapter is to move management laws to Web services architecture to make use of widespread Web service technologies in the realm of distributed systems [AFP07]. RBSLA has been found to be the suitable tool to extend management laws to XML-based Web service extensions.

Although RBSLA seems quite similar to management laws in many aspects, they fundamentally tackle triggers and time issues in totally different ways and it is closer to the platform-specific layer than our management laws, which abstract from the technical perspective in a computational independent view. Therefore, the next section is devoted to discussing this issue and to presenting a pragmatic solution for it.

5.4.1. The Detailed Translating Mechanisms

The focus of this section is using RBSLA language for authoring formal rules for management laws via a set of uniform translation rules. More specifically, rules written in management laws are taken as input, parsed into RBSLA.

As mentioned before, a management law has various primitives, each of which has a clause that contains statements. A statement can be an event, a condition, or an action. Each may contain partner references, methods, and parameters. Moreover, statements may contain arithmetic, comparison, and list operations.

Following the generic pattern for management law presented in the previous chapter and as recalled in Table 4-8, the EBNF syntax for the management law is shown in Table 5-4.

```
Management  law name
partners
      {management  interfaces}*
types {{par}+:data_type}*
rules
when trigger
who conditions //on partner preferences and histories
at_time conditions //on time issues
Manage {operations}*
else {operations}*
end law
```

Table 5-4. The EBNF syntax for the management law.

Reaction RuleML (version 0.2), which evolved from the reaction rules event handling in RBSLA (i.e. ECA RuleML), integrates external API calls and service invocations as Boolean-valued atomic functions <Atom> if calls return a Boolean value or as functional expressions <Expr> if calls return one or more objects. The translation rules are shown in Table 5-5.

Management law	RBSLA
When event	<on> <!-- event --> </on>
event	<on > <Atom><!-- event --></Atom></on >
Who conditions on partner preferences and histories	<if> <!-- condition --> </if>
at-time condition on time issues	<if> <!-- condition --> </if>
condition	<if > <Atom><!-- condition --></Atom></if >
Manage action	<do> <!-- action --> </do>
action	<do> <Atom><!-- action --></Atom></do>
X.Y ({parm: data type}*)	
X Partner reference	<Ind uri="*X*"/>
Y method	<oid> <!-- method name --> </oid>
***P* ≡ ({parm: data type}*)**	<Var>*P*</Var>
List operations	
In // Set operation	<Rel per="value">In</Rel>
comparison	
≤	<Rel per="value">LessEqual</Rel>
<	<Rel per="value">Less</Rel>
>	<Rel per="value">More</Rel>
≥	<Rel per="value">MoreEqual</Rel>
=	<Rel per="value">Equal</Rel>
Arithmetic operations	
+	<Rel per="value">Add</Rel>
-	<Rel per="value">Sub</Rel>
*	<Rel per="value">Mult</Rel>
/	<Rel per="value">Div</Rel>

Table 5-5. The translation rules.

[MLaws Event → RBSLA Event]: Both management laws and RBSLA global reaction rules are based on the ECA concept; therefore, they share the idea of anchoring on events by assuming that each management law trigger refers to a corresponding RBSLA trigger. **Example:** Recalling the management rule for the request activity as detailed in the previous chapter:

```
management law RequestML
partners MgCt:MaCuReMI; MgPr:MaPrReMI
when MgCt.Request(i,Qt)
who MgPr.NamePrv()in MgCt.PreferListPrv()
 at-time
       MgCt.deadline()≥ MgPr.TmRepLm()and
       MgCt.OrdHrs()in MgPr.WrkHrs()  and
       MgCt.RqDay() in MgPr.WrkDays()
 manage MgPr.ManageRequest()
 else   MgCt.Add(HistPrv,Cancel)
end
```

Table 5-6. Management law for Request activity.

Event in RBSLA represents in <on> tag. *MgCt* is an object reference to the customer interface (denoted by its URI) that can be translated into <oid> as individual constant <Ind>. The operation *Request* should be translated into the relation tags <Rel> in which the option per ="effect" can be used to interpret both the *value* and the (side)-effect of them. The variables i and Qt are represented by variable tag <Var>. The complete example for event translation is shown in table 5-7.

Event in MLaws	Event in RBSLA

when MgCt.Request(i,Qt)	<on>
	  <Atom>
	    <oid><Ind uri= "**MgCt**"/></oid>
	    <Rel per="effect">**Request**</Rel>
	    <Var>**i**</Var>
	    <Var>**Qt**</Var>
	  </Atom>
	</on>

Table 5-7. Example for Event translation.

[MLaws Who clause → RBSLA]: As explained in the previous sections, the valuation and assessment of history of previous operations and preferences of providers/customers can be checked by using the if condition.

The full example with explanation is shown Table 5-8.

Who condition in MLaws	**Who condition in RBSLA**

who MgPr.NamePrv()
in
MgCt.PreferListPrv()
	<if>
	<Atom>
	  <!-- Boolean relation, which is interpreted -->
	  <Rel per="value">in</Rel>
	  <!-- First argument of in function -->
	  <Expr>
	    <!-- uri pointer to external object or class -->
	    <oid><Ind uri="MgPr"/></oid>
	    <!-- method/function call of the object/class -->
	    <Fun per="effect">NamePrv</Fun>
	    <!-- arguments; here no argument -->
	  </Expr>
	  <!-- second argument of in function -->
	  <Expr>
	    <oid><Ind uri="MgCt"/></oid>
	    <Fun per="effect">PreferListPrv</Fun>
	  </Expr>
	</Atom>
	</if>

Table 5-8. Example for "WHO" condition translation.

[MLaws Timing Constraints → RBSLA Timing Constraints]: The **at-time** management law clause can be translated into a condition clause of RBSLA language by using the <if> tag. Multiple conditions can be connected via the <and> tag. The function tag <fun> is used to call external operations from interfaces and compare them, using comparison operators. In the example, there is deadline, work hours, and work days as conditions, illustrated in Table 5-9.

[MLaws Manage clause → RBSLA actions]: Finally, the manage clause represents no difficulty, since it corresponds clearly to actions to be performed and thus can be codified directly, using an RBSLA actions clause with the ECAs rule. In RBSLA, tag <do> is used to represent the action and, like the other part in RBSLA, the same tag is used to represent the partners and operations. The example for condition is shown in Table 5-10.

The full example for management law in RBSLA can be presented as shown in table 5-11.

Time in MLaws	**Time conditions in RBSLA**

at-time
MgCt.deadline() ≤
MgPr.TmRepLm()
and
MgCt.OrdHr()**in**
MgPr.WrkHrs()
and
MgCt.RqDay()**in**
MgPr.WrkDays()
	<if>
	  <Atom>
	    <Rel use="value">**LessEqual**</Rel><Expr><oid><Ind use="**MgCt**"/></oid><Fun use= "effect">**deadline**</Fun></Expr>
	    <Expr><oid><Ind per="**MgPr**"/></oid><Fun per= "effect">**TmRepLm**</Fun></Expr>
	  </Atom>
	  <Atom>
	    <Rel per="value">**in**</Rel><Expr><oid><Ind uri= "**MgCt**"/></oid><Fun per= "effect">**OrdHr** </Fun></Expr>
	    <Expr><oid><Ind per="**MgPr**"/></oid><Fun per= "effect">**WrkHrs**</Fun></Expr>
	  </Atom>
	  <Atom>
	    <Rel per="value">**in**</Rel><Expr><oid><Ind uri= "**MgCt**"/></oid><Fun per="effect">**RqDay**</Fun> </Expr>
	    <Expr><oid><Ind per="**MgPr**"/></oid><Fun per= "effect">**WrkDays**</Fun></Expr>
	  </Atom>
	</if>

Table 5-9. Example for At-Time translating.

Manage condition in MLaws	**Manage condition in RBSLA**

manage MgPr.ManageRequest()
	<do>
	  <Atom>
	    <oid><Ind uri="**MgPr**"/></oid>
	    <Rel per= "effect">**ManageRequest**</Rel>
	  </Atom>
	</do>

Table 5-10. Example for Manage translating.

```
<Rule execution ="active">
   <label><Expr><Rel>name</Rel><Ind>RequestML</Ind><Expr/></label>
   <on>
       <Atom>
          <oid><Ind uri= "MgCt"/></oid>
          <Rel per= "effect">Request</Rel>
          <Var>i</Var>
          <Var>Qt</Var>
       </Atom>
   </on>
   <if>
       <And>
        <Atom>
          <Rel per="value">in</Rel><Expr><oid><Ind per="MgPr"/></oid><fun per=
            "effect">NamePrv</fun></Expr>
          <Expr><oid><Ind per"MgCt"/></oid><fun per= "effect"> PreferListPrv
            </fun></Expr>
        </Atom>
        <Atom>
          <Rel per "value">LessEqual</Rel><Expr><oid><Ind per"MgCt"/></oid><fun
            per= "effect">deadline </fun></Expr>
          <Expr><oid><Ind per"MgPr"/></oid><fun per=
            "effect">TmRepLm</fun></Expr>
        </Atom>
        <Atom>
          <Rel>in</Rel><Expr><oid><Ind uri= "MgCt"/></oid><fun per=
            "effect">OrdHr</fun></Expr>
          <Expr><oid><Ind per"MgPr"/></oid><fun per=
            "effect">WrkHrs</fun></Expr>
        <Atom>
          <Rel>in</Rel><Expr><oid><Ind uri= "MgCt"/></oid><fun per=
            "effect">RqDay</fun></Expr>
          <Expr><oid><Ind per"MgPr"/></oid><fun per= "effect">WrkDays
            </fun></Expr>
        </Atom></And>
   </if>
   <do><Atom><oid><Ind uri="MgPr"/></oid><Rel per= "effect">
          ManageRequest</Rel> </Atom>
   </do>
   <elseDo> <Atom><oid><Ind uri="MgCt"/></oid><Rel per= "effect">
          CancelRequest</Rel> </Atom>
   </elseDo>
```

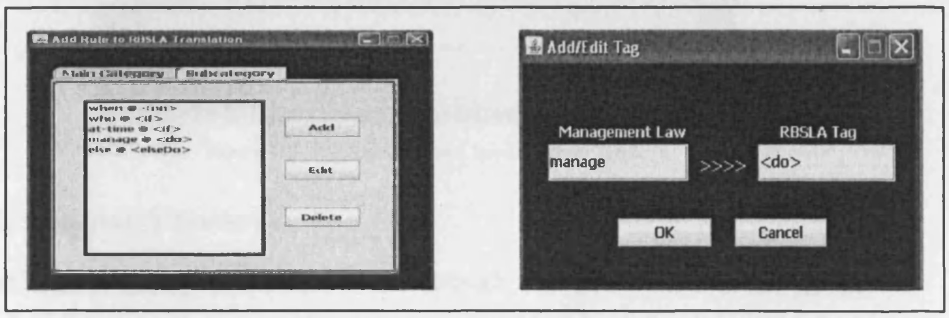
Table 5-11. Example for translating management law to RBSLA.

5.5. A Supporting Tool for Automating the Translation

To translate management law to RBSLA language, a tool was designed using Java technology. The tool allows the designer to specify a management law. Once this law has been checked syntactically, it will be translated automatically into corresponding RBSLA code following the above detailed steps.

Following the aforementioned rules, the user interface depicted below is implemented. It includes the main category interface and their subcategories.

The main category form consists of the main primitives of management law and the corresponding primitives in RBSLA syntax, as illustrated in figure 5-4. These primitives can be updated using the add/edit form tag.

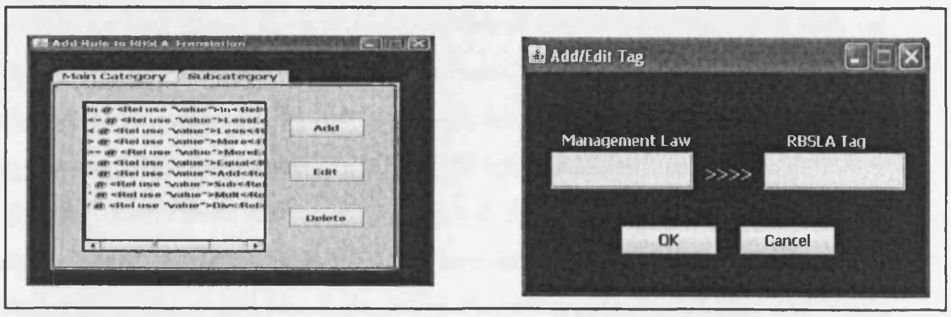


```
when @ <on>
who @ <if>
at-time @ <if>
manage @ <do>
else @ <elseDo>
```

Management Law		RBSLA Tag
manage	>>>>	<do>

Figure 5-4. Management tool add/edit form.

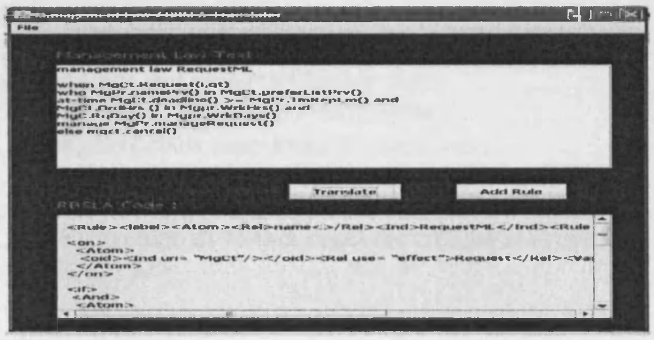
The subcategory form consists of the arithmetic, comparison, and list operations in management law and the corresponding operations in RBSLA syntax, which also can be updated using the add/edit form tag as shown in figure 5-5.



```
in @ <Rel use "Value">In</Rel
<= @ <Rel use "Value">LessE
< @ <Rel use "Value">Less</R
> @ <Rel use "Value">More</R
>= @ <Rel use "Value">MoreE
= @ <Rel use "Value">Equal</R
+ @ <Rel use "Value">Add</Re
- @ <Rel use "Value">Sub</Rel
* @ <Rel use "Value">Mult</Re
/ @ <Rel use "Value">Div</Rel
```

Figure 5-5. Management tool add/edit sub category form.

The management law translator form consists of two panes as shown in figure 5-6. The first one allows for editing and updating management laws; the second one is for presenting the resulting output of any translation into RBSLA.



```
management law RequestML

when MgCt.Request(i,qt)
who MgPr.namePrv() in MgCt.preferListPrv()
at-time MgCt.deadline() >= MgPr.TmRepLm() and
MgCt.OrdHrs () in MgPr.WrkHrs() and
MgCt.RqDay() in MgPr.WrkDays()
manage MgPr.manageRequest()
else mgct.cancel()
```

```
<Rule><label><Atom><Rel>name<:>/Rel><Ind>RequestML</Ind><Rule

<on>
 <Atom>
  <oid><Ind uri= "MgCt"/></oid><Rel use= "effect">Request</Rel><Va
 </Atom>
</on>

<if>
 <And>
  <Atom>
```

Figure 5-6. The management law translator form.

5.6. Chapter Summary

Tackling the quality of (Web) services through management concerns in general has been shown to be essential for the success and wide embracing of the service technology at a large scale [Hil93, CD01, KKL03, MS04, TGR+04]. Handling management concerns in a flexible manner, still at the business level, was the main focus in the previous chapter.

In this chapter, we pushed management laws one step further toward applying them at the technology/implementation level, without compromising the features on which we laid stress: flexibility, separation of concerns. Toward that end, we reviewed the RuleML language elements and then the RBSLA concepts were discussed. More precisely, in our approach, we proposed to move management rules toward RBLSA representation (RuleML-serialization).

For the interfaces of both management and coordination laws, we proposed to translate them to corresponding WSDL-based service interfaces. In this sense, the (management-driven) services are associated with interfaces and also rule-based behavioural (RBSLA-based) rules. This approach has been applied to the PC Selling case study. The translation rules have been presented and illustrated by an example for each clause. Finally, the tool for automatic translation has been designed.

CHAPTER SIX

Chapter 6

Conclusion

As service technology and its Web-Service standards are maturing, world-wide (cross-)organizations are increasingly embracing this technology at a rapid pace. This wide acceptance has already resulted in an abundance of functionality-similar services. One may look just to the "exponential" number of online services for flight tickets, accommodation or even complete vacation packages. Indeed, most existing Web-Service standards (e.g. WSDL for service description or BPEL / WS-CDL for composition) support only functional features in a rigid and static manner.

Nevertheless, to stay competitive in such a globalized and highly volatile economy, organizations opting for this advanced yet emerging service technology are forced to go beyond rigid functionality-based services to attract more requestors and thus increase their benefits. For that aim, the quality of service has been placed at the centre of this technology alongside functionalities. Service availability and throughput, time response to a specific service query, or cost management of services have been, among others, explored to support the quality-of-service in Web-Services. These qualities have been mainly captured as agreements between requestors and providers, in so-called SLA contracts [JM02] (see Chapter Three for an extensive survey).

Despite these advances in providing service requestors with quality-of-service to boost service functionalities and facilitate selecting or composing the right services, we have observed that severe limitations are still hindering the handling of quality-of-service and management concerns in general in service-driven business applications. Among these preoccupations on which this thesis has been contributing, we recall the following. Firstly, we found that most of the qualities addressed are technology-centric low-level ones. For instance, by establishing an agreement between customer and service provider, the two parties are at the start more concerned about deadlines and preferences than low-level time-response and availability. Secondly, we further observed that once specific low-level quality-of-services are agreed on, there is no room to dynamically change or adapt them. Thirdly, there is no transparency on what concerns the way of bringing together such non-functional qualities and service functionalities. In other words, the separation of concerns has been completely missing in existing approaches to the handling of qualities in Web-Services.

These shortcomings, akin to the state-of-the-art handling of management concerns in service-driven applications, have been steering the main objectives of this thesis. More precisely, in this thesis we have been concerned with approaching management concerns at the business and conceptual levels while engineering complex and service-driven business applications. The thesis presents an integrated approach for addressing management concerns covering phases of business requirements, as well as the smooth and compliant mapping of the later phases toward service technology.

The remainder of this conclusion first highlights the main scientific contributions that have been achieved. Secondly, we point out some of the shortcomings of the proposed approach, and then we project how to address them in future investigations.

6.1. Thesis summary and main achievements

With the growing development and acceptance of the service-oriented paradigm, currently developed (cross-)organization business processes and applications are based on service-oriented architecture and enabling online Web services. Although awareness of that central role of qualities has been recognized by both researchers and academia as early as the first release of Web services, the handling of qualities until now remained empowered by and centred on that technology.

As we reported at different occasions in this thesis, this technology-driven handling of qualities and their inherent IT-SLA remains beneficial and essential in pushing toward providing, invoking and/or composing better customized services. Nevertheless, relying on just machine-dependent qualities (e.g., time response, availability, throughput, etc.) presents several severe drawbacks in the quest of developing, providing and/or requesting high quality services. As we presented in the thesis's fourth and fifth chapters, coping with just machine-based qualities implies ignoring the business-level quality-driven requirements inherent in any service-driven business applications at hand. This is a serious limitation as the service-oriented paradigm aims at abstracting from any platform and promoting business applications specificities. While requesting an airline ticket, hotel room, or a PC, for instance, "*the faster*" Web service responses represent only a last selection criteria; instead, we all are interested in qualities like discounts, preferences, flexibility, variability, and so on. Secondly, when it comes to separation of concerns, adaptability and all technology-driven solutions become overcome.

This thesis proposed to leverage the handling of management concerns (i.e., all quality-related issues) to the application level where it should belong, while developing service-driven business applications. We addressed this leveraging in a way that promotes separation of concerns and adaptability. Without delving again into the detail of the approach we described in the previous chapters, we restrict ourselves at this conclusion to reemphasizing the main achievements underlying the proposed approach:

Business characterization of qualities: Through the analysis of several service-driven applications, we proposed ways the qualities should be understood at the application level and their main features. In this respect, we distinguished time-dependent constraints such as agreed-on deadlines, partner preferences, and degree of reputation and trust, among others.

Business activity-based handling of concerns: In contrast to most existing approaches to quality in service-oriented applications that are process coarse-grained- based, we motivated and presented how a more fine-grained activity-based treatment of qualities and other concerns bring more flexibility and mastering of the business application at hand. To be more specific, we recall some of these advantages including: (1) The tractable handling at the activity level of different (management and coordination) concerns, even if the process itself is huge and complex; (2) the wide and optimal use of different resources as the effective request and the release depend only on which activities currently are running; (3) The ability to customize the business process once all activities are addressed (and not before in the fixed way).

ECA-Business rules pattern for qualities: To cope inherently with adaptability and evolution while tackling the characterized business-level qualities, we presented how event-driven (cross-services) business rules represent the most suitable business artefact. We proposed a generic (cross-organizational) ECA pattern for describing intuitively the business-level qualities we referred to as ECA-driven management concerns. We then applied to the PCs Selling case study.

ECA-Business rules pattern for coordination concerns: As we discussed along this thesis, among the main drawbacks of existing approaches to qualities—beside the rigid Web technology-centricity—remain their lack of transparency and tangling of different concerns. To avoid these serious limitations, we proposed, in the same spirit as we did for management concerns to explicitly extract and describe coordination-based functionality concerns through ECA-driven business rules. A simplified pattern inspired from coordination laws proposed within the group then is forwarded.

Conceptualization of ECA rules as management and coordination laws: To bridge the gap between the ECA-driven business level and any WS technology-based implementation, we proposed to move the business description toward service-oriented conceptualization. We proposed for management concerns to consider their corresponding ECA-driven qualities-based rules into suitable behavioural architectural connectors, which we referred to as management laws. In the same spirit, we proposed to recapitulate coordination laws to consider functionality concerns in a precise, yet flexible and evolving, manner.

RBSLA-based WS implementation of both laws: To enhance the practicability and its compliance to Web technology, yet still remain rule-driven, we presented how to move from both management and coordination laws toward

RBSLA-based service implementation. More specifically, while different law interfaces are captured using WSDL standards, the ECA rule-based connectors of both management and coordination laws are translated into RBSLA (and RuleML) rules in a preserving manner.

Conducting of case study on PCsSelling: In chapter four, beside demonstrating the forwarded concepts and primitives of the proposed approach, we carried out the whole PCs Selling business process with all its possible business activities. At the RBSLA level, in chapter five, we also automated all worked out management laws for different business activities of this case study.

Implementation of tools supporting the laws – RBSLA mapping: Since, RBSLA is associated with an advanced language and software environment, we concentrated on how to automate the proposed translating steps from management laws toward it. We implemented a translator to achieve this task and applied it to the case study.

6.2. Shortcomings and projected further work

We have to acknowledge that the "engineering road" toward *effectively serving any requestors with best customized services and on-time* is an aspiration that remains difficult to reach. We consider as barriers at least the following open issues:

- Early validation/verification: Despite the fact that RBSLA supports several (programming) logic-based environments, we are aware that such late testing and validation could be costly and very limited in effectiveness. We suggest achieving formal validation at the early business and conceptual levels, through suitable operational formalisms with supporting tools such as Graph-transformations [HHL04] and Coloured Petri Nets [Jen92], just to name these two we motivate later.
- Supporting environment for all phases: Though we associated the forwarded approach with management laws automated mappings to the RBSLA language that itself is associated with complete tools, this

automation tackles just the last phase in the approach. As will be detailed more hereafter, the business and conceptual phases as main corners in the approach also need to be systemized by adequate tools as we described below.

- Runtime adaptability: With ECA business rules and their governing conceptual management and coordination laws, we showed how *design time* adaptability and evolution are assured by construction when adopting our approach. Unfortunately, when it comes to adaptability on-the-fly, the approach needs to be upgraded significantly. We suggest some directions and ideas to bring such runtime adaptability to the approach.
- WS Standards adaptation/extension: Although we presented benefits from the WSDL standard and showed how to capture RBSLA rules as behavioural-intensive Web services, we could not develop the composition further using BPEL standard. We report on a possible vision on enriching BPEL with qualities as we developed in this thesis.

In the following, we analyse these open research directions one by one while suggesting focussed ideas and fresh visions on how they could be tackled by me at the postdoctoral level and/or by (collaborating with) any interested researcher(s)/ practitioner(s).

6.2.1. Early formal validation / verification

The conceptualisation of management/coordination laws is a milestone for ensuring preciseness and rigor. Nevertheless, to detect inconsistencies, conflict, or even misconceptions, we require the execution of such conceptual modelling at an abstract level; that is, before investing in any specific WS-based deployment. In such a way, time and cost will spared, yet effectiveness and reliability will be scored.

For management laws, which are ECA-driven (transitional) rule-based architectural connectors, several operational and executable semantics for governing them may be potential candidates. Herewith, I sketch just two possibilities: adopting Coloured Petri nets [Jen92] and graph transformation [HHL04]. Petri nets in general and coloured Petri nets in particular are graphical, coping with both type and instance level. They are associated with advanced software tools for editing and simulating any conceptual model. They support good analysis techniques, such as reachability, invariants, and even temporal properties. They have been adopted for Web services specification and verification [YK04].

The interpretation of management laws to CPNets seems at first glance to be straightforward, such as events/messages to corresponding places, conditions to transition conditions, and actions to output arcs. The second complement to CPNets possible operational rule-based semantics is graph transformations, by which nodes are states and rules are transformations. First attempt interpretation could consist of capturing law interfaces as nodes and laws themselves as rule transformations.

We should reemphasize that these possibly could be executable formal interpretations of management laws, and others—such as temporal semantics, state machine, and rewriting techniques—could also be adopted depending on the profile of the investigator.

6.2.2. Integrated tools supporting the approach

Though we implemented a first prototype for automating the translation of management laws toward RSBLA, as well as the adoption of the advanced RBSLA environment, we should recognize that much work remains ahead toward full automated support of the approach. In this direction, we particularly suggest that the following complementary tooling software could boost the practicability and the wide embracing of the approach significantly:

- Graphical editor and analyser for ECA business rules. This tool should automate the description of management and coordination of ECA-based rules.
- Tool for manipulating and evolving management laws. This tool first should be able to translate any automated ECA-driven rule for

management (from the above tool) into corresponding management laws. Then, we should be able to evolve/adapt such laws in a systematic and graphical manner. The group development environment for CommUnity [FL03, WO04] could be a possible source of inspiration.

- Tool supporting validation/verification. As we developed in the previous open issue, depending on the adopted executable formal interpretation, associated tools either should be developed from scratch or adapted from existing ones.

6.2.3. Runtime adaptability with techniques akin to AOP

As we pointed out, the approach we proposed permits coping with adaptability at design time and in a constructive manner. Nevertheless, in some service-driven applications, it is highly required to enforce any management/coordination laws at runtime time. Since aspect-oriented techniques are the most emerging techniques for coping with such adaptability on-the-fly, we envision adopting this paradigm in the future for leveraging the approach to deal with runtime evolution.

Indeed, aspect-oriented programming (AOP) was forwarded first by [KLM+97], as a consequence to the limitations of the object paradigm in factoring out cross-cutting concerns (e.g., Persistence, Management, Security, etc.). AOP allows extracting cross-cutting concerns from different code units (e.g., components, modules, or classes) and externalizing them in so-called *advices*, as encapsulated behavioural units ready to be "injected" into specific positions in concerned units. While the right positions, where these advices have to be *woven*, are referred to as joinpoints, the different ways of combining such advices before superposing them on respective units is referred to as pointcuts. All these primitives and mechanisms have been implemented for the first time on the AspectJ language [KHH+01].

Concerning the explicit and dynamic handling of business rules as advices coupled by non-intrusive weaving, the JasCo language [SVJ03] remains the most suitable. Moreover, this language has been leveraged to cope with dynamic multi-concerns in Web Services through a variant called WSML [KL03].

We argue that to cope with runtime adaptability, this JasCo language could be a good starting point for such investigations. A possible way to achieve that could consist of capturing our management laws as advices and dynamically weaving them onto corresponding activities. But, of course, more and deeper explorations are required to realize such moving effectively.

6.2.4. Composition through standards extensions

We presented how activities, after being modelled using management and coordination laws, are put together flexibly to build any business process at hand. Nevertheless, what we could not develop further concerns the ability of composing RBSLA rules as activity-centric Web services into more complex BPEL-like business processes. That is, though we hinted that RBSLA rules should be regarded as "behavioural" Web services, the main remaining open question is how to compose them to build processes using the BPEL or WSCI standards.

As a first approach to this question, we think that by capitalizing on the above aspect-oriented techniques, we could achieve that process-centric composition by extending BPEL in this respect. We should be inspired by the work, for instance, on AO4BPEL [CM04, CM07], which allows weaving aspects as rules on BPEL specifications. However, due to the complexity of our management and coordination rules, deep investigations are required on this interesting research stand.

6.2.5. Consideration of further concerns with security on top

Last but not least, we claim that after this separation of management concerns along all "business conceptual-deployment" development phases, the same experience could be applied straightforwardly to other concerns. Among these potential and pressing concerns, we cite in particular the security in service-driven business applications. Indeed, different recent explorations present that security also is more a business matter than a technical issue. Role-based access, access through trust, and so on, are just snapshots confirming this trend [MSS+04]. In other words, security concerns are governed by evolving policies that depend on customer activities, profiles, and surrounding environments.

6.3. Closing remarks

Before concluding this thesis, we would like to reemphasize some important scientific lessons we learned during this thesis. More precisely, we argue that the following lessons are of great advantages towards pushing forwards this emerging service-oriented paradigm. The first lesson is really “separate concerns”. Indeed, although separation of concerns is one of the main goals in software-engineering, we found that in service-driven applications it is far from being respected.

For instance, as we already pointed out, when tackling qualities, even concerns such as basic functionalities either are ignored or mixed up with qualities. The second lesson is “business and business and business”; that is, to tackle any service-driven business application, we argue that to result in flexible, evolving, and sustainable implementation, we have to put all efforts on the early phases of business and inherent conceptualization. Finally, business rules could be essential in the handling of any concerns. All that is required is to adjust them judiciously.

Bibliography

[ACD+04] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S. and Xu, M.: *Web Services Agreement Specification (WS- Agreement), Version 1.1(Draft 20),* 2004.

[ACK04] Alonso, G., Casati, F., Kuno, H. and Machiraju, V.: *Web Services - Concepts, Architectures and Applications.* 354 pages, 2004. Springer Verlag.

[AF01a] Andrade, L.F. and Fiadeiro, J.L.: Coordination Technologies for Managing Information System Evolution. In *Advanced Information Systems Engineering, LNCS,* 2068: pages 374–387, 2001. Springer Verlag.

[AF01b] Andrade, L.F. and Fiadeiro, J.L.: Coordination: The Evolutionary Dimension. In *Technology of object-oriented languages and systems; Tools 38 Components for Mobile Computing*: pages 136–147, 2001. Zurich, Switzerland: IEEE Computer Society.

[AF02] Andrade, L.F. and Fiadeiro. J.L.: Coordination Architecture for Evolvable Event- Based Systems. In *International conference on distributed computing systems workshops*: pages 571–572, 2002. Vienna: IEEE Computer Society.

[AF03] Andrade, L.F. and Fiadeiro, J.L.: Service-Oriented Business and System Specification: beyond object-orientation. In H. Kilov and K. Baclwaski (eds), *Practical Foundations of Business and System Specifications*: pages 1–23, 2003. Kluwer Academic Publishers.

[AF06] Al-Ghamdi, A. and Fiadeiro, J.L.: Architectural Handling of Management Concerns in Service-Driven Business Processes. *MSVVEIS 2006*: pages 111–120, 2006.

[AFG+02] Andrade, L.F., Fiadeiro, J.L., Gouveia, J., Koutsoukos, G. and Wermelinger, M.: *Coordination for Orchestration, LNCS,* 2315: pages 5–13, 2002. Springer-Verlag.

[AFL+03] Andrade, L.F., Fiadeiro, J.L., Lopes, A. and Wermelinger, M.: Coordination for Distributed Business Systems. In J. Eder, R. Mittermeir and B. Pernici (eds.), *Proc. of Information Systems for a Connected Society*: pages 27–37, 2003. University of Maribor Press.

[AFP07] Al-Ghamdi, A., Fiadeiro, J.L. and Paschke, A.: RBSLA based Implementation for Architectural Management Laws. *4th International IEEE Conference on Innovations in Information Technology (Innovations'07), Dubai, United Arab Emirates*: pages 556–560, 2007.

[AFW01] Andrade, L.F., Fiadeiro, J.L. and Wermelinger, M.: Enforcing Business Policies through Automated Reconfiguration. In *Proc. of the 16th Intl. Conf. on Automated Software Engineering*: pages 426–429, 2001. Los Alamitos: IEEE Computer Society Press.

[AG97] Allen, R. and Garlan, D.: A Formal Basis for Architectural Connection. *ACM Transactions on Software Engineering and Methodology*, 6(3): pages. 213–249, 1997.

[Ars01] Arsanjani, A.: *A Pattern Language for Adaptive Manners and Scalable Business Rule Design and Construction*: pages 370, 2001. IEEE.

[ARW+03] Al-Ali, R., Rana, O., Walker, D., Jha, S. and Sohail, S.: G-QoSM: Grid service discovery using QoS properties. *Computing and Informatics Journal, Special Issue on Grid Computing,* 21(5), 2003. http://www.cse.unsw.edu.au/~nrl/pub/papers/cijnl.pdf

[Ash94] Ashcroft, M. (Ed.): Service Level Agreements. *Proceedings of a Workshop held in Stanford, Lincolnshire, 25 May*, 1994. Capital Planning Information Ltd.

[BCT+03] Benatallah, B., Casati, F., Toumani, F. and Hamadi, R.: *Conceptual Modeling of Web Service Conversations*, *LNCS*, 2681: pages 448–467, 2003.

[BHH00] Barroca, L., Hall, J. and Hall, P.: An introduction and history of software architectures, components, and reuse. In L. Barroca, J. Hall and P. Hall (eds.), *Software architectures: Advances and applications*: pages 1–11, 2000. London: Springer.

[BHL01] Berners-Lee, T., Hendler, J. and Lassila, O: The Semantic Web. *Scientific American*, 284(5): pages 34–43, 2001.

[BK05] Bajec, M. and Krisper, M. Methodology and Tool Support for Managing Business Rules in Organisations. *Information Systems*, (30), pages 423–443, 2005.

[Bol06] Boley, H.: The rule-ml family of web rule languages. In *4th Int. Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro*, 2006.

[BPT98] Belaid, D., Provenzano, N. and Taconet, C.: Dynamic Management of CORBA Trader Federation. *4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*, 1998. http://www.usenix.org/publications/library/proceedings/coots98/full_papers/belaid/belaid.pdf

[BRG00] The Business Rules Group: *Defining Business Rules - What Are They Really?* http://www. businessrulesgroup.org/first paper/br01c0.htm, July 2000.

[CC03] Cox, A. and Chicksand, L.: The Impact of the Internet on Marketing and Sales. In R. ul-haq (ed.), *QMRIJ Report*, 2003. The Birmingham Business School, The University of Birmingham.

[CCD+03] Castellanos, M., Casati, F., Dayal, U. and Shan, M.C.: Intelligent Management of SLAs for Composite Web Services, *LNCS*, 2822: pages 158–171, 2003.

[CD01] Cox, J. and Dale, B.G.: Service Quality and E-commerce: an exploratory analysis. *Managing Service Quality*, volume 11, No.2: pages 121–131, 2001.

[CDK+02] Curbera, F., Duftler, M., Khalaf, R., Nagy, W. and Mukhi, S.W.N.: Spotlight-Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 3(4), 6, pages 86–93, 2002.

[CDS+03] Cibrán, M.A., D'Hondt, M., Suvée, D., Vanderperren, W. and Jonckers, V.: JAsCo for Linking Business Rules to Object-Oriented Software. In *Proc.CSITeA, Rio De Janeiro, Brazil*: pages 1–7, 2003.

[CGM+04] Chinnici, R., Gudgin, M., Moreau, J.-J., Schlimmer, J. and Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0 part 1: Core language. *W3C Working Draft*, 2004.

[CGS01] Casati, F., Georgakopoulos, D. and Shan, M.C.: Technologies for E-Services. *2nd International Workshop on Technologies for E-Services, LNCS*, 2193: pages 16–29, 2001.

[CHT03] Channabasavaiah, K., Holley, K. and Tuggle, E.M.: *Migrating to a service-oriented architecture*, 2003. http://www-106.ibm.com/developerworks/library/ws-migratesoa. (Last accessed: 17 Jan. 2008).

[CLS+03] Chen, Z., Liang-Tien, C., Silverajan, B., and Bu-Sung, L.: UX -An Architecture Providing QoS-Aware and Federated Support for UDDI. *The 2003 International Conference on Web Services (ICWS'03), Las Vegas, Nevada, USA,* June 2003, http://www.ntu.edu.sg/home5/PG04878518/Articles/ICWS03_Paper.pdf

[CM04] Charfi, A. and Mezini, M.: Hybrid Web Service Composition: Business Processes Meet Business Rules. In *Service Oriented Computing*; *ICSOC04*: pages 30–38, 2004. New York, NY: ACM.

[CM07] Charfi, A., and Mezini, M.: AO4BPEL: An Aspect-oriented Extension to BPEL. *World Wide Web Journal: Recent Advances on Web Services (special issue)*, 10: pages 309–344. 2007.

[CSD+03] Casati, F., Shan, E., Dayal, U. and Shan, M.C.: Business-Oriented Management of Web Services. *Communications ACM*, 46: pages 55–60, 2003.

[CSS+85] Czepiel, J.A., Solomon, M.R., Surprenant, C. and Gutman, E.G.: Service Encounters: an Overview. In Czepiel, J.A., Solomon, M.R. and Surprenant, C.F., *The Service Encounter*, Chapter 1, 1985. Lexington, MA: Lexington Books.

[Dat00] Date, C.J.: *What not How: The Business Rules Approach to Application Development*: pages 144, 2000. Addison-Wesley Publishing Company.

[DDK+04] Dan, A., Davis, D., Kearney, R., Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M. and Youssef, A.: Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, 43: pages 136–158, 2004.

[DFG98] Dib, N., Freer, J. and Gray, C.: Service-level agreements at the Huddersfield NHS Trust. *International Journal of Health Care Quality Assurance*, volume 11, No.3: pages 96–101, 1998.

[DHP02] Diao, Y., Hellerstein, J.L. and Parekh, S.: Using fuzzy Control to Maximize Profits in Service Level Management. *Systems Journal*, volume 41, No. 3: pages 403–420, 2002.

[Dou03] Barry, D.K.: *The Savvy Manager's Guide to Web Services and Service-Oriented Architectures*, 2003. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[EFB01] Elrad, T., Filman, R.E. and Bader, A.: Aspect-Oriented Programming. *Communications of the ACM*, 44(10): pages 29–32, 2001.

[EGR89] Edwardson, B., Gustavsson, B.O. and Riddle, D.I.: An Expanded Model of the Service Encounter with Emphasis on Cultural Context. *Research Report, volume 89, No. 4,* 1989. CTF Research Centre, University of Karlstad.

[Fia02] Fiadeiro, J.L.: Coordination Technologies for Just-in-Time Integration. In United Nations University, *International Institute for Software Technology; Formal methods at the crossroads from panacea to foundational support 10th anniversary colloquium of UNU/IIST*, 2002. Lisbon: Berlin.

[Fia03] Fiadeiro, J.L.: Service-oriented business and system specification: beyond object-orientation. In H. Kilov and K. Baclwaski (eds.), *Practical Foundations of Business and System Specifications*: pages 1–23, 2003.

[FK02] Farrell, J.A. and Kreger, H.: New Developments in Web Services and E-commerce Web Services management approaches. *IBM System Journal*, Volume 41, Number 2: pages 212–227, 2002. Available at. http://www.research.ibm.com/journal/sj/412/farrell.html. (Last accessed: 12 Jan. 2008)

[FM97] Fiadeiro, J.L. and Maibaum, T.: Categorical Semantics of Parallel Program Design. *Science of Computer Programming* 28: pages 111–138, 1997.

[GC92] Gelernter, D. and Carriero, N.: Coordination Languages and their Significance. *CACM* 35(2): pages 97–107, 1992.

[Git03] Gittlen, S.: Getting to the Heart of SLM, *Editorial, Network World.* Available at: www.nwfusion.com, January 2003.

[GKW+02] Gouveia, J., Koutsoukos, G., Wermelinger, M., Andrade, L.F. and Fiadeiro, J.L.: *The Coordination Development Environment, LNCS,* (2306): pages 323–326, 2002. Springer-Verlag.

[GR04] Governatori, G. and Rotolo, A.: Modelling Contracts Using RuleML. In T. Gordon, (ed.). *Jurix 2004: The Seventeenth Annual Conference*, December, 2004.

[GR05] Gartner Group: Plummer, D.C.: *Gartner's Positions on the Five Hottest IT Topics and Trends in 2005*. www.gartnergroup.com, accessed 2005-05-12. NATIS, Y.: Service-Oriented Architecture Scenario, 2003-04-16: AV- 19-6751 http://www.gartner.com/resources/114300/114358/114358.pdf, 2006-12-16.

[Gra05] Graham, I.: *Service Oriented Business Rules Management Systems*. TriReme International Ltd, 2005.

[HC07] Heckel, R. and Cherchago, A.: Structural and behavioural compatibility of graphical service specifications. *Journal of Logic and Algebraic Programming*, Volume 70, Issue 1: pages 15–33, 2007.

[HHL04] Hausmann, J.H., Heckel, R. and Lohmann, M.: Model-based Discovery of Web Services. *ICWS 2004*: pages 324–331, 2004.

[Hil93] Hiles, A.: *Service Level Agreements - Measuring Cost and Quality in Service Relationships*, 1993. London: Chapman & Hall.

[IEC04] International Engineering Consortium: *Service-Level Management Definition and Overview*, available at: http://www.iec.org/online/tutorials/service_level, 2004.

[Jen92] Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and practical Use. Volume 1: Basic Concepts. *EATCS Monographs in Computer Science*, 26, 1992.

[JM02] Myerson, J.: 01 Apr 2002 Updated 29 Oct 2004: *Use SLAs in a Web Services context, Part 1: Guarantee your Web service with a SLA.* Available at http://www-128.ibm.com/developerworks/library/ws-sla/

[Jos06] Joseph, B.: Service Oriented Architecture (SOA) A New Paradigm to Implement Dynamic E-business Solutions. *Ubiquity* 7, (30), 2006. http://www.acm.org/ubiquity/views/pf/v7i30_soa.pdf (Last accessed: 01Jan. 2008).

[Kar04] Carter, F. Apr. 5, 2004, *Managing The Bottom Line, Service-level management is essential to deploying business-critical Web Services systems*, Available at http://webservices.sys-con.com/read/44352.htm?CFID=261718&CFTOKEN=10C34C34-F49A-9338-5665FD5131A63A4D

[Kat93] Katz, S.: A Superimposition Control Construct for Distributed Systems. *ACM Transactions on Programming Languages and Systems*, 15(2): pages 337–356, 1993.

[Kay02] Kaye, D.W.: *Strategies for Web Hosting and Managed Services*, 2002. Wiley.

[KHH+01] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. and William G. Griswold.: An Overview of AspectJ. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'01)* LNCS 2072: pages 327–355, 2001.

[KKL+02] Keller, A., Kar, G., Ludwig, H., Dan, A. and Hellerstein, J.L.: Managing Dynamic Services: A Contract Based. Approach to a Conceptual Architecture. *IEEE/IFIP network operations and management symposium, Florence, Italy,* 2002. IEEE.

[KKL03] Kalepu, S., Krishnaswamy, S. and Loke, S.: Verity: A QoS Metric for Selecting Web Services and Providers. *1st Web Services Quality Workshop (WQW 2003),* in conjunction with *IEEE Computer Society 4th International Conference on Web Information Systems Engineering (WISE 2003), Rome, Italy,* December 2003. http://alarcos.inf-cr.uclm.es/wqw2003/kalepu%20ABSTRACT.pdf

[KL02] Keller, A. and Ludwig, H.: Defining and Monitoring Service Level Agreements for Dynamic e-Business. *Systems Administration Conference, 2002.* Philadelphia, PN: USENIX Association.

[KL03] Keller, A. and Ludwig, H.: 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): pages 57–81. 2003.

[KL04] Kardasis, P. and Loucopoulos, P.: Expressing and organising business rules. *Information and Software Technology,* 46(11): pages 701–718, 2004

[KL05] Kardasis, P. and Loucopoulos, P.: A Roadmap for the Elicitation of Business Rules in Information Systems Projects. *Business Process Management Journal,* 11(4): pages 316–348, 2005.

[KLM+97] Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J-M. and Irwin, J.: Aspect-Oriented Programming. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP), LNCS,* pages 220–242, 1997.

[LKD03] Ludwig, H., Keller, A., Dan, A., King, R. and Franck, R.: A Service Level Agreement Language for Dynamic Electronic Services. *Electronic Commerce Research,* 3(1/2): pages 43–59, 2003.

[LSE03] Lamanna, D.D., Skene, J. and Emmerich, W.: SLAng: A language for defining Service Level Agreements. *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03), San Juan, Puerto Rico*: pages 100, 2003.

[Lud02] Ludwig, H.: *Web Service Level Agreement (WSLA) Language Specification.* IBM T.J. Watson Research Centre, IBM Corporation, 2001, 2002, 2003. http://www.research.ibm.com/wsla, (Last accessed: January 20, 2008).

[LVA02] Li-jie Jin, Vijay Machiraju and Akhil Sahai: *HP Software Technology Laboratory,* 21 June 2002. Available at http://www.hpl.hp.com/techreports/2002/HPL-2002-180.pdf

[MB03] Meredith, L.G. and Bjorg, S.: Contracts and Types. *Communications-ACM,* 46(10): pages 41–48, 2003.

[MK96] Magee, J. and Kramer, J.: Dynamic structure in software architectures. *Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE4), ACM Software Engineering Notes*: pages 3–14, San Francisco, October 1996.

[MN02] Mani, A., and Nagarajan, A.:Understanding quality of service for web services. http://www-106.ibm.com/developerworks/webservices/library/wsquality.html,2002.

[MN05] Mendling J., and Nüttgens, M.: A Comparison of XML interchange formats for business process modelling. In: Fischer L (ed) Workflow handbook. Future strategies, pages 185–198. 2005.

[MPM+04] Martin, D.L., Paolucci, M., Mcilraith, S.A., Burstein, N.H., Ncdermott, D.V., Mcguinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K.P. (eds.): *Bringing Semantics to Web Services: The OWL-S Approach,* volume 3387 of Lecture Notes in Computer Science, 2004. Springer.

[MPX04] McKenzie, F.D., Petty, M.D. and Xu, Q.: Usefulness of Software Architecture Description Languages for Modeling and Analysis of Federates and Federation Architectures. *SIMULATION* 80: pages 559–576, 2004.

[MS04] McConnell, J. and Siegel, E.: *Practical Service Level Management: Delivering High-Quality Web-Based Services*, 2004. Cisco Press.

[MSS+04] Mendling, J., Strembeck, M., Stermsek, G., and Neumann, G.: An Approach to Extract RBAC Models from BPEL4WS Processes. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'04).* IEEE Computer Society: pages 81–86, 2004.

[Oas07] OASIS.: *UDDI Specification,* http://www.uddi.org/specification.html, Copyright (c) OASIS Open 2006. All Rights Reserved. (Last accessed: 17 Apr. 2007).

[OYP03] Orriens, B., Yang, J. and Papazoglou, M.: A Framework for Business Rule Driven Service Composition, *LNCS,* 2819, 14–27, 2003.

[Pap03] Papazoglou, M.P.: Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*: pages 3–12, 2003.

[Pas05] Paschke, A.: RBSLA - A declarative Rule-based Service Level Agreement Language based on RuleML. *International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC 2005), Vienna, Austria*, 2005.

[Pas07] Paschke, A.: *Rule-Based Service Level Agreements - Knowledge Representation for Automated e-Contract, SLA and Policy Management*, 2007. Munich: IDEA Verlag GmbH.

[PDK05] Paschke, A., Dietrich, J. and Kuhla, K.: A Logic Based SLA Management Framework. *Semantic Web and Policy Workshop (SWPW), 4th Semantic Web Conference, Galway, Ireland*, 2005.

[Pio02] Piotr SZYMCZYK.: DEVELOPING SERVICE LEVEL AGREEMENT FOR OUTSOURCED PROCESSES AND SYSTEMS. *International Carpathian Control Conference ICCC' 2002.*

[PKB07] Paschke, A., Kozlenkov, A. and Boley, H.: A Homogenous Reaction Rules Language for Complex Event Processing. *International Workshop on Event Drive Architecture for Complex Event Process (EDA-PS 2007), Vienna, Austria*, 2007.

[Pra03] Pratt, K.T.: Introducing a Service Level Culture. *Facilities*, volume 21, No.11/12: pages 253–259, 2003.

[PW92] Perry, D.E., and Wolf, A.L.: Foundation for the Study of Software Architectures. *ACM SIGSOFT Soft. Engineering Notes* 17(4): pages 40–52, 1992.

[QDS04] Quartel, D., Dijkman, R. and van Sinderen, M.: Methodological Support for Service-oriented Design with ISDL. In *Service Oriented Computing; ICSOC 04*: pages 1–10, 2004. New York, NY: ACM.

[Ran03] Ran, S.:A Model for Web Services Discovery With QoS. *ACM SIGecom Exchanges,Vol. 4, No. 1,* pages. 1–10., 2003.

[RD05a] Rosenberg, F. and Dustdar, S.: Business Rules Integration in BPEL - A Service-Oriented Approach. In *Proceedings of the 7th International IEEE Conference on E– CommerceTechnology (CEC'05)*, 2005.

[RD05b] Rosenberg, F. and Dustdar, S.: Towards a Distributed Service-Oriented Business Rules System. In *Proceedings of the 3stIEEE European Conference on Web Services (ECOWS'05)*, 2005.

[RND05] Rosenberg, F., Nagl, C. and Dustdar, S.: Applying Distributed Business Rules - The VIDRE Approach. *IEEE SCC 2006*: pages 471–478, 2006.

[Rul05] RuleML. :*The Rule Markup Initiative*, 2005. http:// www.ruleml.org/ (accessed March 2006).

[RW02] Rosca, D. and Wild, C.: Towards a flexible deployment of business rules. *Expert Systems with Applications,* 23(4): pages 385–394, 2002.

[SDM02] Sahai, A., Durante, A. and Machiraju, V.: Towards automated sla management for web services.
Hewlett-Packard Labs Technical Report HPL-2001-310 (R.1), 2002.

[SG96] Shaw, M. and Garlan, D.: *Software architecture: Perspectives on an emerging discipline,* 1996. Upper Saddle River, NJ: Prentice Hall.

[SJ04] Sedighi, A. and Johnson, E.: *Classification of the Current Constraint and Capabilities Protocols in Describing Web Services,* 2004. Palo Alto: TIBCO Software.

[SLE04] Skene, J., Lamanna, D. and Emmerich, W.: Precise Service Level Agreements. In *26th International Conference on Software Engineering IEEE: ICSE 2004*: pages 179–188, 2004. Edinburgh, Scotland: Los Alamitos Calif.

[SN00] Schneider, J. and Nierstrasz, O.: Components, scripts, and glue. In L. Barroca, J. Hall and P. Hall, *Software architectures: Advances and applications*: pages 13–25, 2000. London: Springer.

[Soa01] *SOAP Version 1.2 Working Draft, World Wide Web Consortium (July 2001),* http://www.w3.org/TR/2001/WD-soap12-20010709/. (Last accessed: May 31, 2007)

[SRA+03] ShaikhAli, A., Rana, O.F., Al-Ali, R. and Walker, D.W.: UDDIe: An Extended Registry for Web Services, *2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops), Orlando, Florida,* January 2003, http://csdl.computer.org/comp/proceedings/saint-w/2003/1873/00/18730085abs.htm

[SVJ03] Survee, D. and Vanderperren, W., and Jonckers, V.: JAsCo: an Aspect-Oriented Approach Tailored for Component based Software Development. In *Proc. of 2nd international conference on Aspect-oriented software development (AOSD'03),* pages 21–29, 2003. ACM Press.

[TA03] TeleCom Asia.: Service Level Management: *End User's Drive Need to Monitor QoS and Content.* Available at: www.telecomasia.net, 2003.

[TAA+01] Tsur, S., Abiteboul, S., Agrawal, R., Dayal, U., Klein, J. and Weikum, G.: Are Web Services the Next Revolution in e-Commerce? In *Proc.of the 27th International Conference on Very Large Data Bases*: pages 633–636, Rome, Italy, September 2001.

[TGR+04] Tian, M., Gramm, A., Ritter, H. and Schiller, J.: *A Survey of current Approaches towards Specification and Management of Quality of Service for Web Services.* Freie Universität Berlin, Institut für Informatik Takustr. 9, D-14195 Berlin, Germany. Available at http://page.mi.fu-berlin.de/~tian/pdf/tian_et_al_wsqos_approaches_PIK032004.pdf

[TGR04] Tian, M., Gramm, A., Ritter, H. and Schiller, J.: *Efficient Selection and Monitoring of QoS-aware Web Services with the WS-QoS Framework.* Available at http://page.mi.fu-berlin.de/~hritter/publications/TGRS04.pdf

[TPP+03] Tosic, V., Pagurek, B., Patel, K., Esfandiari, B. and Ma, W.: Management Applications of the Web Service Offerings Language (WSOL). In *Advanced Information Systems Engineering, LNCS,* 2681: pages 468–484, 2003.

[TSP+04] Tsai, W.T., Song, W., Paul, R., Cao, Z. and Huang, H.: Services-Oriented Dynamic Reconfiguration Framework for Dependable Distributed Computing. In *Proc. of the IEEE COMPSAC 2004*: pages 554–559, 2004.

[TW01] Taveter, K. and Wagner, G.: Agent-Oriented Enterprise Modeling Based on Business Rules. In *Proceeding of the 20^{Th} International Conference on Conceptual Modelling (ER'01): pages* 527–540, 2001.

[TWB03] Tabet, S., Wagner, G. and Boley, H.: MOF-RuleUML: The Abstract Syntax of RuleML as a MOF Model. In N. Lenehan, (ed.), *Integrate*, 2003.

[UDD07] *Universal Description, Discovery, and Integration.* UDDI.org Consortium, http://www.uddi.org. (Last accessed: May 31, 2007)

[UM00] Nilsson, U. and Maluszynski, J.: *Logic, Programming and Prolog (2 ed.),* 2000. John Wiley & Sons Ltd.

[Von01] Von Halle, B.: *Business Rules Applied: Building Better Systems Using the Business Rules Approach,* 1st edition, 2001. Wiley& Sons Inc.

[W301] W3 Consortium. : *Web Services Activity Statement,* available at: www.w3.org/2002/ws/Activity, 2001.

[W307] World Wide Web Consortium (W3C): *Simple Object Access Protocol v1.2 (SOAP),* 2007.

[Wag02] Wagner, G.: How to design a general rule mark-up language? In *Workshop XML Technologien fuer das Semantic Web (XSW), Berlin,* June 2002.

[WCL+05] Weerawarana, S., Curbera, F., Leymann, F., Storey, T. and Ferguson, D.F.: *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More,* 2005. Prentice Hall PTR.

[Web00] *Web Services architecture overview - the next stage of evolution for e-business,* http://www.IBM.com/developerworks/web/library/w-ovr/, September 2000.

[Web01] *Web Services Description Language (WSDL) 1.1 W3C Note, World Wide Web Consortium (March 2001),* http://www.w3.org/TR/wsdl.

[Web08] Web services architect, Part 2: *Models for dynamic-business.* http://www-106.ibm.com/developerworks/webservices/library/ws-arc2.html (Last accessed: Jan 01, 2008)

[WL04] Wan-Kadir, W.M. and Loucopoulos, P.: Relating evolving business rules to software design. *Journal of Systems Architecture,* 50 (7): pages 367–382, 2004

[WO04] Wermelinger, M., and Oliveira, C.: The CommUnity Workbench. *ICSE*, pages 709–710, 2004.

[Wus02] Wustenhoff, E. (Editor): *Service Level Agreement in the Data Center*, 2002. Sun Microsystems Inc.

[YK04] Yi, X. and Kochut, K.J.: A CP-nets-based Design and Verification Framework for Web Services Composition. In *Proceedings of 2004 IEEE International Conference on Web Services*: pages 756–760, 2004. IEEE Computer Society.

[YTS+08] Yuhas, J., Turner, D., Speake, A., et al.: *Program Manager, Microsoft Corporation, Service Management Functions, Service Level Management.* Available at http://www.microsoft.com/technet/itsolutions/cits/mo/smf/smfslamg.mspx, 2008.

[ZBN+04] Zeng, L., Benatallah, B., Ngu, A.H. and Dumas, M.: QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5): pages 311–327, 2004.

[Zho06] Zhou, N.: Programming finite-domain constraint propagators in action rules. Theory and Practice of Logic Programming (TPLP), 6(5): pages 483–508, 2006.

[ZLB04] Zirpins, C., Lamersdorf, W. and Baier, T.: Flexible Coordination of Service Interaction Patterns. In *Service Oriented Computing; ICSOC 04*: pages 49–56, 2004. New York, NY: ACM.

Appendix A

List of Abbreviations

ACT	▶	Activity
ADLs	▶	Architecture Description Languages
AOP	▶	Aspect-Oriented Programming
BPEL	▶	Business-Process Execution Language
BR	▶	Business Rule
BRs4ACT	▶	Business Rules for Activity
BS	▶	Banking Service
CORBA	▶	Common Object Request Broker Architecture
CS	▶	Customer Service
DCOM	▶	Distributed Component Object Model
ECA	▶	Event Condition Action
Funct	▶	Function
HTTP	▶	Hyper Text Transfer Protocol
IntegConcerns	▶	Integration of Concerns
Jess	▶	Java Expert System Shell
KR	▶	Knowledge Representation
Manag	▶	Management
NSP	▶	Network Service Provider
PS	▶	Provider Service
QoS	▶	Quality-of-Service

RBSLA	▶	Rule Based Service Level Agreement
SLA	▶	Service-level Agreements
SOA	▶	Service-Oriented Architecture
SOAP	▶	Simple Object Access Protocol
SOC	▶	Service Oriented Computing
SP	▶	Service Provider
SS	▶	Shipment Service
UDDI	▶	Universal Description, Discovery, and Integration
UML	▶	Unified Modeling Language
ViDRE	▶	Vienna Distributed Rule Engine
WS	▶	Web Services
WSDL	▶	Web Services Description Language
WSLA	▶	Web Service Level Agreement
WSOL	▶	Web Service Offering Language

Appendix B

Glossary

Activity: a process comprises two types of rules: an interaction rule and a management rule
Banking services (BSs): This service also is crucial as it interacts with all other services to accomplish any required payment.
Business activity Any task related or taking part in a given business process
Business processes Inter-related partially-ordered business activites defining a specific process
Business rule Constraints and policies for doing business.
Business SLAs refer to agreements on how a specific service is delivered, and to the semantics of the service rather than to system or application level metrics
Cancellation: This optional business activity is performed when the customer wants just a part of the requested quantity and more precisely accepts part of the offer. Such a cancellation is possible only before delivering the initial request; otherwise, penalties are applied.
Coordination Laws Set of primitives in terms of behavioural architectural connectors. It is composed of (1) interfaces with required events, messages and attributes to participate in an interaction; and (2) rules expressed in the form ECA.
Coordination the business rules that specify how the entities can interact
Customer services (CS): This provides PC Selling with a front end that handles interactions with the customers (i.e., end-users). That is, CS allows customers to post their requests, buy PCs, pay their dues, and cancel/accept offers.
Delivery: This activity begins when the customer accepts the proposed offer. It notifies the provider to start delivering the PCs agreed on.
ECA business rules On the occurrence events (E), check the holding of constraints (C) and then perform related actions (A)
High-level management concerns Deadlines, preferences, trust
Low-level management qualities Availability, time-response, throughput, cost

Management laws Similar in form to coordination laws but focuses on the management concerns. In the rule we have clauses such as: (1) **At-time** for deadlines and **Who** for preferences
Offer: In reply to the customer's request, this activity consists of an offer from the provider, including some necessary details such as the price.
Payment: This activity deals with the execution of all the different aspects of paying, refunding, penalties, etc.
PCs-Selling case-study This is the running case-study of this thesis. It concerns a service-oriented business applications dealing with the online-selling of PCs to customers
Phase-ACT It is the first phase in our conceptual-model to management and interaction concerns. It concerns the detailed informal description of any activity in a given (service-oriented) business process
Phase-BRs4ACT It is the second phase and allows the informal definition of any business rules governing the behaviour of a given business activity
Phase-ConcernsBPs This last phase at the conceptual-level allows inter-relating different (multi-concern) activities to form the complete business process
Phase-Func.BRs&Laws This phase extracts the disciplined formulation of functionality concerns in any informal business rule in terms of coordination laws
Phase-IntegConcerns@ACT This phase addresses the integration of both functionalities and management concerns. That is, for a given business activity we integrate all coordination and management laws to reflect the activity behaviour (at the running configutation)
Phase-Manag.BRs&Laws This phase extracts the disciplined formulation of management concerns in any informal business rule in terms of management laws
Provider services (PS): This represents the milestone business entity in this application. It provides the customer services with tailored offers satisfying their demands. It controls the delivering of PCs to customers and also plays a key role in payment.
Request: This activity is the first step in this process-driven application and corresponds to the customer's request for a number of PCs.
Rule Based Service Level Agreement language (RBSLA) is based on RuleML.

With this language SLAs can be implemented in machine readable syntax.
RuleML belongs to the XML-based standard for expressing rules on the Web.
Service Level Agreement (SLA) is defined as a formal contract between a service provider and a client guaranteeing quantifiable performance at defined levels.
Service Level Management (SLM) is the process of managing (composed) services so that they can fulfill SLA requirements
Service-driven business applications Any software-intensive business application that will be implemented using service technology (i.e. Web-Service standards)
Shipment services (SS): This service intervenes when there is a need for shipping the goods to the customer.
Shipment: This activity concerns a case in which the customer's place of residence requires a shipment.
Software architecture is a high-level software design dealing with the structure and organization of large software systems.
Web services (WS) are network-addressable software units (e.g., components, modules, programs); that is, they are developed to be used on the Internet.

Appendix C

Management Law Language Syntax

```
Management  law name
partners
      {management  interfaces}*
types {{par}+:data_type}*
rules
when trigger
who conditions //on partner preferences and histories
at_time conditions //on time issues
Manage {operations}*
else {operations}*
end law
```