University Pierre et Marie Curie – Paris 6

LIP6 – UMR 7606
RSR Department – MoVe Team

Thales Communications France
SC2 Service

A qualitative approach for decision making under non-functional constraints during agile service composition

THESIS

submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy of the
University Pierre et Marie Curie – Paris 6
(in Computer Science)

by

Pierre Châtel

Committee

Reviewers:	Luis Martínez López	Professor, University of Jaén, Spain
	Laurence Duchien	Professor, University Lille 1
Advisors:	Jacques Malenfant	Professor, University Pierre et Marie Curie – Paris 6
	Isis Truck	Assistant professor, University Vincennes - Saint-Denis – Paris 8
Members:	Florence Sèdes	Professor, University Paul Sabatier – Toulouse 3
	Emmanuel Chailloux	Professor, University Pierre et Marie Curie – Paris 6
Industrial supervisor:	Hugues Vincent	R&T Team Leader, Thales Communications France

École Doctorale d'Informatique, Télécommunication et Électronique de Paris

Contents

Chapter 1

Introduction

COMPUTER SYSTEMS, especially distributed ones, are evolving. Their complexification often goes hand in hand with this evolution. Thus, the sustainability of software infrastructures, especially on the Web, is directly linked to its capacity to integrate multiple changes of context in order to accompany theses changes.

Deployed architectures must be flexible, within a context of software developments rationalization. This thesis thus aims to solve some scientific and technical hurdles related to the implementation of real agile distributed systems, a prerequisite for their sustainability, while seeking to preserve or even improve their performance.

Moreover, as we shall see in the next section, a number of programming paradigms have successively established the conceptual and technological ground for our work to germinate. Work which, while falling into the inheritance of these paradigms, aim to respond to the unprecedented challenges posed by these complex distributed architectures: especially the dynamic composition of services under non-functional constraints .

1.1 From objects to services

One of the first relevant paradigms to address these new challenges happens to be Object-Oriented Programming (OOP) [Jacobson, 1991], which has roots in the 1960s. This paradigm involves the definition and assembly of software building blocks called objects: an object representing mostly a concept, an idea or a physical entity. The strength of the OOP approach lies in its ability to reconcile software and user levels through object modeling, it thereby simplifies the development of complex computer systems. Its broad adoption, however, has begun in the early 1990s, slowly pushed by the increased visibility of the Unified Process [Jacobson et al., 1999], including the famous "Rational Unified Process" (RUP) from IBM.

Subsequently, the component paradigm [Szyperski et al., 1999], including established technologies like JEE and CORBA, is built around key OOP concepts, such as encapsulation, plus contractualization of interactions, composition by third parties and components distribution. A component has well specified interfaces and can be deployed independently of the rest of the application.

Component-based software engineering can thus be seen as OOP applied, not directly in the code, but at the software architecture level: it has enabled the gathering of coherent and reusable objects.

Facilitated by the irresistible rise of the Internet, the world of distributed programming is still in mutation, with the gradual adoption of the service paradigm and implementation of software architectures based on these services ("Service Oriented Architectures", or SOA) [Papazoglou et Georgakopoulos, 2003].

SOA highlight a theoretically loose coupling between service consumers and producers. They decouple a distributed application into two layers. A layer of services offers specified by their interfaces and possibly semantic information in a directory service. The second layer is the physical implementation of services on machines connected to the Internet and accessible using standard protocols.

Developing an application can then be seen as the description of the combination of a set of services (provided by a directory), of granularity equivalent to that of components, and whose composition during execution corresponds to the calculation required by the application. This composition, often called "orchestration", will then be made on the basis of information brought by a business process, whose execution will first seek to link abstract service calls to compatible implementations.

From a technological standpoint, another key SOA feature lies in the fact that they are based on standard protocols and data representations, usually using XML languages, thus facilitating interoperability. It is in this context that the concept of Web service, as defined by the W3C[1] is embodied, as is its cohort of associated technologies and languages, such as SOAP and WSDL.

A computer system based on an SOA is made available as reusable services that can be dynamically discovered and composed by loose coupling. Its is therefore differentiated from fully integrated solutions of the "black box" type, such as ERP or other software. This architecture tries to respond to today's needs in term of flexibility, reusability, and fast adaptability in software engineering. For these reasons, our approach will be based on the paradigm of service.

1.2 Birth of a problematic

In an industrial context, this thesis has a broad application spectrum, ranging from conventional Systems of Systems (SoS) to Command and Control (C2) applications deployed on tactical military operations and crisis management frameworks. It can also be applied to pervasive computing.

In all these areas, we have to manage the dynamic composition of services with a strong emphasis on quality and minimum guarantees. The non-functional aspects that affect the level of service rendered or perceived, such as response times of applications, or more generally the technical or business Quality of Service (QoS) offered by these applications, are of crucial importance. Architectures based on services should then be equiped to meet these different requirements. Other features are also essential in these applications:

- mobility, making the appearance and disappearance of services very frequent;

[1] *World Wide Web Consortium*

- redundancy, which ensures that the same service can be offered by many physical implementations at the same time. Taking into account that all these implementations have a certain probability of being destroyed that varies widely depending on the type and the operational phase in which the device is;

- heterogeneity, since the same service can be offered by numerous devices. For example, digital imaging can be made by the low resolution camera built into the helmet of a soldier or the high resolution camera of a tank or Unmanned Aerial Vehicle (UAV).

From these characteristics emerges the general problem of implementing an agile composition of services, which, to be useful, must take into account the heterogeneity of the context and multiple related non-functional constraints.

1.3 Research objectives and pursued approach

Several obstacles come along the path towards agile service composition. Thus, many non-functional properties to be treated can extend from the technical characteristics of a service (such as its availability or offered bandwidth), to higher-level non-functional properties not defined over a finite known in advance set of QoS dimensions. These are intimately linked to the competence (or business) domain of the service. Moreover, in the vast majority of distributed systems based on business processes, the link to service producers is statically fixed, for technical or human reasons, through the indication of their physical location on the network. This requires prior knowledge of available services at process writing time.

Therefore, particularly to overcome this lack of flexibility and to align ourselves with real agility needs of modern distributed systems, as highlighted by the previous problematic, we pursue the following objectives:

First objective: promote the reactivity of complex distributed systems

Semantic SOA [Vitvar et al., 2007] (SSOA) is a recent trend located at the intersection of the Semantic Web [Berners-Lee et al., 2001] and SOA [Papazoglou, 2008]. It offer tools that can facilitate the description of services business properties. The first one is to allow the definition and reuse of high-level concepts from business ontologies, specifically established for each considered area of expertise. From these concepts, it is possible to semantically annotate service offers and requests, and, as such, to raise their abstraction level [Peer, 2002, Eberhart, 2004, Sivashanmugam et al., 2003, Patil et al., 2003].

We then propose the use of SSOAs in our context since they put forward such a high level of decoupling between service consumers and producers. Consequently, they will allow the publication and the coexistence of a large number of functionally equivalent services, but most certainly exhibiting distinct non-functional (QoS) characteristics.

In order to take advantage of loose coupling property of SSOAs, and to achieve this first goal in terms of responsiveness of complex distributed systems, we propose the establishment of a mechanism for *late binding of services* at business processes execution time, based on current QoS values of services.

From a technical standpoint, this approach is based on a two layers abstract software architecture:

- A service filtering and business processes execution layer. Filtering is the static stage in which available services are associated to the process to be executed, according to their relevance to strict functional requirements and non-functional expressed therein.

- And a layer for monitoring the first layer and making binding decision according to the availability of services previously filtered and their *current QoS values* obtained by an external supervision framework. This layer will be responsible for making the best binding decisions between running services and processes, according to user-defined criteria.

We follow in fact the approach of reflective languages, who arrange such an architecture in a core layer (execution) and a meta-layer (or meta-level) for controlling the core layer. As part of this thesis, we especially focus on the management of late binding moments between processes and services during composition, so as to include actual "fresh" QoS values of services.

Second goal: improve the performance of complex distributed systems

If the late binding of services mechanism implemented by the first part of our approach can effectively make an informed decision when choosing a service to bind to a process, the criteria that will guide this decision to a local optimization on the basis of current values of QoS remains to be defined.

However, and even more so in SSOAs, the concept of performance is inherently multifaceted and dependent on the business domain of applications : it can't therefore only be based on a fixed pre-established set of absolute rules. It is therefore necessary to provide the user with the means to characterize indirectly, for its business, the concept of performance so that it can then be integrated at the core of the decision-making process.

The second angle of our approach is therefore to develop a qualitative model of user preferences used to maximize the utility (i.e usefulness) of links between service producers and consumers.

We therefore propose a formalism that will allow the programmer of a business process to express its "decision policies" in linking and using physical services based on their current QoS values. These *user preferences*, will therefore allow to characterize an implication relationship between the actual values of non-functional properties of service and its transient utility level (for business process binding) compared to its peers.

The notion of performance as an objective to be achieved during service composition, is then expressed through the less subjective and more easily manipulated notion of *service utility* in the composition, and by extension the utility of a business process as a whole as the sum of all local utilities. The smooth execution of a distributed system, as desired by the user when defining its preferences on services, thus finds itself dependent on the ability of orchestration to direct his choice according to subtleties from its own execution domain.

Third objective: ensuring the simplicity and genericity of the proposed approach

The final challenge of this thesis will ensure that the formalisms and technical achievements, from our implementation of this approach, are easy to use and sufficiently generic to be easily adaptable to every area of business application and to be able to support their developments.

Indeed, as preference elicitation experts are not available in our context, and in order to meet our goal of simplicity, preferences will present the double characteristic of being *qualitative*, in order to manage inherent ambiguities and uncertainties in modeling, and secondly will allow the definition of potential contradictions between several non-functional properties offered by one service (for example, and very schematically, *price* vs. *quality*).

1.4 Scientific contributions

Our scientific contributions are naturally organized around the previously defined research objectives and will therefore crystallize under three different names in our SSOA context : *active*, *useful* and finally *agile* service composition.

Indeed, the full realization of the first objective in terms of responsiveness of complex distributed systems requires an adaptation of the approach proposed to the singular case of SSOAs, and to incorporate decision concepts which, to date, have not been studied much. Therefore, **our first contribution to the field of service composition is to develop an *active* approach, in multiple steps, able to filter and select services based on their non-functional characteristics at static and dynamic times**. This contribution will be detailed in Chapter **??** of this manuscript.

In response to performance issues mentioned above, **a second scientific contribution will lie in the implementation of a *useful* service composition**, since it seeks to maximize the utility of process/services bindings by exploiting the peculiarities of this new user preferences formalism. This contribution will be detailed in the chapter 6 of the manuscript.

Finally, **both active and useful approaches are being combined within a single contribution in terms of *agile* service composition**. Presented in chapter 7, it involves jointly exploiting, within a service orchestration based on BPEL, the late binding of services and our new qualitative model for user preferences. We also introduce more advanced concepts such as managing the global QoS of business processes during at late binding time.

1.5 Document setup

This manuscript focuses on two parts: at first we perform a *context and state of the art* so as to position our work compared to the existing approaches and identify potential technological and conceptual shortcomings which need to be addressed. This first part is accompanied by the presentation of the *use case* in which our work will be illustrated. Then, we successively addresses our *contributions* in terms of active useful and agile composition of services (see chapters 5, 6 and 7). The first contributions are followed by a formalization of the LCP-net preference model (see chapter 8) we have introduced for useful composition, as well as their technical implementation (see chapter 9).

1.6 Associated publications

International Journal

- Pierre Châtel, Isis Truck, Jacques Malenfant. *LCP-Nets: A linguistic approach for non-functional preferences in a semantic SOA environment.* Journal of Universal Computer Sciences (JUCS), 2010(Special Issue on Information Fusion and Logic-based Reasoning Approaches for Decision Making Under Uncertainty).

International conferences

- **Submitted** : Pierre Châtel, Jacques Malenfant, Isis Truck. *Qos-based late-binding of service invocations in adaptive business processes.* The 8th International Conference on Web Services (ICWS). 2010.

- Pierre Châtel, Isis Truck, Jacques Malenfant. *A linguistic approach for non-functional constraints in a semantic SOA environment.* in FLINS'08. 2008. Madrid, España.

- Pierre Châtel. *Toward a Semantic Web service discovery and dynamic orchestration based on the formal specification of functional domain knowledge.* in ICSSEA 2007 : 20th International Conference on Software & Systems Engineering and their Applications. 2007. Paris, France.

- Pierre Châtel. *Une architecture pour la découverte et l'orchestration de services Web sémantiques.* in JFO 2007 - Les Ontologies : mythes,réalités et perspectives. 2007. Sousse, Tunisia.

International Workshop

- Bao Le Duc, Pierre Châtel, Nicolas Rivierre, Jacques Malenfant, Philippe Collet, Isis Truck. *Non-functional Data Collection for Adaptive Business Process and Decision Making.* in MW4SOC'09 workshop. 2009: Vienna, Austria.

Part I

Context and state of the art

Chapter 2

Technological context

The contributions of this thesis build over a rapidly evolving conceptual and terminological basis, grounding our work into a precise industrial and technical context. Among the major preceding work, several took place within European or French national projects, such as ITEA S4ALL, CARROLL and DYONISOS. Our work has been developed wihtin the SemEUsE French ANR project, in a core partnership between Thales, LIP6 and Orange Labs. This chapter first covers more precisely concepts from the Semantic Service-Oriented Architectures, and then quality of service concepts applied to the service-oriented approach.

2.1 Service-Oriented Architectures

Service-oriented architecture have brought to distributed computing the loose-coupling between service providers and clients, through the use of standards and the integration of mature concepts coming from object-orientation and component-based architectures introduced in this section.

2.1.1 Base concepts

Objects

From object-orientation, SOA keeps essentially the concepts of information hiding, the clean separation between the implamenation and the use of objects through methods, and polymorphism, the ability to use in the same way object of different implementations.

Components

Software components strive for more decoupling by making interfaces the key conept, insisting on having explicit offered but also required interfaces. Interconnexion between components then boils down to connect required interfaces to offered ones, externally to the implementation of the components.

Services

In the same vein as components, service-oriented architectures strive for even more abstraction, being defined independently of the execution platforms and using higher-level interconnection protocols. Services can be then implemented directly with underlying components, or through orchestrations using SOA standard languages. To date, the proposed languages for orchestration are still insisting on an early, static binding of services called in the orchestration to the orchestrated business process. One of the goal of this thesis is to put forward a late-binding of service calls, by deciding right before the call which service to use.

Service Composition

Service composition put into collaboration existing services to provide new more complex composite services. Composing services implies to put them into a workflow implementing the desired behavior. Choreography and orchestration currently compete to define these workflows. Our thesis puts itself in the setting of orchestration, where a unique orchestrator organizes a traditional execution control flow, including calls to other services.

2.1.2 Web Services

Web services form a subset of SOA were the iteroperability is obtained through the use of standards of the Web, essentially XML-based formalisms to express interfaces (WSDL), discover each others from directories (UDDI), exchange data (SOAP), negociate service-level agreements (SLA), etc. (see figure 2.1).

2.1.3 Semantic Web Service

Semantic Service-Oriented Architectures position themselves at the frontier of Web Services and the Semantic Web. It forsters the systematic use of semantic metadata, in the form of ontologies for instance, to perform the tasks of service discovery, service interconnection (e.g. to translate data from offered formats to required ones), etc. Standards begin to emerge, such as SAWSDL, to annotate interfaces and other service descriptions with semantic concepts.

2.1.4 Web Service Composition

WS-BPEL is currently the *de facto* standard language for orchestration, putting together the above standards and many other WS-* ones to define business processes. Several interpreters, or orchestration engines, have been developed, such as ActiveBPEL, ODE and Orchestra. Though most engines now tend to adopt the recent WS-BPEL 2.0 standard, their support for the novel extended activity construct, essential to the implementation of our late-binding, is still limited. The choice of Orchestra later in the thesis is essentially due to its beteter support of this new standard.

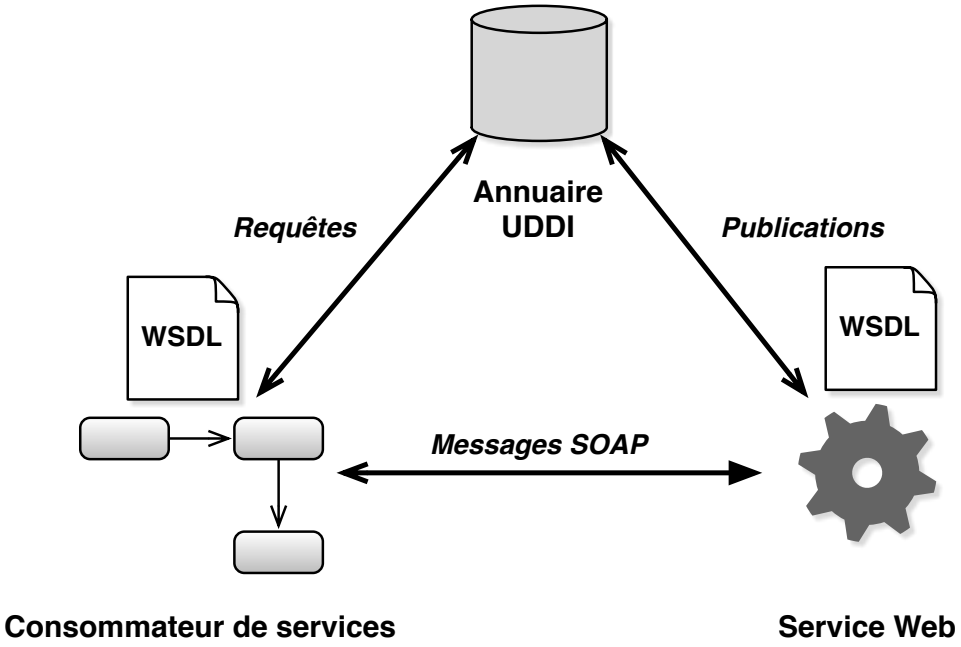

Figure 2.1: Architecture des services Web selon le W3C.

Alternative approaches to composition

Although predominant, alternatives to WS-BPEL for orchestration still exist, among which the translation of orchestration into the programming language Java trades a more static definition of the business process for efficiency.

2.2 Business Processes

Business processes consist of a set of activités organized toward the goal of transforming inputs into outputs. Their definition spans the whole software development, from design to implementation, using different set of standards at each step. Design languages include BPMN, a notation that recalls flowcharts, while the predominant implementation language is still WS-BPEL, as noted above.

2.3 Non-functional Properties and Constraints

In the context of SemEUsE, our late-binding of service calls uses non-functional or quality of service properties, and more precisely the current values from the different candidate services, to select the one to be called. This section therefore covers the core concepts of QoS: properties themselves, ontologies to describe them, and the intergration of QoS in SOA through QoS contracts and QoS-based composition. Because of the proliferation of services, and its high variability, the QoS is becoming a key notion in SOA. More and more, business processes express the required QoS from services, and services publish in directory their QoS commitments. Both can be expressed using

contracts, the matching between required and offered QoS then amounts to proving the conformity between the associated contracts.

2.3.1 QoS Taxonomies and Ontologies

In an open context such as SOA, introducing QoS lends itself to the same interoperability problems as experienced at the functional level to describe services. Different users may express QoS properties in different ways, and therefore a semantic approach, with ontologies, can help in matching needs and offers using different vocabularies. In our context, a semantic matching called filtering is done when looking for services on directories, We further use this semantic matching at run-time to connect decisions upon QoS dimensions expressed in the terms of the business process to the corresponding dimensions in the services expressed in the terms of the different service providers.

Several work propose such ontologies, with different standard underlying formalisms. OWL is a major standard to this end. It appears behind a lot of proposals.

2.3.2 Policies

Policies are means that express laws governing the way the QoS is delivered externally to the system itself. WS-Policy is the major standard in this area for SOA; Alternatives existn, but overall the kind of rules that can be expressed looks for more static ways to respond to QoS deviating from agreed contracts.

2.3.3 Quality of Service Contracts

A lot of work has been done since twenty years on QoS contracts, which has been gradually integrated to the SOA. The essential compromise here is between the expressiveness of contracts, in terms of the complexity of constraints, and the complexity of the reasoning algorithms needed to decide the conformity of two contracts. Most current proposals use linear and even simpler orms of constraints, for which conformity is easily tractable. In the context of SOA, WS-agreements is the key standard, though the standard does not define the constraint language itself, left simply to its extensions.

2.4 Conclusion

This chapter has presented the major industrial *de facto* standards that were underlying the implementation choices, and sometimes have forced conceptual approaches over others, given the nature of the project within which our work had to be done. In a SSOA setting, our late-binding extends the existing orchestration language WS-BPEL with a selection of services from their current QoS values. QoS ontologies help in matching business process requirements to service offers, by properly connecting dimensions expressed using different vocabularies.

Chapter 3

State of the art

THIS CHAPTER introduces the related work to the contributions of this thesis. We have categorized them in three parts: the linguistic reasoning approach, the compact modeling of preferences and the dynamic composition of services.

3.1 On a linguistic approach to reason

To put in place an agile composition of services, the decision framework calls for a qualitative assessment of the mutli-criteria problem arising from the use of different QoS dimensions which can contradict each other among candidate services. Indeed, users rarely have clear-cut ideas but rather approximative decision criteria that better translates to qualitative reasoning. In this section, we present the notions of fuzzy logic and computing with words to that end.

3.1.1 Fuzzy Modeling Concepts

This subsection takes up essentially the section 2.2 of the JUCS paper.

3.1.2 Fuzzy Reasoning

Fuzzy reasoning is based on a Generalized Modus Ponens, for which many different forms of implication (*Kleene-Dienes* évoqué ci-dessus, *Reichenbach*, *Lukasiewicz*, etc.).

3.1.3 2-tuples formalisms

This subsection also takes up essentially the section 2.2 of the JUCS paper.

The use of linguistic variables in reasoning implies their fusion, aggregation, comparison, etc., for which several models have been proposed. Among them, formalisms based on 2-tuples distinguish themselves by founding a form of computing with words without loss of precision. Herrera and Martínez [Herrera et Martínez, 2000], Truck and Akdag [Truck et Akdag, 2006] as well as Wang

and Hao [Wang et Hao, 2006], among some others, have proposed different 2-tuples models sharing this goal.

3.2 On the compact modeling of preferences

Preference modeling is required to attack multi-criteria decision problems; To set preference modeling in a qualitative framework, we consider two approaches: a linguistic and a graphical one.

In recent years, linguistic approaches have been applied to multi-criteria decision making. Based on a linguistic modeling of the different criteria, the process then goes to the aggregation of utility values coming from different experts to get a collective assessment for each alternative, and then to the use of these to rank the alternatives. To put such an approach in place requires to define suitable sets of linguistic variables and to choose an appropriate aggregation operator. Indeed, representation models and aggregation operators from 2-tuple models pave the way to a finer ranking, thanks to the precision allowed in the aggregation by such models.

Compared to GAI networks [Gonzales et Perny, 2005], the desired simplicity of *CP-nets (ie. CP-nets [Boutilier et al., 2004], UCP-nets [Boutilier et al., 2001], TCP-nets [Brafman et Domshlak, 2002]) in terms of elicitation or reasoning is particularly relevant on the Web in general, and particularly in the context of SSOA where the preferred solution should be found among a set of combinatorial possibilities. * CP-nets can be easily integrated into systems where users' preferences should be captured using only a few questions in order to quickly the "best" item among the available ones (services in the SSOA context) according to these preferences.

Its in the context where more time can be spent on elicitation (for example problems of configuration, equitable distribution of resources or even combinatorial escalation), in order to obtain a more detailed description of preferences, that GAI decompositions by additive cardinal utility may be more appropriate than purely ordinal models through their supposed greater descriptive power [Queiroz et al., 2007], and avoid the well-known theorem of Arrow's impossibility under collective decision problem [Pini et al., 2005].

However, we must not forget that the use of cardinal utility is not the exclusive domain of GAI networks. Indeed, its interests in the elicitation and implementation of preferences have been studied in the context of compact *CP-net preferences models with UCP-nets [Boutilier et al., 2001].

GAI networks also have several approaches for the integration of preferences expressed by various users in the context of a collective decision through the search for a compromise solution [Queiroz et al., 2007]. However, this is an issue that has not been much studied in connection with *CP-nets: while explorated by Rossi *et al.* through their *mCP-net* model [Rossi et al., 2004], it is hardly applicable to common use cases of our work in the SSOA context.

Finally, we note also that compact approaches for graphical representation of preferences (*CP-net or GAI-net), do not integrate a linguistic approach. This however allows, in our context, to elegantly take into account continuous value domains. Both approaches are evolving so far, in different conceptual fields, although they share common application goals when it comes to bridging the gap between informal business knowledge of users on one hand, and a more " formal" representation of that knowledge that is interpretable by a machine on the other. We thus try to reconcile these two approaches in our of useful composition contribution (see chapter 6).

3.3 On dynamic service composition under non-functional constraints

Compared to the very general definition of service composition that we have set in chapter 2, *dynamic* services composition provides an additional indication as to how and when to do it. Although there is no commonly accepted definition of this concept, most of the work that we address in this section share a set of common characteristics that can determine its outline:

- Contrary to a " classical" composition of services where links between processes and services are statically indicated by a human operator, the dynamic composition approach relegates this *binding decision* to the SOA support framework through algorithms able to take into account various factors.
- Processes non-functional constraints as well as services non-functional properties, indicated through their respective QoS contracts, establish most (if not all) of these factors.
- Service *recomposition*, at execution time, can perform the adjustments necessitated by changes in the QoS of services, following their initial composition and initial binding decisions made before the execution of the business part (or payload) of processes.

3.3.1 Dynamicity Management by recomposition

In the following works, service composition is seen primarily as an initial *off-line* step for selecting one or more global assignments of services to business processes.

During composition, these works will consider all services available for each service call site of a process. A service is then selected for the global assignment if its own QoS, added to that of previous binding choices, does not violate the constraints imposed by the process.

To verify this property, they use QoS aggregation functions to compute the *global value of a specific QoS property*, from the multiple *local* QoS values of services. Basically, for a linear business process P of one branch (no alternative or loop), in order to obtain the global value A_P of a specific QoS dimension A, one must aggregate values of A local to each bound service S:

$$\forall S \in (S_1, ..., S_n), value(A_P) = \sum_{i=1}^{n} value(A_{S_i})$$

In this formula, $\sum$ stands for the mathematical formula used for the aggregation of values of A.

For example, for a business process named *process* which includes a global non-functional constraint on the maximum execution time t :

$$t_{process} \leq 40ms$$

Several global assignments can be determined (see figure 3.1). The execution of a process on the basis of these assignments is then supposed to meet its QoS constraints.

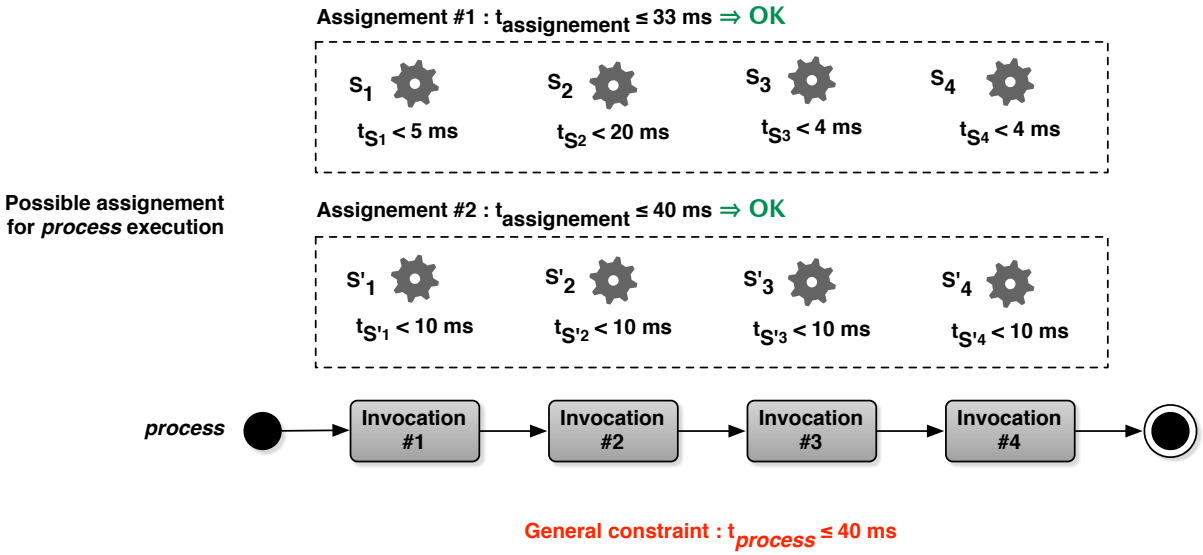

Figure 3.1: Dynamic service composition: assignments computation.

The following works, provide their own definitions of x for each type of QoS they treat. The most advanced are no longer limited to a linear approach, but incorporating the management instructions control flow (usually used in a business process) in the calculation of aggregation: for example, through the use of average values QoS to handle conditional branches.

The works considered below offer their own definitions of $\sum$ for each type of QoS they deal with. The most advanced of these works are no longer limited to a linear approach, but incorporate control flow management instructions in the aggregation calculation: for example, by averaging QoS values of conditional branches.

A comprehensive assessment of all possible services assignments is particularly expensive. This reinforces the *static* aspect composition of these approaches. These works thus add a *recomposition* stage of process restructuring, triggered during its execution: this is a re-allocation of services to different sites of appeals process in order to meet again the non-functional constraints. The cost varies depending on the extent of the reorganization: a reconstruction of the whole process, it is total, for a limited "reconfiguration", it can be minimized.

The high cost of a comprehensive and *a priori* assessment of all possible combinations of service assignments to a process reinforces the static characteristic of these composition approaches. These works then propose to supplement them by a *recomposition* process, triggered during business processes execution: that is, computation of new assignments of services for each of its call sites, in order to again fulfill its non-functional constraints. The cost of a recomposition varies on its extent : for a recomposition on the whole process, it is total [Ben Mokhtar et al., 2007, Canfora et al., 2006], but, for a limited "reconfiguration", it can be minimized [Colombo et al., 2006, Halima et al., 2008].

COCOA framework

The COCOA ("COnversation-based service COmposition in pervAsive computing environments") framework applies the principles of composition and recomposition of services to pervasive computing [Mokhtar et al., 2005, Ben Mokhtar et al., 2007].

It provides a set of aggregators that can address the *availability*, *latency* and *cost* of services according to some specific structures of a business process (loops, conditional tests, etc...). The framework will use these formulas during composition and recomposition with the services available in the environment, *but only when a QoS violation is detected*.

These works also adjoin, to the service selection step, a logical reasoning based on an OWL-S ontology. Ontological concepts used by service consumers, correlated to those of service providers, will then influence service compositions as far as services QoS values. Indeed, ontologies define a specific frame for defining new non-functional features by extending the basic model proposed by COCOA. In these works, OWL-S ontologies are used to describe the entirety of service offers or requests, and not only to annotate business offerings like SAWSDL does. However, we find that the SAWSDL specification quickly established itself as the dominant standard in the majority of recent SSOA works, which will probably lead to some kind of update of COCOA .

Canfora *et al.*

Recent works by Canfora *et al.* put forward a comprehensive approach to address the problem of dynamic service composition [Canfora et al., 2005b, Canfora et al., 2006]. It is declined, from an architectural standpoint, by a *QoS evaluator* and a *binding component* (called "binder").

Composition is then based on characteristics at the border of functional and non-functional domains, mostly business related. To define these characteristics, authors introduce a new language for describing QoS properties and a second one dedicated on defining the mathematical formulas necessary for data aggregation, based on previous work of Cardoso [Cardoso, 2002]. The *QoS evaluator* is then responsible of interpreting the definitions formulated in both languages.

The *NP-hard* problem of the comprehensive computation of all possible service assignments is treated, in turn, by the use of genetic algorithms [Canfora et al., 2005a] where other works have deployed integer programming techniques [Zeng et al., 2004].

Recomposition is declined here over the "services rebinding" and "process replanning" concepts. Based on noticing the volatility of effective QoS values and their evolution over time compared to those contracted during the initial composition, the authors have developed a component capable of performing, in some cases, a localized *service recomposition* during the execution of business processes. The idea is to avoid costs linked to calculation of new compositions by limiting the needed rebinding to a specific branch of the running process. The reconstruction will then be triggered:

- after finding a QoS constraint violation by a service which was previously bound on its QoS commitments;
- in some cases preemptively before the violation takes place, but only if considered as highly likely.

SCENE

The SCENE ("Service Composition ExecutioN Environment") framework consists of a software architecture and a language defined as an extension of BPEL [Colombo et al., 2006]. The operating principle of the framework is to allow description of policies/methods for *reconfiguration* that will be triggered following violation of one or more QoS rules. Reconfiguration is actually a specific terminology used in this work to define a *partial recompositon* of a process, normally less expensive than a full recompositon. It is also possible to compare the expressivity degree of these rules to that of QML non-functional constraints.

At runtime, thanks to the specifically developed software architecture in SCENE, violations (if any) are detected, and recompositon policies are triggered.

Halima *et al.*

This is another approach, proposed by Halima *et al.*, for process reconfiguration [Halima et al., 2008]. It is particularly dynamic since the authors recommend directly observing exchanged messages in a distributed application based on Web services to conduct punctual re-configurations, limited in time and space (it is only in extreme cases that the entire process would have to be reconstructed).

As in previous work, re-configuration requires the prior definition of policies, with the particularity that these policies can trigger the deployment of new services over the network.

Halima *et al.* also introduce the concept of *self-healing processes*, which remains largely dependent on the support framework and not on processes, as might be suggested by the concept name. We note however that this type of work is on the border of the recompositon concept as defined above, and a more *active* approach as advocated in our works.

These works however remain limited by a *treatment of failures* logic, and not on an *agile adaptation* to non-functional changes of services properties.

WSQoSX

The work underwent by Berbner *et al.* on WSQoSX ("Web Service Quality of Service Architectural Extension"), is based on the principle of a "real" dynamic service selection, actually carried out during their orchestration, through a proxy that sits between a process orchestration engine and available services [Berbner et al., 2007].

This selection is conducted based on informations statically defined in QoS contracts attached to available services, and not on the current QoS of services previously selected. If a service does not fulfill his contract (if an error occurs), a new service selection is triggered, local to its call site in the current business process. This first mechanism is followed, if insufficient, by *replanning* (thus re-compose) the entire process.

3.3.2 Composition by generating alternative and equivalent processes

Chiu *et al.*

The dynamic composition based on non-functional criteria, proposed by Chiu *et al.*, is limited to two dimensions: *"accuracy"* and *"time"*. They are incorporated in decision making, during service composition, before execution of a business process [Chiu et al., 2009].

Service composition is also based on a very limited set of predefined QoS dimensions: thus, values aggregation is done through a set of mathematical formulas similar to those established in COCOA.

These works are focused on the generation of *alternative workflows* capable of meeting the same abstract functional requirements: they are established before business process execution, in order to be able to switch between alternatives at execution time, if too many services doesn't meet their contracts.

Chafle *et al.*

A similar approach is found in Chafle *et al.* works: they seek to limit the impact of service recomposition by the "offline" computation of several functionally similar business processes. It is then possible to classify them by their relevance to non-functional initial constraints [Chafle et al., 2006].

In case of failure, the algorithm developed by the authors passes to the next available process and, if none process is left, uses an expensive global recomposition of services.

CoRE framework

As for COCOA, use of semantic information has been implemented in the SOA context by CoRE ("Component Runtime Environment") [Fujii et Suda, 2009].

This implementation is then carried through by the SeGSeC ("Semantic Graph based Service Composition") component, but on a much larger scale than in COCOA. The goal is to create one or more business process alternatives from scratch, based on user queries formulated in natural language. These works suggest that creation of alternatives is facilitated by business semantics, and achieved by the computation of correspondences between these semantic concepts.

This kind of approach, based on natural language, is particularly dependent on the quantity and quality of information provided by users themselves, the execution context of the distributed application, and the number of services available. Indeed, insofar as a business process skeleton is not defined by users before service composition, SeGSeC will require a large number of functionally equivalent services to reach its full potential in creating processes: each of these services is a possible alternative to meet the needs of users and create a logical sequence based on semantics.

3.3.3 Dynamic composition directed by CP-nets preferences

Schröpfer *et al.*

The idea of matching CP-nets and non-functional properties, in order to perform a dynamic service composition, has already been studied by Schröpfer *et al.*. The authors define preferences through CP-nets, in order to select the " best service" based on its QoS. Moreover the use of TCP-nets allow them to integrate the particular concept of *compromise*, which may be necessary for the expression of really useful preferences [Schröpfer et al., 2007].

Compared to our problem and the dynamic approach we advocate, it seems that two items are missing in their approach: integration of the concept of current QoS of services during decision making (it is exclusively based on non-functional contracts); and the explicit definition of a method to deal adequately with continuous and *qualitative QoS domains* (such as *latency*, *bandwidth*, etc...) . To manage these values, they are subject to the same constraints of UCP and TCP-net, on which they rest: it is necessary to make a partitioning of strict domains, with *artificial borders*, to obtain variables that can be manipulated in TCP-net preference tables, discrete by nature (for example: if $0 \leq$ bandwidth < 10 then it is called *"low bandwidth"*).

Furthermore, compared to previous work on recomposition, those conducted by Schröpfer *et al.* focus more on the stage of selecting services based on preferences, that on the implementation of a framework supporting this selection. It thus seems necessary to integrate them with more global approaches in order to exploit them properly.

TCP-Compose*

The works of Santhanam*et al.* developed an algorithm *TCP-Compose** for the efficient composition of Web services, based on qualitative preferences. To achieve this, they rely on TCP-net networks citesanthanam2008tcp, allowing them to have models whose properties are quite similar to those of Schröpfer *et al.*.

However, their algorithm proposes to tackle the computation of a composition *as a whole* and not, unlike the latter, to multiple *individual service selections*. This ambitious approach is then based on the *dominance testing* of assignments to a TCP-net that should include all the required non-functional properties for composition.

This has nevertheless a significant impact on the dynamic nature of the composition since it becomes indivisible from the nature of the TCP-Compose* comparison algorithm and by the dominance calculation of CP-nets. It then should probably be accompanied by a recomposition, although it is not envisaged by the authors.

3.3.4 Summary

The different approaches to dynamic services composition in non-functional constraints that we have addressed in this state of the art exhibit advanced features and are the result of a still very active scientific community. However, relative to our specific problem of implementation of an agile composition of services, their limitations are often similar: service composition is done relatively early in the life cycle of business processes, and recomposition (ie. composition triggered at execution

on errors and exceptions) as the main principle of adaptation to context evolutions at runtime is irrelevant in the case where non-functional characteristics of services vary within bounds of acceptable QoS as defined in their QoS contracts.

Indeed, in this case, execution errors that trigger processes reconfigurations are not detected, or only after accumulation. This observation will lead to the distinction between *active* composition (as our contribution in chapter 5) and *dynamic* composition.

3.4 Conclusion

Regarding the *linguistic approach to reason* and *compact modeling of preferences*, it appears that a combination of both approaches within one formalism has not yet been made and applied to our problem of taking into account non-functional constraints during service composition of services. It however fosters a better management of user preferences by providing formal tools to properly cross the gap between continuous QoS value domains and linguistic terms of everyday language that are fuzzy by nature. Thus, this approach will be implemented during *useful service composition* (see chapter 6).

Following the state of the art on *dynamic service composition under non-functional constraints*, appears a certain lack of support for multiple variations of the *current QoS* of components (services) of a distributed application, when they are below the threshold of a blocking error. The dynamic integration of these current values during service orchestration, conditioned by user preferences, should however have the advantage of allowing greater adaptivity of applications to their environments. This adaptivity could also be particularly useful in our use case (see chapter 4). Moreover, this technique could overcome the accumulation of small deviations from local non-functional properties of orchestrated services. This observation is the basis of our contribution in terms of *active service composition* (see chapter 5).

Chapter 4

Use case in crisis management: fire fighting

THIS USE CASE concerns a particular facet of environmental crisis management: fire-fighting in a civil context, on large fires in natural environments (ie. forests). Because of its inherent characteristics, we see that it is an ideal medium for the presentation of scientific contributions identified in this manuscript, namely : active composition (see chapter 5), useful composition (see chapter 6) and agile composition (see chapter 7).

Moreover, it is clear that such fires occur more frequently in our latitudes, a direct consequence of the combination of multiple climatic and human factors. To be fought effectively, they require the deployment of substantial resources and, above all, a "smart" composition of available air and land resources at the scene. Thus, on the "old continent", major fires induce the rapid mobilization of various, international means in order to respond appropriately to the fire and safeguard the environment : the real challenge is therefore to implement innovative approaches to the composition of these resources in order to take best advantage of them.

In this context, and compared to usual approaches for managing environmental crisis, we argue that an improved treatment of the situation can be obtained by dynamically using available resources in a partially automated fashion. The aim of our work is not to replace the classic chain of command and control which requires human intervention, but rather to complement this action by the automation of certain targeted procedures. There are indeed, in this type of professions, many intervention procedures (protocols or processes) resulting of many years of field practice that it is reasonable to follow scrupulously. Spurred by standardization initiatives of the European Union, there is also an increasing automation of these processes. It is therefore necessary to consider the availability of resources, often only known when the crisis occurs. which in consequence thwarts any provisioning attempts at process design time. Another constraint is the availability and QoS of resources *during* process execution: in this use case, some services related to physical resources are involved in the field and wear out with time, as such their evolving QoS must be integrated in decision making. Therefore, a high level of agility is required during crisis management, to ensure full mobilization of available resources, *which leads us to the implementation of an active composition of services.*

Also emerging in this context is a need for managing intervention priorities at different levels. For example, a hierarchy must be made between the defense of goods, people and the natural environment: in the case of a forest fire, away from towns and homes, it would seem logical to promote greater environmental protection. Theses *preferences* can be only be specified at process deployment time, according to the specific characteristics of the crisis, and can't be "pre-wired" in intervention protocols. These *non-functional user preferences* can be specified from a command center. It is thus emphasized a direct reconciliation between the concept of priority and the *utility* of mobilized resources: the most useful resources to solve the crisis, according to the semantics of deployed preferences, will be prioritized when composing services.

The goal of this use case is specifically to illustrate the importance of:

- The composition of the various stakeholders, which should result in the dynamic use of the available resources.
- The separation between two main moments during intervention. First, the specification *before crisis* of business processes related to the usual protocols and their functional and non-functional requirements. Then, *during crisis*, the execution of these business processes based on various services (UAVs, fire-fighters, trucks, etc.) actually available at launch, then dynamically selected during the process execution on the basis of their current QoS values.
- The importance of the current QoS of services during orchestration, integrated in binding decisions through user preferences, in order to maximize the utility (and therefore effectiveness) of the overall process.

Semantic SOA approach

First and foremost, a peculiar aspect of our approach is to associate each *physical resource* (UAVs, fire-fighters, trucks, etc.) to a *service* within the SOA framework, on the internal command and control network. The interfaces of these services is used to send them orders. Their offers are published in a registry available to all: they expose both their static functional and non-functional characteristics. These services are then consumed by *business processes* which correspond to fire-fighting protocols.

Key features of Semantic SOAs also address common interoperability and service composition problems in a crisis management context:

- The formalization of domain knowledge under knowledge bases (eg. ontologies). This gives a semantic model of each domain or a model of common knowledge between all contributors.
- The use of such high level knowledge in the specification of service offerings and business processes, as meta-data.
- The use of these meta-data at runtime, which can significantly expand the scope and relevance of service discovery and filtering when the crisis occurs: the junction between a service producer and consumer is then carried out dynamically on the basis of semantic criteria which are more flexible than syntactic criteria.

However, if the SSOA approach itself actually brings a certain level of agility when responding to the crisis, it is not without constraints and must be consolidated to be really usable in the field. More specifically, it is not enough to exploit semantic information once and for all at static service filtering, we must instead use the elasticity of the link between producers and consumers to add dynamicity. In response to these limitations, we then propose:

- the implementation of an architecture for truly dynamic orchestration of services since they can appear and disappear at any time during the execution (newcomers, equipment destruction, etc.).

- taking into account current QoS value, through user preferences, during execution of business processes and the development of tools to effectively discriminate between services on the basis of this information.

Summary

In this chapter we have laid the groundwork for a simplified use case of environmental crisis management, in the context of large forest fires. It allows, declined in multiple sub-scenarios to illustrate the various facets of our scientific contributions on the composition of services, but also gives details on steps prior to composition: the definition of service offerings and business processes using semantic annotations, as well as user preferences to be exploited at execution. This use case shows that it is possible to forge close links between the usual problems of crisis management and our work on SSOAs.

Part II

Contributions and implementation

Chapter 5

Active service composition

Although SSOAs lay the groundwork of our approach, they remains silent on specific issues raised by our context : services used by distributed applications are rarely known at the time of writing business process, it is also necessary to show great agility at execution time to tardily incorporate during composition, dynamically discovered services, taking into account many non-functional constraints. Although these issues arise here directly from our application context, one can easily imagine their relevance beyond these borders.

Thus, the need to address these issues is reflected by the implementation of an *active services composition* through the description and implementation of a *late binding of services approach*, at orchestration [Châtel et al., 2010a]. Are then defined in this chapter concepts and programming abstractions to implement a late and effective *binding decision* of service providers to consumers. Similarly, we highlight the importance of binding decisions that can (or must) be made at processes execution, based on their needs and non-functional characteristics of currently available services. The approach to make these binding decisions, and the choice of the most opportune time to trigger them, is at the core of our active services composition.

As part of the state of the art of this manuscript, we discussed many pre-existing work in the field of service composition under non-functional constraints. They are presented as "dynamic" to the extent they are able to perform, during the deployment process, a set of binding choices, based on processes non-functional constraints and static QoS contracts of services.

These binding choice being carried out in bulk at process loading time, thus before the execution of their business part, are challenged by *recomposition* only when *obvious and damaging violations of non-functional contracts are made*. We can compare this mode of operation to that of an "exception handling" mechanism. We can classify these dynamic composition approaches as *"defensive"*. The recomposition of services, a particularly costly (in time) process despite optimizations, is often presented as the only way to adapt to current environmental constraints, and is only triggered in response to an error during process execution.

We set ourselves apart from these traditional approaches in that the recomposition is only regarded as a last resort solution. Without waiting for his intervention, active composition will select *during* service orchestration, for each invocation site, the most suited service *at this moment*, according to its current QoS values and local non-fonctional constraints. This selection mechanism is an integral part of our late binding of services approach.

Indeed, local service selection at each invocation site is done *inside a group of functionally equivalent services*. These groups are predetermined through a filtering mechanism. As such, the relevance of this selection and binding decision can only be greater if based on current QoS, *collected as late as possible*. Therefore, we argue that the decision must be made on the fly, at the extremum of the life cycle of a process: in this case during its orchestration. Our approach late binding approach is especially justified, in our environmental crisis use case, by the frequent fluctuations of current QoS level of services and relatively long execution duration of business processes.

However, the impact of this approach on SSOA middlewares performances should not be overlooked. To minimize it, we propose the definition of a comprehensive approach for late binding of services. This approach includes preliminary steps to late binding, so as to bypass, at service binding and invocation time, delays related to the preliminary service filtering algorithms and monitoring framework setup.

Summary

In this chapter we presented the foundations of our contribution for the active composition of services. The aim of this approach is to address a very real need, that of improving the adaptability of complex computer systems. As the embodiment of these complex systems, distributed applications based on SOA and especially SSOA are in fact increasingly deployed in the field.

In this context, dynamicity and flexibility of an application is assessed against its ability to respond quickly to constant changes in its execution context, which is greatly facilitated by the late binding of services based on their current QoS values. Moreover, implantation of this late binding has been completed and will be presented to the chapter 9.

The performance of an application is evaluated on its ability to choose and use the most “relevant” services for its execution, which we discuss in chapter 6.

Chapter 6

Useful service composition

We seek to improve the performance of complex distributed systems throughout their execution. This improvement involves consideration of users' non-functional preferences during the final stage of composition: the late selection and invocation of a service. This recognition will help maximize the *utility* of the binding between a process and a service, as well as improve global process performance by a multitude of local optimizations at each service call site.

This chapter therefore aims to lay the conceptual foundations of a *useful composition* of services, especially by introducing a new preferences formalism, LCP-nets ("Linguistic Conditional Preference networks") [Châtel et al., 2008, Châtel et al., 2010b], which will allow the programmer of a business process to express beforehand its decision policy in the binding and use of physical services based on their effective (or current) QoS values obtained during process execution.

However, if LCP-nets are particularly suited to the realm of SSOA, their formalism is not related to any specific multi-criteria decision field. Sharing the same philosophy, the mechanism by which preferences are expressed, processed and evaluated, can also be put to work outside this specific context.

In addition to its *genericity*, we also illustrate the *ease of use* of the formalism. Indeed, since preference elicitation experts are absent from our context, it has the double characteristic of being *qualitative* through the use of linguistic concepts in order to better manage the inaccuracies and uncertainties inherent in modeling such preferences, it also has the ability to express contradictions between several non-functional properties offered by one service.

We chose to base our modeling on the *CP-net family (CP-nets, UCP-nets, TCP-nets, etc.) of compact preference definition languages. This choice was made because these models offers significant advantages in our application context, such as ease of use for non-expert modelers, have a relatively low computational cost, a strong structure, and show extension potential to support additional properties that could be crucial with respect to our context. Other formalisms have been considered, such as GAI networks, but these approaches are not adapted to our context due to the major effort on the elicitation phase and the cost in terms of reasoning they involve. Conversely, the GAI-net are especially suited to discriminate on a very large amount of alternatives, which is not the case here.

The vast majority of QoS properties that we have to handle are defined on continuous domains, with numerical values. *We then propose to conduct a systematic discretization (a fuzzy partitioning)*

of continuous domains by the use of fuzzy linguistic terms [Zadeh, 1975]. This approach avoids erecting strict and artificial value barriers on continuous domains, allowing a smooth transition from one class of values to another. Moreover, this approach avoids loss of information related to discretization, since it maintains continuous interval calculations for low-level inference, but still show linguistic terms associated to classes at the higher user level.

Therefore, in an environment where users must express their preferences between values of continuous domains, the qualitative nature of fuzzy sets, with smooth transitions, proved to be a better and easier solution than crisp values for capturing their preferences. Indeed, giving particularly accurate numbers to express a perception (or preference in our case) is not always possible [Zadeh, 1975, Degani et Bortolan, 1988, Herrera et Martínez, 2000, Truck et Akdag, 2006]. *A second proposal of this thesis is thus to retain the UCP-net concept of utility while extending the qualitative approach to expressing its values.*

Finally, one of the main aspects of LCP-net is the transposition performed during compilation, between a user-oriented graphical model to an "internal" representation more suited to decision-making with preferences. In particular, the various tables of an LCP-net are translated Fuzzy Inference Systems (FIS) [Jang, 1993] on which will be made the logical reasoning. Evaluation of a LCP-net is then based on the evaluation of its SIFs, using the theoretical framework of fuzzy logic.

Summary

We have highlighted a scientific contribution of this thesis on the establishment of a new variant of CP-nets, known as LCP-nets. They then benefit from a marriage between:

- *a linguistic approach*, implemented through fuzzy modeling of concepts;
- *a graphic approach*, promoted by *CP-nets;
- *properties specific to UCP-nets and TCP-nets*, such as the possibility to evaluate the utility of an assignment or define tradeoffs.

We have assumed for the time being that utilities of various LCP-nets are always expressed using the same set of linguistic terms, but this restriction could well be lifted by the implementation of a multi-granular approach [Herrera et al., 2002]. Similarly, the assumption on the positioning of linguistic terms (centered on 0.5, the terms distributed in an equidistant manner) could also be eliminated by implementing approaches to deal with unbalanced terms sets [Herrera et al., 2008].

Furthermore, an interesting improvement would be to compare the speed of calculations, considering only 2-tuples [Herrera et Martínez, 2000] in LCP-nets, under an *end-to-end linguistic treatment.*

Chapter 7

Agile service composition

Following detailed but separate presentations of *active* and *useful* service composition, the purpose of this chapter is to address the integration of these two contributions in a unified approach, deployed for service orchestration during the execution of a BPEL business process. This contribution is called *agile composition* of services, which faithfully reflects the ability of a distributed application built on its foundations to adapt dynamically to changes in its environment, for example by being able to take into account the evolution of surrounding services (disappearances, disconnects, changes in QoS, etc.), while taking into account non-functional constraints imposed on it as well as user preferences, expressed using LCP-net.

This ability may be crucial during crisis situations, particularly in our use case. It will then help select the best available services in terms of their QoS (including UAVs, fire fighters and trucks) at the time of the outbreak of the crisis and throughout its course, while allowing late adjustments through user preferences related to business processes. These preferences (contextual by nature) will then direct the selection of services, at every stage of the crisis management process, based on different QoS criteria that can be monitored.

The mechanism of active service composition, presented in chapter 5, introduced the abstract concept of late binding of services and the various steps that precede its onset. Useful composition, presented in chapter 6, is closely related to the LCP-net formalism. The purpose of this section is then to address more practical issues of their *joint* integration in a non-functional framework, during service orchestration based on the execution of a BPEL business process.

From a technical standpoint, in an SOA, the definition of a business process carried out by following the base WS-BPEL 2.0 specification (ie. without exploiting its extension mechanism) does not allow the introduction of late binding of services: its syntax implies that only one service must be explicitly assigned to each invocation site service. If WS-BPEL 2.0 provides an effective mechanism for extending basic language activities, even defining totally new ones, no extension for late binding and invocation of services have been predefined. Thus, we define an extended BPEL activity named *lateBindingInvoke*, whose implementation details are discussed in chapter 9.

Furthermore, the clear benefit of allowing the factorization of preference models, in order to avoid unnecessary repetitions and rationalize development costs, appeared during the joint integration of late binding and LCP-nets. As such, rather than having to repeat a preference applicable to all sites in each local LCP-net , a plausible alternative would be to factor these common preferences into a

shared net-LCP and then add, on each call site, the *local modifications* to apply to it. This ability to define user preferences incrementally, by local extensions or modifications, has been added to the LCP-net formalism by introducing the concept of preference *fragments*. Fragments are entities, constructed on the basis of an existing LCP-net, which unambiguously describe *additions*, *deletions*, or *modifications* of any element in their base LCP-net. Moreover, another advantage of this approach, particularly relevant for SSOAs, is the improvement of the readability of preference.

In the context of the agile service composition, we further seek to integrate LCP-net user preferences and their fragments in the orchestration of services. For this integration to succeed, we must minimize the constraints that must undergo a developer of business process BPEL to integrate them. We propose to develop a high-level XML representation of LCP-nets and their fragments in order to equalize the abstraction level of preferences with that of BPEL. Preferences and fragments described in this language are called *HL*-LCP-nets ("High Level LCP-nets") and *HL*-LCP-frags (" High-Level LCP-net fragments").

With this syntax, preferences become independent of the underlying modeling and decision framework that is presented as a *virtual machine on low-level LCP-nets* in chapter 9. We use the primitives of this virtual machine to establish the structural operational semantics of *HL*-LCP-nets by a set of production rules that establish connections between their abstract representation and calls to these primitives.

Summary

In this chapter we have defined the necessary programming abstractions to jointly use active and useful service compositions. This approach, called *agile composition*, presents its own complexities. For instance, we introduce the ability to define preferences incrementally through the concept of *preference fragment*.

It also seemed appropriate to offer further integration of preferences into BPEL business processes. The idea being to move towards LCP-nets considered as first class entities of these processes. This led to the introduction of a higher-level representation of LCP-nets and their fragments: *HL*-LCP-nets and *HL*-LCP-frags, including an XML syntax that allows a perfect coexistence with BPEL 2.0 processes.

Thanks to the genericity of this formalism, it was also possible to integrate global QoS properties of processes into the preferences, and therefore, in the binding decision itself. In the longer term, *contextual data* (such as the extent of fire, drought, etc..) or *technical QoS* provided by the SSOA support framework could also be integrated, by following a similar methodology.

Chapter 8

LCP-net formalization

8.1 Introduction

In this thesis, we have proposed a graphical model to represent linguistic preferences (see chapter 6) and we exhibited it in a specific use case (see chapter 4). We now consolidate this contribution in formalizing it through a set of notations and computation rules in order to guarantee its durability and its reusability to other multi-criteria decision contexts.

8.2 Preliminary notations

Following TCP-net definitions, we now introduce our LCP-nets in a formal manner. Let:

- V_i be a variable ($i \in \{1, \ldots, p\}$): e.g. security,
- D_i be the definition domain of V_i: e.g. $[0, 100]$,
- T_{V_i} be the linguistic term set associated to V_i: e.g. $\{S_{none}, S_{full}\}$,
- LV (a linguistic variable) be the following triplet: $LV = \langle V, D, T_V \rangle$:
  e.g. $\langle$security,$[0, 100]$,$\{S_{none}, S_{full}\}\rangle$.

As in the UCP-net formalism, preferences are expressed through utilities in our framework. But they are expressed through linguistic variables, as the other variables. They take their values in the single triplet $\langle V_U, D_U, T_{V_U} \rangle$ defined once for all:

e.g. $\langle$utility,$[0, 1]$,{very_low,low,medium,high,very_high}$\rangle$.

One utility is a triplet $LV_U = \langle V_U, D_U, S_{V_U} \rangle$, with $S_{V_U} \in T_{V_U}$, e.g. $\langle$utility,$[0, 1]$,low$\rangle$.

A conditional preference table $CPT(LV)$ associates preferences over D for every possible value assignment to the parents of LV denoted $Pa(LV)$. In addition, as in the TCP-nets formalism, each undirected edge is annotated with a conditional importance table $CIT(LV)$. A CIT associated with

such an edge (LV_i, LV_j) describes the relative importance of LV_i and LV_j given the value of the corresponding importance-conditioning linguistic variable LV_k.

Graphically, a preference table (CPT or CIT) is a tuple of triplets, i.e. a table with N dimensions. N is the number of the linguistic variables interrelated with LV ($N = |Pa(LV)|$) and a utility S_{V_U} is defined in each case.

Thus a preference table may be represented by the tuple

$\langle LV_i, LV_{i'}, \ldots, LV_{i''\cdots'}, LV_{U_1}, LV_{U_2}, \ldots, LV_{U_\eta} \rangle$
with $\eta \in \{2 * N, \ldots, H\}$ and $H = |T_{V_i}| \times |T_{V_{i'}}| \times \ldots \times |T_{V_{i''\cdots'}}|$.

For example, a preference table is the tuple:

$\langle \langle$resolution,$[320, 1024]$,$\{R_L, R_H\}\rangle$, $\langle$bandwidth,$[56, 4096]$,$\{B_L, B_M, B_H\}\rangle$,
$\langle$utility,$[0, 1]$,very_high$\rangle$, $\langle$utility,$[0, 1]$,very_low$\rangle$, $\langle$utility,$[0, 1]$,high$\rangle$, $\langle$utility,$[0, 1]$,low$\rangle$,
$\langle$utility,$[0, 1]$,very_low$\rangle$, $\langle$utility,$[0, 1]$,very_high$\rangle\rangle$.

More precisely, a preference table is equal to

$$
\begin{array}{l}
\langle\ \langle S_{V_{i_1}}, S_{V_{i'_1}}, \ldots, S_{V_{i''_1\cdots'}}, S_{V_{U_1}} \rangle \\
\quad \langle S_{V_{i_2}}, S_{V_{i'_2}}, \ldots, S_{V_{i''_2\cdots'}}, S_{V_{U_2}} \rangle, \\
\quad \ldots \\
\quad \langle S_{V_{i_\eta}}, S_{V_{i'_\eta}}, \ldots, S_{V_{i''_\eta\cdots'}}, S_{V_{U_\eta}} \rangle \\
\rangle
\end{array}
$$

So we get η tuples $\langle S_{V_i}, S_{V_{i'}}, \ldots, S_{V_{i''\cdots'}}, S_{V_U} \rangle$ with $\min(\eta) = 2 * N$ and $\max(\eta) = H$. The reason why the minimum for H is equal to $2 \times N$ is because it is necessary that $|T_V| \geq 2$ in order to be able to express a preference (!).

Following the same example and considering that resolution and bandwidth are interrelated, the associated preference table can be defined as these six ($\eta = 6$) tuples:

$\langle \langle R_L, B_L, very\_high \rangle$, $\langle R_L, B_M, very\_low \rangle$, $\langle R_L, B_H, high \rangle$,
$\langle R_H, B_L, low \rangle$, $\langle R_H, B_M, very\_low \rangle$, $\langle R_H, B_H, very\_high \rangle \rangle$.

This is to be read as six rules implying two different linguistic variables, i.e. six triplets $\langle V, D, S_V \rangle$ and six triplets $\langle V_U, D_U, S_{V_U} \rangle$:

R1. If we have $\langle$resolution,$[320, 1024]$,$R_L\rangle$ and $\langle$bandwidth,$[56, 4096]$,$B_L\rangle$
then we have $\langle$utility,$[0, 1]$,$very\_high\rangle$;

R2. ...

...

R6. If we have $\langle$resolution,$[320, 1024]$,$R_H\rangle$ and $\langle$bandwidth,$[56, 4096]$,$B_H\rangle$ then we have $\langle$utility,$[0, 1]$,$very\_high\rangle$.

8.3 Formalism and dominance testing on an LCP-net

Thus an LCP-net $\mathcal{L}$ is a tuple $\langle SL$,cp,i,ci,cpt,cit$\rangle$ where:

- SL is a set of linguistic variables $\{LV_1, \ldots, LV_p\}$, e.g. $SL = \{\langle$security,$[0, 100]$,$\{S_{none}, S_{full}\}\rangle$, $\langle$bandwidth,$[56, 4096]$,$\{B_L, B_M, B_H\}\rangle$, $\langle$resolution,$[320, 1024]$,$\{R_L, R_H\}\rangle\}$,
- cp is a set of directed cp-arcs. A cp-arc $\langle\overrightarrow{LV_i, LV_j}\rangle$ is in $\mathcal{L}$ iff the preferences over the values of LV_j depend on the actual value of LV_i. For each $LV \in SL$, $Pa(LV) = \{LV' | \langle\overrightarrow{LV', LV}\rangle \in$ cp$\}$,
- i is a set of directed i-arcs. An i-arc $(\overrightarrow{LV_i, LV_j})$ is in $\mathcal{L}$ iff $LV_i \triangleright LV_j$,
- ci is a set of undirected ci-arcs. A ci-arc (LV_i, LV_j) is in $\mathcal{L}$ iff we have $\mathcal{RI}(LV_i, LV_j | LV_k)$, i.e. iff the relative importance of LV_i and LV_j is conditioned on LV_k, with $LV_k \in SL \setminus \{LV_i, LV_j\}$. We call LV_k the *selector set* of (LV_i, LV_j) and denote it by $\mathcal{S}(LV_i, LV_j)$,
- cpt associates a CPT with every linguistic variable $LV \in SL$, where $CPT(LV)$ is a mapping from $D_{Pa(LV)}$ (i.e., assignments to LV's parent linguistic variables) to D_U,
- cit associates with every ci-arc between LV_i and LV_j a CIT from $D_{\mathcal{S}(LV_i, LV_j))}$ to orders over the set $\{LV_i, LV_j\}$.

The CPT (attached to an LV) supplies a local utility for this LV. This local utility denoted by lu is computed thanks to an inference engine using the aforementioned rules.

So we get on the one hand an LCP-net that defines the preferences and on the other hand p local utilities denoted by the set:

$$LU = \bigcup_{i=1}^{p} \{lu_i\}$$

Then each node of $\mathcal{L}$ is associated with a weight w, i.e. we obtain a weight vector W:

$$W = \bigcup_{i=1}^{p} \{w_i\}$$

$\mathcal{L}$ can now be represented as the tuple $\langle SL$,cp,i,ci,cpt,cit,$W\rangle$ assuming that values taken by W depend on the node depth.

The algorithm for computing W can be based on a BUM (Basic Unit-interval Monotonic) family function. A BUM function f_{BUM} is a mapping from $[0,1]$ to $[0,1]$ and assumes the following properties:

- $f_{BUM}(0) = 0$
- $f_{BUM}(1) = 1$
- f_{BUM} is increasing (i.e. if $x > y$ then $f_{BUM}(x) \geq f_{BUM}(y)$)

So weights W are computed thanks to f_{BUM} in the following manner:

$$w_i = f_{BUM}(i/p) - f_{BUM}((i-1)/p)$$

The chosen f_{BUM} function can be $f_{BUM}(x) = x$ (in this case, all weights equal $(1/p)$ with p the number of nodes) ; or $f_{BUM}(x) = x^3$ (in that case, w_1 is very small compared to w_p) ; or $f_{BUM}(x) = \sqrt{x}$ (in that case, w_1 is the greater weight). To be able to analyze the choice of f_{BUM}, we can compute a measure of *orness* on this weight vector :

$$orness(W) = \frac{1}{p-1}\sum_{i=1}^{p}(p-i)w_i$$

This measure, between 0 and 1, allows us to express to which extent the aggregator using these weights resembles an OR. For example, when $f_{BUM}(x) = x$, $orness(W) = 0.5$. But when w_1 is much bigger than the "following" weights, $orness(W)$ tends towards 1.

As in the CP-nets, the deeper we go, the smaller the weights: we will then choose a vector W whose measure *orness* is between 0.5 and 1[2], i.e. $f_{BUM}(x) = \sqrt{x}$ or $\sqrt[3]{x}$, etc.

Assign weights to nodes of a graph is slightly different from a classical weight assignment to values. The difference is in the order of the values. In an LCP-net graph several nodes can have the same depth, so the order is not total. That is why assigning only w thanks to a BUM function, even appropriately chosen, doesn't permit to completely answer our problem, since nodes of the same depth will be discriminated.

We apply a BUM function such as the associated w be decreasing ($w_i > w_{i+1}$, avec $i \in [1,p]$). Then for every node of the same depth, we sum their associated weights and make an equirepartition of the obtained sum between these nodes.

Thus, every constraint is fulfilled, by constructing weights through f_{BUM}:

- $\sum_{i=1}^{p} w_{i,l_i} = 1$ with l_i the depth of node i, $l_i \in [1,L]$ and $L \leq p$
- $\forall i \in [1,p], \forall l_i \in [1,L], \begin{cases} w_{i,l_i} > w_{i+1,l_{i+1}} & \text{if } l_i \neq l_{i+1} \\ w_{i,l_i} = w_{i+1,l_{i+1}} & \text{otherwise} \end{cases}$

[2]In our implementation, the obtained weight vector verifies this criteria.

W is combined with LU to obtain the global utility associated to an outcome o denoted GU_o.

$GU_o = \Delta(LU_o, W)$, with Δ a pondered aggregator such as an OWA operator [Yager, 1988] (for example a simple weighted average aggregation).

A local utility is either a linguistic term or a number corresponding to the defuzzification (through operator d) of the subset: $lu = f_{S_{V_U}}$ or $lu = d(f_{S_{V_U}})$ with:

$$f_{S_{V_U}} = f_{S_{V_U}}(y) = \begin{cases} \perp(f_{S^1_{V_U}}(y), \ldots, f_{S^\eta_{V_U}}(y)) & \text{if the } \eta \text{ rules are independent} \\ \top(f_{S^1_{V_U}}(y), \ldots, f_{S^\eta_{V_U}}(y)) & \text{otherwise} \end{cases}$$

with $y \in D_U$, $\perp$ a triangular conorm and $\top$ a triangular norm.

For sake of simplicity, we assume that $lu_i = f_{S_{V_{U_i}}}(y)$. GU_o is thereof either a linguistic term or a number. We assume that in the case where it is a linguistic term, it is always possible to find a defuzzification operator d that provides for a number.

Considering that the rules are independent and applying the generalized modus ponens [Zadeh, 1975],

$$f_{S_{V_U}}(y) = \sup_{(x_1,\ldots,x_N)\in D_1\times\cdots\times D_N} \Big\{ \top\Big[g(f_{S_{V'_1}}(x_1), \ldots, f_{S_{V'_n}}(x_N), \Phi(g(f_{S^1_{V_1}}(x_1), \ldots, f_{S^1_{V_N}}(x_N)), f_{S^1_{V_U}}(y))\Big] \\ \vee \ldots \vee \top\Big[g(f_{S_{V'_1}}(x_1), \ldots, f_{S_{V'_n}}(x_N), \Phi(g(f_{S^\eta_{V_1}}(x_1), \ldots, f_{S^\eta_{V_n}}(x_N)), f_{S^\eta_{V_U}}(y))\Big]\Big\}$$

with $f(x)$ the membership function of element x, Φ any fuzzy implication, V' the real variables observed (retrieved, given by the user), $S_{V'_1}$ the linguistic term associated to the first variable (V'_1) observed and g an aggregation operator such as a triangular norm (min for example). Thus an outcome o is actually a tuple $\langle S_{V'_1}, \ldots, S_{V'_p}\rangle$.

In order to prove the dominance testing property, we shall prove that an outcome o can always be found as being strictly preferred to another outcome o'.

Theorem.
Given an LCP-net $\mathcal{L}$ and a pair of outcomes o and o', we have that $\mathcal{L} \models o \prec o'$ iff GU_o is weaker than $GU_{o'}$.

Proof.

$$\frac{\text{inference from } CPT(LV_1^o) \quad \dots \quad \text{inference from } CPT(LV_p^o)}{\mathcal{L} \vdash lu_1^o = f_{S_{V_{U,1}^o}} \quad \dots \quad lu_p^o = f_{S_{V_{U,p}^o}}}$$

$$\frac{\mathcal{L} \vdash lu_1^o = f_{S_{V_{U,1}^o}} \quad \dots \quad lu_p^o = f_{S_{V_{U,p}^o}}}{\mathcal{L} \vdash LU_o = \{lu_1^o, \dots, lu_p^o\}}$$

$$\frac{\text{inference from } CPT(LV_1^{o'}) \quad \dots \quad \text{inference from } CPT(LV_p^{o'})}{\mathcal{L} \vdash lu_1^{o'} = f_{S_{V_{U,1}^{o'}}} \quad \dots \quad lu_p^{o'} = f_{S_{V_{U,p}^{o'}}}}$$

$$\frac{\mathcal{L} \vdash lu_1^{o'} = f_{S_{V_{U,1}^{o'}}} \quad \dots \quad lu_p^{o'} = f_{S_{V_{U,p}^{o'}}}}{\mathcal{L} \vdash LU_{o'} = \{lu_1^{o'}, \dots, lu_p^{o'}\}}$$

$$\frac{\mathcal{L} \vdash LU_o = \{lu_1^o, \dots, lu_p^o\} \qquad \mathcal{L} \vdash LU_{o'} = \{lu_1^{o'}, \dots, lu_p^{o'}\}}{\mathcal{L} \vdash \Delta(LU_o, W) < \Delta(LU_{o'}, W)}$$

$$\frac{\mathcal{L} \vdash \Delta(LU_o, W) < \Delta(LU_{o'}, W)}{\mathcal{L} \vdash d(GU_o) < d(GU_{o'})}$$

$$\frac{\mathcal{L} \vdash d(GU_o) < d(GU_{o'})}{\mathcal{L} \vdash GU_o \prec GU_{o'}}$$

$$\frac{\mathcal{L} \vdash GU_o \prec GU_{o'}}{\mathcal{L} \models o \prec o'}$$

□

8.4 Conclusion

If it is possible to represent an LCP-net as a set of nodes, arcs and CPTs, this chapter has shown that, mathematically, it's a tuple consisting of LVs, three types of arcs, two types of tables and a weight vector. Furthermore, an LCP-net corresponds, after translation, to a set of LVs, FIS and weight associated with nodes that allow automated reasoning about these preferences and lay the foundations for their implementation.

Chapter 9

Implementation

9.1 Introduction

This chapter presents the implementation of our contributions in terms of *active* and *useful composition* of services [Châtel et al., 2010a]. Their joint deployment is the technical foundation for a true agile composition of services. It is done through the implementation of a *Java framework* for modeling and reasoning on LCP-net, and the integration of late binding of services in a specific orchestration engine for *Web services*.

Several choices have been made to carry out our implement work and are detailed in the chapter, as well as all technical information necessary to their understanding.

9.2 LCP-nets modeling and reasoning framework

The implementation of the framework was completed in its entirety in Java. The choice of this language was naturally made in our SSOA context where it prevails, but also for reasons of ease of integration into the broader framework of the SemEUsE project. This choice is also justified by the use, within our framework, of external Java components: in particular a fuzzy inference engine, which only offers a Java API.

Furthermore, developments have been made on the Eclipse platform: we have used its IDE, but also some of its core libraries, such as EMF which greatly facilitated the development of our own graphics editor for LCP-nets.

Finally, all code and templates for the implementation of the framework is publicly available at `http://code.google.com/p/lcp-nets` under GPLv3 license: the graphical editor and the LCP-net inference engine are both available for download.

9.2.1 Structured data model for preferences and fragments

All LCP-net or fragment "entities", as they have been presented so far in this manuscript, have been modeled using the *EMF framework*. An EMF model has been established to en-

compass not only the concept of LCP-net that we introduced, but also generic CP-nets as defined by Boutilier *et al.* [Boutilier et al., 2004], UCP-nets [Boutilier et al., 2001] and TCP-nets [Brafman et Domshlak, 2002]. This choice was made, rather than focusing solely on the LCP-net in order to facilitate framework extensions. We also implemented, as a side project, modeling and decision on UCP-net. This global model is also a good illustration of the close relationship of these various formalisms.

ecore data model

A data model of LCP-net has been established as an *ecore* file, specific to EMF, and serialized by the XMI OMG standard.

From an *ecore* model to a Java representation

From this ecore model and available EMF tools, a corresponding *object model* in Java has been obtained.

9.2.2 LCP-nets tooling

Our framework has two essential features: modeling LCP-net preferences and making decisions based on these models. We offer special tools to accompany them.

Preference modeling is accessible not only through the primitives of a LCP-net virtual machine (see below), but also graphically through an LCP-net editing tool we have developed. Is can be distributed as Eclipse plugin.

LCP-net virtual machine primitives

The skeleton of a comprehensive API for creating and manipulating LCP-net has been automatically obtained from the EMF preference model. We expanded this skeleton to make it functional and established an LCP-net Virtual Machine (VM) on top of it.

This concept of virtual machine makes sense when related to the high level *HL*-LCP-net preference language. A metaphor would be to compare these high-level preferences to *.java* source files, whereas XML serialization produced by our framework (ie. VM) are associated to java *.class* compiled files.

Graphical editor for preferences and fragments

The graphics editor developed in this framework allows for easier definition of LCP-net and their fragments, this definition is carried out under a tree form.

This editor has been created on the basis of the underlying LCP-net API, it allows the obtention of a serialization of models as easily exchangeable XML files, even more so in the SSOA context, whit no extra effort from the user.

9.2.3 Computing and reasoning on LCP-nets

The preference evaluation process, from which the overall utility of an assignment of values to all nodes of a LCP-net is obtained, is a multi-criteria decision problem, where each criterion is a variable (ie. a node) of the preferences network. The purpose of this section is to dwell on points of specific interest to the implementation of this process.

Choosing a fuzzy inference engine

The fuzzy inference process itself is based on the use of an external logic reasoner (or "inference engine") specialized in fuzzy logic: *jFuzzyLogic*[3] has been chosen. Several criteria have directed this choice compared to other alternatives such as *mbFuzzit*[4], *NRC FuzzyJ Toolkit*[5], or *Free Fuzzy Logic Library*[6], and are detailed in the main manuscript.

Limitations of current implementation

However, the use of JFuzzyLogic also leads to limitations in the broader decision framework. In particular, this library requires us to convert potential fuzzy inputs of the inference process as crisp values, which means defuzzification prior to their injection into the inference engine. Indeed, in its current implementation, JFuzzyLogic only supports the definition of Fuzzy Inference Systems with input and output typed as being real.

Such limitations could be partially lifted by the development of our own fuzzy inference engine based, for example, on fuzzy 2-tuples to carry out his calculations without loss of information induced by multiple fuzzification/defuzzification steps [Herrera et Martínez, 2000].

9.3 Late binding of services framework

Our efforts for the implementation of this framework have focused on the integration of decision making over LCP-nets, based on QoS values obtained by monitoring Web services, when tardily binding them a to a BPEL business process. To enable this integration, according to the constraints of SemEUsE and SSOA, different technical choices needed to be made.

The Orchestra service orchestration engine has been chosen in SemEUsE. Its particularity is to propose an implementation of WS-BPEL 2.0 and its mechanism for extending activities, on which we build our late binding of service by defining two new distinct activities :

1. **a *lateBindingConfigure* activity:** placed at the beginning of business processes in order to be evaluated before the business code.

2. **a *lateBindingInvoke* activity** which is related to a *lateBindingConfigure* twin activity, through a unique identifier used by both activities. This activity has the responsibility to

[3] http://jfuzzylogic.sourceforge.net
[4] http://mbfuzzit.sourceforge.net/en/mbfuzzit_software.html
[5] http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJDocs/index.html
[6] http://ffll.sourceforge.net/index.html

perform the actual selection and invocation of a service based on its current QoS and LCP-nets preferences.

9.4 Conclusion

In this chapter, we discussed the implementation of the LCP-net modeling and decision framework, as well as the implementation of late binding of Web services, through the various technical options that have been made to meet the specific constraints of the SSOA and SemEUsE contexts. However, much effort has been made to allow for future developments: if some choices have actually been made according to specific and punctual need, the implementation of too specific solutions has been avoided in many cases, especially on the LCP-net framework itself. Indeed, LCP-nets can, by nature, be applied to other contexts of multi-criteria decision making.

Chapter 10

Conclusion and prospects

After three years of thesis, it is now possible to look back at the work that was done. Performed under CIFRE agreement, this thesis has started on issues specific to large distributed systems of today, such as those commonly handled in the SC2 service at my host company, Thales Communications France, and on which the LIP6 MoVe team has recognized expertise. The SemEUsE industrial and academic ANR project then allowed us to frame this work to the SSOA context and to a specific use case, the management of environmental crisis.

Later in this chapter, we present a summary of contributions made in this thesis and the executed work as a whole, followed by perspectives in the short and long term that we have identified.

Executed work

The work presented in this thesis has been made to address a very specific problem: how to implement a particularly dynamic and flexible services composition that cantake into account many non-functional constraints. We articulate our answer around the concepts of *active*, *useful* and *agile* composition.

Active, so as to distinguish this first contribution from existing approaches in terms of *dynamic* service composition. Indeed, we propose a mechanism able to effectively complement these more traditional approaches in order to improve the adaptability to changes of complex distributed systems (see chapter 5). This mechanism is based on a transposition of the late binding principle, well known in object-oriented programming, to SSOAs. It fits into a broader framework, capable of filtering available service offerings from directories and establish monitoring of services to provide QoS measures.

Thus, when executing a business process using the programming abstractions that we have developed, our active composition framework is able to bind tardily and effectively a service consumer call site from a process to an effective service producer. Being late to make binding decisions helps ensure that the service selected to meet the needs of the process is available and that its *current* QoS values have been taken into account in the decision. This is a fundamental departure from dynamic composition approaches (see chapter 3) who usually hold a first pass of *static* composition

of services (based on non-functional *constraints*) before the execution of business processes : once this composition has been fixed, new services and QoS variations of pre-binded services will not be taken into account, unless that there is a "fatal" error during process execution. It is in fact a mechanism similar to exceptions management.

Moreover, late binding of service under non-functional constraints has been applied to the management of environmental crisis use case, and more precisely to fire fighting in a civilian context, on major fire in natural environments (see chapter 4). It is indeed, by its intrinsic characteristics, an excellent support to the presentation of this contribution.

Useful, since maximizing the *utility* of each service binding is important and will determine the overall performance of a business process during its execution. The semantics we give to this notion of utility is one that has been highlighted in our new LCP-net formalism for expressing user preferences (see Chapter 6). These preferences will then be taken into account when making binding decisions.

Indeed, to allow the selection of services in our multi-criteria context, this formalism is applied to the elicitation of non-functional preferences, graphically established *between* the services QoS properties and their values (the user then manipulates simple nodes, arcs and tables). Once elicited, it will yield a total order on each set of candidates services based on the proportional value of the binding of each of these services to their respective consumers.

Although independent of any specific field of application, LCP-nets are particularly suited to the context of Semantic SOA as they enable users to efficiently partition the continuous domains of QoS variables with appropriate linguistic terms, and express the utility of assignments in a qualitative rather than numerical fashion, the latter often proving to be artificial. However, the LCP-net language and inference framework that we have established on its basis (see chapter 9) could very well be made and applied to other contexts of multi-criteria decision making, as long as they exhibit similar constraints.

In this context, we have consolidated this language through the proposed formalization (see Chapter 8). It aims to ensure its sustainability and reusability and is made through a set of notations and calculation rules. It shows that, mathematically, an LCP-net is a tuple consisting of linguistic variables, three types of arcs, two types of tables and a weight vector.

Agile, since our contributions on the active and useful composition of services make sense when deployed together within a single approach. The agile composition thus presents itself as the "sum" of the previous two (see chapter 7): it is essentially the skillful use of LCP-net user preferences during services late binding, although specific issues have been addressed and resolved.

Thus, it is within this common implementation that the need for incremental definition of preferences became apparent. This finding led to the definition of preference *fragments* and the introduction of a higher-level XML representation LCP-nets: *HL*-net-LCP and *HL*-LCP-frags. Their formalization allowed to finely treat problems of integrating these preferences into a host language, namely BPEL.

As part of agile composition, it also seemed appropriate to offer further integration of preferences into BPEL business processes. The idea was to move towards LCP-nets considered as first-class entities (citizens) in processes definitions. Thanks to the genericity of our formalism, it was also

possible to integrate global QoS properties of process into preference models and, therefore, into binding decisions themselves. This ability complements dynamic composition in a relevant way, in order to reflect the reality of process execution in the subsequent choice that need to be made: for instance, the ability to bind a service slower but more accurate if previous calls occurred faster than expected. The classic dynamic composition does not achieve this kind of compromise, yet it appears that service consumers QoS constraints are often pessimistic compared to actual execution.

Short-term perspectives

In the short term, different work prospects, for which preliminary discussions have been conducted, are possible : whether to extend the experimental validation of our contributions, to generalize the concept of late binding to other technical frameworks, or to lay the groundwork for building valid LCP-nets.

On an extensive experimental validation

In this thesis, the principle of active service composition that we introduced has been applied to the very specific use case of large-scale fires management. In order to perform an extensive validation, careful study of the behavior of the late binding component for Web services, when faced with a wider range of network characteristics, should be undertaken. This study is all the more necessary that these are often usual conditions, related to the nature of networks in general and the Web in particular. These conditions can, for instance, induce significant delays in QoS values delivery from the monitoring framework, which are essential to make binding decisions.

To conduct this validation, which by nature cannot be comprehensive, require to develop new use cases and applications for our software implementation. This should be facilitated by the very nature of our framework : although developed in a specific context, its key principles of late binding and multi-criteria decision based on user preferences are not related to a specific use case.

During these experiments, the collection of performance values and observation of the actual behavior of late-binding and decision components should support the validation of our contributions.

On the generalization of late binding to other technical frameworks

We also need to generalize our work to other technical and technological frameworks. Indeed we placed ourselves, throughout this thesis, under the SSOA context, largely for industrial purposes. This work, although tedious, would be very useful.

It would be reasonable to generalize the concept of service and apply late binding to other categories of applications or distributed systems, from the most basic and centralized (client / server) systems, to the most advanced and distributed ones (eg. peer-to-peer), while exploring other technologies, like distributed objects with RMI and CORBA distributed components.

However, if we have reason to believe that this concept of late binding could be effectively transposed to other technical contexts, the usefulness of this transposition is higher if it's backed-up by real use cases.

On building valid LCP-nets

The operations proposed and implemented by the VM (see section 9.2.2) do not guarantee the obtention of *valid* LCP-net, since the VM seeks to build the most effective in memory representation of LCP-nets. For example, the addition of a fragment of LCP-net to an existing LCP-net does not only handle valid LCP-nets. Checking the validity of LCP-nets afterward is not an easy task. An alternative approach would be to define a new set of elementary operators, manipulating and returning only valid LCP-nets, that would guarantee that the construction of always valid final LCP-nets. Thus, an *atomic* valid LCP-net would consist only of a node and its associated CPT. Then we would define the basic operators on which to specify preconditions, postconditions and invariants. Therefore, we consider:

- n a node and V its linguistic variable ;
- SN the set of nodes ;
- (s, t) an arc between origin node s and destination node t. If it's a ci-arc, s can be exchanger with t ;
- SA the set of arcs (cp, i and ci) : $SA = \mathsf{cp} \cup \mathsf{i} \cup \mathsf{ci}$;
- CPT the set of CPTs, and CIT the set of CITs. For a $cpt \in CPT$, $\dim(cpt) = p$ means that cpt has p dimensions ;
- SL the set of LCP-nets.

The invariants of these operators would be :

- The total number of arcs (cp, i, ci) does not go over the number of (s, t) couples where $(s, t) \in SN$ and $s \neq t$;
- Mutual exclusion of arc types :
  - if $(s, t) \in \mathsf{cp}$ then $(s, t) \notin \mathsf{i}$ and $(s, t) \notin \mathsf{ci}$;
  - if $(s, t) \in \mathsf{i}$ then $(s, t) \notin \mathsf{cp}$ and $(s, t) \notin \mathsf{ci}$;
  - if $(s, t) \in \mathsf{ci}$ then $(s, t) \notin \mathsf{i}$ and $(s, t) \notin \mathsf{cp}$.
- The total number of dimensions for the CPT linked to node s equals 1 + the number of cp-arcs entering in s ;
- The size of each CPT dimension is smaller or equal to the number of domain values of the associated linguistic variable ;
- There is no conditional cycles in the graph ;
- There is at least one node (so at least one CPT) and 0 to n arcs ;
- There is at least as many table as there is nodes in the graph, that is to say exactly nb_nodes CPT et nb_ciarcs CIT.

For example, implementing the basic node addition operator, would be to add a new element to take into account in the preferences graph, that is to say creating a node, its linked CPT, and one or more arcs (more precisely from 1 to k, where k is the number of linked nodes).

Its preconditions would be :

- the new node n to add: $n \notin SN$;
- there is a least one LCP-net: $\exists \mathcal{L} \in SL$.

Its postconditions would be :

- the node has been added : $n \in SN$;
- a new CPT is linked to the node: $\exists cpt \in CPT$ with $\dim(cpt) = 1$.
- (at least) one new arc appears: $\exists (s, t) \in SA$;
- (the) LCP-net(s) is (are) modified: $\mathcal{L} \notin SL, \exists \mathcal{L}' \in SL$.

This work could lead us in the long term to define an algebra over LCP-net.

Long-term perspectives

We present in this section broader perspectives, which are part of the extension of our contributions, but for which further study should be conducted. They open, hopefully, new axis of research.

On a deeper work around LCP-nets

The algebra of LCP-net mentioned above would resume this formalization in a broader theoretical framework totaly independent of the use case of this thesis. Typically, such work would lead us to define an abelian group $(SL, *)$ with $*$ a law of internal composition. Specifically, the composition of two (or more) LCP-net would be relevant if one wishes to combine two (or more) preferences network on the same decision. For example, when two users must use the same service chosen from a common set with common preferences: it would be necessary in this case, to combine the two LCP-net defined by the two clients to reach a common decision.

Furthermore, an interesting improvement would be to compare the speed of calculations, considering only 2-tuples in LCP-nets, under an *end-to-end linguistic treatment.*

Indeed, when using 2-tuples (either the formalism of Herrera and Martínez, Wang and Hao or Truck and Akdag), we frees ourselves from fuzzy subsets, at least during calculation, while retaining linguistic semantics attached to handled objects. It should therefore be seen whether these numerical calculations are lighter than those associated with fuzzy susbsets. This seems likely given that only calculations over indices, or $\min(1 - x + y, 1)$ implications are performed, whereas fuzzy subsets end-to-end manipulation involves calculations of line intersections (if fuzzy subsets are trapezoidal), which may be considered as longer.

This would involve significant work on modeling an *ad hoc* inference over these 2-tuples formalisms, which is an interesting research track, regardless of LCP-nets.

On coordinating late binding decision making by sharing decision agents

This thesis has introduced programming abstractions for performing, during agile service coordination, multiple *local* binding decisions between service consumers and producers: this decision is actually being made *ceteris paribus* at each service call site of a business process, according to local user preferences and current QoS values of concerned services; *ie* independently of other past and future decisions.

The transition from one set of *independent* local decisions, to a global coordination of this set at process level, is justified by the issue of efficient management of its *global QoS*. A first step toward taking these constraints into account has been proposed in this thesis by using variables of LCP-net as a way of reintroducing the current global QoS values of processes in binding decisions. However, despite this reflexivity, these decisions are largely local.

It becomes clear that a *global and coordinated decision* (on the whole process) will require the introduction of specific programming abstractions, because it should not come at the expense of agility of the composition, as is the case in traditional approaches. This raises the question of choosing a suitable abstraction, to which the principle of factorization of LCP-nets that we have already discussed provides a partial answer. Indeed, it seems appropriate to accompany this factorization of decision models by a factorization of *decision agents* themselves. If the current implementation uses local decision agents binded to each service call site, they could be extracted and *shared between sites*.

The possibility of using these *global decision agents*, independent of program structure, together with a *sequential decision approach* [Bellman, 1957, Littman, 1996] could be an axis of development of our work over the long term.

Bibliography

[Bellman, 1957] Bellman, R. (1957). Dynamic Programming, Princeton. *Princeton University Press*.

[Ben Mokhtar et al., 2007] Ben Mokhtar, S., Georgantas, N., et Issarny, V. (2007). COCOA: COnversation-based service COmposition in pervAsive computing environments with QoS support. *The Journal of Systems & Software*, 80(12): 1941–1955.

[Berbner et al., 2007] Berbner, R., Spahn, M., Repp, N., Heckmann, O., et Steinmetz, R. (2007). WSQoSX-a QoS architecture for Web service workflows. *Lecture Notes in Computer Science*, 4749: 623.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., et Lassila, O. (2001). The Semantic Web. *Scientific American*.

[Boutilier et al., 2001] Boutilier, C., Bacchus, F., et Brafman, R. I. (2001). UCP-Networks: A directed graphical representation of conditional utilities. In *Proc. of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 56–64.

[Boutilier et al., 2004] Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., et Poole, D. (2004). CP-nets: A tool for representing and reasoning with conditional *Ceteris Paribus* Preference Statements. *Journal of Artificial Intelligence Research*, 21: 135–191.

[Brafman et Domshlak, 2002] Brafman, R. I. et Domshlak, C. (2002). Introducing variable importance tradeoffs into CP-nets. In *Proc. of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 69–76.

[Canfora et al., 2006] Canfora, G., Di Penta, M., Esposito, R., Perfetto, F., et Villani, M. (2006). Service composition (re) binding driven by application-specific qos. *Lecture Notes in Computer Science*, 4294: 141.

[Canfora et al., 2005a] Canfora, G., Di Penta, M., Esposito, R., et Villani, M. (2005a). An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, page 1075. ACM.

[Canfora et al., 2005b] Canfora, G., Penta, M., Esposito, R., et Villani, M. (2005b). QoS-aware replanning of composite web services. In *Proceedings of the IEEE International Conference on Web Services*, pages 121–129. Citeseer.

[Cardoso, 2002] Cardoso, J. (2002). Quality of service and semantic composition of workflows. *Department of Computer Science. Athens, GA, University of Georgia*, 215.

[Chafle et al., 2006] Chafle, G., Dasgupta, K., Kumar, A., Mittal, S., et Srivastava, B. (2006). Adaptation in web service composition and execution. In *International Conference on Web Services (ICWS), Industry Track*.

[Châtel et al., 2010a] Châtel, P., Malenfant, J., et Truck, I. (2010a). Soumis : Qos-based late-binding of service invocations in adaptive business processes. *The 8th International Conference on Web Services (ICWS)*.

[Châtel et al., 2008] Châtel, P., Truck, I., et Malenfant, J. (2008). A linguistic approach for non-functional preferences in a semantic SOA environment. In *Computational Intelligence in Decision and Control, Proceedings of the 8th International FLINS Conference*.

[Châtel et al., 2010b] Châtel, P., Truck, I., et Malenfant, J. (2010b). LCP-nets: A linguistic approach for non-functional preferences in a semantic SOA environment. *J.UCS Journal of Universal Computer Science*.

[Chiu et al., 2009] Chiu, D., Deshpande, S., Agrawal, G., et Li, R. (2009). A Dynamic Approach toward QoS-Aware Service Workflow Composition. In *Proceedings of the 2009 IEEE International Conference on Web Services-Volume 00*, pages 655–662. IEEE Computer Society.

[Colombo et al., 2006] Colombo, M., Di Nitto, E., et Mauri, M. (2006). Scene: A service composition execution environment supporting dynamic changes disciplined through rules. *Lecture Notes in Computer Science*, 4294: 191.

[Degani et Bortolan, 1988] Degani, R. et Bortolan, G. (1988). The Problem of Linguistic Approximation in Clinical Decision Making. *International Journal of Approximate Reasoning*, 2: 143–162.

[Eberhart, 2004] Eberhart, A. (2004). Ad-hoc invocation of semantic Web services. In *In IEEE International Conference on Web Services*, San Diego, California ICWS 2004.

[Fujii et Suda, 2009] Fujii, K. et Suda, T. (2009). Semantics-based context-aware dynamic service composition.

[Gonzales et Perny, 2005] Gonzales, C. et Perny, P. (2005). GAI networks for decision making under certainty. In *Multidisciplinary IJCAI-05 Workshop on Advances in Preference Handling*. Citeseer.

[Halima et al., 2008] Halima, R., Drira, K., et Jmaiel, M. (2008). A QoS-Oriented Reconfigurable Middleware for Self-Healing Web Services. In *Proceedings of the 2008 IEEE International Conference on Web Services*, pages 104–111. IEEE Computer Society.

[Herrera et al., 2008] Herrera, F., Herrera-Viedma, E., et Martínez, L. (2008). A fuzzy linguistic methodology to deal with unbalanced linguistic term sets. *IEEE Transactions on Fuzzy Systems*, 16(2): 354–370.

[Herrera et Martínez, 2000] Herrera, F. et Martínez, L. (2000). A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems*, 8(6): 746–752.

[Herrera et al., 2002] Herrera, F., Martínez, L., Herrera-Viedma, E., et Chiclana, F. (2002). Fusion of Multigranular Linguistic Information based on the 2-tuple Fuzzy Linguistic Representation Model. In *Proceedings of IPMU 2002*, pages 1155–1162.

[Jacobson, 1991] Jacobson, I. (1991). *Object-oriented software engineering.* ACM New York, NY, USA.

[Jacobson et al., 1999] Jacobson, I., Booch, G., et Rumbaugh, J. (1999). *The unified software development process.* Addison-Wesley.

[Jang, 1993] Jang, J. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3): 665–685.

[Littman, 1996] Littman, M. (1996). Algorithms for sequential decision making. *Brown University, Providence, RI.*

[Mokhtar et al., 2005] Mokhtar, S., Liu, J., Georgantas, N., et Issarny, V. (2005). Qos-aware dynamic service composition in ambient intelligence environments. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, page 320. ACM.

[Papazoglou, 2008] Papazoglou, M. (2008). *Web services: principles and technology.* Pearson Prentice Hall.

[Papazoglou et Georgakopoulos, 2003] Papazoglou, M. P. et Georgakopoulos, D. (2003). Service-oriented computing. *Communications of the ACM*, 46.

[Patil et al., 2003] Patil, A., Oundhakar, S., et Sheth, A. (2003). Semantic annotation of Web services. Rapport technique, LSDIS Lab, Department of Computer Science, University of Georgia.

[Peer, 2002] Peer, J. (2002). Bringing together Semantic Web and Web services. In *In Proceedings of the First International Semantic Web Conference*, Sardinia, Italy.

[Pini et al., 2005] Pini, M., Rossi, F., Venable, K., et Walsh, T. (2005). Aggregating partially ordered preferences: impossibility and possibility results. In *Proceedings of the 10th conference on Theoretical aspects of rationality and knowledge*, page 206. National University of Singapore.

[Queiroz et al., 2007] Queiroz, S., Gonzales, C., et Perny, P. (2007). Decision collective avec des reseaux GAI. *Congres de la Societe Francaise de Recherche Operationnelle et d'Aide a la Decision.*

[Rossi et al., 2004] Rossi, F., Venable, K., et Walsh, T. (2004). mCP nets: representing and reasoning with preferences of multiple agents. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 729–734. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Schröpfer et al., 2007] Schröpfer, C., Binshtok, M., Shimony, S. E., Dayan, A., Brafman, R., Offermann, P., et Holschke, O. (2007). Introducing preferences over NFPs into service selection in SOA. In *Proc. Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'07).*

[Sivashanmugam et al., 2003] Sivashanmugam, K., Verma, K., Sheth, A. P., et Miller, J. A. (2003). Adding semantics to Web services standards. In *In Proceedings of the International Conference on Web Services, ICWS'03*, pages 395–401, Las Vegas, Nevada, USA.

[Szyperski et al., 1999] Szyperski, C., Bosch, J., et Weck, W. (1999). Component Oriented Programming. *Lecture Notes in Computer Science*, 1743: 184–184.

[Truck et Akdag, 2006] Truck, I. et Akdag, H. (2006). Manipulation of Qualitative Degrees to Handle Uncertainty : Formal Methods and Applications. *Knowledge and Information Systems (KAIS)*, 9(4): 385–411.

[Vitvar et al., 2007] Vitvar, T., Moran, M., Zaremba, M., Haller, A., et Kotinurmi, P. (2007). Semantic SOA to promote integration of heterogeneous B2B services. In *IEEE Joint Conference on E-Commerce Technology (CEC'07) and Enterprise Computing, E-Commerce and E-Services (EEE'07), IEEE, Tokyo, Japan.*

[Wang et Hao, 2006] Wang, J. et Hao, J. (2006). A new version of 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems*, 14(3): 435–445.

[Yager, 1988] Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1): 183–190.

[Zadeh, 1975] Zadeh, L. (1975). The Concept of a Linguistic Variable and Its Applications to Approximate Reasoning. *Information Sciences, Part I, II, III*, 8,8,9: 199–249, 301–357, 43–80.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., et Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5): 311–327.

Abstract

Service composition implementation, in a Web and business context, opens many investigation and improvements prospects. The contributions in this thesis are then intended to perform a particularly dynamic and flexible composition, able to take into account multiple non-functional constraints. They revolve around the concepts of active, useful and agile composition.

Active composition is able to effectively complement dynamic approaches, in order to improve their capability to adapt to non-functional changes. It relies on a transposition of the late-binding principle to the SSOA context, then able to integrate the current QoS values of services at execution time. **Useful composition** is linked to our new *LCP-net* formalism for expressing user preferences. The elicitation of non-functional preferences, established *between* services QoS properties and their values, will afterward provide a total or almost total order over each candidate services sets, during service selection. Finally, **agile composition** is the "sum" of the last two, where we lay the groundwork for global QoS management during processes execution.

Keywords: Service Oriented Architecture, service composition, non-functional constraint, qualitative approach, linguistic approach, multi-criteria decision making, fuzzy Logic, preference modeling, late binding.