Mälardalen University Press Dissertations
No. 147

FORMAL APPROACHES FOR BEHAVIORAL MODELING AND ANALYSIS OF DESIGN-TIME SERVICES AND SERVICE NEGOTIATIONS

Aida Čaušević

2014

School of Innovation, Design and Engineering

ISBN 978-91-7485-128-1
ISSN 1651-4238
Printed by Arkitektkopia, Västerås, Sweden

Mälardalen University Press Dissertations
No. 147

FORMAL APPROACHES FOR BEHAVIORAL MODELING AND ANALYSIS OF DESIGN-TIME SERVICES AND SERVICE NEGOTIATIONS

Aida Čaušević

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid Akademin för innovation, design och teknik kommer att offentligen försvaras onsdagen den 15 januari 2014, 09.00 i Pi, Högskoleplan 1, Västerås.

Fakultetsopponent: Professor Ina Schieferdecker, Fraunhofer Fokus

Akademin för innovation, design och teknik

Abstract

During the past decade service-orientation has become a popular design paradigm, offering anapproach in which services are the functional building blocks. Services are self-containedunits of composition, built to be invoked, composed, and destroyed on (user) demand.Service-oriented systems (SOS) are a collection of services that are developed based onseveral design principles such as: (i) loose coupling between services (e.g., inter-servicecommunication can involve either simple data passing or two or more connected servicescoordinating some activity) that allows services to be independent, yet highly interoperablewhen required; (ii) service abstraction, which emphasizes the need to hide as manyimplementation details as possible, yet still exposing functional and extra-functionalcapabilities that can be offered to service users; (iii) service reusability provided bythe existing services in a rapid and flexible development process; (iv) service composabilityas one of the main assets of SOS that provide a design platform for services to be composed anddecomposed, etc.One of the main concerns in such systems is ensuring service quality per se, but alsoguaranteeing the quality of newly composed services. To accomplish the above, we consider two system perspectives: the developer's and the user's view, respectively.In the former, one can be assumed to have access to the internal service representation:functionality, enabled actions, resource usage, and interactions with other services.In the second, one has information primarily on the service interface and exposed capabilities(attributes/ features).Means of checking that services and service compositions meet the expected requirements,the so-called correctness issue, can enable optimization and possibility toguarantee a satisfactory level of a service composition quality.In order to accomplish exhaustive correctness checks of design-time SOS,we employ model-checking as the main formal verification technique, which eventually providesnecessary information about quality-of-service (QoS), already at early stages of system development.~As opposed to the traditional approach of software system construction,in SOS the same service may be offered at various prices, QoS, and other conditions,depending on the user needs.In such a setting, the interactionbetween involved parties requires the negotiation of what is possible at request time,aiming at meeting needs on demand.The service negotiation process often proceeds with timing, price, and resource constraints,under which users and providers exchange information on their respective goals, until reachinga consensus. Hence, a mathematically driven technique to analyze a priori various ways to achieve such goals isbeneficial for understanding what and how can particular goals be achieved.

This thesis presents the research that we have been carrying out over the past few years, which resulted in developingmethods and tools for the specification, modeling, and formal analysisof services and service compositions in SOS. The contributions of the thesis consist of: (i)constructs for the formal description of services and servicecompositions using the resource-aware timed behavioral language called REMES; (ii) deductive andalgorithmic approaches for checking correctness of services and service compositions;(iii) a model of service negotiation that includes different negotiation strategies, formally analyzedagainst timing and resource constraints; (iv) a tool-chain (REMES SOS IDE) that provides aneditor and verification support (by integration with the UPPAAL model-checker) to REMES-based service-oriented designs;(v) a relevant case-study by which we exercise the applicability of our framework.The presented work has also been applied on other smaller examples presented in the published papers.

ISBN 978-91-7485-128-1
ISSN 1651-4238

Abstract

During the past decade service-orientation has become a popular design paradigm, offering an approach in which services are the functional building blocks. Services are self-contained units of composition, built to be invoked, composed, and destroyed on (user) demand. Service-oriented systems (SOS) are a collection of services that are developed based on several design principles such as: (i) loose coupling between services (e.g., inter-service communication can involve either simple data passing or two or more connected services coordinating some activity) that allows services to be independent, yet highly interoperable when required; (ii) service abstraction, which emphasizes the need to hide as many implementation details as possible, yet still exposing functional and extra-functional capabilities that can be offered to service users; (iii) service reusability provided by the existing services in a rapid and flexible development process; (iv) service composability as one of the main assets of SOS that provide a design platform for services to be composed and decomposed, etc. One of the main concerns in such systems is ensuring service quality per se, but also guaranteeing the quality of newly composed services. To accomplish the above, we consider two system perspectives: the developer's and the user's view, respectively. In the former, one can be assumed to have access to the internal service representation: functionality, enabled actions, resource usage, and interactions with other services. In the second, one has information primarily on the service interface and exposed capabilities (attributes/features). Means of checking that services and service compositions meet the expected requirements, the so-called correctness issue, can enable optimization and possibility to guarantee a satisfactory level of a service composition quality. In order to accomplish exhaustive correctness checks of design-time SOS, we employ model-checking as the main formal

verification technique, which eventually provides necessary information about quality-of-service (QoS), already at early stages of system development. As opposed to the traditional approach of software system construction, in SOS the same service may be offered at various prices, QoS, and other conditions, depending on the user needs. In such a setting, the interaction between involved parties requires the negotiation of what is possible at request time, aiming at meeting needs on demand. The service negotiation process often proceeds with timing, price, and resource constraints, under which users and providers exchange information on their respective goals, until reaching a consensus. Hence, a mathematically driven technique to analyze a priori various ways to achieve such goals is beneficial for understanding what and how can particular goals be achieved.

This thesis presents the research that we have been carrying out over the past few years, which resulted in developing methods and tools for the specification, modeling, and formal analysis of services and service compositions in SOS. The contributions of the thesis consist of: (i) constructs for the formal description of services and service compositions using the resource-aware timed behavioral language called Remes; (ii) deductive and algorithmic approaches for checking correctness of services and service compositions; (iii) a model of service negotiation that includes different negotiation strategies, formally analyzed against timing and resource constraints; (iv) a tool-chain (Remes SOS IDE) that provides an editor and verification support (by integration with the Uppaal model-checker) to Remes-based service-oriented designs; (v) a relevant case-study by which we exercise the applicability of our framework. The presented work has also been applied on other smaller examples presented in the published papers.

Populärvetenskaplig sammanfattning

Under det senaste årtiondet har ett tjänstorienterat paradigm blivit alltmer populärt i utvecklingen av datorsystem. I detta paradigm utgör så kallade *tjänster* den minsta funktionella systemenheten. Dessa tjänster är konstruerade så att de kan skapas, användas, sammansättas och avslutas separat. De ska vara oberoende av varandra samtidigt som de ska kunna fungera effektivt tillsammans och i samarbete med andra system när så behövs. Vidare ska tjänsterna dölja sina interna implementationsdetaljer i så stor grad som möjligt, samtidigt som deras fulla funktionalitet ska exponeras för systemdesignern. Tjänsterna ska också på ett enkelt sätt kunna återanvändas och sammansättas i en snabb och flexibel utvecklingsprocess.

En av de viktigaste aspekterna i tjänsteorienterade datorsystem är att kunna säkerställa systemens kvalitet. För att åstadkomma detta är det viktigt att få en djupare insikt om tjänstens interna funktionalitet, i termer av möjliga operationer, resursinformation, samt tänkbar interaktion med andra tjänster. Detta är speciellt viktigt när utvecklaren har möjlighet att välja mellan två funktionellt likvärda tjänster som är olika med avseende på andra egenskaper, såsom responstid eller andra resurskrav. I detta sammanhang kan en matematisk beskrivning av en tjänsts beteende ge ökad förståelse av tjänstemodellen, samt hjälpa användaren att koppla ihop tjänster på ett korrekt sätt. En matematisk beskrivning öppnar också upp för ett sätt att matematiskt resonera kring tjänster. Metoder för att kontrollera att komponerade tjänster möter ställda resurskrav möjliggör också resursoptimering av tjänster samt verifiering av ställda kvalitetskrav.

I denna avhandling presenteras forskning som har bedrivits under de senaste åren. Forskningen har resulterat i metoder och verktyg för att specificera, modellera och formellt analysera tjänster och sammansättning av tjänster. Arbetet i avhandlingen består av (i) en formell definition av tjänster och sammansättning av tjänster med hjälp av ett resursmedvetet formellt specifikationsspråk kallat REMES; (ii) två metoder för att analysera tjänster och kontrollera korrektheten i sammansättning av tjänster, både deduktivt och algoritmiskt; (iii) en modell av förhandlingsprocessen vid sammansättning av tjänster som inkluderar olika förhandlingsstrategier; (iv) ett antal verktyg som stödjer dessa metoder. Metoderna har använts i ett antal fallstudier som är presenterade i de publicerade artiklarna.

To my family

Acknowledgments

Almost six years ago when I decided to start with my Ph.D. studies, someone told me that getting a Ph.D. degree is a long and tedious journey. But no matter what that person has told me, I have decided to take the chance and accept the challenge. Through the past six years I have really learned that it is not the easiest job in the world to be a Ph.D. student, but I have to say, for me it was the best. It was very exciting and vibrating to learn new stuff, to have opportunity to publish my work, present it at international conferences and workshops, to share my knowledge and thoughts with fellow Ph.D. students, to learn from seniors (professors, lecturers, etc.). I have got not only a chance to meet new people, but also to see new countries, cultures, learn new languages. All of the sudden all became close, and the most important possible to reach. I have learned that the whole joy is not in the final destination, the Ph.D. title, but along the way towards completing Ph.D. studies.

There are many people that have made this journey to be as it was for me. The most important figures are of course my supervisors. First of all I would like to thank to my main supervisor Paul Pettersson, for giving me the opportunity to become a Ph.D. student and believing that I have lived up to the challenges that this position has carried. Second, I want to thank to my assistant supervisor Cristina Seceleanu who has not only served as my supervisor, but also as friend, always there with a warm word of praise and encouragement. I am grateful to you for all challenges that you have put me through. I owe you a great debt of gratitude for your guidance and for never accepting less than my best efforts.

Also I would like to thank to colleagues from my research group Aneta Vulgarakis, Jagadish Suryadevara, Leo Hatvani, Eduard Paul Enoiu, and Raluca Marinescu for all support, discussions, reviews and comments.

Outside of the thesis work I have also been involved in teaching. Many thanks to people that I have had pleasure to work with: Ivica Crnković, Frank Lüders, Jan Carlson, Séverine Sentilles, Andreas Johnsen, Jiale Zhou, and Mehrdad Saadatman.

I wish to thank to teachers, lectures, and professors at MDH: Hans Hansson, Sasikumar Punnekkat, Gordana Dodig-Crnković, Mats Björkman, Eun-Young Kang, Thomas Nolte, Emma Nehrenheim, Dag Nyström, Lars Asplund, Radu Dobrin, Damir Isović, Björn Lisper, Kristina Lundqvist, Mikael Sjödin, Jan Carlson, and Daniel Sundmark, for giving me the knowledge and vision to become a better Ph.D. student.

I would like to thank to the whole administrative and research coordination staff at the department for making my life easier, in particular Carola Ryttersson, Gunnar Widforss, Susanne Fronnå, Malin Rosqvist, Anna Juto Andersson, Jenny Hägglund, Malin Åshuvud, Ingrid Runnérus, Sofia Jäderén, and Malin Swanström.

A Ph.D. position does not include work only, but also a lot of fun at coffee breaks, lunches, and travels. I would like to thank to Abhilash, Alessio, Andreas G., Aneta, Anita, Anton, Antonio, Barbara, Batu, Bob, Cristina, Dag, Damir, Daniel, Eddie, Federico, Frank, Fredrik, Gabriel, Giacomo, Guillermo, Hüseyin, Irfan, Jagadish, Jan, Josip, Juraj, Lars, Leo, Luka, Mehrdad, Meng, Mikael, Mohammad, Moris, Nikola, Nima, Omar, Radu, Rafia, Raluca, Saad, Sara Abbaspour, Sara Afshar, Svetlana, Thomas, Tibi, Saad, Séverine, and many others for making life at MDH more interesting and enjoyable.

Thanks to my Bosnian friend, Ajla Ćerimagić, for being always there despite the distance between us. For encouraging me to never give up and to follow my dreams.

To my dear brother Adnan and his wife Belma. Thank you for believing in me, for your love, support, and encouragement.

Veliko hvala mojim roditeljima, Edini i Mujagi. Ono što danas jesam, osoba koja sam postala, mogu zahvaliti samo vama. Vi ste bili uvijek oni koji su mi govorili da sve što poželim mogu samo upornošću i trudom postići. Hvala Vam što ste mi uvijek vjerovali i podupirali moje namjere, bez obzira koliko se sulude u tom momentu činile. Znam da je mami oduvijek bila želja da budem doktorica, evo želja joj se ispunila. Doduše, ne liječim ljude, ali mogu pomoći oko računara.

Finally, my deepest gratitude goes for my dear husband Adnan and daughter Alina. Adnan, thank you for supporting, maybe at that point in time, a crazy idea to leave all we have had before coming to Sweden

and joining me at this journey. Thank you for your unselfish and unconditional love, your understanding, simply thank you for being around. Alina, I thought I have had everything in my life, before you entered into it. With you, all my life has completely changed, in a positive way. You have thought me to be organized, to prioritize my time, to cherish each moment spent together. Now, I cannot imagine my life without you, your smile, your love, your eyes. I love you my child!

Aida Čaušević
Västerås, October, 2013

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade the service-oriented paradigm has become a popular software development approach that provides a way to implement distributed, loosely coupled, and platform independent systems. The paradigm has been introduced as an answer to the need of handling a significant growth of software functionality, by packing it into services and making it accessible through a networked infrastructure. A service is assumed to be an autonomous piece of software providing its functionality via well-defined interfaces that expose the services' characteristics, such as, response time, capacity, etc. Services have become available via either open or proprietary network protocols, and accessible within closed corporate Intranets, or throughout open protocols using Internet. The service-oriented approach has also brought a way to integrate and connect heterogeneous applications and available resources, in most cases on demand. Constructs to build systems in such a way can be seen as means to support complex and dynamic interactions among possibly large numbers of parties that interact in order to achieve well-defined goals.

One can view service-oriented systems (SOS) as a solution to bridge the gap between business models and existing technical solutions. From a technical perspective, SOS enable the use of services that provide reusable functionality via a well-defined interface, which are discoverable, and capable of being invoked and composed when needed. Moreover, SOS promote development of new applications based on the functionality available in already existing services. From a business perspective,

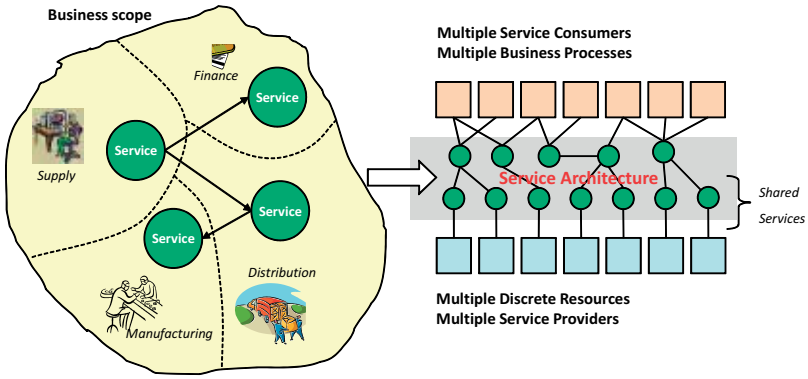

Figure 1.1: An illustration of an applied service-oriented architecture (SOA) on a business model (Source: Tieto AB)

SOS are expected to provide a way to expose legacy functionality to remote clients throughout existing or third-party software assets and at the same time to reduce the overall IT expenses [2, 3].

Figure 1.1 depicts a solution in which a business model is structured and exposed to the user using a service-oriented architecture (SOA). SOA organizes such a system as a set of capabilities that are offered as services. A service is available for use by multiple service consumers, and at the same time it is able to serve multiple business processes. It virtualizes how a specific capability is performed, and where and by whom the resources are provided, enabling multiple service providers to participate together in shared business activities.

SOS assume services as their basic functional units, independent of any specific implementation platform, capable of being published, invoked, composed and destroyed on demand. In such systems, it is challenging to ensure the expected level of quality-of-service (QoS) required in case the user needs to select one of many functionally similar services. To guarantee the required level of QoS, some of the existing SOS frameworks provide formal analysis of a mathematical model of the SOS [4–7]. In most cases building the formal model to be analyzed is not a straightforward process and it requires a user to master not only specification, but also transformation techniques.

The design and analysis of SOS needs to cater for two different perspectives: the developer's and the user's. Assuming the former, one needs to gain insight into the service functionality representation, enabled actions, resource annotations, and possible interactions with other services, all represented as a *service behavioral description*. For the user's view, such a description is not needed, instead the service interface needs to be visible. The SOS paradigm assumes that new systems and applications are built by reusing already existing services, providing the reusable functionality via well-defined interfaces. Once systems and applications are built, it becomes crucial to be able to check the fulfilment of defined requirements of the employed services, both in isolation, as well as in the context of the newly created service compositions. An important aspect, many times ignored, is the service's resource usage. Any analysis approach that abstracts from service resource constraints might produce analysis results that are insufficiently correct, or reliable.

The goal of this thesis is to provide methods and tools for the specification, modeling, and formal analysis of services and service compositions in SOS. Relying on the fact that SOS have similar characteristics with component-based systems (CBS), one could think of reusing an existing component-based framework for designing service-oriented software. Embracing this view, in this thesis we introduce an extension of the existing behavioral modeling language REMES, which has been designed to fit a component-based design perspective [8, 9]. Our extension enhances REMES to enable the graphical description of internal service behavior, in terms of actions, resource annotations, timing constraints, possible interactions with other services, etc., but also lets the designer to specify its interface as a set of service attributes (i.e., service type, service capacity, time-to-serve, status, service precondition, and postcondition, respectively). REMES is a state-machine based behavioral language suitable for abstract modeling, with support for hierarchical modeling, has an input/output distinction, a well-defined formal semantics, and tool support for modeling and formal analysis of SOS [10, 11]. A REMES service can be described in terms of modes that can be either atomic if they do not contain any submode, or composite if they contain a number of submodes, but can also be employed in various types of compositions, resulting in more complex services. The language supports sequential, parallel, or synchronized composition of services that is enabled through the special type of REMES mode, called AND/OR mode. In CBS the system architecture is imposed by the component model's

rules of inter-connection, yet for SOS there is no assumed underlying component model to define an architecture, so composition can be handled by operators (beside parallel composition) with formal semantics, which can be used to model service compositions. Our extensions introduce service-oriented features, aiming at making REMES suitable for behavioral modeling and analysis of SOS, too. Thomas Erl recognizes two stages during service life-cycle [12]. The first deals with service candidates, at design-time, where a developer can change and improve both functional and extra-functional properties of a service [12]. The second assumes a service that is already visible to service users and ready to be deployed. In this thesis we focus on service candidates that can still be analyzed in order to predict their possible future behavior. However, in the remainder of the thesis we call them services, assuming only design-time services.

Nowadays, one of the best known and most used formal analysis technique is model-checking [1]. The essence of model-checking is its ability to automatically verify finite-state system properties for all system behaviors. The analysis process starts with an automata model of a system describing possible system behaviors fed into a model-checking tool or a verifier, together with a desired property. Properties to be examined are typically expressed in a temporal logic. The tool automatically passes through the system's state space in an exhaustive manner, and provides an answer regarding the defined property. In case that the property is satisfied, the tool finishes the verification successfully, otherwise, it reports one of the traces that violates the property as a counter-example to the model. For reachability properties that check whether a given state formula possibly can be satisfied by any reachable state, a trace is reported when the property is satisfied. The benefit of such an analysis process is the fact that one can refine the model and reapply model-checking as many times as needed. In this thesis we apply model-checking techniques for the formal analysis of services and service compositions, given that the formal semantics of REMES language is defined in terms of timed automata (TA) and priced timed automata (PTA).

One of the main principles of SOS is the idea of composing services by discovering and selectively invoking them rather than building the whole application from scratch, at design-time. Therefore, as soon as services are connected, the validity and correctness of the result need to be analyzed. For instance, let us assume that a user needs a service that

is composed from several navigation services, where some services return a route length in miles and some in kilometers. If the developer omits to introduce a service that converts length from one metrics to the other, the error can be detected by formally checking the correctness of the actual composition, as soon as the composition is formed. Also, services that are functionally similar might differ in extra-functional attributes, such as time and resource-usage making them more (or less) suitable for particular users and applications. In such cases, it is also beneficial to be able to provide information regarding the minimum (or maximum) time needed for a service or a service composition to finish the given task, or the minimum (or maximum) total resource consumption of a service or service composition. In order to make services (or service compositions) comparable with respect to resource consumption, we assume a cost modeled by a weighted sum of the consumed resources.

To verify the correctness and quality of services and service compositions we use the forward analysis technique based on the computation of the strongest postcondition of a REMES service with respect to a given precondition [13]. To prove the correctness of a REMES service in isolation, we check that the calculated strongest postcondition is no more than the given requirement. The strongest postcondition technique assumes Dijkstra's and Sholten's strongest postcondition semantics [13] that lets us reduce proving correctness of services and service compositions to boolean implications. The actual strongest postcondition is then calculated algorithmically, with services modeled as PTA. We consider the service resource consumption in REMES as a cost variable in PTA, and alongside our strongest postcondition calculation, we include, in our algorithms, well known approaches for computing the minimum and maximum reachability cost [14].

In SOS the same service may be offered at various prices, QoS and other conditions, depending on the customer needs. In such a setting, the interaction between parties involves the negotiation of what is possible at request time, aiming at meeting needs on demand. Therefore, the a priori analysis of possible negotiation strategies facilitates insights into what can be achieved under each strategy, and possibly compute optimal values of price, resource consumption, etc., or maximized value of the utility function (a weighted sum of negotiation preferences) for all possible behaviors of the involved parties. Within SOS, services can act as clients, mediators, or providers, respectively, as depicted in Figure 1.2. The role of a client service is to require a service that performs

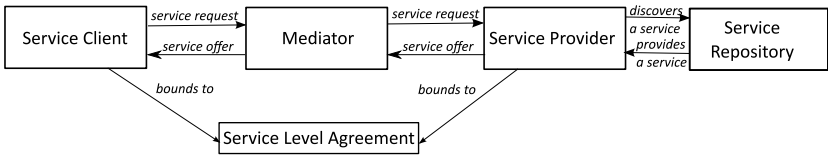

Figure 1.2: A model of the negotiation process

a specific task within given resource limits. The mediator initiates and steers the communication, that is, the negotiation process between the client and provider, helping them to reach the agreement. The service provider creates a counteroffer, based on the client's request and on the available services.

The negotiation process is an iterative process that, if successful, ends up with service level agreement (SLA). SLA is a contract between the client and the provider that sets boundaries on both the functional and extra-functional properties of a service, which are to be guaranteed, defines the cost of a service delivery and possible penalties in case that the contract is broken. In this thesis we propose an analyzable negotiation model between service clients and service providers. The model is based on the set of Remes interface operations that support Remes service composition, which we have recently proposed [15].

In brief, the contribution of this thesis is a framework for specification, modeling and formal analysis of services and service compositions in SOS, which includes: (i) an extension of a suitable behavioral language, called Remes [8], to describe functional and extra-functional properties (i.e., timing-, and resource-wise behavior) of services and service compositions (via hierarchical composition textual language), associated with analysis techniques for various properties (functional, extra-functional, timing, etc.); (ii) a Hoare-triple model of service correctness equipped with correctness check via model-checking algorithms of networks of PTA, which compute the strongest postcondition of given automata networks together with the minimum (or maximum) cost of the service resource consumption; (iii) an analyzable negotiation model between service clients and service providers; (iv) a design tool for graphical modeling of service-based systems accompanied with textual service description supporting modeling in our proposed behavioral language; (iv) validation of the framework on relevant case-studies;

In the rest of this chapter, we provide the outline of the thesis (Sec-

tion 1.2).

1.1 Thesis Outline

The thesis is organized in two parts. The first part provides a summarized description of the research. Chapter 1 describes the motivation for the conducted research. Chapter 2 introduces important technical concepts used throughout the remainder of this thesis. Chapter 3 formulates the main research goal, introduces the research subgoals, and the research method that we use. Chapter 4 describes the research results and recapitulates the research goals. Chapter 5 surveys related work. Finally, Chapter 6 concludes the thesis, summarizes the contributions and outlines future work.

The second part consists of a collection of peer-reviewed conference, and workshop papers, presented below, contributing to the research results.

Paper A. "Towards a Unified Behavioral Model for Component-Based and Service-Oriented Systems". Aida Čaušević, Aneta Vulgarakis. In Proceedings of 2nd IEEE International Workshop on Component-Based Design of Resource-Constrained Systems (CORCS2009), pages 497-503, IEEE Computer Society Press, Seattle, USA, July, 2009.

Summary: This paper overviews the general characteristics of both SOS and CBS, pointing out the similarities and differences between them. We show how an existing component framework could be effectively used to model and analyze SOS constituent services. We assume the existing model REMES as being the underlying model of modeling functional and extra-functional behavior of services, as well as their interface assumptions and guarantees. For this to become applicable, we first identify the specific constructs that we need to equip REMES with, such that our goal is achieved. The benefit of REMES language is mainly the fact that it is abstract enough and ready to use even when no detailed system description exists. The modeling part includes also resource annotations on corresponding edges and modes. By transforming REMES to PTA, one can conduct rigorous, formal analysis on REMES models against functional as well as extra-functional (timing, resource-aware) properties. The model also benefits from a recently im-

plemented tool-chain for simulation and automatic transformation into PTA. The paper's small case-study is used to illustrate the modeling process within REMES.

Contribution: This paper was written with equal contribution from both authors. My responsibility was related to the description of SOS, identifying their characteristics, and the necessary concepts that would be needed for SOS modeling in behavioral language called REMES . With Aneta Vulgarakis I have shared responsibility for modeling an illustrative example of an ATM machine in REMES.

Paper B. "Modeling and Reasoning about Service Behaviors and their Compositions". Aida Čaušević, Cristina Seceleanu, Paul Pettersson. In Proceedings of 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA2010); Formal Methods in Model-Driven Development for Service-Oriented and Cloud Computing track, pages 82-96, Lecture Notes in Computer Science, Springer, Heraklion, Greece, October, 2010.

Summary: In this paper, we introduce necessary extensions to the already existing behavioral language, called REMES, to make it fit to the service-oriented paradigm. In REMES, the smallest unit used to represent a single service, is a mode. We add to a REMES mode service specific attributes deemed useful for service discovery, and we also define and describe the semantics of REMES service compositions. The notion of mode is extended with attributes such as: service type, service capacity, time-to-serve, service status, service pre-, and postcondition. When all these attributes are published, a service becomes visible and ready to be used or composed with other services to achieve the given user requirement. To provide means for service composition, the paper proposes a hierarchical textual service composition language. The language facilitates modeling of sequential, parallel or synchronized services. It takes into account the services to be composed, type of binding between them, and a requirement given by the service user. For a small case-study described in this paper, we show the service composition correctness checking by manually calculating the strongest postcondition of a program model expressed in terms of guarded commands language.

Contribution: This paper was written as equal contribution of all

three authors. I have particulary worked on the development of the hierarchical language for service composition, and specified, modeled, and analyzed the correctness of service compositions for an autonomous shuttle system presented as the example in the paper.

Paper C. "Checking Correctness of Services Modeled as Priced Timed Automata". Aida Čaušević, Cristina Seceleanu, Paul Pettersson. In Proceedings 5th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA2012); Quantitative Modeling and Analysis track, pages 308-322, Lecture Notes in Computer Science, Springer, Heraklion, Greece, October, 2012.

Summary: In this paper, we describe an algorithm for checking the correctness of services formally defined as PTA, by employing forward analysis technique that assumes computation of the strongest postcondition of automata, with respect to a given precondition. Our algorithm is inspired by already existing approaches for computing the minimum and maximum reachability costs, respectively [16]. Proving the correctness of a service reduces to showing that the calculated strongest postcondition and minimum (or maximum) cost of resource consumption implies a requirement defined by a user. Hence, our algorithm provide automation to calculating the strongest postcondition of a service, such that only the boolean implication remains to be discharged. The approach is demonstrated on a small accompanying example. Also, we illustrate resource consumption calculation using priced zones for a service modeled in the example.

Contribution: I was the main driver and principal author of this paper. I have contributed with developing algorithms for checking the correctness of services. All the coauthors have contributed with valuable discussions and reviews.

Paper D. "An Analyzable Model of Automated Service Negotiation". Aida Čaušević, Cristina Seceleanu, Paul Pettersson. In Proceedings of 7th International Symposium on Service Oriented System Engineering (IEEESOSE13); Service Runtime and Management track, pages 125-136, IEEE Computer Society Press, San Francisco, USA, March, 2013.

Summary: In this paper, we have introduced an approach for model-

ing and analysis of service negotiation between two or more parties based on an iterative form of the Contract Net Protocol for web services. Our model is an analyzable high-level description of the negotiation between service clients and providers, which characterizes SOS. The model has an implicit notion of time, and supports annotations in terms of price, quality, or other parameters, all modeled by the REMES textual service composition language, called HDCL. The crux of the model is that it has a formal timed automata semantics, which lets one verify various model properties, for all possible executions, which is not achievable in principle by any simulation or testing technique. The model is illustrated in the car insurance example, for which we have shown how to analyze the negotiation model against safety properties, but also against specified timing and utility constraints.

Contribution: I was the main driver and principal author of this paper. I have contributed with specifying the negotiation model and later on applying and analyzing it within the given example. All the coauthors have contributed with valuable discussions and reviews.

Paper E. "Distributed Energy Management Case Study: A Formal Approach to Analyzing Utility Functions". Aida Čaušević, Cristina Seceleanu, Paul Pettersson. MRTC technical report, ISSN 1404-3041, ISRN MDH-MRTC-279/2013-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, October, 2013.

Summary: In this paper, we present a case-study where our recently introduced approach for automated service negotiation in REMES has been applied to model and analyze distributed energy management. We have modeled one energy consumer, two energy providers and one mediator that represents the interests of all negotiation participants. In the study we have observed a single day at the energy market in which the energy supply is based on a negotiation carried out between consumers and providers in possibly more than one round, assuming a certain strategy. We have focused on three scenarios: (i) a client cannot exceed a fixed maximum price bound; (ii) a client does not have an imposed maximum price value; (iii) a client adapts the maximum price trying to get as close as possible to the offered price, but without paying more than the initial price's double. The provided study has been analyzed by semantically translating the REMES-based models into a network of TA to

enable model-checking in the UPPAAL tool. As the result of the analysis, we have calculated the value of the optimal utility function described as a as a weighted sum of negotiation preferences, i.e., price and the energy reliability, given as a number. By model-checking, we have obtained the trace that leads to such state. The negotiation model is time constrained, which lets one get an insights in the duration of a negotiation, under specific strategies, which can form a base for strategy comparison. Based on the verification results, the participants have spent the most time to converge towards the final agreed price in scenario 3, possibly due to the fact that the client needed to recalculate new prices compared to the previous offers, and further, the mediator had to ask for the new offers, always from all providers. Also, verification shows that in scenario 1, there exists a case in which no agreement has been reached, since the initial requested and offered prices were too far from each other, and since the customer had an upper bound on the price. We were also able to see who owns the market in which scenario based on the collected analysis results. In scenario 1, despite the introduced maximum price bound, the providers were able to force the final prices in their favor. Similar situation was in scenario 2, but this time with more freedom with respect to the maximum acceptable price on consumer's side. In scenario 3, the price gain was in favor of consumer. Based on these finding, we were able to conclude that overall, the total amount of money spent in the energy negotiation process is very close to the participants' budgets, with an average increase of less than 10% of the initially requested price.

Contribution: I was the main driver and principal author of this paper. I have contributed with specification, modeling, and formal analysis of the case-study. All the coauthors have contributed with valuable discussions and reviews.

Paper F. "A Design Tool for Service-oriented Systems". Eduard Paul Enoiu, Raluca Marinescu, Aida Čaušević, Cristina Seceleanu, In Proceedings of 9th International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA12), volume 295, pages 95-100, Electronic Notes in Theoretical Computer Science, Elsevier, Tallinn, Estonia, May, 2013.

Summary: In this paper, we present a modeling and analysis tool for service-oriented systems. The tool enables the graphical modeling

of service-based systems, within the resource-aware timed behavioral language REMES, as well as the textual system description. We have developed a graphical environment where services can be composed as possibly desired by the user, together with a textual service composition interface in which compositions can also be checked for correctness. We also provide automated traceability between the two design interfaces, which results in a tool that enhances the potential of system design by intuitive service manipulation. The paper presents the design principles, infrastructure, and the user interface of our tool.

Contribution: This paper is a result from the master thesis work conducted by Eduard Paul Enoiu and Raluca Marinescu, which were the main contributors to this paper. I have been the thesis supervisor, while Cristina Seceleanu has been the examiner. Together with Cristina Seceleanu I have been a driver for the thesis work. All the coauthors have contributed with writing, discussions and reviews.

1.2 Publications related to the thesis

Licentiate Thesis

- *Formal Approaches to Service-Oriented Design: From Behavioral Modeling to Service Analysis.* Aida Čaušević. Licentiate Thesis, ISBN 978-91-7485-012-3, Mälardalen University Press, June, 2011.

Journals

- *Applying* REMES *Behavioral Modeling to PLC Systems.* Aneta Vulgarakis and Aida Čaušević. Mechatronic Systems, volume 1, number 1, pages 40-49, Faculty Of Electrical Engineering, University Sarajevo, December, 2009.

Conferences and workshops

- *Analyzing Resource-Usage Impact on Component-Based Systems Performance and Reliability.* Aida Čaušević, Paul Pettersson, and Cristina Seceleanu. Proceedings of International Conference on Innovation in Software Engineering (ISE2008), pages 302-308, IEEE Computer Society, Vienna, Austria, December, 2008.

- *Applying* REMES *Behavioral Modeling to PLC Systems.* Aneta Vulgarakis and Aida Čaušević. 22nd International Symposium on Information, Communication and Automation Technologies (ICAT 2009), IEEE Xplore, Sarajevo, Bosnia and Herzegovina, October, 2009.

- *Behavioral Modeling and Refinement of Services.* Aida Čaušević, Cristina Secelanu, and Paul Pettersson. Proceedings of 21st Nordic Workshop on Programming Theory (NWPT2009), pages 98-102, Lyngby, Denmark, October, 2009.

MRTC reports

- *Formal Reasoning of Resource-Aware Services.* Aida Čaušević, Cristina Secelanu, and Paul Pettersson. MRTC report, ISSN 1404-3041, ISRN MDH-MRTC-245/2010-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, June, 2010

- *Algorithmic Computation of Strongest Postconditions of Services as Priced Timed Automata.* Aida Čaušević, Cristina Secelanu, and Paul Pettersson. MRTC report, ISSN 1404-3041, ISRN MDH-MRTC-263/2012-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, April, 2012.

Chapter 2

Preliminaries

2.1 Service-Oriented Systems

The service-oriented paradigm has been around for almost two decades now. The first recorded successful implementation of service-oriented principles was within Wells Fargo Bank, that in 1995 became the world's first internet bank [17] and that proved the competitive advantages of service-oriented architectures. The paradigm has become very popular since it has offered features such as interoperability, platform independence, access and communication via well-defined interfaces, and in many cases tool support to ease the process of integration for legacy systems [2, 3]. Since then we have witnessed the significant growth in software and system functionality packed in form of services and accessible throughout the network infrastructure, be it open or proprietary network environment. During these years the use of SOS has been spread within telecommunication, health, government domain, car industry and many more [18–22]. All of these domains have easily adopted a newly introduced concept and coped successfully with limitation such as security, verification and validation of the newly composed system, application ownership, issues related to the existing internet protocols, etc., that SOS brought in.

So far, we have seen a gradual evolution from the first generation of SOS, based on monolithic services capable to be reconfigured only at compile time. They have been followed by second generation of services that have brought even more flexibility being able to provide adaptation

and reconfiguration at installation. The third generation has introduced in an autonomic and ad-hoc reconfiguration possibilities [23].

In SOS, services are utilized to offer rapid and low-cost development of distributed applications in heterogeneous environments. Services are assumed to be self-contained (i.e., being able to maintain its own state independently of the application that uses it), platform independent (i.e., being capable to be invoked by any client no matter what network, hardware, and software platform the client uses, but also to have well-defined interfaces that are distinct from service implementation), and dynamically discoverable, invocable, and composable [24].

In the literature there is many informal definitions for the term "software service". In the OASIS Service-oriented Architecture Reference model [25] it is defined as:

> *A mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by an entity - the service provider - for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider.*

One may say that SOS offer cost-efficient software development by reusing functionality from available services. Also, a service becomes a single point of maintenance for a common functionality. Using discovery mechanisms that are dedicated service discovery protocols used to enable automatic detection of the available services based on their published interface information, developers can find and take advantage of existing services, significantly reducing time to develop new systems. If the quality of the already existing and reused services is already guaranteed, the verification process for a newly established services might require a lower effort and less time. Services can be seen as adaptable units, thanks to the clear separation between the service interface and service behavior, making it possible to employ incremental deployment of services.

Although SOS promise huge gains as it is based on principles of coarse-grained, loosely coupled, interoperable, and reusable services, there is also a number of challenges related to design and analysis of such software units. It still remains a challenging task to predict QoS, since the

system's QoS is not a function of the QoS of the services only. It also involves interdependencies between services, resource constraints of the environment, and network capabilities. Additionally, checking the correctness of service compositions lacks appropriate methods and tools especially for extra-functional properties like resource-wise behavior. With a growing number of services, offered by different service providers/vendors, the need to formally define and analyze the service negotiation process has increased. The services available in a service repository might have the same functionality but differ in extra-functional qualities response time, time-to-serve, or price. Assuming that the service is negotiated upon the client's request, establishing guarantees on the provided QoS, under changing conditions of service composition, and possible negotiation constraints, becomes a difficult task.

Nowdays a number of service-oriented approaches exist [4–6, 26–28]. All of them have the basic service-oriented concepts incorporated like discovery mechanisms, support for orchestration and choreography, some predictability for service performance, reliability, etc., but only few can deliver the whole process from creating single service to system development, including some means for analysis. It is obvious that this paradigm of SOS still remains to be fully explored, developed, and utilized.

2.2 REMES: A Resource Model for Embedded Systems

REsource **M**odel for **E**mbedded **S**ystems (REMES), introduced in [8], describes the resource-aware behavior of interacting embedded components. The language was initially designed to fit a component-based design perspective [9]. It is a dense-time state-based hierarchical modeling language, suitable to address functional and extra-functional behavior of components or services.

In REMES, the internal component behavior is described by a *mode* that can be either *atomic* (does not contain submode(s)), or *composite* (contains submode(s)) (see Figure 2.1). Additionally, there is also a special type of mode called *non-lazy* whose semantics will be described in the following. The language supports hierarchy of the arbitrary depth, however in this chapter we show the implementation of a two-level hierarchy only.

The data transfer between modes is done through the *data interface*

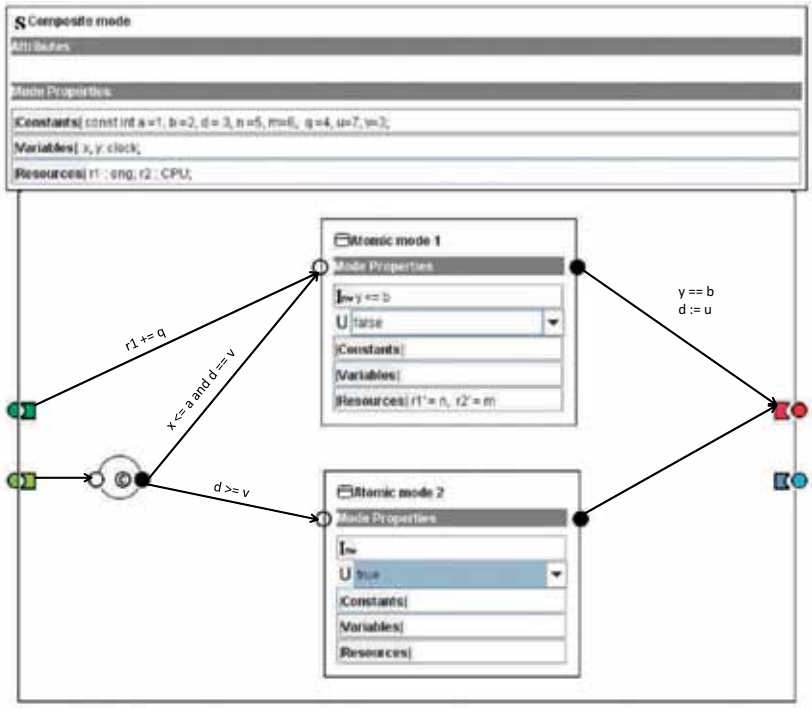

Figure 2.1: A REMES mode

(i.e., typed global variables), while the (discrete) control is passed via the *control interface* (i.e., entry and exit points). A REMES mode has a "run-to-completion" semantics, meaning that no interruptions are supported, and internal loops are not allowed. However, a special type of variable called *history variable* is used to control possible interleavings between computation of two composite modes. On the other hand, iteration is enabled by introducing two kinds of exit points: Write and Exit. The first kind of exit point is used for modeling both internal mode computations that are resumed until the Exit is reached, which signals the termination, but also continuously active behaviors, which might never terminate. A mode describing active behavior can also be exited through the Exit point. REMES assumes local or global variables that can be of types boolean, natural, integer, array, string, list, clock (continuous variables evolving at

rate 1), and of type resource, which are nonnegative real-valued variables that model continuous resource behavior. Resource variables are of type energy (eng), processor load (CPU), bandwidth (bdw), memory (mem), or communication ports. The submode Atomic mode 1 in Figure 2.1 is annotated with its resource-wise continuous behavior, assuming that the corresponding component consumes resources (r1 : eng, r2 : CPU). Such consumption is expressed by the first time derivative of the typed resource variables, respectively r1', r2', which give the rates at which the composite mode consumes resources in time, depending on the executing submode.

The control flow in a composite mode is given by a set of directed *edges*. Edges connect the control points of the submodes, or the composite mode and its submodes. In Figure 2.1, the composite mode takes the edge from control point Init to enter Atomic mode 1, after initialization, and in similar manner, edge from Atomic mode 1 to exit the mode.

The REMES language supports two types of actions, *delay (or timed)* actions and *discrete* actions. The first describe the continuous behavior of the mode, and its execution does not change the mode, while the latter are instantaneous actions, whose execution results in a mode change. A mode can be marked as an *urgent mode*, decorated with letter U, if it is exited instantaneously when possible (Atomic mode 2 in Figure 2.1).

A composite mode executes by performing a sequence of discrete steps, via actions that, once executed, pass the control from the current submode to a different submode. An action, A = (g, S) (e.g., (y == b, d := u) in the figure), is a statement S (in our case d := u), preceded by a boolean condition, the guard (y == b), which must hold in order for the action to be executed and the corresponding outgoing edge taken. A REMES composite mode may contain *conditional connectors* (decorated with letter C) that allow a possibly nondeterministic selection of one discrete outgoing action to execute, out of many possible ones. In Figure 2.1, via C, one of the empty statement actions, ($x \leq a \wedge d == v$) or ($d \geq v$) can be chosen for execution. Modes may also be annotated with *invariants* (e.g., $y \leq b$ in Atomic mode 1), which bound from above the current mode's delay/execution time. Before the invariant stops to hold, the current mode is exited.

To enable formal analysis, REMES models can be semantically transformed into timed automata (TA) [29], or PTA [30], depending on the

analysis goals. The REMES language benefits from a set of tools[1] for modeling, simulation and transformation into TA and PTA, which could assist the designer during system development. For a more thorough description of the REMES model, we refer the reader to [8].

Analysis Model for the REMES language

Let us assume that each REMES service has access to a set of resources $R_1, \ldots, R_n$. In order to analyze various scenarios of the service and service compositions resource consumption, in this thesis we encode the service resource as a weighted sum as shown in Eq. 2.1, and presented in [31].

$$r_{total} \quad = \quad w_1 \times r_1 + w_2 \times r_2 + \ldots + w_n \times r_n, \tag{2.1}$$

in which variable r_{total} stands for the total consumption of resources $R_1, \ldots, R_n$, and variables $r_1, \ldots, r_n$ denote the accumulated consumption of $R_1, \ldots, R_n$, respectively. The constants, $w_1, \ldots, w_n$ (weights), denote the relative importance of $r_1, \ldots, r_n$. The choice of values for the weights is subjective matter and depends both on the application and the analysis goals.

In order to be able to formally analyze REMES services and service compositions, we need a semantic translation of the model. Assuming the rules of transforming REMES models into TA or PTA networks, introduced in previous work on REMES [10, 32–34], then $r_1, \ldots, r_n$ can be modeled as cost variables $c_1, \ldots, c_n$ and analyzed within the PTA framework with UPPAAL or its variant CORA [35–37].

2.3 Formal Modeling and Analysis of Software Systems

Formal methods are a particular kind of mathematically rigorous techniques, described as well-formed statements in a mathematically precise way, in many cases supported by tools, which enable specification, development, and verification of software and hardware systems. Formal verification is a technique that provides means to prove or disprove a system model's correctness with respect to a formally specified property.

[1] The REMES tool-chain is available at `http://www.fer.hr/dices/remes-ide`.

This means that, by formally verifying a system model, one checks that the latter indeed behaves according to a specified property. As a result of formal analysis conducted using formal verification, one can get either qualitative answers (yes/no) that are a result of verification of properties that can be either satisfied, or not; or quantitative analysis results (numbers) that, in our case, represent the minimum (or maximum) value of the accumulated resource usage for reaching a given goal, but in a more general context, could mean reliability estimates, performance estimates, or similar.

Today the two most popular formal analysis methods are model-checking and theorem-proving. In theorem proving a system that is about to be verified is modeled as a set of mathematical definitions in some formal logic. The desired properties of such system are derived as theorems that follow from these definitions. The process of properties derivation can be long and tedious, therefore techniques have been developed to automate much of this process by using computers to handle steps in the proof. On the other hand, model-checking is a verification technique that enables exploration of all possible system states, for a given model. The result of such a exploration for a defined property is returned in a form of "satisfied/not satisfied" answer. It can also often provide a run of the model showing how a given property is satisfied or violated, which brings more insight into the model and provides a chance to improve it if found necessary. A model that describes all the possible system behaviors is represented in terms of an automata model while properties to be verified are expressed in a temporal logic. The essence of model-checking is its ability to automatically verify finite-state system properties for all possible system behaviors. The properties to be examined have to be precisely and unambiguously defined. Since the technique is completely automatic and capable of detecting counter examples, model-checking is also suited for uncovering and correcting errors, if a given model fails to satisfy the specified property. In case that counter example is detected, one might modify the system model and run the model-checking again. Furthermore, in cases when the system's desired behavior is satisfied, one can refine the model and reapply model-checking. Figure 2.2 depicts a generic example of model-checking and includes all steps that the technique follows.

The properties to be examined can be specified using of Computation Tree Logic(CTL) [38]. CTL is a specification language for finite-state systems (Kripke structures) that enable reasoning about sequences of

events. The model-checking problem reduces to checking that for a given model M, initial state $\mathsf{s} \in \mathsf{S}$, where S is the set of all model states, and CTL-formula ϕ, $M, s \models \phi$ is satisfied.

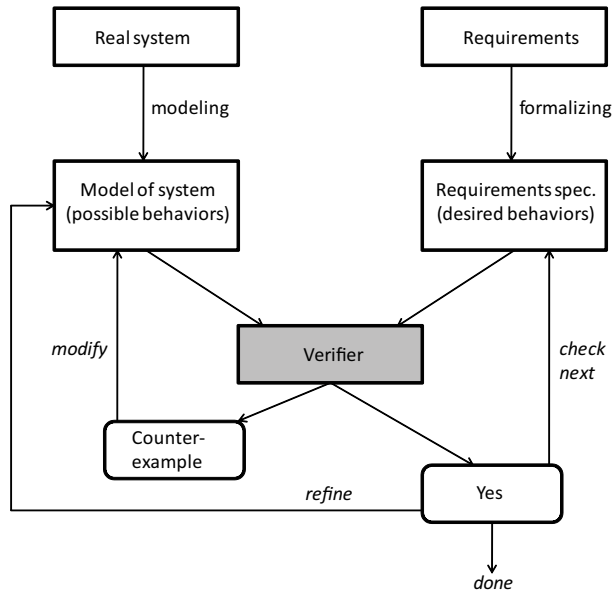

Figure 2.2: Verification methodology of model checking [1]

Services in SOS are assumed to be invoked, composed, and decomposed on a user demand, in many cases in an automatic and ad-hoc manner. Due to such constraints, a designer of such systems is in need to have available methods and tools that support modeling and verification of the system behavior, as soon as it is constructed. In this thesis, we have chosen the framework of TA and PTA as our modeling framework, and the UPPAAL-based tools [2] as the model-checkers for verifying the system's property specified in Timed Computation Tree Logic (TCTL) [39], an extension of CTL [38] with clocks.

In the following, we briefly describe the models of TA [29] and PTA [16, 40], an extension of TA with prices on both location and edges.

[2] For more information about the UPPAAL tool, visit the web page www.uppaal.org.

2.3.1 Timed Automata

The model of timed automata was introduced by Alur and Dill in 1990s [29], as a model for real-time systems. A timed automaton (TAn) is a finite-state machine enriched with a set of clocks. All clocks in a system are synchronized and assumed to be real-valued variables, measuring the time elapsed since their last reset. The UPPAAL tool set extend the standard framework of TA, by introducing data variables. In this thesis we use TA defined in terms of UPPAAL framework [35, 41].

Formally, let us assume a finite set of real valued variables C ranging over x, y, etc., standing for clocks and V a finite set of all data (i.e., array, boolean, or integer). A clock constraint is a conjunctive formula of atomic constraints of the form $x \sim n$ or $x - y \sim n$ for $x, y \in C, \sim \in \{<, \leq, =, \geq, >\}$ and $n \in N$. The elements of $\mathcal{B}(C)$ are called *clock constraints* over C. Similarly, we use $\mathcal{B}(V)$ to stand for the set of *non-clock constraints* that are conjunctive formulas of $i \sim j$ or $i \sim k$, where $i, j \in V$, $k \in \mathbb{Z}$ and $\sim \in \{<, \leq, =, \neq, \geq, >\}$. We use $\mathcal{B}(C, V)$ to denote the set of formulas that are conjunctions of clock constraints and non-clock constraints.

Definition 1 (***Formal Definition of a Timed Automaton***). *A Timed Automaton A is a tuple $(L, l_0, C, V, I, Act, E)$ where:*

- *L is a finite set of locations,*
- *l_0 is the initial location in L,*
- *C is a finite set of clocks,*
- *V is a finite set of data variables,*
- *$I : L \rightarrow \mathcal{B}(C)$ assigns invariants to locations,*
- *$Act = \Sigma \cup \{\tau\}$ is a finite set of actions, where Σ is a finite set of synchronizing actions, and $\tau \notin \Sigma$ denotes internal or empty actions without synchronization,*
- *$E \subseteq L \times \mathcal{B}(C, V) \times \Sigma \times R \times L$ is a finite set of edges, where $\mathcal{B}(C, V)$ denote the set of guards and R denotes the (clock) reset set i.e., assignments to manipulate clock- and data variables.*

In the case of $(l, g, a, r, l') \in E$, we write $l \overset{g,a,r}{\rightarrow} l'$, where l is the source location, l' is the target location, g is a guard, a boolean condition that

must hold in order for the edge to be taken, a is an action, and r is a simple clock reset. ■

The semantics of a TAn is defined in terms of a labeled transition system. A state of a TAn depends on its current location and on the current values of its clocks. So, a state of a TAn is a pair (l, u), where l is a location, and $u : C \to R_+$ is a clock valuation. The initial state (l_0, u_0) is the starting state where all clocks are zero. There are two kinds of transitions: delay transitions and discrete transitions.

*Delay transition*s are the result of time passage and do not cause a change of location. More formally, we have

$$(l, u) \xrightarrow{d} (l, u \oplus d)$$

if $u \oplus d' \models I(l)$ for $0 \leq d' \leq d$. The assignment $u \oplus d$ is the result obtained by incrementing all clocks of the automata with the delay d.

Discrete transitions are the result of following an enabled edge in a TAn. Consequently, the destination location is changed from the source location to the new target location, and clocks may be reset. More formally, a discrete transition

$$(l, u) \xrightarrow{a} (l', u')$$

corresponds to taking an edge $l \overset{g,a,r}{\to} l'$ for which the guard g is satisfied by u. The clock valuation u' of the target state is obtained by modifying u according to updates r such that $u' \models I(l')$.

Definition 2 (*Run of a Timed Automaton*). *A run of a timed automaton $A = (N, l_0, E, I)$ with initial state (l_0, u_0) over a timed trace $\xi = (t_1, a1)(t_2, a2)(t_3, a3) \ldots (t_n, a_n)$ is a sequence of transitions:*

$$\xi = (l_0, u_0) \xrightarrow{d_1} \xrightarrow{a_1} (l_1, u_1) \xrightarrow{d_2} \xrightarrow{a_2} \ldots \xrightarrow{d_n} \xrightarrow{a_n} (l_n, u_n)$$

satisfying the condition $t_i = t_{i-1} + d_i$ for all $i \geq 1$.

■

To model a concurrent real-time system, several TA can be composed by CCS parallel composition operator $A_1 \| ... \| A_n$ of a set of an individual automata $A_1, \ldots A_n$. CCS parallel composition operator allows an individual automaton to carry out internal actions (i.e., interleaving) as well as pairs of automata to perform hand-shake synchronization.

Definition 3 (***Network of Timed Automata***). *Let* $N = \{1, \ldots, n\}$ *and let* $A_i = (L_i, l_{0_i}, C_i, V_i, Act_i, E_i)$ *be a timed automaton for* $i \in N$. *A network of timed automata* $A_1 \| ... \| A_n$ *is defined as the parallel composition of n TA.* ■

The hand-shake synchronization is carried out through input and output actions using synchronization channels. Let us assume that S is a set of channel names and that the set of synchronization actions of the TA network is defined as $\Sigma = \{a? \mid a \in S\} \cup \{a! \mid a \in S\}$. A discrete transition with a synchronization action $a! \in \Sigma$ (send synchronization) is only enabled if another automaton in the network is able to perform simultaneously a complementary action $a? \in \Sigma$ (receive synchronization). The channels are divided into: *binary channels* used to synchronize one send with one single receive synchronization, while *broadcast channels* are used to synchronize one sender with an arbitrary number of receiving automata. A state of a network of TA is defined by the locations of all automata in the network and the values of clocks and discrete variables.

To model atomic sequences of actions, UPPAAL supports a notion of *urgent* and *committed* locations. If a location is marked as an urgent location it must take the next transition as soon as it is enabled, i.e., no time passing is allowed, but this does not prevent other actions from happening. On the other hand, when a committed location is active, a transition from a committed location must be taken immediately, and no other transition in other automaton can be taken in between.

The UPPAAL model-checker provides an engine for verification of temporal properties, such as safety and liveness properties, specified in a subset of Timed Computation Tree Logic (TCTL) [39]. To visualize counter examples produced by the model-checker, one can use the simulator.

An Illustrative Example of a Timed Automata

An example of a simple network of TA modeled in UPPAAL is depicted in Figure 2.3. Consider the timed automaton of Figure 2.3 b). It consists of 2 locations l_0 and l_1, where one of the locations is marked as initial (l_0). The edge from location l_0 to location l_1 will only be taken if the guard $x \geq 1$ holds. Additionally, clocks in the automata may be reset on edges. For example, when following the edge from l_1 to l_0 both clocks x and y are reset to 0. Real-valued clocks x and y, initially set to zero, evolve

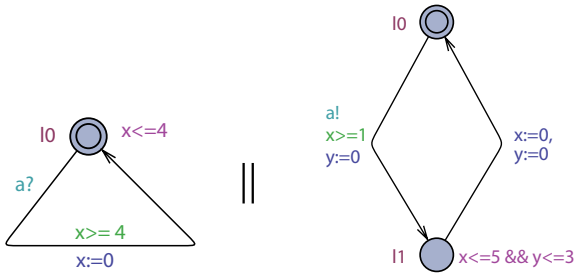

Figure 2.3: A timed automata

continuously at the rate 1. A control node l_1 is labeled with an invariant ($x \leq 5 \wedge y \leq 3$) that defines the maximum time allowed to be spent in that node. The TA in Figure 2.3 a) may stay in location l_0 as long as the invariant $x \leq 4$ is satisfied. The automata shown in Figure 2.3 synchronize via synchronization actions, i.e., by sending and receiving events through a channel a. Sending and receiving via a channel a is denoted by a! and a?, respectively.

2.3.2 Priced Timed Automata

Priced (or weighted) timed automata are timed automata decorated with prices (or costs) on both locations and edges. The cost that annotates an active location represents the cost of a delay transition and it is the product of the duration of the delay and the cost rate of the active location. On the other hand, the cost that annotates an edge represents the cost of the discrete transition and it is given by the cost of the edge.

Definition 4 (***Formal Definition of a Priced Timed Automaton***)**.** *A linearly Priced Timed Automaton (PTA) over clocks C and actions Act is a tuple $(L, l_0, C, V, I, Act, E, P)$, where $(L, l_0, C, V, I, Act, E)$ is a TA and $P : (L \cup E) \to \mathbb{N}$ is a cost function that assigns price-rates (or cost-rates) to locations and prices (or costs) to edges.* ■

Each run in PTA has a global cost, which is the accumulated price along the run of every delay and discrete transition.

The semantics of PTA is defined in terms of priced transition systems over states of the form (l, u), where l is a location, $u \in \mathbf{R}^C$ are clock valuations, and the initial state is (l_0, u_0), where u_0 assigns all clocks in C to zero. In this model, there are two kinds of transitions: *delay transitions* and *discrete transitions*. In delay transitions,

$$(l, u) \overset{d,p}{\rightarrow} (l, u \oplus d),$$

where $u \oplus d$ is the result obtained by incrementing all clocks of the automata with delay d, and $p = P(l) * d$ is the cost of performing the delay. Discrete transitions

$$(l, u) \overset{a,p}{\rightarrow} (l', u')$$

correspond to taking an edge $l \overset{g,a,r}{\rightarrow} l'$ for which the guard g is satisfied by u. The clock valuation u' of the target state is obtained by modifying u according to updates r. The cost $p = P((l, g, a, r, l'))$ is the price associated with the edge.

Definition 5 (***Run of a Priced Timed Automaton***). *A timed trace of a PTA is a sequence of transitions:*

$$\xi = (l_0, u_0) \xrightarrow{a_1,p_1} (l_1, u_1) \xrightarrow{a_2,p_2} \ldots \xrightarrow{a_n,p_n} (l_n, u_n)$$

and the cost of performing $\xi = \sum_{i=1}^{n} p_i$.

■

For a given goal state (l, u), the minimum cost of reaching (l, u) is the infimum of the costs of the finite traces ending in the (l, u), while the maximum cost of reaching the goal state (l, u) is the supremum of the costs of the finite traces ending in (l, u).

Properties of PTA can be specified in Weighted Computation Tree Logic (WCTL) [42], which is an extension of TCTL with resetting and testing cost variables. Let AP be a set of atomic propositions, $a \in AP$, P is a cost function, c ranges over $\mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. Then, a WCTL formula ϕ is defined by the following grammar:

$$WCTL \ni \phi \quad ::= \quad true \mid a \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid E\phi \; U_{P \sim C} \phi \mid A\phi \; U_{P \sim C} \phi \tag{2.2}$$

Here, A and E are the universal and existential path quantifiers of WCTL, respectively, and $U_{P\sim C}$ is the "until" temporal modality. The first operator is either A ("for All paths"), or E ("there Exists a path"). We interpret formulas of WCTL over labeled PTA, that is, PTA having a labeling function that associates with every location l a subset of atomic propositions. The satisfaction relation of WCTL is defined over configurations of the labeled PTA. More details on WCTL can be found in [42].

An Illustrative Example of a Priced Timed Automata

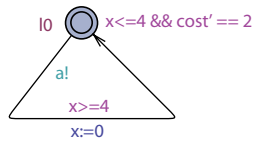

Figure 2.4: A priced timed automaton

Let us assume that the PTA in Figure 2.4 is a clock that periodically synchronizes (every 4 time units, which represents the clock period) with another PTA via channel a. Moreover, we assume that the periodic synchronization consumes a certain amount of energy, modeled here as the cost variable *cost*, which evolves at rate 2. The special variable *cost* is increased by the price per time unit for staying in the location l_0 ($cost' == 2$ indicates that the energy consumption is 2 units per time unit in location l_0).

2.3.3 Formal Analysis of REMES Models

Based on the TA or PTA model as the semantic translation of REMES models we are able to perform feasibility and optimal or worst-case resource consumption analysis. Feasibility analysis checks whether the accumulated values of the resources consumed during all possible system behaviors are within the available resource amounts provided by the system platform. Optimal or worst-case resource consumption analysis returns the cost of the "cheapest", and/or most "expensive" trace of the

TA network that will eventually reach some designated goal expressed as TCTL property.

Considering the TA or PTA model of REMES models, one might express possible analysis goals by the following WCTL properties that TA or PTA networks can be checked against:

$$AF_{t \leq t_{max}}\ final \tag{2.3}$$
$$AG\ (v \Rightarrow AF_{cost \leq n} final) \tag{2.4}$$
$$EF_{cost \leq n}\ final \tag{2.5}$$
$$AG\ (v \Rightarrow EF_{cost \leq n} final) \tag{2.6}$$
$$EF\ (t \leq\ t_{max} \wedge\ contract \wedge\ u \geq val) \tag{2.7}$$

Properties (2.3), (2.4), (2.6) are liveness properties, while properties (2.5) and (2.7) are reachability properties. We say that the first two properties specify *strong feasibility* of the model: property (2.3) requires that for all execution paths, the target location *final* is eventually reached within time t less or equal to the maximum allowed time t_{max}; property (2.4) states that, for all paths, it is always the case that, once in location v, the *cost* of eventually reaching location *final* will be no more than n, regardless of how location *final* is reached. Property (2.5) models *weak feasibility*, meaning that the target location *final* may be reached within a total cost of n. Property (2.6) states that for all paths, it is always the case that once location v is reached, there exists a way by which location *final* will be eventually reached within cost n. The last property (2.7) states that there exists a path in which a contract can be signed (*contract* holds) within the given timing constraints, that is, $t \leq t_{max}$, and maximized utility function ($u \geq val$). It is important to point out that model-checking WCTL formulae is decidable just for one-clock PTA [43]. For other PTA, one can only verify reachability properties of the form given by property 2.5.

In this thesis we assume that the cost function is described as $cost = w_1 \times c_1 + w_2 \times c_2 + \ldots + w_n \times c_n$, where $c_1, \ldots, c_n$ are constants, then in our analysis the above presented properties involve a single cost variable that stands for the accumulated resource consumption of all involved resources. This means that, semantically, the different resources become undistinguishable in these cases (the sum only accounts for all resource values).

Chapter 3

Research Goals and Methodology

The research presented in this doctoral thesis is conducted in the area of service-oriented software development, and it has been driven by problems coming from the domain of SOS. The list includes issues such as service composition, service resource limitation, correctness checking for services and their compositions, service negotiation, and formal analysis of such systems. We tackle the problems at the design level only, so services are design-time units. An important challenge is thus to develop appropriate languages, algorithms, and tools to support modeling, composition, negotiation, and formal analysis of service behaviors.

In this chapter we present the scope of our work by formulating the main research goal, the set of smaller research goals that reflect the idea behind the overall thesis goal, which we actually address in the thesis, as well as the employed research methodology.

3.1 Problem Description

In Chapter 1, we have argued that the development of new applications out of existing or new services is a challenging task. This claim is justified by the fact that new applications and systems are built on demand, often using services provided from possibly different vendors, which should in many cases be compatible with legacy systems. Ad-

ditionally, various available services might provide similar functionality, but differ in resource consumption, price, or reliability. The functional correctness (meeting requirements), but also QoS of selected services, as well as service compositions should be predicted as early as possible in order to prevent possible problems at run-time. Also, it is beneficial to analyze negotiation with different vendors about the terms and conditions of service provision, enabling the open competition between service providers that in many cases leads to more favorable agreements for the service users.

Based on the above arguments, we identify our main research goal coming from the domain of SOS, as the follows:

Provide methods and tools for specification, modeling, and formal analysis of services and service compositions in SOS.

The goal is broad and admits various answers, so in order to address the goal we formulate four smaller research goals (see section 3.2) that we focus on in this thesis.

3.2 Research Subgoals

In this section we present four research subgoals that serve our main research goal, and which we address in this thesis.

Research subgoal 1.

The service-oriented paradigm is a design paradigm that relies on a specific set of design principles such as: service-reusability, autonomy, discoverability, abstraction, statelessness, etc. [12]. Services are the main units of SOS, which can be created, invoked, composed, and decomposed on a user request. Each service provides a predefined collection of capabilities that are grouped together and they relate to the functional context (surrounding environment) in which a service can be invoked. Consequently, all services should advertise their capabilities via well-defined service interfaces that might include information such as type of the service, time-to-serve, status, etc.. Also, since services are platform independent and loosely coupled, it is possible to compose them in more than one way, usually on user demand. In this thesis, we adopt

the developer-user perspective with respect to the information needed for a service: the developer has access to the internal description of a service behavior, whereas the user needs only the service interface description. Hence, our aim is to provide a complete and unambiguous service behavior description that includes modeling both the internal state changes of each specific entity of the system's architecture, and the external interface behavior.

An important target is to ensure the required QoS that is expected by the user when deciding which service to select out of available services (possibly delivering the same or similar functionality). Therefore, it is essential to be able to apply formal analysis techniques (in particular model-checking) on a complete behavioral model of a service, or connected services, at design-time. Some of the existing SOS approaches support formal analysis to guarantee the expected level of QoS [4–7], but few cater for both functional as well as extra-functional (timing and resource-usage) behavior. Accordingly, the first research subgoal can be subdivided into the following two subgoals:

> *To provide a language for describing the relevant features of SOS high-level models, and corresponding tool support.*
>
> *(RSG1A)*

> *To develop a formal model for services, amenable to formal analysis, which provides support for describing the service function, timing, and resource-usage aspects.*
>
> *(RSG1B)*

Research subgoal 2.

Developing systems that do not always have a predefined architecture raises some concerns regarding the quality and correctness of the employed services. When building new services from the scratch one should be able to predict the future behavior of the service, even more so if the service needs to interact with other services in order to carry out the given task. It is obvious that in such a dynamic environment it is not sufficient to check only the correctness of single services, but also to be able to verify the functional and extra-functional correctness of possible service compositions.

Considering the fact that services and service compositions sometimes are embedded into larger systems that need to run on limited resources, it could become important to ensure that the resource-usage of a service, be it isolated or interconnected, is kept within existing bounds. To address such requests at early design stages, one needs powerful analysis techniques that encompass both functional but also extra-functional service behavior. Such motivation justifies our second research subgoal:

> *To provide means to formally analyze functional, and extra-functional properties of services and service compositions within our framework.*
>
> *(RSG2)*

Research subgoal 3.

In principle, services in SOS might be offered at various prices, QoS, and other conditions depending on the customer needs. In such a setting, the interaction between involved parties requires the negotiation of what is possible at request time, aiming at meeting needs dynamically. SOS assume the involved parties to have one of the following roles: client, mediator, or provider. The role of a client service is to request a service that has certain functionality and characteristics, which should deliver its function within given resource limits. The mediator initiates and steers the communication, that is, the negotiation process between the client and the provider, helping them to reach an agreement. The service provider creates a counteroffer, based on the client's request and available services.

For a negotiation to take a place, a negotiation mechanism should exist. The mechanism must be public and defines the rules of the possible client-mediator-provider interaction, and the space of the possible actions that the participants can take, assuming that each participant adopts a private strategy in order to maximize the individual gain. The negotiation process is an iterative process that, if successful, finishes with a formal contract called Service Level Agreement (SLA). SLA is a contract between the client and the provider, which sets boundaries on both functional and extra-functional properties to be guaranteed, defines the cost of a service delivery, and possible penalties in case that the contract is broken.

As the number of services offered by different providers is growing, the need to formally define and analyze the service negotiation process

has increased. Given that service provision is being negotiated upon the client's demand, establishing guarantees of provided QoS, under (most likely) agile conditions of service composition and possible negotiation constraints, becomes a challenging task.

Taking into account the above, a mathematically driven technique to analyze various ways to achieve the client's and provider's goals is beneficial. For instance, one can compute the minimum price for reaching an agreement within a given time constraint, while maximizing the utility function (a weighted sum of negotiation preferences) for all involved parties. Such analysis might expose different configurations that can provide valuable inputs to both service designers as well as service users. Assuming particular offer and counter-offer, strategies of the involved services, as well as a negotiation protocol, the formalization of the negotiation between clients and providers basically results in a "negotiation interface" that iterates until an agreement is reached. Our aim is to provide an analyzable negotiation model between service clients and providers fit to the above described needs. To meet these needs, we need to address the subgoal below:

> *To provide an analyzable model that describes possible interactions between services and their providers, through successive selection of actions.*
>
> *(RSG3)*

Research subgoal 4.

The usefulness, applicability, and scalability of modeling languages and analysis methods for SOS can be exercised by performing their validation against measured, quantified behavioral properties. To be able to validate the applicability of our design framework, we must make sure that it is suitable to real-life problems. Thus, our fourth research goal is:

> *To apply the proposed design framework on a real-life service-based system in order to check its suitability in practice.*
>
> *(RSG4)*

3.3 Research Methodology

Research is a systematic, methodical and ethical process of enquiry and investigation, which aims at solving practical problems and increases knowledge about the topic of research [44]. In order to adequately address the main research goal, it is vital to adopt an appropriate research methodology and set of research methods, suitable for a given setting. The research process that is used in this thesis includes the following steps based on the ones proposed by Shaw [45]:

- Identification and formulation of a general research problem based on the current trends and demands from the SOS community;
- Transferring the problem to a research setting, refining and narrowing down the general problem by expressing it in terms of selected research subgoals;
- Analysis of the current state-of-the-art with respect to the defined research subgoals;
- Answering the research subgoals by presenting the achieved research results;
- Validation of our research results in order to examine the validity of conducted research in both research and real-life settings.

The research process that is used in this thesis is presented in Figure 3.1. It consist of seven steps described in the following. We have considered the general research problem, the need to be able to specify, design, and formally analyze services and service compositions, and have transferred the problem to a research setting as described in section 3.1. In order to understand the problem better, we have performed information gathering and state-of-the-art investigation covering the previous work conducted on the research problem (see paper A). Next, based on the findings from the state-of-the-art investigation, we have been able to identify research subgoals as presented in section 3.2. In the next phase of our research we have moved to addressing the research subgoals by developing solutions and presenting achieved research results through several iterations where the achieved results have been improved through discussions and analysis. In papers B, C, and D we have presented our

research results on developing formal approaches for behavioral modeling and analysis of services and service compositions in SOS. In paper F, we present a tool for modeling and analysis of SOS.

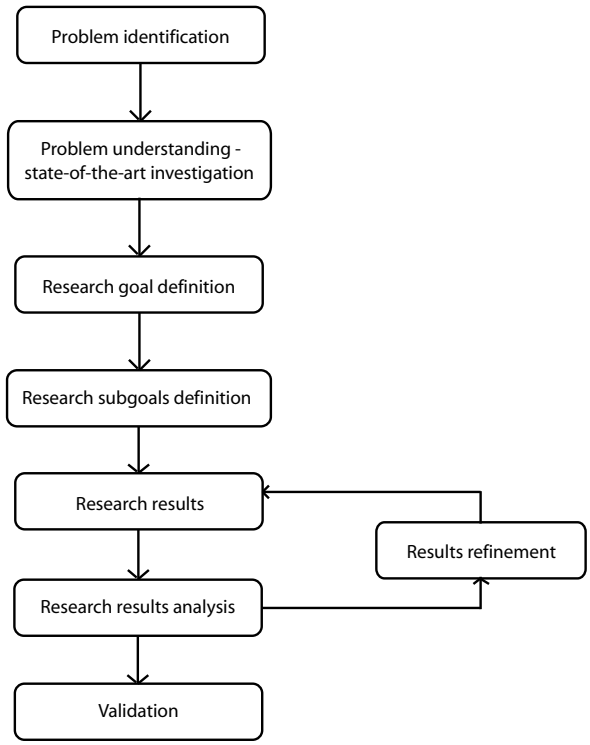

Figure 3.1: Research process steps

The last stage of our research process is validation. All presented results have been analyzed and exemplified as shown in the included papers. In papers B, C, and D toy examples and a simplified example motivated by reality, are used. The approach presented in paper B, is demonstrated on an adapted version of an intelligent shuttle system, for which we have computed resource consumptions, and performed a service composition correctness verification for respective shuttle compositions. Paper C includes an illustrative example of our proposed approach for which we have analyzed resource usage/cost calculation using symbolic

states of the example service. In paper D, we illustrate the presented approach of automated service negotiation in the car insurance example, for which we have shown how to analyze the proposed negotiation model against safety properties, but also against specified timing and utility constraints. Finally, in paper E, we show how to model and formally analyze a distributed energy management in an open energy market, similar to one described by Mobach [46]. The salient point of our negotiation model, which enables its validation, is the fact that the model can be analyzed against several requirements, such as price, time, and reliability, in order to check whether the available energy and proposed prices can satisfy a client's needs. Also, we have been able to calculate the value of the optimal utility function, described as a weighted sum of energy price and energy reliability (modeled as a number), and model-check the trace (a sequence of actions) that leads to such state. Our behavioral language and the associated analysis techniques have been compared to the existing related work and have been shown to be applicable to modeling and analysis of a distributed energy management.

Chapter 4

Research Contributions

In this chapter, we present the contributions of this thesis, and we relate them to the subgoals formulated in Chapter 3. The details of each contribution are presented in the appended papers, to be found in the second part of this thesis.

Research subgoal 1.

The first subgoal is represented by two complementary subgoals, as follows:

> *To provide a language for describing the relevant features of SOS high-level models, and corresponding tool support.*
>
> *(RSG1A)*

> *To develop a formal model for services, amenable to formal analysis, which provides support for describing the service function, timing, and resource-usage aspects.*
>
> *(RSG1B)*

Contribution: A language for behavioral modeling of SOS. To address RSG1A and RSG1B, we extend the already existing dense-time hierarchical modeling language called REMES (a REsource Model for Embedded Systems) initially designed to fit a component-based embedded systems design perspective [8,9]. The main motivation for the extension relies on the fact that SOS share similar characteristics with CBS, as presented in paper A. Our extensions exploit such advantages of the model,

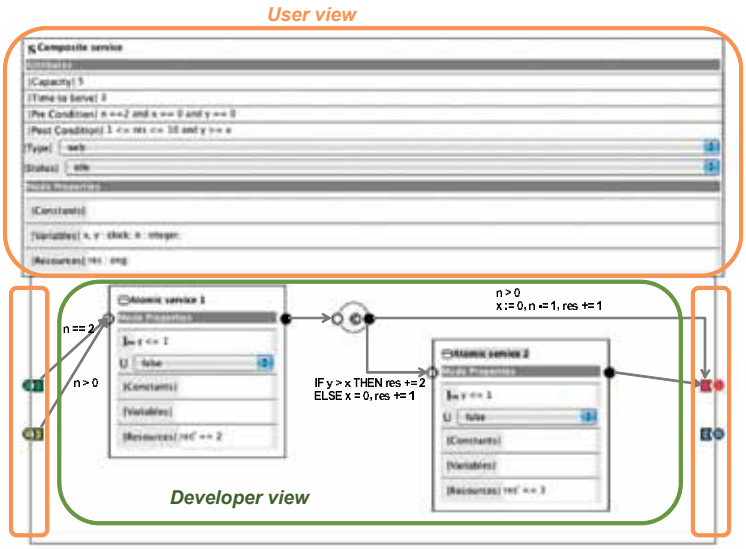

Figure 4.1: A user and developer perspective in a REMES composite service.

and also introduce service-oriented features, aiming at making REMES suitable for behavioral modeling and analysis of SOS, too. Our choice of REMES is also justified by its support for formal analysis, since the REMES language has a formal semantics defined in terms of TA [29], or PTA [30], and depending on the analysis type [33], one can choose between analyzing networks of TA or PTA.

REMES language is well-suited for abstract modeling, it is hierarchical, has an input/ouput distinction, a well-defined formal semantics, and tool support for both component-based [1] and service-oriented [2] system modeling and analysis [10,11]. We extend the language with various constructs and features deemed necessary for a service-oriented perspective, as described in the following. In service-oriented version of the REMES language, we consider two system views: the users's and the developers's view, respectively, as depicted in Figure 4.1. In the former, one has

[1] The REMES tool-chain is available at http://www.fer.hr/dices/remes-id.
[2] More information available at http://www.idt.mdh.se/personal/eep/reseide/

information primarily on the service interface and exposed attributes (or features). In the latter, one can be assumed to have access to the internal service representation: functionality, enabled actions, resource usage, and possible interactions with other services.

A REMES service can be described graphically (as a mode), or textually by a list of attributes (i.e., service type (a web, network, or embedded service), capacity (the maximum number of messages per time unit), time-to-serve (the worst-case time needed to respond and serve a given request), status (the current service status, i.e., passive, idle, active), service precondition, and postcondition exposed at the interface of the REMES service (see Figure 4.1). A service precondition is a predicate that constrains the start of the service execution, and must be true at the time a REMES service is invoked ((n == 2 and x == 0 and y == 0) in Figure 4.1). A postcondition is the output guarantee, also a predicate, which must hold at the end of a REMES service execution for a service to be correct with respect to the function, timing, and resource usage requirements ((1 $\leq$ res $\leq$ 10 and y $\geq$ x) in Figure 4.1). A service modeled in REMES can be atomic (Atomic service 1 and Atomic service 2 in Figure 4.1), composite (Composite service in Figure 4.1) or employed in several types of compositions, resulting in new and more complex services.

Let us assume an example of a composite service that models a web service depicted in Figure 4.1. Composite service contains two subservices Atomic service 1 and Atomic service 2. The service has two entry points, Init and Entry point, where Init entry point is visited when the service executes for the first time, and where all variables are initialized. Timed behavior in REMES is modeled by global continuous variables of type clock, evolving at rate 1 (variable x and y in Figure 4.1). Each service can be annotated with the corresponding resource usage, modeled by the first time derivative of the real-valued variables that denote resources and that evolve at positive integer rates (variable res in Figure 4.1). Discrete resources are allocated through updates, e.g., res += 1 in Figure 4.1. To describe the continuous behavior of the service, delay (or timed) actions are used, while the instantaneous actions are modeled via discrete actions. The execution of the first type of actions do not change the mode in a service, while the second are represented as annotations of the edges, and their execution results in a mode change.

A composite service executes by performing a sequence of discrete steps, via actions that, once executed, pass the control from the current subservice to a different subservice. An action, A = (g, S) (e.g., (n > 0,

x := 0, n -= 1, res += 1) in Figure 4.1), is a statement S (in our case x := 0, n -= 1, res += 1), preceded by a boolean condition, the guard (n > 0), which must hold in order for the action to be executed and the corresponding outgoing edge taken. A REMES composite service may contain conditional connectors (decorated with letter C) that allow a possibly nondeterministic selection of one discrete outgoing action to execute, out of many possible ones. In Figure 4.1, via C, one of the three available actions can be chosen for execution. Services might also be annotated with invariants (e.g., y $\leq$ 1 in Atomic service 1), which bound from above the current service's delay (or execution time). Before the invariant stops to hold, the current mode is exited.

Service manipulation is supported via REMES interface operations that include: service creation, deletion, composition, and replacement. When composing services, it is also beneficial to form lists of services a priori (s_list) that can be managed by similar interface operations as for simple services such as: create, delete, or reorder list of services. Examples of a service creation and adding a service to a list are shown below. In the example of a service creation we assume that attributes service capacity and service time-to-serve might be irrelevant for a given service and therefore we allow in the service specification to omit their definition (e.g., capacity == infinity).

Create service: *create service_name*
$[pre] : service\_name =$ NULL
create : $Type \times\ N\ \times N\ \times\ ''passive'' \times (\Sigma \rightarrow bool) \times (\Sigma \rightarrow bool) \rightarrow\ service\_name$
$\{post\} : service\_name \neq$ NULL and $Type\ \in$ {web service, network service, embedded service, ...} and $capacity\ \geq\ 0$ and $time - to - serve\ \geq\ 0$ and $status\ =\ ''passive''$

Add service to a list: *add service_name, s_list*
$[pre] : service\_name \notin s\_list$
add : $s\_list \rightarrow s\_list$
$\{post\} : service\_name \in s\_list$

To facilitate modeling of nested sequential, parallel or synchronized services and their compositions, we also introduce a hierarchical textual dynamic [3] composition language (HDCL).

$$\begin{aligned} \text{DCL} \quad &::= \quad (\text{s\_list}, \text{PROTOCOL}, \text{REQ}) \\ \text{HDCL} \quad &::= \quad (((\text{DCL}^{+}, \text{PROTOCOL}, \text{REQ})^{+}, \text{PROTOCOL}, \text{REQ})^{+},\ \ldots\ ) \end{aligned} \tag{4.1}$$

[3] Here the term dynamic should be understood as "on demand" or "on the fly".

In Eq. 4.1, DCL (Dynamic Composition Language) describes the basic service composition mechanism, while HDCL describes the hierarchical composition. Our service composition textual description requires the existence of a service list (s_list), a service interaction protocol (PROTOCOL), and a service requirement (REQ). To express the need that one or more DCLs are required to form HDCL we use the positive closure operator. PROTOCOL defines the type of binding between services, which can be modeled by a unary or binary operator described in Eq. 4.2.

$$\begin{aligned} \mathsf{PROTOCOL} ::= \mathsf{unary\_operator}\ &service\_name \mid \\ &service_m\ \mathsf{binary\_operator}\ service_n \end{aligned} \tag{4.2}$$

The requirement REQ is a predicate ($\Sigma \rightarrow \mathsf{Bool}$) that might comprise both functional and extra-functional properties of the composition. It identifies the required attribute constraints, capabilities, characteristics, or qualities of a system composed of services, such that it exhibits the value and utility requested by the user. Σ is the polymorphic type of the state that includes both local and global variables. HDCL allows creating new services by composing existing services via binary operators, as well as adding and/or deleting services from lists. The unary and binary operators supported by the language (as PROTOCOL) are defined in Eq. 4.3.

$$\begin{aligned} \mathsf{Unary\_operator} &::= \mathsf{exec-first} \\ \mathsf{Binary\_operator} &::= \ ; \ \mid \ \| \ \mid \ \|_{SYNC-and} \ \mid \ \|_{SYNC-or} \end{aligned} \tag{4.3}$$

The unary operator exec-first to specifies which service should be executed first in a composition and only when that respective service finishes and establishes its postcondition, other services can become active. To model synchronized behavior in REMES we use a special kind of mode, called an AND/OR mode. The mode is made of "paths" of interconnected submodes that can be executed in parallel, as in Figure 4.2. By the semantics of the mode, in such a mode, the services finish their execution simultaneously, from an external observer's point of view. However, if the mode is employed in design as an AND mode, the subservices are activated at the same time, while an OR mode assumes that one or at

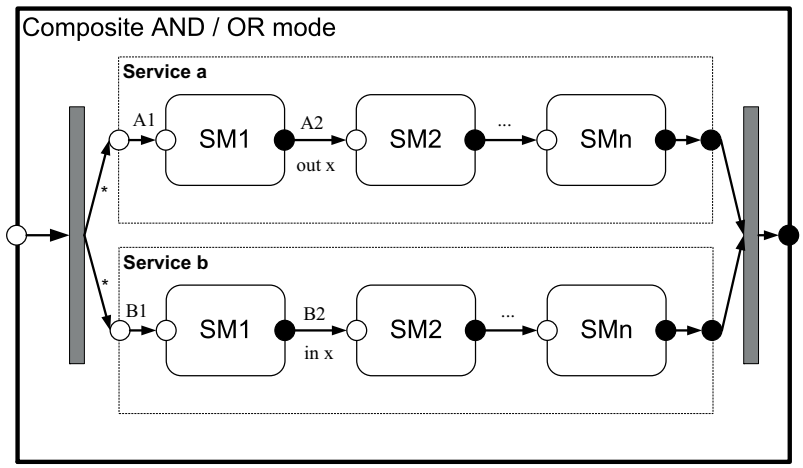

Figure 4.2: An illustration of an AND/OR REMES mode

most all subservices are activated based on the truth value of the guards of the actions on the incoming edges. Services that belong to this type of REMES mode and that have to synchronize their behavior at the end of their execution interact via the PROTOCOL expressed as either of the following two operators:

- $\|_{SYNC\text{-}and}$ - all services follow their exit edges at the same time when the mode finishes its execution;
- $\|_{SYNC\text{-}or}$ - the mode finishes its execution as soon as one service traversed an exit edge.

The detailed description can be found in paper B.

Contribution: Tool support for designing SOS. The current tool enables graphical modeling of service-based systems, with the resource-aware timed behavioral language REMES, as well as the textual system description (in the language presented above). The textual description is an alternative to the graphical description, deemed useful especially when the design tends to become complex. To enable the automated design and analysis of our kinds of SOS, REMES is backed by a tool that comprises: (i) a graphical editor used to display the service diagram,

where services can be created, composed, and displayed as desired by the designer, (ii) a console view that is an alternative description of the graphical design, which supports the textual description of the system, including service declarations, list of services, and their compositions[4], and (iii) an automated traceability between the two design interfaces, which provides support for tracing the error from one model to the other [34, 47]. Using the textual description of the system and textual service composition interface, service compositions can also be checked for correctness with respect to given requirements. We have automated the transformation of REMES models into TA networks in order to enable the formal verification and validation of given models, to reduce error proneness in the design process. Another benefit is the fact that the designer does not need to be a verification expert in order to verify the modeled system. All details regarding the tool implementation can be found in paper F.

[4]More information available a http://www.idt.mdh.se/personal/eep/reseide/

Research subgoal 2.

> *To provide means to formally analyze functional, and extra-functional properties of services and service compositions within our framework.*
>
> *(RSG2)*

Contribution: A service and service composition correctness check. To verify the service and service composition correctness, we use the forward analysis technique based on the computation of the strongest postcondition (sp) of a REMES service (Service) with respect to a given precondition (s_pre) [13]. The correctness verification of a service with respect to the service's postcondition (s_post) and service precondition (s_pre) reduces to the following boolean implication:

$$\mathsf{sp.Service.s\_pre} \quad \Rightarrow \quad \mathsf{s\_post} \tag{4.4}$$

where (s_pre) is required by the service client, and (s_post) is offered by the service provider, and should be guaranteed after the service is executed. If the implication is verified and s_post is no more than the given Service requirement, one can conclude (or guarantee) that the provided service fits with the client requirement, before the service has been executed. To calculate the sp of a REMES service, we have used two methods: (i) a well-known deductive method described in paper B, starting from the guarded command language (GCL) [48] description of a REMES service [15], and (ii) an algorithmic method described in paper C, which computes automatically the sp of a REMES service described as PTA [49]. The algorithmic version includes also the minimum (or maximum) resource-usage trace computation, while performing strongest postcondition analysis.

In the algorithmic computation of the sp a service is described as a set of priced symbolic states of form (A, π), where A is a set of states, and π assigns non-negative costs to all states in A, as shown in Figure 4.3. The algorithm employs two lists, WAITING (contains the states to be examined) and PASSED (populated with already examined states). At each iteration, the algorithm selects a priced symbolic state (A, π) from WAITING. If (A, π) is a goal state (note that in PTA describing a REMES service, the goal state is determined by a unique location, (A_f, π_f) in Figure 4.3) not contained in a goal state previously stored in the sp, it is added to the calculated postcondition sp. Otherwise, if it is not a goal

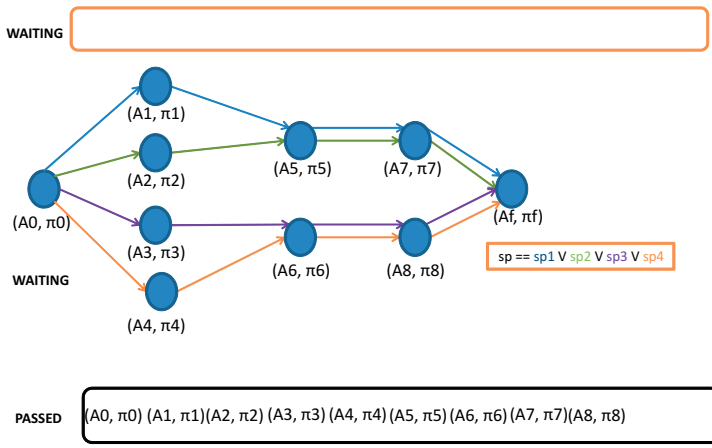

Figure 4.3: An example of the algorithmic strongest postcondition calculation

state and not contained in a symbolic state previously stored in PASSED, it is added to PASSED, and all its successor states are added to WAITING. When WAITING is empty, the strongest postconditions calculated for each path reaching the goal state are returned. In Figure 4.3 we assume that there exist four paths that reach the goal state, and their respective strongest postconditions form the strongest postcondition of a REMES service ($\mathsf{sp} == \mathsf{sp1} \vee \mathsf{sp2} \vee \mathsf{sp3} \vee \mathsf{sp4}$).

To be able to verify service compositions, by the methods above, we give semantics to sequential, parallel, and parallel with synchronization service composition, respectively. The benefit of this language is that, after each composition, we require that one checks whether the given requirement is satisfied, by forward analysis, e.g., by calculating the strongest postcondition of a given composition with respect to a given precondition.

Research subgoal 3.

> *To provide an analyzable model that describes possible interactions between services and their providers, through successive selection of actions.*
>
> *(RSG3)*

Contribution: A service negotiation model with formal semantics. To provide a systematic and an analyzable way to model a service negotiation process, we propose the REMES HDCL negotiation model. The model is based on the set of REMES interface operations and the hierarchical language HDCL, which supports REMES service composition, which we have recently proposed [15].

In the model described in paper D, we assume an iterative form of the Contract Net Protocol (CNP) between web services, described in REMES HDCL. In the CNP the roles of the manager that can be seen as a negotiation mediator, and the contractor (a service provider) are defined as assumptions. The manager gets a request from a client and aims at finding an appropriate contractor to fulfill the request via call for proposals (CFP). Furthermore, the CFP is evaluated by contractors and a response is being sent back. The manager evaluates the received proposals and based on the evaluation decides which proposals to accept and which to reject. In the iterative CNP, the communication between manager and possible contractors repeats until the consensus is reached. In each round the contractors aim to improve on their previous proposals, in order to make them more suitable to the manager. The goal is to settle an agreement that will satisfy requirements of all involved parties.

The model's benefit is that once the negotiation process is finished and a service has been provided, one can easily check if the service delivers the original qualities by employing the service correctness check previously described. Moreover, the model can be seen as a "negotiation service" that iterates over client-provider choices, automatically, until a consensus is reached. The fact that the REMES model can be translated to TA and analyzed with the UPPAAL model checker brings additional insight into the negotiation process and its possible outcomes if compared to a simulation-based approach [46].

Research subgoal 4.

> *To apply the proposed design framework on a real-life service-based system in order to check its suitability in practice.*
>
> *(RSG4)*

Contribution: Validation of REMES language for SOS. Our approach has been applied on three simple, yet relevant research examples: an ATM scenario (depicted in paper A), an intelligent shuttle system (shown in paper B), and an insurance scenario (presented in paper D). The examples an the application results enable a deeper understanding of the service-based behavior in isolation, as well in more or less complex service compositions, while also providing a basis to exercise the applicability of our framework. In the given examples we have modeled behavior of different types of services, analyzed their performance, checked their correctness, and simulated a service negotiation process, by translating REMES models into TA and PTA networks.

In paper E, we have provide a more detailed validation of the REMES-based negotiation model on a more complex and realistic example of distributed energy management. In the energy management system an energy consumer (i.e., client) creates a request and communicates with energy provider via a mediator in order to get a given request served. A request carries information about an amount of energy to be provided, expected price per unit of energy, and information regarding the acceptable energy reliability. The supply of energy is based on a negotiation carried out between consumers and providers in possibly more than one round, assuming a certain scenario. The negotiation relies on providers' advertisements that specify type of energy to be sold (i.e., energy coming from diesel generators, wind turbines, etc.), available amount of energy, its reliability, and price per unit of energy. The energy consumer is assumed to have a varying energy demand over a day, while at the same time providers have varying energy capacity available over a period of time. In the case-study we have been able to analyze three different scenarios of a service provision using a service negotiation process: (i) a scenario in which a client uses maximum bound on the price, (ii) a scenario in which there is no maximum bound on the client's acceptable price, and (iii) a scenario in which a client adapts the price based on the offers given by a provider.

In this case-study we have modeled the described negotiation model using our textual composition language HDCL, and analyzed it against

price, time, and reliability requirements, in order to get information whether the available energy and given prices can satisfy the client's needs. We have focused on calculating the optimal values of utility function (a weighted sum of negotiation preferences) with respect to the price and the energy reliability (modeled here as a number), and we have also model-checked the trace that leads to such state. Since the negotiation model is time constrained, we have been able to compute time needed to reach an agreement for energy supply over a day, which will satisfy expectations of all negotiation participants.

Chapter 5

Related Work

This chapter relates the work in this thesis to relevant research areas. It is subdivided into a number of sections in which we provide comparisons with work of fellow researchers, for each area, respectively.

5.1 Modeling and Analysis of SOS

Two popular commercial solutions for modeling SOS are SOMA [50] and SOMF [51]. SOMA [50] is a method developed by IBM for designing and building SOA-based solutions. The method includes techniques for design and analysis, implementation, testing, and deployment of services and corresponding policies required for a successful design of reusable SOA solutions. On the other hand, Service-Oriented Modeling Framework (SOMF) has been introduced by Michael Bell as the modeling language suitable for developing distributed software systems [51]. SOMF is constructed around eight models of implementation such as: design, discovery, analysis, quality assurance model, etc. Each of these models identifies the methodology, process, platform, best practices, and disciplines by which a practitioner is able to accomplish a modeling task during a project. A service in SOMF is classified by its contextual and structural attributes into: an atomic, a composite, a cluster, and a cloud service. The framework offers a number of modeling practices that contribute to a successful service-oriented life cycle development and modeling during a project. Compared to the described approaches,

our approach is more abstract yet rigorous, dealing purely with design level services, without an actual relation to an implementation model.

There is a number of academic solutions that address modeling and analysis of SOS. In the following, we mention several of them, pointing out their advantages and drawbacks with respect to their modeling and analysis capabilities. The Sensoria Reference Modeling Language (SRML) is a high level modeling language describes service-oriented architectures [52]. The language enables modeling of composite services via a distributed service orchestration that is given formal semantics in terms of configuration graphs and state transition systems. The behavior of services is described by well-defined interfaces (requires-, and provides-interfaces). In SRML, properties of required and provided services are specified in temporal logic (UCTL branching time temporal logic that is an action or state based logic, originally introduced to express properties of UML statecharts [53]) and can be analyzed over orchestrations defined in terms of state transition systems using the UMC model-checker [54]. Time-related properties of services can be analyzed using the stochastic process algebra PEPA [55]. The analysis of SRML models provide timing analysis of possible delays that might occur while providing a service, but also performance, responsiveness, and sensitivity analysis are possible. The SRML language is equipped with constructs for correctness check of the requested (or provided services). The model also enables a static analysis of the existing services. The benefit of such a language is the fact that it provides a rich set of constructs to enable both qualitative and quantitative analysis, but the fact that the user needs to master several techniques and formalisms to explore the full potential of such an approach might become a disadvantage.

Rychlý describes the service behavior as a component-based system for dynamic architectures [27]. The specification of services, their behavior, and hierarchical composition are formalized within the π-calculus. Similar to our approach, this work emphasizes the behavior in terms of interfaces, (sub)service communication, and bindings, yet we can also cater for service descriptions including timing and resource annotations [56].

5.2 Checking Properties of Isolated and Composed Services

Beek at al. [57] present a comprehensive survey of relevant approaches that accommodate modeling and analysis of service compositions [4–7]. Regarding service modeling, all these approaches are solid; however, with respect to verifying service correctness [58–60] (usually by employing formal methods), such approaches show limited capabilities to automatically support these processes. Foster et al. present an approach for modeling and analysis of web service compositions [28]. The approach takes BPEL4WS service specification and translates it into Finite State Processes (FSP), and Labeled Transition Systems (LTS), for analysis purposes. The drawback of the approach is the tedious transformation process necessary to obtain the analysis model, especially in cases when the user is not familiar with different notations and approaches required in this process. Díaz et al. describe a process of automatic translation of BPEL and WS-CDL service models to TA in order to enable analysis via Uppaal model-checker [58]. Even though the given model-checker is a powerful analysis tool, the described approach is limited to checking only service timing properties. Narayanan et al. show how semantics of OWL-S, described using first-order logic, can be translated to Petri-nets and then analyzed as such [59]. The analysis of such models includes reachability analysis and some liveness properties checking, including deadlock freedom of service compositions.

Compared to these approaches, compositions of Remes models, but also atomic Remes services, can be deductively reasoned about (although, as for now, we still miss the interface correctness tool support), or can be automatically translated to TA [29] or PTA [30], and analyzed with Uppaal, or Uppaal Cora tools, for functional but also extra-functional behaviors (timing and resource-wise behaviors).

5.3 Service Negotiation

Lapadula et al. provide a description of modeling publication, discovery, negotiation, deployment, and execution of service-oriented applications in COWS [61]. COWS is a WS-BPEL-inspired process calculus, which can be seen as a lower level modeling language suitable for specifying, combining, analyzing services, while modeling their dynamic behavior.

For the analysis purposes, the language can be translated to the CMC model-checker. In comparison to this approach, our approach offers a service model with semantics defined in UPPAAL's TA, which allows investigating both functional as well as extra-functional negotiation properties. Sierra et al. describe a formal model for negotiation between autonomous agents in service-oriented environments [62]. The service-oriented model is a modified version of the general negotiation model proposed by Faratin et al. [63]. The paper includes several negotiation steps, such as generation of the initial offer, evaluation of the incoming proposals, and generation of counter proposals. However, no analysis support has been described. Comuzzi et al. present an automated approach to web service QoS negotiation [64]. The negotiation is performed via a Negotiation Broker to which both a consumer and a service provider notify their preferences on QoS attributes and negotiation strategies, by specifying the value of a relatively small set of parameters. In some later work Comuzzi et al. propose a semantic-based framework to support negotiation processes in Service-Oriented Architectures (SOA) [65]. The benefit of this approach is that the framework allows the service client and a service provider to express their capabilities in terms of the negotiation protocols they are able to support and the actions they are able to perform, and based on that the framework decides on a negotiation protocol to be used. Again, both papers describe a rich theoretical foundation for automated negotiation processes, but compared to our approach lack the formal analysis of negotiated QoS.

Benkner et al. [22] describe the GEMSS Grid infrastructure, based on standard Web services technology, that enables parallel application available on clusters to be exposed as QoS-aware Grid services. The infrastructure enables clients to dynamically negotiate provided QoS constraints with respect to response time and price, using SLAs. The negotiation is based on request-offer model where the central role in the process is given to the QoS manager that receives requests from a client, checks whether client's request can be met using the application performance model, and generates a corresponding offer. The model enables also renegotiation in cases that some parts of the requests are fit and some need to be adjusted based on the client's restrictions. The main benefit of this infrastructure is recognized to be in medicine, where Grid technology can provide medical practitioners with access to advanced simulation and image processing services for improvement of pre-operative planning and near real-time surgical support.

Resinas et al. provide a description of automated agreement negotiation system based on a bargaining protocol called NegoFAST-Bargaining [66]. The architecture includes a rich environment to first identify the key element in the negotiation process, then model it together with its corresponding processes, and finally create the scenarios to be validated. The framework can be seen as a rich theoretical basis that developers can use when building their negotiation processes, and might be extended such that it supports validation of different negotiation scenarios with respect to functional, and extra-functional properties as proposed in our approach. Paurobally et al. describe a way to deploy multi-agent negotiation techniques to facilitate dynamic negotiation for Grid resources in order to provide an adaptive and autonomous Grid [67]. Moreover, they describe the deployment of CNP and its corresponding strategies for negotiation between web services. The approach offers a rich environment to model the negotiation process using CNP, but in terms of the analysis it is limited to monitoring the modeled system.

Chhetri et al. describe an agent-based negotiation framework that supports provision and maintenance of SLA for web service compositions [68]. In this work they propose an approach to provide service compositions during the service negotiation process in case that no single service that can satisfy user demands exists. The concept includes a third party agents, called Negotiators, that in case a single service that can satisfy user demands look for suitable services that can create service compositions. Such a composition is further analyzed by Coordinator agent using implemented decision support strategies in order to guarantee that the proposed composition is fit for a user requirement. The overall idea behind this approach emphasizes the overall idea of SOS, where services are discovered, composed, and provided on demand. However, the approach could be improved by adding a stronger analysis engine that would provide better environment to analyze proposed service compositions.

Chapter 6

Conclusions and Future Work

The objective of the research presented in this thesis is to develop methods and associated tools for the specification, modeling, and formal analysis of services and service compositions in SOS. Our main focus is on the behavioral aspects of services and challenges associated with analyzing such models. In the thesis, we have extended the resource-wise hierarchical timing behavioral language, called REMES, and provided associated analysis techniques aiming at supporting services and service compositions in SOS. We have illustrated our approach on several small examples, but also on a more realistic and complex case-study of distributed energy management. In the following section, we present a brief overview of all thesis contributions.

6.1 Summary of Thesis Contributions

In this work, we have presented research that addresses the formulated research goals of Chapter 3, and which can be summarized in the following concrete lines of contribution:

REMES **behavioral language for service-oriented environments.** To support modeling of independent services, we have extended the existing behavioral modeling language REMES, which has been designed

to fit a component-based design perspective [8, 9]. A service in REMES is described both graphically as a mode with explicit entry- and exit points, and textually by a list of attributes within the service interface that describes service characteristics in detail, and makes a service discoverable by potential service users. We distinguish between two different service design perspectives: the developer's and the user's. In the former, developers are assumed to have access to the service functionality representation, enabled actions, resource annotations, and possible interactions with other services, all given as the service's behavioral description; in the user's view such a description is not needed, instead the service interface needs to be visible. These described constructs set the ground for formal analysis of services and their compositions, since the language supports modeling both single and composed services. This can be achieved via hierarchical composition language that allows to create new services, using binary operators, as well as adding and/or deleting services from lists. In addition, it allows serial, parallel and parallel with synchronization service composition.

Checking the correctness of REMES services. To support the the correctness check of REMES services we use the forward analysis technique based on the computation of the strongest postcondition of a REMES service with respect to a given precondition [13]. To calculate the strongest postcondition of a REMES service, we have used two methods: (i) a well-known deductive method that starts from the guarded command language (GCL) [48] description of a REMES service, and (ii) an algorithmic method that automatically computes the strongest postcondition of a REMES service described as PTA. The algorithmic version includes also the minimum (or maximum) resource-usage trace computation, while performing strongest postcondition analysis. The approach makes checking the correctness of more complex services feasible, and awaits implementation in the UPPAAL CORA tool.

Service negotiation model in REMES. Available services might deliver similar or the same functionality but differ in extra-functional characteristics. Moreover, they might be offered to service users at different prices, time, or any other condition. Therefore, it is often the case that service providers and clients need to negotiate in order to support a systematic and analyzable way to model a service negotiation process, we have introduced the REMES HDCL negotiation model. The model

is based on the set of REMES interface operations and the hierarchical language (HDCL) that supports REMES service compositions. The main benefit of such a model is that once the negotiation process is finished one can easily check whether the service really delivers the original qualities by employing the previously described service correctness methods. In addition, the model can be seen as a "negotiation service" that iterates over client-provider choices, automatically, until a satisfactory agreement for all involved parties is reached. Additionally, the fact that the REMES model can be translated to TA and analyzed with the UPPAAL model checker brings more insight into the negotiation process and its possible outcomes.

In the proposed negotiation model, we have assumed an iterative form of the Contract Net Protocol (CNP) between web services, described in REMES HDCL, but the model could be extended towards supporting other protocols, too. Moreover, in our model we have implemented two negotiation strategies, price- and time-driven with marginal cost. The model has been applied in a car insurance example, for which we have shown how to analyze the model against safety properties, but also against specified timing and utility constraints represented as a weighted sum of negotiation preferences.

Tool support for modeling and analysis of services and service compositions. In this thesis we have developed a Java-based, standalone tool for describing REMES services and service compositions. The tool enables graphical modeling of service-based systems, as well as a textual service and service composition description, in REMES. The tool consists of an editor for new service creation, as well as the design of service compositions; in addition to the editor, a console view is provided for the textual description of the system, including service declarations, list of services, and their compositions. The tool enables the design of service compositions as possibly desired by the user, together with a textual service composition interface in which compositions can also be checked for correctness. We also provide automated traceability between the two design interfaces, which enhances the system design process with intuitive service manipulation. With respect to the formal analysis, we provide an automatic translation of the graphical service description into TA networks, where described models can be formally analyzed.

The approach validation. The approach described in this thesis has

been applied on three simple research examples: an ATM scenario, an intelligent shuttle system, and a car insurance scenario. For these examples, we have analyzed the performance of service-based systems, have carried out correctness checks, and simulated service negotiation processes. However, we have applied our proposed negotiation model to a more complex real-life case-study, that is, a distributed energy management, where we have analyzed the proposed model with respect to the optimal values of utility function (weighted sum of negotiation preferences) with respect to the price and the energy reliability. In this case, we have been able to derive the time required to close an agreement for different modeled scenarios. Each of these studies has helped us to better understand the capabilities and limitations of our framework when applied to modeling of service behaviors, as well as of behaviors of service compositions. We have also got insight into how to extend our work such that it becomes more complete and adequate for modeling and analysis of real SOS systems.

6.2 Future Research Directions

We have identified several possible directions that our research could follow in the future. As we can see in the available literature, many of the academic approaches have been enriched such that they enable translation or connection to WS-BPEL language [69]. Our aim is to provide a connection to WS-BPEL language, too, such that the large analysis spectrum covered by our approach reaches and becomes accessible to a broader research community.

The current trend in SOS leads to service clouds, both private and public. It would be interesting to investigate how our approach could fit into modeling and more importantly analyzing available services in service clouds. Another interesting direction might be connected to applying our negotiation model on a service provision process in service clouds.

The analysis techniques presented in this thesis are based on formal techniques, more specifically on service verification by model-checking, where services are described as networks of TA and PTA. We are also interested in applying other analysis techniques in order to uncover a larger spectrum of possible errors, and also improve scalability. One of the interesting techniques that we have in mind is software testing for

services.

At last, all the analysis presented in this thesis applies to the service design-time models. We envision an increased value of our framework, if we enhance it with capabilities to perform run-time SOS analysis too. In such a context, we could get better insight into services and their composition behavior, when already deployed and used in the target environment.

Bibliography

[1] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series).* The MIT Press, 2008.

[2] Eric A. Marks and Michael Bell. *Service-Oriented Architecture (SOA): A planning and Implementation Guied for Bussiness and Tchnology.* John Wiley & Sons, Inc, Hoboken, New Jersey, April 2006.

[3] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures and Applications.* Springer Publishing Company, Incorporated, 1st edition, 2010.

[4] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. *BPEL4WS, Business Process Execution Language for Web Services Version 1.1.* IBM, 2003.

[5] Nickolas Kavantzas, David Burdett, Greg Ritzinger, Tony Fletcher, Yves Lafon, and Charlton Barreto. Web services choreography description language version 1.0. World Wide Web Consortium, Candidate Recommendation CR-ws-cdl-10-20051109, November 2005.

[6] Object Management Group (OMG). *Business Process Modeling Notation (BPMN) version 1.1.*, January 2008.

[7] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[8] Cristina Seceleanu, Aneta Vulgarakis, and Paul Pettersson. Remes: A resource model for embedded systems. In *Proc. of the 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2009)*. IEEE Computer Society, June 2009.

[9] Aneta Vulgarakis, Cristina Seceleanu, Paul Pettersson, Ivan Skuliber, and Darko Huljenic. Validation of embedded systems behavioral models on a component-based ericsson nikola tesla demonstrator. In *11th InternationalConference on Quality Software (QSIC 2011)*. IEEE, July 2011.

[10] Dinko Ivanov, Marin Orlic, Cristina Seceleanu, and Aneta Vulgarakis. Remes tool-chain - a set of integrated tools for behavioral modeling and analysis of embedded systems. In *Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010)*, September 2010.

[11] Eduard Paul Enoiu, Raluca Marinescu, Aida Causevic, and Cristina Seceleanu. A design tool for service-oriented systems. In *9th International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA 2012)*. ENTCS, March 2012.

[12] Erl Thomas. *SOA Principles of Service Design*. Prentice Hall PTR, 2008.

[13] Edsger W. Dijkstra and Carel S. Scholten. *Predicate calculus and program semantics*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[14] Kim Guldstrand Larsen and Jacob Rasmussen. Optimal conditional reachability for multi-priced timed automata. In Vladimiro Sassone, editor, *Foundations of Software Science and Computational Structures*, volume 3441 of *Lecture Notes in Computer Science*, pages 234–249. Springer Berlin / Heidelberg, 2005.

[15] Aida Causevic, Cristina Seceleanu, and Paul Pettersson. Modeling and reasoning about service behaviors and their compositions. In *Proceedings of 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA 2010), Formal Methods in Model-Driven Development for*

Service-Oriented and Cloud Computing track. Springer LNCS, October 2010.

[16] Kim Guldstrand Larsen and Jacob Illum Rasmussen. Optimal reachability for multi-priced timed automata. *Theor. Comput. Sci.*, 390:197–213, January 2008.

[17] Michael L. Ronayane and Erik S. Townsend. A case study: Distributed object technology at wells fargo bank. White paper, A Cushing Group, Inc., October 1996.

[18] Tzu-Hsiang Yang, Yeali S. Sun, and Feipei Lai. A scalable healthcare information system based on a service-oriented architecture. *J. Med. Syst.*, 35(3):391–407, June 2011.

[19] Thomas Magedanz, Niklas Blum, and Simon Dutkowski. Evolution of soa concepts in telecommunications. *Computer*, 40(11):46–50, 2007.

[20] G. Gehlen, E. Weiss, and A. Quadt. Service oriented middleware for automotive applications and car maintenance. In *Proceedings of the 1nd Workshop on Wireless Vehicular Communications and Services for Breakdown Support and Car Maintenance*, pages 42–46, Nicosia, Cyprus, Apr 2005. RWTH Aachen University.

[21] Burbeck Steve. The tao of e-business services: the evolution of web applications into service-oriented components with web services. *IBM DeveloperWorks*, 2000.

[22] S. Benkner, G. Engelbrecht, S.E. Middleton, I Brandic, and R Schmidt. End-to-end qos support for a medical grid service infrastructure. *Journal of New Generation Computing*, 2007.

[23] Fitzgerald Brian and Olsson Carl Magnus. The software and services challenge. Technical report, Contribution to the preparation of the Tehnology Pilar on 'Software Grids, Security, and Dependability', EU 7th Framework Programm, 2006.

[24] Dimitrios Georgakopoulos and Michael P. Papazoglou, editors. *Service-Oriented Computing*. MIT Press, Cambridge, MA, 2008.

[25] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown, and Rebekah Metz. Reference model for service oriented architecture 1.0, October 2006.

[26] Johannes Maria Zaha, Alistair P. Barros, Marlon Dumas, and Arthur H. M. ter Hofstede. Let's dance: A language for service behavior modeling. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 145–162. Springer, 2006.

[27] Marek Rychlý. Behavioural modeling of services: from service-oriented architecture to component-based system. In *Software Engineering Techniques in Progress*, pages 13–27. Wroclaw University of Technology, 2008.

[28] Howard Foster, Wolfgang Emmerich, Jeff Kramer, Jeff Magee, David Rosenblum, and Sebastian Uchitel. Model checking service compositions under resource constraints. In *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 225–234, New York, NY, USA, 2007. ACM.

[29] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[30] Rajeev Alur. Optimal paths in weighted timed automata. In *HSCCć01: Hybrid Systems: Computation and Control*, pages 49–62. Springer, 2001.

[31] Aneta Vulgarakis. *A Resource-Aware Framework for Designing Predictable Component-Based Embedded Systems.* PhD thesis, Mälardalen University, June 2012.

[32] Marin Orlić. *Resource usage prediction in component-based software systems.* Phd thesis, Faculty of electrical engineering and computing, University of Zagreb, November 2010.

[33] Dinko Ivanov. Integrating formal analysis methods in progress ide. Master of science thesis, Malardalen Research and Technology Centre, Vasteras, Sweden, June 2011.

[34] Predrag Filipovikj. Connecting a design framework for service-oriented systems with uppaal model-checker. Master of science thesis, Malardalen Research and Technology Centre, Vasteras, Sweden, June 2013.

[35] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Uppaal Ů a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control: verification and control*, pages 232–243, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.

[36] Uppaal tool. `http://www.uppaal.com`.

[37] Uppaal Cora tool. `http://www.cs.aau.dk/~behrmann/cora/`.

[38] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Logic in Computer Science, 1990. LICS '90, Proceedings., Fifth Annual IEEE Symposium on e*, pages 414 –425, jun 1990.

[39] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Inf. Comput.*, 104:2–34, May 1993.

[40] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-Cost Reachability for Priced Timed Automata. In Maria Domenica Di Benedetto and Alberto Sangiovanni-Vincentelli, editors, *Proceedings of the 4th International Workshop on Hybris Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Sciences, pages 147–161. Springer–Verlag, 2001.

[41] Johan Bengtsson, W.O. David Griffioen, Kåre J. Kristoffersen, Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Automated verification of an audio-control protocol using Uppaal. *The Journal of Logic and Algebraic Programming*, (0):163 – 181, 2002.

[42] Thomas Brihaye, Veronique Bruyère, and Jean-Francois Raskin. Model-checking for weighted timed automata. In *Proceedings of FORMATS04*, number 3253 in Lecture Notes in Computer Science, pages 277–292. Springer–Verlag, 2004.

[43] Patricia Bouyer, Kim Guldstrand Larsen, and Nicolas Markey. Model-checking one-clock priced timed automata. In *Proceedings of the 10th international conference on Foundations of software science and computational structures*, FOSSACS'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag.

[44] Colin Neville. Introduction to research and research methods. University of Bradford, School of Management, July 2007.

[45] Mary Shaw. The coming-of-age of software architecture research. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, page 656, Washington, DC, USA, 2001. IEEE Computer Society.

[46] David Mobach. *Agent-Based Mediated Service Negotiation*. PhD thesis, Vrije University, 2007.

[47] Raluca Marinescu and Eduard Enoiu. A design framework for service-oriented systems. Master of science thesis, Malardalen Research and Technology Centre, Vasteras, Sweden, July 2011.

[48] Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, 1975.

[49] Aida Causevic, Cristina Seceleanu, and Paul Pettersson. Checking correctness of services modeled as priced timed automata. In *Proceedings of 5th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. LNCS Proceedings (Springer Verlag), October 2012.

[50] Norbert Bieberstein, Robert G. Laird, Keith Jones, and Tilak Mitra. *Executing SOA: A Practical Guide for the Service-Oriented Architect*. IBM Press books, Upper Saddie River, NJ, USA, 2008.

[51] Michael Bell. *Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley and Sons publishing, Hoboken, NJ, USA, 2008.

[52] José Luiz Fiadeiro, Antónia Lopes, and Laura Bocchi. Algebraic Semantics of Service Component Modules. In J. L. Fiadeiro and P. Y. Schobbens, editors, *Algebraic Development Techniques*, volume 4409 of *LNCS*, pages 37–55. Springer, 2007.

[53] Stefania Gnesi and Franco Mazzanti. A model checking verification environment for uml statecharts. In *PROCEEDINGS OF XLIII CONGRESSO ANNUALE AICA*, 2005.

[54] João Abreu, Franco Mazzanti, José Luiz Fiadeiro, and Stefania Gnesi. A model-checking approach for service component architectures. In *Proceedings of the Joint 11th IFIP WG 6.1 International Conference FMOODS '09 and 29th IFIP WG 6.1 International Conference FORTE '09 on Formal Techniques for Distributed Systems*, FMOODS '09/FORTE '09, pages 219–224, Berlin, Heidelberg, 2009. Springer-Verlag.

[55] Jane Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA, 1996.

[56] Aida Causevic, Cristina Seceleanu, and Paul Pettersson. Formal reasoning of resource-aware services. Technical Report ISSN 1404-3041 ISRN MDH-MRTC-245/2010-1-SE, Mälardalen University, June 2010.

[57] Maurice H. Ter Beek, Antonio Bucchiarone, and Stefania Gnesi. Formal methods for service composition. *Annals of Mathematics, Computing & Teleinformatics*, 1(5):1 – 10, 2007. In: Annals of Mathematics, Computing & Teleinformatics, vol. 1 (5) pp. 1 - 10. Technological Education Institute of Larissa (TEIL), Greece, 2007.

[58] Gregorio Díaz, Juan José Pardo, María-Emilia Cambronero, Valentin Valero, and Fernando Cuartero. Automatic translation of ws-cdl choreographies to timed automata. In Mario Bravetti, Leïla Kloul, and Gianluigi Zavattaro, editors, *EPEW/WS-FM*, volume 3670 of *Lecture Notes in Computer Science*, pages 230–242. Springer, 2005.

[59] Srini Narayanan and Sheila A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, New York, NY, USA, 2002. ACM.

[60] Gwen Salaün, Lucas Bordeaux, and Marco Schaerf. Describing and reasoning on web services using process algebra. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 43, Washington, DC, USA, 2004. IEEE Computer Society.

[61] Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. Service discovery and negotiation with cows. *Electron. Notes Theor. Comput. Sci.*, 200:133–154, May 2008.

[62] Carles Sierra, Peyman Faratin, and Nicholas R. Jennings. A service-oriented negotiation model between autonomous agents. In *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent Rationality*, pages 17–35, London, UK, 1997. Springer-Verlag.

[63] Peyman Faratin, Carles Sierra, and Nick R. Jennings. Negotiation decision functions for autonomous agents. *International journal of robotics and autonomous systems*, 24:3–4, 1998.

[64] Marco Comuzzi and Barbara Pernici. An architecture for flexible web service qos negotiation. In *Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference*, EDOC '05, pages 70–82, Washington, DC, USA, 2005. IEEE Computer Society.

[65] Marco Comuzzi, Kyriakos Kritikos, and Pierluigi Plebani. A semantic based framework for supporting negotiation in service oriented architectures. In *CEC*, pages 137–145, 2009.

[66] Manuel Resinas, Pablo Fernandez, and Rafael Corchuelo. A bargaining-specific architecture for supporting automated service agreement negotiation systems. *Science of Computer Programming*, 77(1):4 – 28, 2012. System and Software Solution Oriented Architectures.

[67] Shamimabi Paurobally, Valentina A. M. Tamma, and Michael Wooldridge. A framework for web service negotiation. *TAAS*, 2(4), 2007.

[68] Mohan Baruwal Chhetri, Jian Lin, SukKeong Goh, Jian Ying Zhang, Ryszard Kowalczyk, and Jun Yan. A coordinated architecture for the agent-based service level agreement negotiation ofweb service composition. In *Proceedings of the Australian Software Engineering Conference*, ASWEC '06, pages 90–99, Washington, DC, USA, 2006. IEEE Computer Society.

[69] Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, 2007.