Preface

Service Oriented Architecture (SOA) is the paradigm for software and system specification, design, implementation and management, that pretends to shape and dominate IT and business landscapes in the near future. SOA has departed from the initial hype phase and ceased to be a simple buzzword long time ago, while entering a relatively mature phase with numerous companies offering SOA products and services. The scientific community has kept apace and continues to explore creative and inspiring approaches at an astonishing pace that further enrich this approach.

SOA initially promised and also thereafter successfully delivered several basic functionalities: unified and standardized description, discovery, communication, and binding of autonomous and self contained software entities, called services, at an unprecedented scale. They have furthermore enabled dynamic and complex enterprise interactions, previously unthinkable or commercially unattainable, and at the global level.

However, very soon it also became clear that functional interoperability offered by the first generation of SOA standards and products failed to satisfy several important requirements, such as efficient discovery and matching of business and/or technical requirements between service requestors and service providers. The increasing number of offered services further exacerbated this problem: it was not clear how to choose adequate interaction partners (services) among many of them offering approximately "the same" functionality; how to select optimal partners according to a given criteria or their combination (e.g., price or performance); or how to be able to specify "soft" or "non-functional" requirements on the requestor side and match them with provided properties on the service side. In other words, the issues of specifying a whole range of properties orthogonal to pure functional description (what the service should do) remained generally unaddressed.

Parallel to these developments, the decade-old idea of a component marketplace was brought to life once again as the service marketplace under the umbrella of emerging SOA standards. Here also, very soon it was only too clear that matching and searching for composition partners or building an application based on third party services depends heavily on many other properties beside the pure functional ones.

The properties which are orthogonal to functional properties (**what** is the service doing) and describe the nature, mechanism, or context of the service execution (**how** and **under which conditions** is the service doing), have been given different names in different disciplines and by different people, including "non-functional properties", "extra-functional properties", "quality of service properties" or "service level agreement properties." In this book, the term non-functional properties will be mostly used, although other designations may also appear. Notable examples of non-functional properties are security, reliability, availability, timeliness, location, price, performance, et cetera.

This book offers a selection of chapters that cover three important aspects related to the use of non-functional properties in SOA: requirements specification with respect to non-functional properties, modeling non-functional properties, and implementation of non-functional properties.

Each software project begins with requirements specification phase. Hidden, unspecified requirements present a constant source of errors, frustrations, and costly workarounds required to fix them. This problem is further exacerbated in heterogeneous and dynamic SOA environment, where frequent changes of processes and technologies dictate adaptive and tool supported requirement specification. In the first section of the book, four approaches for capturing non-functional requirements in SOA will be presented. They build a foundation for successful modeling and execution of complex SOA projects. In Chapter 1, Bode and Riebisch present a novel architectural design method supporting specification of non-functional requirements in the design phase and, more importantly, traceability: mapping of requirements to software solutions. Gross, Yu, and Song argue in Chapter 2 that the true challenge of modeling non-functional requirements is how to support them in different platforms or application domains. For that purpose the authors present a platform and a development method supporting goal and scenario oriented modeling and analysis of non-functional requirements. In Chapter 3, Becha, Mussbacher, and Amyot present an aspect-oriented approach for analyzing non-functional requirements in SOA applications. Finally in Chapter 4 Rodríguez et al. describe a novel tool for capturing security requirements in software product lines.

Modeling non-functional properties is the critical step for achieving successful realization of complex SOA projects. Issues of reliability, availability, security, or quality of service are often subsumed under the general term of Service Level Agreements (SLA). The second section of the book discusses approaches for formal and tool supported modeling of SLAs containing non-functional properties. In Chapter 5, Perino et al. provide an overview of non-functional properties in SOA. The authors propose the basic set of non-functional properties (policy, security, transaction and management), each with the corresponding set of attributes. Moayerzadeh and Yu argue in Chapter 6 that although widely used, basic SOA principles such as abstraction, discoverability, reusability, and composability are rarely collected and systematically organized. They propose a goal-graph representation of SOA principles which can be used in system design. In Chapter 7, Achilleos, Yang, and Georgalas describe a model-based framework for engineering non-functional properties in the context of pervasive service creation. Shekhovtsov et al. present in Chapter 8 an approach of using non-first-normal-form tables for modeling quality of service in SOA. They argue that it is very suitable for communicating application design issues to stakeholders with the business background. Ortiz and Hernández argue in Chapter 9 that the combination of model-driven and aspect-oriented methods provides useful foundation for development of high-quality SOA systems. The authors propose a method for integrating non-functional properties into SOA model-driven development process using aspect-oriented methods.

The final, third section of the book discusses practical application of methods for implementing non-functional properties in SOA environments. Methods such as aspect oriented programming (AOP), model driven architecture (MDA) or control theory are presented and applied to diverse properties (e.g., security) in various domains (e.g., biomedicine). In Chapter 11 Salinas and Salinas present and apply an extended version of an aspect-oriented framework for software product lines that exploits aspect-oriented software development techniques in order to model variability of non-functional properties in SOA from early development stages. Satoh et al. discuss in Chapter 12 the problem of very late and missing specification of security properties in SOA development, because of which developers in the downstream development phases must manage different security requirements and configurations ad-

hoc and manually. The authors then propose a model-driven process which can be extended to multiple specification and development phases for definition of various security properties, such as business security requirements or platform security properties. In Chapter 13 Diao, Hellerstein and Parekh explore scalability of SOA applications. They propose a methodology for scaling SOAs based on the control engineering theory and demonstrate the benefits achieved in an industrial setting. Stantchev and Tamm argue in Chapter 14 that, with massively distributed architectures becoming more prevalent, the assurance of availability and dependability for distributed applications becomes an even more challenging and nontrivial task. The authors describe an approach for addressing non-functional properties in SOA based on reference models such as ITIL and the SOA life cycle. Finally, in Chapter 15 Liu et al. provide an illustrative case study of applying functional and QoS properties in the field of SOA-based biomedical multimedia processing applications.

The book will thus gradually guide the reader through all steps of SOA application development, starting with requirement specification, over non-functional property modeling, to their implementation. Focusing state-of-the art research results in one place, the book can serve both as a practical reference manual as well as advanced scientific source. Finally, the authors discuss open issues, and propose future exciting questions yet to be explored.

Nikola Milanovic
Model Labs - Berlin, Germany