Foreword

A service system is a system of systems that depends on distributed control, cooperation, cascade effects, orchestration, and other emergent behaviours as primary compositional mechanisms to achieve its purpose. A service system's purpose, structure, and number of components are increasingly unbounded in their development, use, and evolution. Service systems support the development of Web-scale service-based applications that are characterised by unbounded numbers and combinations of software-intensive, geographically dispersed, and globally available services.

Service-based applications are qualitatively different from traditional large-scale applications. They are typically realized by creating alliances between service providers, each offering services to be used or syndicated within other external services. They usually comprise aggregations of end-to-end processes that cross organizational borders and can deliver full or partial service application solutions on the Web.

Service development is continuously in flux. The pace of development has accelerated greatly due to the advanced requirements of service systems and the increasing complexity of application architectures (e.g., heterogeneous, virtualised, and increasingly cloud-based). The standardization of core Internet services (often called software or application, platform, infrastructure-as-a service) in conjunction with the advent of cloud technologies offers new possibilities for developing Web-scale service-based applications because of their flexibility and "on-demand" nature.

The foundation of Cloud Computing is the delivery of dynamically scalable and often virtualized resources that are provided as a service over the Internet to multiple external clients, while its focus is on the user experience. The essence is to decouple the delivery of computing services from their underlying technology. At its core, Cloud Computing is a service delivery model designed to help organizations procure and consume service-based solutions as needed, based on some metering and pricing scheme. It allows organizations to move costs from capital expenditures to operating expenditures while allowing them to become more flexible and service oriented. It is therefore obvious that both Cloud Computing and SOA share concepts of service orientation.

When we compare SOA with Cloud Computing solutions, we notice that SOA is a conceptual software architecture pattern, while the cloud can be viewed as a target deployment platform for that architecture pattern. In particular, Cloud Computing provides the ability to leverage on-demand new computing resources and platforms that SOA applications require but an organization does not own. In fact, we observe that Cloud Computing and SOA have important overlapping concerns and common considerations. The combination of Cloud Computing and SOA facilitates service deployment and collaboration with partners over the Web to deliver global SOA solutions to multiple clients across multiple geographic locations. Overall, we see different types of service providers (commercial, public, or community organized) that need to be considered when designing these Web-scale service-based applications.

It is precisely the nexus of service-oriented and Cloud Computing approaches and the migration of legacy applications to these environments that is being explored in this book. An important characteristic of service orientation is that it enables application developers to dynamically grow application portfolios by creating compound SOA-application solutions that inter-mix internally existing organizational software assets with external services that possibly reside in remote locations. So far, the development of service-based applications is still far from being fully achieved. It is mainly based on ad hoc and haphazard techniques. No effective, easy-to-use, focused methodological support is provided to assist in coping with many of the intricacies of flexible service provisioning. I am especially aware of the need to build this area on strong practical foundations. This book provides just such a foundation for software engineers, application developers, and enterprise architects.

You have picked up the right book for just about any topic you wish to learn about in legacy migration to modern computing environments and platforms. The editors of this book have brought together insights, best practices, and a variety of research approaches that address all aspects of legacy migration to service-oriented and Cloud Computing environments, including business drivers. They show how migration to service-oriented and Cloud Computing environments fits in the broader context of modern software application development.

This book covers an impressive number of topics and presents a wealth of research ideas, techniques, practical experiments, use cases, and strategies in relation to modern service provisioning that will excite any researcher (or practitioner) wishing to understand legacy migration to service-oriented and Cloud Computing systems. Important topics that are covered in this book include: re-engineering and wrapping techniques, integration, leverage and evolution of legacy systems, model-driven software migration and development, data migration to cloud environments, examining the trade-off between development/re-design effort, and performance/scalability improvements for cloud application development, collaborative cloud application development, and adaptation of existing service-oriented systems to a RESTful architecture. The book chapters are organized in three logical parts: Migrating to SOA Environments, Migrating to Cloud Environments, and, Frontier Approaches to Service-Oriented Systems.

It is pleasant to see that diverse and complex topics relating to legacy migration and service provisioning are explained in an eloquent manner and include important references to help the interested reader find out more information about these topics. All in all, this is an inspiring book and an invaluable source of knowledge for advanced students and researchers working in or wishing to know more about this exciting field.

I commend the editors and the authors of this book on the breadth and depth of their work and for producing a well thought out and eminently readable book on such a complicated topic.

Michael P. Papazoglou
European Research Institute in Service Science, The Netherlands

Michael P. Papazoglou *holds the Chair of Computer Science at Tilburg University. He is also the Executive Director of the European Research Institute in Service Science (ERISS) and the Scientific Director of the EC's Network of Excellence, S-Cube. He is also an Honorary Professor at the University of Trento in Italy, and Professorial Fellow at the Universities Lyon 1 (France), Univ. of New South Wales, and RMIT Melbourne (Australia), Universidad Rey Juan Carlos, Madrid, and Universitat Politècnica de Catalunya, Barcelona (Spain). He has acted as an Advisor to the EU in matters relating to the Internet of Services and as a reviewer of national research programs for numerous countries in Europe, for the Middle East and Asia, for the NSF* *and NSERC in North America, and the ARC in Australia. His research interests lie in the areas of service-oriented computing, Web services, business processes, distributed computing systems, large scale data sharing, and manufacturing execution systems and processes. He has published 22 books, monographs, and international conference proceedings, and well over 250 journal and international conference papers. He serves on several committees around the globe and is frequently invited to give invited talks and tutorials on service-oriented computing in international conferences. He is the editor-in-charge of the MIT Press book series on Information Systems as well as the founder and editor-in-charge of the new Springer-Verlag book series on Service Science.*