Preface

As service-oriented computing (SOC) gains wide acceptance, the need to understand the lifecycle of SOC-based software becomes inevitable, not only for developers, but also for users. SOC exploits both web services and service-oriented architectures (SOA) as fundamental elements for developing applications. Web services are autonomous, platform-independent computational elements that can be described, published, discovered, and composed using standard protocols to build applications across various platforms and organizations. SOA is a software architecture that identifies relevant participants and interactions that facilitate service publication, service discovery, service binding, and service composition. Currently, SOA technology is commonly treated as “implementing SOA with web services,” which realizes SOC. Similar to the object-oriented paradigm, where software engineering has played a significant role both in describing software lifecycles and in providing fundamental and advanced technologies, SOC has become a new paradigm that is required of software engineering technologies. The development of service-oriented applications does not only depend on constructing service providers, but also considers service composition and delivery. Both service requesters and service providers, along with developers, will benefit from understanding the SOA lifecycle.

At present, research is ongoing in both industry and academia to address issues associated with the SOC domain, including service discovery, service composition, and service management. SOC fundamentally changes the way software applications are designed, constructed, delivered, and consumed, and it supports the development of interoperable, configurable and evolvable composite applications. It has emerged as a major trend of the computing paradigm in the past few years.

To apply service-oriented computing, we indentified high-level requirements for service-oriented environments that are realized by a service bus or a service engine to integrate and link all necessary components and services together. We have accomplished this by consolidating and summarizing extant industrial products and academic research:

- **Compatible with industrial standards:** service-oriented environments should be established based on standard specifications such as SOAP, WSDL, UDDI, BPEL, WS-policy, WS-security, OWL, et cetera. Standardization can enable the technology neutrality and interoperability that are very critical concerns for service-oriented computing.
- **Ease of integration with resources:** service-oriented environments should be capable of integrating data, applications, and processes. An important mission of the SOA is to enable multiple organizations or multiple departments in an enterprise to collaborate consistently. This inter- or intra-integration should be achievable as a result of technology neutrality and standards.
• **Service utility:** end users should be able to consume services on multiple devices (such as desktop computers, notebooks, handheld devices and cellular telephones), at any time, in any place. Thus, services should be hosted in the cloud (public or private) and exposed as standardized interfaces that allow various remote accesses via adequate authentication and authorization.

• **Context awareness and pluggability:** services delivered to the user should meet the user’s personal needs, and services should be easily plugged into or out of the current working space that the service consumer is using. Accordingly, dynamic context and service selection, binding and re-binding should be considered during service delivery to achieve better adaptability.

For realizing above high-level requirements, we identified six research themes related to the concepts, issues, and methodology of service-oriented computing, including:

• **Service design and provision:** to build an SOA-based business solution, we need to first identify, design, and build services. In this phase, identifying business services and corresponding IT services is a critical concern. Next, we must design and build service implementations based on the IT service specifications. Choosing extant service components, wrapping or adapting existing software components into services, and establishing new services from the scratch are all common development strategies. Accordingly, a set of service components (with the same or different interfaces) will be prepared while above development strategies are performed well. The primary issues that should be addressed in the service design and provision phase include the following:
  ◦ What are the service-oriented development processes and methods to be used? Should any new design patterns be specified to assist with building service-based/service-oriented software?
  ◦ How should one develop service components in a more systematic manner? We should consider a means to model services from various viewpoints and explore the usage of formal specification for services and their integration with ontology.
  ◦ How can we uncover the services embedded in legacy systems, databases, and web sites? What are the necessary service wrapping tools and procedures?

• **Service delivery:** most services, though varied in type and possessing different capabilities, need to be delivered to the requester’s site. There are several requirements that should be addressed in the service delivery phase, including negotiating service contracts or service-level agreements, fulfilling the client’s requests based on the contract, and rendering service results via appropriate channels (for software agents) or user interfaces (for end users). The main issues that would be addressed in service delivery phase include the following:
  ◦ How to negotiate and establish Web Services Level Agreements and contracts?
  ◦ How to compose remote services, local services, and UI components at the same time?
  ◦ How to infuse user interactions into SOA and provide on-demand service user interfaces?

• **Service search and selection:** usually, multiple published services offer seemingly similar features but with some variations (e.g., different service interfaces, different attributes, different quality-level). A service requester can search, select, and bind services that satisfy his or her intentions. A service discovery facility can enhance the effectiveness of other activities (e.g., service design and provision, as well as service composition). Main issues that would be addressed in service search and selection phase include the following:
How to transfer the original Web services into semantic Web services? How to annotate and interpret semantic information?

How to capture the users’ intension for making appropriate service requests? The query process should include refining users’ query, generating query forms, and building users’ personal ontology. Techniques such as goal formulation, ontological reasoning, case-base reasoning will be involved in this process.

How to locate suitable services according to the user’s request? Issues of semantics, uncertainty, and quality of web services should be addressed in the matchmaking process.

**Service composition:** to fulfill a business solution or a user’s request, combining multiple existing service components into a composite service is always beneficial. These composite services can be, in turn, recursively combined with other services into higher level solutions. Techniques for dynamic service composition and semi-automatic service composition can be applied in this phase. The main issues that should be addressed in the service composition phase include the following:

- How should one fulfill the service transaction, rollback and compensation tasks in a complicated service flow? Service composition causes transactions to span in multiple web services. A service flow engine should be integrated with transaction-related functionalities to ensure safety in real business operation.
- How should one efficiently realize QoWS-aware composition? Quality of web services is always an important concern when composing services. QoWS will affect the activities of selecting services, binding services, and re-planning composition paths. Challenges include how to provide an adequate model to define QoWS metrics, how to measure or calculate QoWS at runtime, and how to consolidate QoWS information to allow real-time alerts or offline process improvement.
- How should one compose service dynamically per user request? Dynamic service composition focuses on translating users’ needs into a service composition plan based on rule-based reasoning, state machine, and service patterns. The composition should include details about how service-to-service communications, collaborations, and flows are performed.

**Service management:** services are monitored and controlled by configuring and collecting metrics, detecting failures at service levels and recovering the operational state of the malfunctioning service, as well as tuning service performance automatically or semi-automatically. One important goal in this phase is to ensure high reliability or high availability of the target system. The main issues that should be addressed in the service management phase include the following challenging questions.

- How does one acquire and compute the quality of service, react appropriately to an abnormal status, and induce a common cause of performance decay for process improvement?
- How can one increase and maintain service reliability and availability?
- How should one mine service patterns through the analysis of users’ behavior to adjust composite service dynamically and adaptively according to context changes or requirement changes?

**Service verification and validation:** verification and validation (so-called “V&V”) are important concepts in software engineering that aim to improve the quality of software products. Both dynamic testing and static checking mechanisms are common means to perform verification, which focuses on “building the product right,” and to perform validation, which focuses on “producing
the right product.” V&V is also significant for developing SOA-based applications since we need to confirm that the service (either an atomic service or composite service) can offer the functionality and quality-levels it declares, and that it can satisfy the requester’s demands. Main issues that would be addressed in service verification and validation include:

- The lack of source code about the service implementation makes it difficult to use white-box testing techniques and code-based test coverage criteria to test atomic web services.
- The lack of control over the services being composed happens when a service provider may change the services without notifying the service requester.
- The dynamic service binding can pose challenges in testing service compositions. Integration testing of service compositions needs to test all possible bindings in order to ensure that the dynamic behavior of service compositions is correct.
- The cost of testing service compositions can be a concern since service providers may charge a fee for each service invocation.
- The notions of concurrency, synchronization, and compensation also make service compositions difficult to understand and test. Currently, the supporting tools for testing these notions are still inadequate.

Above six research themes can be also treated as main activities or steps in the SOA lifecycle for developing and operating SOA-based systems. Firstly, service design and provision will be performed. Secondly, services will be delivered to the service requester. Thirdly, service management mechanism will be applied to ensure the success of service delivery. These three activities are the elementary tasks in the lifecycle. Service composition can be treated as an advanced service provision style to allow developers combine multiple component services at design time, or as an advanced service delivery style to search and select appropriate component services as well as to generate composite services dynamically at run time. Service verification and validation techniques can be embedded into both service design and service composition activities to improve the quality of either atomic services or composite services.

To provide a clear and consolidated view for current academic and industrial efforts in SOC, this book features fourteen chapters in eight parts to detail extant and promising methods, tools, and technologies along with the lifecycle of web services. The eight parts include introduction and foundation, service provision and design, service delivery, service search and selection, service composition, service management, service validation and verification, and SOA-based applications. In addition, this book provides a glossary section that includes definitions or descriptions of more than ninety terms in SOC domain.

SECTION 1: INTRODUCTION AND FOUNDATION

Chapter 1: Service-Oriented Computing: From Web Services to Service-Oriented Components

In this chapter, Chollet et al. present the different technologies implementing the new paradigm of SOA: Web Services, UPnP (Universal Plug And Play), DPWS (Devices Profile for Web Services), as well as service-oriented component OSGi (Open Service Gateway Initiative) and iPOJO. These technologies have been developed and adapted to multiple domains: application integration, pervasive computing, and dynamic application integration. Authors provide comprehensive analysis on these technologies and also propose in-depth summary and strong comparison.
SECTION 2: SERVICE PROVISION AND DESIGN

Chapter 2: Semi-Automated Lifecycles for Eliciting Requirements for Service-Oriented Environments

From a process and workflow point of view, Blake addresses the evolution of traditional software engineering approaches to support the conceptualization of abstract services that overlap multiple organizations. He stresses that traditional software engineering lifecycles must be enhanced with emerging processes related to the development applications for service-oriented environments. This chapter also discusses state-of-the-art approaches that elicit information about the requirements for service-oriented architectures. These approaches tend to leverage existing requirements engineering approaches to suggest aggregate service-based capabilities that are most effective for a particular environment.

Chapter 3: Quality of Service and Extra-Functional Properties for Web Services: A Model-Driven Approach

Stressing the importance of requirements specifications, Ortiz presents a model-based approach to deal with specifications and implementations of Quality of Service (QoS) and extra-functional properties (EPF) in a loosely coupled manner. The presented approach draws on two main software techniques, Model-Driven Architecture (MDA), which transforms models of QoS and EPF into code, and Aspect-Oriented Programming (AOP), which systematically handles scattered and tangled code of QoS and EPF. The chapter provides a case study to show results in increasing the system’s modularity and thus reducing implementation and maintenance costs.

Chapter 4: The Foundations of Service Eco-Systems

In this chapter, Ghose seeks to design the formal basis for practical tools to support the design and maintenance of service eco-systems. He proposes a high-level Business Service Modeling Language (BSRL) that supports the modeling of the full spectrum of services, spanning from web services on the one extreme to abstractly defined business services on the other. Based on this language, the chapter describes a taxonomy of relationships between services in an eco-system. Besides, the author particularly addresses the question of which combination of functionalities might be packaged as a service, thus leading to a set of inter-related components constituting a service eco-system.

SECTION 3: SERVICE DELIVERY

Chapter 5: A QoS-Aware Service Bus with WSLA-Based Monitor for Media Production Systems

Due to enormously increasing distribution channels, media companies need to distribute their (television) programs through a variety of media, such as TV broadcasts, DVDs, the Internet, and mobile phones. Based on their industrial participations, Chen et al. describe a solution that was developed to address issues caused by multiple delivery pathways. It offers a media asset management system that
is used to support media content production and distribution. Notably, this work implements service oriented architecture (SOA) that relies on an enhanced enterprise service bus (ESB). The enhanced ESB, referred to here as a QoS-Aware service bus (QASB), makes it possible to designate which of the available transcoding servers will perform a required task, thus providing a service selection scheme that improves the efficiency of media content production and distribution processes. This system was built for Taiwan’s Public Television Service (PTS) in January 2010 and is currently providing complete support to the company’s daily operations.

SECTION 4: SERVICE SEARCH AND SELECTION

Chapter 6: A Goal-Driven Approach for Service-Oriented Systems Design

This chapter introduces a goal-driven approach for modeling service requests in service-oriented systems. This approach is proposed to solve three research issues: how to represent a service request, how to map the request to predefined models, and how to generate a series of actions to satisfy these models. A goal model is used for user requirements elicitation, analysis, and representation. The service request from users is extracted and mapped to these goal models by a case-based method. A planning mechanism is designed to produce a sequence of activities to satisfy the extracted goals. A general architecture for an intention-aware service-oriented system is proposed for demonstrating how to apply the proposed approach.

SECTION 5: SERVICE COMPOSITION

Chapter 7: Bridging the Gap between Business Process Models and Service Composition Specifications

This chapter presents a flexible approach for aligning business process models with workflow specifications. Buchwald et al. propose an additional model layer, namely Business-IT-Mapping Model (BIMM), to maintain the complex dependencies that exist between high-level business process models (as used by domain experts) and technical workflow specifications (i.e., service composition schemas). Examples from practical settings illustrate the high effectiveness of this approach with respect to the maintenance of service-oriented applications.

Chapter 8: A Model-Driven Approach to Service Composition with Security Properties

Producing secure composite services is always a technical challenge that is difficult to tackle. With the emphasis in security problems, Chollet and Lalanda present a model-driven approach to service composition dealing with nonfunctional aspects. They propose a model-driven approach to facilitate the composition of heterogeneous and secured services. Authors also provide strong background work on security and model-driven topics for readers’ reference.
Chapter 9: Adaptive and Dynamic Service Compositions in the OSGi Service Platform

Toward the challenges raised in dynamic service composition, Touseau et al. provide an overview on dynamic service oriented platforms, giving special focus on the OSGi service platform. Authors present principles and mechanisms that can help to handle “dynamcity”, and supply information on the dynamic service-based component models targeting the OSGi platform. These models allow the realization of applications that are adaptive upon dynamic scenarios where service availability is uncertain.

SECTION 6: SERVICE VERIFICATION AND VALIDATION

Chapter 10: Service Composition Validation and Verification

Verification and validation techniques that can find defects to increase quality of software systems attract enormous interest and attention both in industry and academia. Palomo-Duarte analyzes plenty of approaches to web service and web service composition verification and validation. According to his full survey and in-depth analysis, there is a long way ahead to provide good support for testing techniques in WS-BPEL, the de facto service composition specification. Industrial tools include very limited testing support (mainly basic functional testing) whereas academia is taking the right steps in most directions, adapting techniques and facing trends. However, automatic tools supporting for V&V of service composition are remaining as a challenge in mainstream development tools.

SECTION 7: SERVICE MANAGEMENT

Chapter 11: Towards a High-Availability-Driven Service Composition Framework

Although availability of Web services plays a crucial role in building robust service-oriented applications, it has been largely neglected, especially for service composition. In this chapter, Lee et al. propose a service composition framework that integrates a set of composition-based service discovery mechanisms, a user-oriented service delivery approach, as well as a service management mechanism for composite services. Authors also conduct well-covered literature surveys for web service availability.

SECTION 8: SOA APPLICATIONS

Chapter 12: SOA Designed Health Care System for Taiwan Government

In Taiwan, the government agencies have promoted SOA proactively for a long period and have developed e-Government service platform for integrating a variety of government e-services. The Information and Communications Research Laboratories (ICL) of the Industrial Technology Research Institute (ITRI) is also committed in SOA research and has emphasized on the establishment of SOA service system
regarding medical information and health care. Based on their successful experiences, Shi et al. describe how to construct health care applications based on SOA and Web services technology and propose a web service converged management platform, namely Web Services Security Gateway (WSSG), to decrease the complexity of various security policy management by separating the security control and system development.

**Chapter 13: Case Study on SOA Implementation Framework and Applications**

The Enterprise Service Bus (ESB) is an important middleware technology that provides a set of infrastructure capabilities that enable the integration of services in SOA, such as event-driven processing, content-based routing and filtering, as well as complex data transmission. Innovative Digitech-Enabled Applications & Services Institute (IDEAS) of Institute for Information Industry (III) in Taiwan has implemented an ESB that comprises four modules, including Event-Service, Message-Service, Data-Service, and Process-Service. From a practitioner’s point of view, Weng et al. depict how to design and implement the ESB in depth and also discuss three industrial applications atop the ESB, including an e-learning system, a smart store system and a medical system.

**Chapter 14: Abstract Service for Cyber Physical Service Composition**

In this chapter, Zhao et al. propose an approach that models cyber physical system functionalities as services to solve the collaboration problem using semantic web services. The approach extends the existing OWL-S framework to address the natures of the cyber physical systems and their functionalities, which are different from software systems and their functionalities. The extension addresses the abstract service, resource contention, physical constraint, and context requirement issues. A case study is also provided to illustrate the proposed approach.

With a brief introduction to the outline of this book, the readers should be able to see the goals of this book, which are: to quickly inform the readers to grasp the current research status of SOC from the software engineering point of view, and also to lead the user to be aware of the importance of practicing software engineering in this new paradigm. Finally, we would like to express our sincere gratitude to everyone who has contributed to this book, including the authors and the reviewers.

*Jonathan Lee*
National Central University, Taiwan

*Shang-Pin Ma*
National Taiwan Ocean University, Taiwan

*Alan Liu*
National Chung Cheng University, Taiwan