Preface

Services computing is an emerging discipline cross-cutting the science, engineering and technology. It bridges the gap between Business Services and IT Services. The scope of services computing covers the whole lifecycle of services innovation research and practice that includes services modeling, creation, deployment, discovery, composition, analysis, and management. The goal of services computing is to facilitate the application of loosely-coupled services and computing technology for building systems more efficiently and effectively. The core technology suite includes Service-Oriented Architecture (SOA) and Web services. SOA is a common platform for implementing large scale distributed applications by composing services, which are platform independent components running on different hosts of a network. It offers native capabilities, such as publication, discovery, selection and binding for creating new applications by combining services as basic building blocks. A repository of existing services independent of the underlying infrastructures can be discovered and composed in an application. The requester and the provider exchange messages via the network through standard protocols.

SOA is now being deployed in mission-critical applications in domains that include space, health-care, electronic commerce, telecommunication, and military. Many critical systems require multiple high assurance, including reliability, safety, dependability, security, and availability. Failures of such systems may cause the loss of human lives and finance. For example, the reliability of aircraft/spacecraft navigation and guidance control systems can affect human lives; the correctness and timeliness of military command and control systems can be crucial to the success of defense missions; the failure of a medical process-control system can cause death or injury to the patient; the failure of a banking system can cause property losses for many clients; the failure of a security management system in a network server can cause chaos and result in financial or intellectual property losses; the failure of railroad control systems can cause delays and subsequent financial

losses or can even lead to catastrophic life threatening failures. In modern human society, our reliance on computer systems can be observed in our daily lives. From the current trend, our reliance on high assurance systems will grow at an increasing pace. Thus, there is a pressing need for developing computer systems whose quality can be guaranteed to a high degree; otherwise, we will risk the well-being of societies at the hands of computer hardware and software failures or misuses by human intruders. Existing methods dealing with such constraints may be not readily applied in service-oriented environment. Different from traditional computer-based systems, services are typically third-part entities. There is no standard way to define high assurance properties in service specifications. Service interfaces normally focus on the descriptions of functional aspects, such as input, output, pre/post conditions (IOPE). The high assurance properties of a service are generally unclear or defined in an ad hoc manner in the service interfaces. This poses new challenges on service discoveries with high assurance requirements.

A successful service needs to provide the required functionality and the necessary Quality of Service (QoS). The QoS parameters are typically specified in service level agreements (SLAs) that the service provider needs to guarantee and their violation will be penalized appropriately. The QoS constraints that a service provider guarantees may include run-time properties, such as timeliness, transaction rate, and availability, as well as design-time properties, such as language of service and compliance. Such high assurance guarantees are difficult to ensure when services are spatially distributed over a network subject to active attacks, network congestion, and link delays, which may pose a formidable challenge in delivering services that meet the SLAs.

There are a number of important issues in high assurance services computing:

- How to describe, assess, and ensure Quality of Service in service-oriented systems?
- How to manage and evaluate dependability of service compositions from individual services?
- How to analyze and assess the trustworthiness of service requestors and service providers?
- How to facilitate service creations and executions?
- How to verify service behavior and service level agreement?
- How to engineer service-oriented systems?
- How to test service applications?

This book is a collection of fourteen chapters solving some of these problems.

About This Volume

Chapter 1 defines separate levels of Quality of Service (QoS) assurance within a service-oriented architecture. Each of these levels includes replication options

that can bring substantial benefits toward high assurance of run-time related non-functional properties (NFP) in complex environments. Experimental results based on architectural translucency in health care applications showed an increase of 50% on the NFP levels with more stable QoS levels. The NFP representation has been formalized for automating runtime assurance and matching between required and provided QoS levels. System reconfiguration techniques for the different levels within an SOA will dynamically adapt the architecture so that it provides QoS assurance at different loads.

Chapter 2 considers the challenges of assessing highly critical net-centric systems. A trustworthiness ontology is developed to capture the trustworthiness aspects and their correlations as well as to model various classes of system entities and their integrations. The ontology provides information to guide the trustworthiness analysis and data collection. Based on the ontology, a trustworthiness assessment framework is developed. In the framework, systematic steps are formulated to achieve trustworthiness assessments. Techniques and tools to perform the assessments in each step are incorporated in the ontology to allow the actual analysis and derivation of assessment results. A holistic assessment technique is developed to provide a single overall measure of the trustworthiness of a system or a subsystem.

Chapter 3 presents a monitoring architecture for managing trust rules in service interactions. The trust rules identify the contexts of trust concerns and snapshot system events encapsulating a service outcome that is crucial to the target system. The proposed architecture, called Trust Architecture for Monitoring, may reside in each service provider, which allows the analysis of the trustworthiness of users based on trust rules and calculation schemes. A service requestor is penalized for the violation of trust rules and rewarded otherwise, which thus facilitates the quantification of its trustworthiness. Incorporating the recommendations from similar service providers may help collaborative decision making. The performance overhead of the architecture has been evaluated based on the monitoring of a prototype trust-aware file-sharing grid.

Chapter 4 addresses the key policy challenges of human interoperability enterprise (HIE) and highlights major steps that can lead to the development of a holistic interoperability policy framework for engineering high-assurance systems. The human performance criteria for high-assurance and trustworthy systems are elaborated. The HIE systems are designed by integrating core technology components and methodologies drawn from the area of human cognitive engineering. The key challenges and elicit solutions of HIE systems are closely related to the technological areas including Human-Centered Computing, Information, Knowledge and Intelligence Management, service-oriented architecture, and behavioral sciences.

Chapter 5 describes the architecture of the Service Execution Environment that hides the complexity of the communication environment and the Service Creation Environment to help service developer in evaluating the quality of an orchestration of telecom-IT services. Both static and dynamic non-functional properties are aggregated by the Aggregator service that calculates the overall aggregated non-

functional properties of a service composition designed by the developer, relying also on the Monitor manager which provides live values of dynamic non-functional properties such as response time.

Chapter 6 introduces a performance measurement framework for cyberphysical systems. The framework includes a cyberspatial reference model for establishing the identity and location of servers and clients in distributed high-assurance service systems. It also defines a set of service performance indices to measure the reliability, availability, safety, security and timeliness properties. An application neutral, yet operational definition of value useful in high assurance service systems is developed for defining their respective value propositions.

Chapter 7 applies graph grammars for verifying the behavior of service-oriented systems. The behavior verification problem is cast to a visual language parsing problem. A behavior graph is parsed with user-specified rule-based constraints/properties expressed by a graph grammar. A parsing result indicates whether the observed behavior satisfies its requirements or not. A parsing error represents a potential problem in the service behavior. The approach allows developers to check the acceptable sequence of message exchanges between services confirming to some requirements/specifications.

Chapter 8 provides a distributed service-oriented asynchronous framework in an event-driven formal synchronous programming environment. This model-driven framework is based on a synchronous programming language SOL (Secure Operations Language) that has capabilities of handling service invocations asynchronously and provides strong typing to ensure enforcement of information flow and security policies. The clients' requirements and the service level agreements can be ensured in the service-oriented systems that have been formally verified. An infrastructure for deploying and protecting time- and mission-critical applications on a distributed computing platform is developed especially in a hostile computing environment, such as the Internet, where critical information is conveyed to principals in a manner that is secure, safe, timely, and reliable.

Chapter 9 offers a coordination model for building dynamically adaptive service oriented systems. Each service is situated in and coordinated by an active architectural context, which mediates the interactions among the services. The architecture of service oriented applications is self-adaptive for bridging the gaps between environment, system and application goals with an ontology-based approach. An access control model is proposed for secure service coordination logic as well as keeping service autonomy discretionarily with a decentralized authorization mechanism. Three classes of trust relationships are also identified for a trust management framework to help the understanding and assurance of the trustworthiness of service oriented applications.

Chapter 10 develops a generalized and comprehensive framework to evaluate and maximize diversity for general service-oriented systems. The dependability attributes of individual service components under diverse operational conditions are evaluated. The internal assessments of services are linked to their external dependability attributes. The preferences of a specific set of stakeholders can also be

used to assess the relative importance and trade-off among dependability attributes. The evaluation framework also includes an overall methodology that maximizes system diversity using a mathematical optimization technique for ensuring system dependability via diversity maximization that combines collective strengths of individual services while avoid, complement, or tolerate individual flaws or weaknesses.

Chapter 11 transforms the BPEL processes into Unified Modeling Language (UML) sequence diagrams for consistency analysis. Since sequence diagrams are intuitive and show temporal-based execution naturally, they help to ease the learning curve of BPEL's nomenclature and reduce errors. Two examples have demonstrated the discovery of certain errors in the sequence diagrams with tool support.

Chapter 12 specifies both structurally and behaviorally the Enterprise Web-Oriented Architecture (EWOA) and analyzes its software quality attributes. The specification of the EWOA is based on a generic model of the Enterprise Service-Oriented Architecture. The EWOA style consists of a set of design principals based on REST and Web 2.0, a set of architectural elements of infrastructure, management, process, and a set of software quality attributes. Based on the analysis of the security and manageability issues of EWOA, the pure RESTful system architecture with RESTful QoS governance and a hybrid approach with both REST and SOAP for enterprise are proposed.

Chapter 13 outlines a service oriented architecture for the Peer-Assisted ContenT Service (PACTS) that is a video on demand streaming system. The PACTS organizes elements of traditional video streaming and peer to peer computing into loosely-coupled composable middleware services and distributing them among participating entities for high-quality low-cost video streaming at a large scale and in real time. The implementation of PACTS has demonstrates effectively offload server's bandwidth demand without sacrificing the service quality and in dynamic settings with system churns. It shows significantly reduces bandwidth utilization at the server by leveraging peer assistance. The service level agreement specification is modeled to differentiate QoS to end users based on their bandwidth contributions to the system to derive the minimum and maximum QoS level given a bandwidth budget at the server side.

Chapter 14 proposes a Model-based Adaptive Test (MAT) for multi-versioned software based on the Coverage Relationship Model (CRM) for case selection and ranking technique to eliminate redundant test cases and rank the test cases according to their potency and coverage. It can be applied in various domains, such as web service group testing, n-version applications, regression testing, and specification-based application testing. Two adaptive test cases ranking algorithms are provided by using the coverage probability. Experiments are conducted using the proposed techniques. The experiment results indicate that the CRM-based test case selection algorithm can eliminate redundant test cases while maintaining the quality and effectiveness of testing.

This book is intended particularly for practitioners, researchers, and scientists in services computing, high assurance system engineering, dependable and secure systems, and software engineering. The book can also be used either as a textbook for advanced undergraduate or graduate students in a software engineering or a services computing course, or as a reference book for advanced training courses in the field.

Acknowledgements

We would like to take this opportunity to express our sincere appreciation to all the authors for their contributions and cooperation, and to all the reviewers for their support and professionalism. We are grateful to Springer Publishing Editor Susan Lagerstrom-Fife and her assistant Sharon Palleschi for their assistance in publishing this volume.

Jing Dong
Raymond A. Paul
Liang-Jie Zhang

http://www.springer.com/978-0-387-87657-3

High Assurance Services Computing
Dong, J.; Paul, R.; Zhang, L.-J. (Eds.)
2009, XII, 324 p. 130 illus., Hardcover
ISBN: 978-0-387-87657-3