Preface

Information technology is rapidly evolving to provide an environment where applications will be highly adaptable, interoperable, and portable. In this new era of information technology, computing resources will be infinitely scalable and pervasive; dynamic discovery of heterogeneous resources to compose complex applications will be routine; business and scientific applications will be context-aware and intelligent and adapt themselves autonomically to the varying ecosystems; interactive applications will be accessible from anywhere and anytime; and seamless federation between on-premises and public computing infrastructures will be common.

Furthermore, in this new era of computing, predictive analysis of huge amounts of unstructured data will be the norm; networks will be software controlled and rented as a service to any customer; provisioning of scalability, security, high availability, bandwidth, latency, fault-tolerance, and elasticity will become configurable properties of the application architecture; software applications will continue to function efficiently even if the underlying physical hardware fails or is replaced; data encryption modes will support remote computing without decryption at reasonable cost; and advanced computing will be energy efficient.

The requirements for this next generation of computing have been the primary drivers to generate momentum for research into advanced service architectures that have enabled the design, development, and implementation of smart and adaptive service-driven applications.

Our mission for this comprehensive two-volume *Handbook of Research on Architectural Trends in Service-Driven Computing* is to provide a collection of innovative and advanced research manuscripts that explore the challenges of and recommend solutions to architecting and implementing adaptable, scalable, and dependable systems, thereby enabling enterprises to efficiently align their robust information technology with their agile business requirements and foster the next generation in service-driven computing.

The handbook discusses concepts and delineates the recent advances in architecture methodologies and implementation techniques that will enable effective semantic description of services, semantically aware service discovery, dynamic adaptability to the environment and user needs, robust pervasive services, context-aware service reconfiguration capabilities, testability and reliability of adaptive services, ubiquitous access to services, and seamless integration in the service-driven ecosystem.

By incorporating research ideas, architectural practices, frameworks, platforms, and implementation methodologies recommended by researchers, academicians, enterprise architects, and practitioners from around the world, this handbook strives to serve as a valuable information technology reference for organizations, researchers, students, enterprise and integration architects, SOA practitioners and
evangelists, software developers and implementers, and software engineering professionals engaged in the research, design, development, and implementation of next generation architectural practices, tools, and techniques that will enable smart services and seamless enterprise integration.

This reference handbook is structured as a two-volume book. The first volume is comprised of 15 chapters that address the aspects of dynamic and adaptive architectures for service-driven computing. The second volume is comprised of 14 chapters and covers 2 other aspects: integration in the service-driven ecosystem and service-driven computing in the Cloud.

While Service-Oriented Computing (SOC) is closely aligned with SOA, Service-Driven Computing involves the use of software services that conform to standard service architecture methodologies, such as Service-Oriented Architecture (SOA), Resource-Oriented Architecture (ROA), etc. to drive computing solutions that enable building massively distributed software systems for this new generation of context-aware, dynamically adaptive, self-organizing, self-healing, and ubiquitous applications. In Chapter 1, I (Raja Ramanathan) introduce service-driven computing, explore the challenges, and discuss significant solutions proposed by researchers to mitigate them. Specifically, this chapter delves into composing adaptive services dynamically, supporting context-awareness and autonomic capabilities in services, verification of dynamic service compositions, and extending the service-driven paradigm to the Cloud.

Loose binding between services enables the creation, destruction, and updating of services dynamically at runtime to move them into different contexts and to different providers. A series of such reconfigurations contributing to the evolution of the architecture may lead to incorrect architecture configurations that were not considered during the system design phase. Dynamic aspects of behavior of software systems in dynamically reconfigurable runtime architectures can result in significant architectural violations during runtime. In Chapter 2, Marek Rychly describes and compares possible measures to prevent architectural violations in dynamic service and component models. The author evaluates the applicability of those measures in combination with advanced features of reconfigurable runtime architectures such as ad hoc reconfiguration, service or component mobility, composition hierarchy preservation, and architectural aspects.

In the centralized architecture of BPEL engine solutions, process instances are isolated from each other as well as from any other potential sources of information, thus preventing them from finding relative data at runtime to adapt their behavior in a dynamic manner. In Chapter 3, Michael Pantazoglou, George Athanasopoulos, Aphrodite Tsalgatidou, and Pigi Kouki explain how centralized business process execution engines are not adequate to guarantee smooth process execution in the presence of multiple, concurrent, long-running process instances exchanging voluminous data. The authors present a solution using a distributed architecture with hypercube peer-to-peer topology together with artificial intelligence planning and context-aware computing techniques to enable the scalable execution of service-oriented processes, at the same time supporting their data-driven adaptation. This solution facilitates the discovery of process execution paths at deployment time and improves the overall throughput of the execution infrastructure.

A typical service-driven system includes a large number of interacting services and components. Therefore, a change in the running state of a single component can have a ripple effect on the entire service-driven application. If a component were to shut down for update, other components may not be able to operate effectively. In Chapter 4, Tony Clark, Balbir Barn, and Vinay Kulkarni address the problem of dynamic reconfiguration of component-based architectures. They propose a dynamic architecture that supports changing the behavior and topology of existing components without stopping, updating, and redeploying the system. Their approach involves representing key features of a component in data
that would otherwise be rendered in program code, so that a system can reason and dynamically modify aspects of it. Representation of key features in data form ensures that they can be processed, modified, and replaced without changing the program code.

The growing maturity of smart mobile devices has fostered their prevalence in the enterprise, thereby requiring enterprise business processes to extend themselves to incorporate support for mobile devices. However, process tasks that were originally designed to execute in stationary environments cannot be easily provisioned to run on smart mobile devices. Previously, researchers have proposed fragmentation of the business process to enable mobile support. In Chapter 5, Rüdiger Pryss, Steffen Musiol, and Manfred Reichert oppose the traditional fragmentation approach and instead propose a novel approach to enable the robust and flexible execution of single process tasks on smart mobile devices by provisioning self-healing techniques to address the smooth integration of mobile tasks with business processes.

A category of services based on data propagation among mobile devices is evolving and presents new challenges in terms of reliability, real-time adaptation in dynamic infrastructures, and availability to a large scale of users simultaneously. In Chapter 6, Giovanna Di Marzo Serugendo, Jose Luis Fernandez-Marquez, and Francesco Luca De Angelis explain the concept of Spatial Services using self-organizing services and describe the design of applications using the active tuple space model inspired by chemical reactions. The authors present the architectural approach and execution models to implement the spontaneous self-composition of spatial services and self-management of their non-functional aspects to meet the challenges of this evolving category of services.

The evolution of both distributed systems and mobile computing has resulted in the emergence of ubiquitous (or pervasive) computing. In Chapter 7, Moeiz Miraoui describes dynamic adaptation in ubiquitous services by providing adapted services proactively, without explicit user intervention, and according to the user’s current context. The domain of ubiquitous services requires further research in terms of comprehensive context modeling, architecture of context-aware ubiquitous systems, and dynamic adaptation approaches. In this chapter, the author proposes a conceptual architecture to provide dynamic adaptability in ubiquitous services based on context-awareness and user preferences by using an ontology-based context modeling approach, a multi-agent architecture to support the development of ubiquitous computing applications, and a case-based reasoning method for service adaptation.

It is expected that virtual factories and enterprises of the future will be delivered dynamically as a service, end-to-end along the global value chain. Therefore, it is necessary to model and manage them appropriately for automation and scalability. Typically, human intervention is required to achieve most service-related tasks such as discovery, ranking, invocation, and monitoring. Manual intervention required during integration is not feasible with service-driven systems that are dynamically composed of numerous services. In Chapter 8, Ioan Toma, José María García, Iker Larizio, and Dieter Fensel propose a semantically enabled service-oriented architectural approach and platform using lightweight semantics to address the required automation and scalability in service compositions. The proposed platform provides automatic support for overall service delivery, including various service related tasks such as discovery, ranking, invocation, and monitoring for both WSDL-based and RESTful services.

The grand vision of autonomic computing projects is self-configuring, self-healing, self-optimizing, and self-protecting capabilities for the new generation of services. However, incorporation of self-management features into the software increases its complexity, thereby making it more difficult to validate at development time. In Chapter 9, Tariq King, Peter Clarke, Mohammed Akour, and Anjani Ganti discuss the challenges in validating autonomic services that can automatically integrate with other systems and adapt to changing system environmental conditions. Due to the structural and behavioral
runtime changes caused by dynamic adaptation in autonomic software, they cannot be validated at design time. In this chapter, the authors summarize the state-of-the-art in runtime testing of autonomic systems and provide recommendations for integrating runtime testing approaches into autonomic software using self-testing architectures.

A true service-oriented architecture describes everything, anywhere, anytime as a service. As we tackle the complex problems of the service era, new modeling or programming or both (mogramming) languages are required. In Chapter 10, Michael Sobolewski describes a service-oriented computing platform that provides a service mogramming environment, service context-awareness, and management of everything as a service type within its operating system. The author presents an architecture case study of this platform by describing its metacomputing architecture that enables it to seamlessly unify all existing programming styles and allow the end user to create new compound services at runtime that are both globally and locally distributed federations of services.

Mashups of multi-media contents are becoming aligned with the service-driven trend, which will present new challenges for the agile and economic service provisioning of multi-media over the underlying infrastructure. In Chapter 11, Sang Woo Han and JongWon Kim point out the architectural inflexibility of legacy multimedia systems, which makes them difficult to reuse existing components. As a solution, the authors propose a conceptual model and coordinated workflow-style implementation approach to compose adaptive media-centric services with programmable and virtualized resources, and to create customized media-centric services with agility that can meet the diverse requirements of end users. They also describe a software tool to support easy experimentation of media-centric mashups and validate the same in a realistic testbed.

Traditionally, Quality of Service (QoS) and Web Service standard (WS-*) specifications for non-functional aspects of Web services are scattered and intermingled all over the core service code, thus reducing the reusability of these services and making their maintenance expensive and complex. In Chapter 12, Areeg Samir proposes an adaptive Web service architecture to enhance the reusability of services by using an aspect-oriented approach that separates crosscutting concerns such as QoS and WS-* specifications in aspect Web services and enables integrating them on the fly with the core Web services.

With device diversity, mobile consumers are exposed to a large pool of services that must be discovered in such a way that they best meet user requirements. Mobile service discovery needs to effectively support mobility-related constraints such as availability, heterogeneity, and resource-constrained devices. In Chapter 13, Salma Bradai, Sofien Khemakhem, and Mohamed Jmaiel attempt to categorize service discovery approaches according to several criteria such as architectural choices, service description, and semantic and reasoning techniques, and evaluate the existing approaches in mobile service discovery to identify their strengths and weaknesses, and propose guidelines for future research in service discovery for mobile environments.

Service composition is a complicated task to be handled manually. Automated Web service composition involves organizing, combining, and executing Web services to satisfy user goals through the process of reasoning that resolves constraints between the inputs and outputs of the involved services. In Chapter 14, Tamer Al Mashat, Fatma El-Licy, and Akram Salah present a framework for Web service composition based on semantic specification to establish an ontological agent for automating Web service composition. A Petri nets model is applied to build a formal representation of the structure and behavior of the service and AND-OR graph methodology is used to select between alternatives based on QoS.

Chapter 14 completes Section 1 of the handbook. Section 2 explores integration challenges and proposed solutions in a service-driven ecosystem.
While both SOAP and REST messaging have been used widely to implement Web services, RESTful Web services provide greater development flexibility and their architectural conventions, and best practices can be integrated into Web services incrementally as opposed to the all-or-nothing adoption of SOAP-based Web services. In Chapter 15, Hiranya Jayathilaka, Chandra Krintz, and Rich Wolski discuss the strengths and weaknesses of SOAP- and REST-based Web services and provide evidence and reasoning behind the emergence of REST as the leader for the development of next-generation Web APIs and services. They describe emerging and future research directions in support of REST-based APIs and service development.

SOA has a low semantic gap but introduces coupling between the provider and the consumer that hampers dynamic changes to the interface. On the other hand, REST has a higher semantic gap but attempts to reduce the coupling between provider and consumer to increase scalability and adaptability. In Chapter 16, José Delgado explains why the dynamic Web of today needs a new approach to integration and proposes a new architectural style that combines the best characteristics of the SOA and REST styles. In this architectural style, unlike REST, resources are able to offer a variable set of operations, and unlike SOA, services are allowed to have structure. To minimize resource coupling, this style uses structural interoperability based on the concepts of structural compliance and conformance. The author explains the implementation of this style using a native distributed programming language.

Despite their obsolescence, legacy mainframe systems continue to provide a competitive advantage by supporting unique business processes and acting as a repository for knowledge and historical data in enterprises. However, enterprises would prefer to develop their applications with modern software technology instead of continuing to develop in the mainframe and, at the same time, leverage existing business processes and data from the legacy systems. In Chapter 17, Chung-Yeung Pang proposes an architectural framework and implementation methodology for legacy integration in a service-driven ecosystem. The framework uses an Enterprise Service Bus and intelligent agents that use a rule-based engine and context awareness to easily integrate legacy COBOL applications as services into the system without any programming effort on the mainframe.

There is a need for continuous monitoring of controls and systematic collection and evaluation of relevant data in enterprises to ensure information system compliance with laws and regulations. Since the violation of compliance requirements can lead to significant penalties for an enterprise, compliance management should be taken very seriously. In Chapter 18, Natallia Kokash provides an insight into IT compliance management and discusses the integration and automation of compliance management in the service-driven ecosystem. The author discusses business process management and conceptual models for specifying compliance requirements and details the mechanisms for automating compliance management through formalization of compliance requirements, rule- and event-based monitoring, and integration of compliance governance systems with automated reasoning and verification tools.

Driven by the massive growth in data generated by complex instruments, data-intensive science has emerged as a significant scientific paradigm. In Chapter 19, Lawrence Yao, Fethi Rabhi, and Maurice Peat explore enabling technologies and trends for supporting researchers working on data-intensive analysis processes. They identify that to efficiently support the work of these scientists, software applications should satisfy the essential requirements of interoperability, integration, automation, reproducibility, and efficient data handling. The authors review various enabling technologies such as workflow, service, and portal, and illustrate that no one technology is able to address all of the essential requirements of scientific processes, and therefore propose the use of hybrid technologies to support the requirements of data-intensive research.
The scientific community produces and consumes massive volumes of unstructured and heterogeneous data from various data sources. Scientists need access to distributed computing and effective software tools for describing and querying heterogeneous data, and performing data analysis and visualization with that data. In Chapter 20, Mariana Goranova proposes a scientific data management and visualization system for scientists to perform specialized data browsing, processing, and visualization using a service-driven integration approach. The author discusses the proposed methodology for the specification and access of scientific data from various sources with different formats and transformation of raw data into standard datasets that can be analyzed, processed, and visualized in an effective manner.

The service entities that interact in a service-driven system share knowledge only about the service interface. Although communication protocols can ensure secrecy of messages, those messages can carry complex data and even executable instructions, which makes it difficult to provide security guarantees for both the service consumer and the provider. In Chapter 21, Gabriele Costa, Roberto Mandati, Fabio Martinelli, Ilaria Matteucci, and Artsiom Yautsiukhin point out that in the realm of service compositions, consumer security requires providing strong guarantees that a requested security policy is satisfied. The authors discuss threat models for several different Web service paradigms involving service consumers, providers, and platforms, and propose guidelines for mitigating risks in service interactions in the identified paradigms using contract-based security approaches and its new proposed variants.

This concludes Section 2 of the handbook. Section 3 delves into extending service-driven computing to the Cloud.

Cloud computing enables compute intensive applications, high availability of systems, and cost-effective IT delivery solutions. Cloud computing has revolutionized scalable on-demand information delivery in the form of services. However, in Cloud computing, data ownership and storage locations are not under the control of the enterprise. Moreover, the shared resources nature of Cloud computing delivery hinders data confidentiality and integrity. It is therefore important to provide business decision makers with credible data about the factors to consider in the decision making process for adoption of Cloud computing in their enterprises. In Chapter 22, Omondi John Opala, Shawon Rahman, and Abdulhameed Alelaiwi provide an overview of Cloud computing, present a quantitative non-experimental study to evaluate security, cost-effectiveness, and IT compliance factors that can influence an IT manager’s decision to adopt Cloud computing, and explain the results of a linear regression analysis testing.

The ability to acquire and release resources dynamically and trivially in the Cloud, while being an efficiency and cost-savings aspect for consumers, complicates the resource provisioning and allocation task in the Cloud. Resource under-provisioning can hurt application performance and deteriorate service quality; on the other hand, resource over-provisioning could cost users more and offset Cloud cost advantages. In Chapter 23, Ming Mao and Marty Humphrey explore state-of-the-art research in resource provisioning in the Cloud based on several factors such as the type of workload, VM heterogeneity, data transport requirements, solution methods, and optimization goals and constraints, and recommend guidelines for future research.

Cloud service brokerage refers to the customization, aggregation, and integration of existing Cloud services, possibly by third parties. Cloud service brokerage has been identified as a key concern for future Cloud technology research and development. Cloud brokers need to cater to both horizontal and vertical integration, horizontally between different providers and vertically across the different Cloud layers. In Chapter 24, Frank Fowley, Claus Pahl, and Li Zhang propose a conceptual framework for a Cloud service
Technology obsolescence is a major challenge for the software development industry due to the constant evolution of hardware and software technologies. Traditionally, Model-Driven Architectural (MDA) approaches have been used to alleviate the undesirable effects of technology obsolescence in software. In Chapter 25, Ritu Sharma and Manu Sood propose a solution to the technology obsolescence problem in Cloud SaaS (Software as a Service) by converging three paradigms, namely Cloud computing, SOA, and MDA, to yield Cloud software services that are robust, flexible, and agile.

Platform as a Service (PaaS) provides a convenient software development platform in the Cloud; however, it often comes with constraints in terms of application architecture and provider lock-in. On the other hand, the flexibility of the Infrastructure as a Service (IaaS) model enables application development freedom; however, it necessitates operation at the lower level of virtual machines and snapshots. In Chapter 26, Rostyslav Zabolotnyi, Philipp Leitner, and Schahram Dustdar propose a low-overhead middleware framework that seamlessly enables the federation of Java applications to the Cloud with minimal changes in the application code, minimal overhead, and less Cloud-awareness required from developers. The authors explain how their framework influences Cloud software development and deployment.

Business Intelligence (BI) software is generally available as a monolithic application in the Cloud, providing both computing and reporting functionality internally. Vendors focus on providing existing BI products, such as Software as a Service, rather than attempting to integrate business intelligence functionality with other applications. In Chapter 27, Volker Herwig and Kristof Friess identify common usage scenarios of BI Cloud applications that demand integration of the analysis reports generated in BI software with the user interface of other Cloud applications. The authors discuss the technical concept and illustrate how the required interface can be provisioned.

Cloud computing enables the flexible and on-demand provisioning of computing resources, facilitating cost-savings for enterprises and institutions. However, an increase in the demand for Cloud challenges providers to provision resources cost-effectively and at the same time meet the SLA requirements of the customers. In Chapter 28, Jose Vazquez-Poletti, Rafael Moreno-Vozmediano, and Ignacio Llorente introduce and compare Admission Control algorithms and propose a service model that allows the definition of service-level agreements for the Cloud to maximize the net income derived from provisioning the accepted service requests and minimize the impact of unprovisioned resources.

Open Source Software (OSS) for Cloud infrastructure is attracting attention due to cost efficiencies and quick delivery. However, in general, the reliability of OSS is not well trusted. In this final chapter of the book (Chapter 29), Yoshinobu Tamura and Shigeru Yamada propose new stochastic models for reliability assessment of Cloud OSS. The authors illustrate how reliability analysis can assist in quality improvement in Cloud computing software by introducing a new approach to the traditional Jump Diffusion process that considers the numbers of components and users in the reliability model. They also integrate the reliability model with a threshold-based neural network approach that estimates network traffic in the Cloud. Actual software fault-count data are analyzed to provide numerical evidence of the software reliability assessment.
In this handbook of research, we have attempted to consolidate relevant concepts, ideas, techniques, and tools by providing easy-to-read chapters that cover the basics, state-of-the-art, and novel proposals to ignite a spark of inspiration in the minds of readers. We hope this reference handbook will be pertinent and useful to students, researchers, and practitioners alike, and ultimately help further the research and implementation of smart services and robust enterprise integration.

Raja Ramanathan
Independent Researcher, USA

Kirtana Raja
IBM®, USA

ENDNOTE

1 Disclaimer: Contributions to this reference book are my own and do not necessarily represent IBM’s positions, strategies, or opinions.