Preface

As a professional IT enterprise architect and SOA subject matter expert, I have provided enterprise architecture and development expertise and best practice recommendations to large enterprises in the oil and gas, energy and utilities, financial services, and software product development domains. In my experience, I have observed that most enterprises that venture into the Service-driven architectural methodology for designing and developing their enterprise solutions, do so in bits and pieces; without being guided by best practices in taking a comprehensive view of the service architecture within the overarching enterprise architecture framework, or without realizing the benefit of a structured governance approach right from the inception of the service development lifecycle. This results in the proliferation of “islands of services” that do not integrate with all mission critical applications in the enterprise; thus reducing the potential business benefit derived from implementing a Service-driven model and leading to vendor lock-in in strategic areas of the enterprise and an inflexible enterprise architecture.

There is a dearth of written material that clearly delineates the challenges and architectural best practices applicable to the Service-driven architectural practices for software architecture and enterprise integration. These challenges motivated me to write a book that can cater to a diverse audience of enterprise architects, integration architects, software engineers, technical managers, researchers, and developers, that captures the breadth of my experience and the experiences of brilliant researchers and practitioners from around the world.

The goal of this book is to explore and discuss the challenges of Service-driven architectural approaches that have been a major driver for enabling agile, cost-effective, flexible, and extensible software services and integration solutions; and present architectural approaches, implementation techniques, and best practice recommendations for designing and implementing effective architectural solutions that support the business dynamics of today’s fast-paced enterprises.

The book first lays the groundwork by discussing the principles of Service-driven architectural practices and its application to software architecture and enterprise integration. Subsequently, architectural approaches to tackle the significant challenges of heterogeneous enterprise integration, autonomous mediation of services in a service composition, enrichment of business process models to enable cost-effective process design, policy-driven fault management in long-running business processes, comprehensive security architecture for pervasive service applications, improvement of service composite dependability, holistic governance of the service lifecycle, optimization of the coordination strategies of enterprise services and data, and avoidance of vendor lock-ins in enterprises, are presented. Finally, the book explores how Service-driven approaches can be effectively leveraged in the emerging areas of Mobile Computing and Cloud Computing to meet enterprise initiatives.
In Chapter 1, I have explored the principles of the Service-driven methodology for architecture of software components and services. A Service-driven approach typically uses Service-oriented Architecture (SOA) to design and implement enterprise applications and integration solutions; though more recently, RESTful Web APIs are driving easy to use, agile, and low cost techniques for Service-driven development in the enterprise. Service-driven architectural approaches enable the interaction and interoperability of services within and across enterprises. Well designed, standards-based, Service-driven approaches can provision a flexible infrastructure and processing environment for the enterprise and provide a robust and secure foundation for leveraging services for enterprise integration of data and business processes.

A common misconception is that, because services are defined through the interface and not the implementation, changes in the service implementation will not affect service consumers. This chapter attempts to clarify this misconception and presents a valid case for service versioning. This chapter also delves into Service Mediation which is an abstraction layer that decouples service consumers from service providers and incorporates protocol and message level transformations to enable connectivity between consumers and providers. The mediation layer can also handle cross cutting concerns such as authentication, encryption, and audit logging across all service requests, thereby relieving individual service providers from having to deal with these common concerns.

Enterprise business solutions are created by combining or composing individual fine-grained or coarse-grained application services into business services. Service Compositions can be implemented primarily using orchestration and choreography mechanisms. This chapter explores Service Composition in greater detail. Enterprises have to constantly modify and improve business processes to align with new business decisions, practices, and regulations which make business processes highly dynamic in nature. There is a widespread need for composition techniques that can transparently adapt to unforeseen changes and requirements in the business environment with minimal user intervention.

The Enterprise Service Bus (ESB) provides a Service-driven integration backbone with mediation and orchestration capability. This chapter provides an overview of the ESB. Furthermore, it also delineates Workflow services that can service-enable business processes to create a synergistic ecosystem.

Web Services implemented using APIs have been trending away from using SOAP messaging towards the faster and more direct REST architectural style. Enterprises have historically made use of proprietary APIs; however, in recent years, enterprises are utilizing more Web-based public APIs to enable functionality via the Internet. Mashups, which are novel applications created by aggregating API-specified functionality and data sources have impacted the way software is developed in the enterprise setting. RESTful Web APIs are also pertinent in the areas of Social Media and Cloud applications. This chapter explores the business value of APIs in the enterprise and investigates the value-add to Social Media and Cloud enterprise initiatives and delves into the architectural considerations in implementing Enterprise APIs and their incorporation into the overall Service-driven strategy of the enterprise.

In Chapter 2, Leo Shuster describes the challenges of enterprise integration. Spaghetti integration (or point-to-point integrations) presents one of the most typical integration problems in the enterprise. This chapter classifies integration challenges into organizational, technological, and process related challenges and attempts to provide practical best practice solutions. Various architectural approaches and design patterns have been created to address the integration challenges. They have evolved from simple, proprietary techniques to standards-based mechanisms. Today, Service-driven architectural approaches are the predominantly used integration methodology.

This chapter explores why many enterprise IT shops struggle to adequately integrate their systems. Integrations have to enable heterogeneous systems, processes, and platforms to be able to talk to each
other and accommodate for continuously changing application interfaces and functionality. Enterprises that do not have well defined Service-driven integration approaches in place or address the integration requirement without a strategic plan to manage the integrations in the long term will face the issue of integration proliferation that leads to systems being dependent on each other and completely exposed to one another’s internal changes. Change in one system results in cascading changes through multiple integrations and related systems causing an integration nightmare.

In Chapter 3, Venky Shankararaman and Alan Megargel explore the various architectural patterns for integrating enterprise applications, namely, hub-and-spoke messaging, SOA with ESB, SOA with ESB and BPM, and Event driven architectural styles. The integration patterns examined here vary in terms of complexity versus business benefit. With an ESB, services are developed once and then used by multiple consumers. The ROI (Return on Investment) on service development is quickly realized when two or more composite services reuse an existing service. Besides cost avoidance, another primary goal of an ESB solution is flexibility that enables changes to the services with little or no impact on service consumers. This is made possible by decoupling the front-end service consumers from the backend service providers through data abstraction.

Business processes can be optimized for reuse across an enterprise much like business services exposed via an ESB can be optimized for reuse. The benefit of reusable assets is amplified when BPM and ESB patterns are used together. While BPM and ESB patterns are effective and efficient in executing business processes, there is some limitation in the way processes can be started. They need to be initiated through events or human interaction. Event-driven Architecture is based on the concept that events drive business interactions. More recently, enterprises have become interested in proactively monitoring, analyzing, and acting upon business events in order to support decision making in real time. Event processing can help with detecting event patterns that satisfy constraints and enable enterprises to leverage current event streams in order to predict the future.

In Chapter 4, Prashant Doshi and Nithya Vembu describe establishment of message exchange between related but independently developed services as a key challenge faced during service composition. The challenge lies in resolving the differences in the message schema that are input to and output from the services involved. Data mediation is required to resolve these challenges. They introduce a model for autonomous data mediation that considers the types and semantics of the message elements. It has its roots in schema mapping techniques, but there has been a general scarcity in applying those techniques to the relatively new problem of mediating between services.

Standardized interfaces, platform transparency, and the Web-based loosely-coupled nature of services make their automated composition feasible. Considerable attention has been directed to the problem of automatically determining the flow of control in a service composition with the objective of satisfying the functional and non-functional objectives of the composition. Given that services are usually independently developed, their input and output message schemas are often heterogeneous with structural heterogeneity at the attribute-level, entity-level, and abstraction-level. This chapter introduces a mathematical model of the message-level mediation problem that captures the common types of heterogeneity in service messages and prescribes a translator function that defines the set of syntactic and semantic rules that govern the transformation of message schemas. Mediating message-level heterogeneity is a key step toward successful execution of service compositions.

In Chapter 5, Thomas Bauer, Stephan Buchwald, and Manfred Reichert introduce the concepts of front-loading and look-ahead to present two fundamental techniques for achieving the goal of capturing sufficient process details during the design phase itself. A key objective of Service-driven architectural
approach is to improve the alignment between business and IT. Business process management (BPM), service composition through orchestration and choreography play major roles in achieving this goal. In particular, they enable process-aware integration of people, data, and services. Any Process-aware Information System (PAIS) should meet the needs of all stakeholders involved in the business processes it implements. To meet that requirement, the PAIS must capture all business process model aspects relevant for process automation in a sufficient level of detail during the design phase itself. Furthermore, existing service artifacts should be reused in this context if available. To optimize business-IT alignment and achieve high business value, the business processes implemented in PAIS must be defined by domain experts, and not by IT experts. However, in the current practice, many aspects required by process implementers are missing in the business process models or are only defined in a rather imprecise manner. Consequently, this results in significant delays during process implementation or in imprecise implementation of the business process model.

With contemporary business process modeling tools, front-loading and look-ahead can be realized to some degree for selected process aspects; however, existing process modeling tools do not provide an appropriate basis for business process implementation due to their restricted functionality as well as the incomplete and ambiguous specifications that they capture. This chapter presents appropriate techniques for enabling front-loading and look-ahead for selected process aspects and delves into how executable process models can be derived from business process models, if the latter are enriched with additional information.

In Chapter 6, Stephan Reiff-Marganiec and Manar Ali present a framework to provide autonomous handling of long running transactions based on dependencies which are derived from the workflow. Business processes involve long running activities and require transactional behavior across them. A significant requirement of Long Running Transactions (LRTs) is the preservation of the consistency of the systems being involved in the LRT. The database community has traditionally implemented transactional methodologies to ensure the integrity of data. It would seem appropriate to employ the same techniques to manage the integrity of long running business processes. However, there are a number of new challenges with respect to business processes that necessitates using new techniques. The two most significant challenges are the long running nature of business transactions and the delicate and complex nesting that naturally occurs. Database transactions usually complete within a matter of seconds and the most common solution for addressing the transactional integrity challenge in databases involves some form of resource locking. This is not possible with business transactions as these often span several days or weeks or even years.

This chapter presents a solution for forward recovery from errors by automatic application of compensation to executing instances of workflows. The mechanism is based on propagation of failures through a recursive hierarchical structure of transactional components. A notable feature of this model is the accountability of vital and non-vital components, allowing the process designer to express the cruciality of activities in the workflow with respect to the business logic. Another novel feature is that the approach also permits the workflow designer to specify additional dependencies which will also be enforced. The approach is implemented through workflow actions executed by services and allows management of faults through a policy-driven framework.

In Chapter 7, Aurélien Faravelon and Stéphanie Chollet outline the challenges in access control enforcement in service compositions. Service-driven architectural approaches are a major trend for developing pervasive applications. However, security with respect to access control and data privacy is not currently managed comprehensively from design time through run time. Dynamic service compositions
are complex, especially when they have to enforce an access control policy defined at the application level. Challenges include the need to have policies that are easy to understand even for non-technical users who may be customizing the application security of these pervasive applications; and the need to have both fine-grained and contextual policies so that the privileges and contextual information of users can be managed at runtime and be able to support a highly heterogeneous and dynamic environment for application runtime. This chapter presents a generative approach to manage security in pervasive applications at design time using a model-driven architecture based on models pertaining to access control management that respect the temporal constraints of pervasive applications.

In Chapter 8, I explore the typical challenges faced by enterprises for implementing scalability and fault tolerance in a Service-driven environment and recommend architecture best practices to improve fault-tolerance for services. Service-driven environments are different from traditional user-centric application environments in terms of the magnitude of the rate of consumption requests generated. Consequently, the technologies that were capable of handling traditional user loads are unable to keep up with the increased load requirements associated with a service-driven environment. Service-driven approaches enable to distribute applications or services to multiple locations within an enterprise and off-premises. Enterprises implementing Service-driven solutions face challenges relating to unprecedented scale, high availability, and fault-tolerance. Service availability and fault-tolerance requires the elimination of all single points of failure within a given service and its dependents. Maintaining fault-tolerance in business services is a significant challenge due to their compositional nature, which instills dependencies among the services in the composition. This causes the dependability of the business services to be based on the reliability of the individual services in the composition.

This chapter delves into the architectural approaches, such as, service redundancy and design diversity, scaling, clustering, distributed data caching, in-memory data grid, and asynchronous messaging for improving the dependability of services. It also explores the data scaling bottleneck in data centralization paradigms and illustrates how that presents significant scalability and fault-tolerance challenges in Service-driven environments. Failure recovery such as backward and forward recovery mechanisms as well as exception handling and transactional compensation practices in business processes are also discussed.

In Chapter 9, Leo Shuster presents specific approaches to governing each stage in the lifecycle of a Service-driven environment and provides guidelines on best practices associated with them. A Service-driven approach manages the delivery and reuse of shared services. Since each service is managed independently, its evolution can be described as a well defined lifecycle. A service can pass through each of its lifecycle stages multiple times until it reaches retirement. The service versioning stage starts a new iteration of the entire lifecycle which results in a creation of a new service version. There may be multiple service versions running at the same time. Each of them can reach retirement at different times depending on company’s policies and its ability to migrate consumers to the new version of the service. Different actors participate in different phases of the service lifecycle. These range from a service architect to a developer and support technician. While roles may be persistent across phases, their expected contributions and deliverables change from one phase to another.

Enterprise services are reused across organizational boundaries, while domain services are reused within a single organization, business unit, or functional domain. To pursue the enterprise goal of creating shared services and increasing their reuse requires effective governance of the Service-driven environment. Since services could be owned and consumed by various business units within an organization, all changes have to be carefully choreographed at the enterprise and business unit levels. SOA Governance tools help define, automate, and execute SOA governance processes. SOA governance is critical to the
success of a Service-driven environment. Without its guidance, best efforts will quickly disintegrate in the face of political resistance, lack of cooperation, and unsustainable funding. This chapter delves into program, development, implementation, runtime, and change time governance methodologies, and explores funding models required for maintaining an effective SOA program in the enterprise. To be successful, the Service-driven initiative has to demonstrate value by collecting and communicating metrics to the stakeholders. The chapter attempts to provide guidelines on demonstrating value of the SOA program to enable increased adoption of the Service-driven approach. Current and future state of governance tools and technologies as well as service runtime management is also discussed.

In Chapter 10, Keith Worfolk describes how an organization’s Service-driven and Enterprise Data Management (EDM) strategies can fall short of expectations and potential business value, if key aspects and interrelated components of each strategy are not taken into account in a coordinated manner. Strategies for Service-driven and EDM approaches have traditionally been treated as disparate programs and initiatives within organizations, both from a business requirements perspective as well as from an IT implementation perspective. However, there are critical overlapping and interdependent components, processes, and quality checkpoints of each strategy, for which coordination between them becomes necessary in order to ensure mutual success. Pursuing these initiatives as unrelated approaches and with disconnected objectives and measures of success, often risks the success of both strategies, and is likely to execute poorly and ultimately fail to deliver the expected business results.

This chapter delineates the Service-driven and EDM strategic coordination points, addressing their synergies, dependencies, and interrelated infrastructure and processes. This chapter also prescribes a facilitative Service-driven Data Architecture (SD/DA) framework and Capability Maturity Model (CMM) that can assist organizations in evaluating and optimizing the benefits of their coordinated strategies to improve the overall effectiveness and success of the Service-driven and EDM strategies for the enterprise.

In Chapter 11, Lloyd Rebello provides practical guidance on gaining business flexibility through carefully managed vendor diversification options to enterprises that are implementing Service-driven applications and integration solutions. Vendor dependencies that are not carefully managed can limit an organization’s vendor diversification options that inhibit its business flexibility and put the organization at a negotiation disadvantage for contracts or system enhancements. When vendor products and solutions are tightly woven into key business processes of the enterprise, the organizations no longer have any leverage to negotiate with. The ideal solution starts with service orientation, and the application of a few key principles to maintain vendor boundaries in enterprise systems to build a highly flexible business solution platform. In order to enable flexible, vendor diversifiable enterprise integration, the organization should plan and govern the technology architecture and integration of vendor products in a thoughtful and deliberate manner.

The chapter delves into the dual reinforcing concepts of ownership and control that underpin the vendor diversification opportunities. By presenting a reference architecture that distills these principles into a conceptual model that can be applied to any enterprise, this chapter illustrates the advantages of using vendor diversifiable principles in a Service-driven environment.

In Chapter 12, Longji Tang, Wei-Tek Tsai, and Jing Dong introduce the concepts of Enterprise Mobile Service Computing (EMSC), discuss the opportunities, and address mobile constraints and challenges. Mobile computing is a distributed computing model that allows mobile devices and their applications to connect and interact with other mobile devices and applications in a wireless communication environment. The mobile computing model extends the traditional computing model which requires using stationary computing devices connected to the enterprise network through wires. Mobile devices are becoming the
major interfaces for consuming services, such as, email, internet, entertainment, and social media. EMSC is a recent development in mobile computing, and Enterprise Mobile Service Architecture (EMSA) is a new enterprise architectural approach for mobile system integration in the enterprise.

Mobile devices utilize a broad variety of system and application software. Mobile constraints include aspects relating to mobile hardware, software, networking, and mobility. Many challenges, such as, availability, performance, and security are encountered due to these constraints. Furthermore, mobile operating system constraints greatly impact mobile application software design and the capacity for consuming enterprise mobile services. The ability of a mobile device to communicate with wired information systems and services, such as enterprise applications and services, and Cloud-based services is significant for mobile computing. The communication capabilities of mobile devices directly impact mobile application design. To address these challenges in EMSC, this chapter proposes an EMSA architectural style which provides design guidelines for enterprise mobile application integration.

In Chapter 13, I have focused on exploring the capabilities of the Cloud and the challenges of extending the Service-driven enterprise to the Cloud. Enterprises are constantly interested in improving business agility, growth, and profitability, while at the same time, reducing expenses, and implementing better management of risk and compliance. Today, most enterprises own their IT infrastructures. In the future, it may well be more cost effective to use infrastructure and software provided by entities that are specialized in provisioning infrastructure and services on a need and usage basis. By leveraging economies of scale, such providers will be able to supply the required processing power and software applications and platforms at a lower cost, than could be achieved by enterprises internally. This is the Cloud Computing model.

The Cloud enables ubiquitous, elastic, and on-demand network access, which can be rapidly self-provisioned. Information Technology is beginning to migrate to the Cloud, where dynamically scalable, virtualized resources are provided as a service over the Internet. Cloud Computing is making it possible for enterprises to access and create applications on virtual servers that scale dynamically to meet demand. Service-driven architectural approaches provide a foundation for moving to the Cloud. The Cloud can change the way enterprises think about data, collect data, and manage data. Cloud services are pervasive, thereby lending themselves to connecting across businesses, people, experiences and time. This encourages enterprises to combine data with context information and increase relevancy of the data. The Cloud also shifts the burden of application integration from users to the providers, thus enabling consumers to interact with one composite application rather than having to interact with each of the individual applications or services that comprise the workflow.

There are potentially significant business benefits to building applications in the Cloud. This chapter describes the Cloud ecosystem in terms of the business benefits, service model offerings, standards, deployment models, and challenges; and provides guidelines and best practices for architecting application services that leverage the capabilities of the Cloud. With Cloud Computing, businesses can reap the benefit of agility and cost-effectiveness of technology and enable IT to enhance alignment with the business.

While the Cloud will transform business capabilities and deliver a great deal of benefit to the enterprise, moving from a centralized on-premises IT architecture to a public infrastructure can present new risks to the enterprise. There are several challenges with Cloud Computing in terms of standardization, security, governance, and federation which need to be tackled effectively to make the Cloud a serious enterprise initiative. This chapter explores some of those risks and provides guidelines for mitigating them. Cloud will become essential to enterprises because of its capability to deal with rapid change in external markets. Adopting a service-driven architectural approach will enable IT leaders to address
today’s critical challenges, while at the same time provide for a solid foundation for the enterprise to adopt Cloud for tomorrow.

Service-driven architectural approaches have enabled the delivery of agile and easily maintainable applications and integration solutions to enterprises. With current research and practice focused on dynamic, autonomous, and context-aware service compositions and Mashups, and pervasiveness through the Cloud; Service-driven applications are expected to play a major role in the implementation of robust, adaptable, fault-tolerant, and dynamic enterprise applications that are closely aligned with the ever changing needs of the business.

Raja Ramanathan  
Independent Researcher, USA