BOOK REVIEW

Migrating Legacy Applications:
Challenges in Service Oriented Architecture and Cloud Computing Environments

Reviewed by Pethuru Raj, Infrastructure Architecture, SmartCloud Enterprise+, IBM Global Cloud Center of Excellence (CoE), IBM-India, Bangalore, India

Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments
Anca Daniela Ionita, Marin Litoiu, and Grace Lewis
©2013 by IGI-Global
419 pp.
$195.00
ISBN 9781466624887

It is an unassailable truth that the dynamic and decisive discipline of information technology (IT) has been the precise and perfect business-enabler for the past five decades. Newer and nimbler technologies (information, communication, sensing, perception, integration, etc.) have been consistently emanating and evolving in order to catch up with the rapidly changing business expectations. Once upon a time, a variety of business tasks got automated and accelerated with the available technologies and tools. It was succinctly touted as the technology-driven business era. Today the prevailing scenario is altogether different. There are several business-driven technologies gaining overwhelming market and mind shares. Service oriented architecture (SOA) and Cloud computing are the leading paradigms captivating both academicians as well as industry professionals these days. Both are not competing but coexisting, complementary and collaborative in order to bring in all kinds of desired transformation, optimization, and simplification on geographically distributed and globally diversified business enterprises in the increasingly connected world.

SOA is typically for enterprise application design, integration and composition whereas clouds are being positioned as the next-generation IT infrastructures for deploying, delivering, and managing all kinds of IT services and solutions. Correctly speaking, clouds are consolidated, centralised / federated, virtualized, programmable, extensible, automated and shared IT environments in order to comprehensively establish the goal of providing "IT as a Service". In this scintillating scene, it is very much logical for business entrepreneurs and executives to strategically embark on the long and arduous journey of legacy IT modernization

and migration to SOA and Cloud environments in order to proactively reap all the envisaged benefits (business and technical) with the cool convergence of SOA and Cloud themes.

Legacy systems are business-critical and high-performing systems but are tightly coupled, inflexible, closed, hard-to-maintain, and monolithic. As IT complexity is on the climb, complexity-mitigation techniques (For example, modularity) are being given prime importance. Service-enablement, being the key modularity principle, is being prescribed as the best course of action for these massive systems, which are written and being faithfully maintained over the last four decades. SOA is being favoured because of various advantages including well-established sets of open standards, platform and language independent interfaces, clear separation of service interface and implementation, and loose-coupling among services .Off course, the prescribed movement is not a simple affair as it requires business reanalysis, code reengineering, model-driven automatic transformations, architectural changes and the economic viability.

The first chapter presents the fundamental ideas related to migrating legacy applications to service-oriented systems, and provides an overview of the available approaches that are presented in the book. The idea is to provide a "big picture" while also analysing each chapter and indicating the essential concerns, such as state-of-the- methods, standards, tools, business perspective, practical experiments, strategies, and roadmaps.

The second chapter is for extracting and exposing the a few untold challenges of service orientation. There are a few inflated expectations on SOA, which has a few unfocused areas. Based on a synthesis of two leading efforts, this chapter presents a framework of research challenges for service orientation and focuses on the topics related to the migration and evolution of service-oriented systems. The chapter reviews current progress as well as gaps in addressing those challenges that are derived from the framework.

The third chapter provides a historic overview, focusing on the methods and techniques used in legacy to SOA evolution. The authors have conducted a systematic literature review to collect legacy-to-SOA evolution approaches reported from 2000 to August 2011. To that end, 121 primary studies were found and evaluated using an evaluation framework, which was developed from three evolution and modernization methods widely used in the software reengineering domain. The evaluation constitutes the inventory of current research approaches and methods and techniques used in legacy to SOA evolution. The result of the SLR also identifies current research issues in legacy-to-SOA evolution and provides future research directions to address those research issues.

The fourth chapter talks about how code refactoring and reengineering can be accomplished. The migration costs are prohibitive. The high costs could be easily avoided if the development team were able to reuse the legacy software. Most of the elementary business functions required by the new business processes already exist inside the old legacy modules. They need to be identified, extracted and reused to offset the spiralling cost. But that is not easy as they are intertwined and hence the modules have to be reengineered before being reused. This chapter deals with how this reengineering can be done. The goal is to make modular, flexible, and independent Web services from the monolithic, rigid, and dependent legacy modules. The methods used to achieve this goal are static analysis, code restructuring, code stripping, code transformation, and code *wrapping*.

The fifth chapter focuses on the identification and specification of services based on prior modelled business processes and legacy systems. The resulting service interfaces and service components formalized by using the SOA Modelling Language (SoaML) describe the integration of legacy systems into a service-oriented application landscape. The legacy systems provide services for integration purposes and represent the implementations of service components. Additionally, the resulting architecture allows functionality of legacy

systems to be replaced with functionality provided by external cloud services. Leveraging model driven architecture (MDA) concepts, the formalized service interfaces and service components as part of the service designs can be used to automatically derive service interface descriptions using the Web Services Description Language (WSDL). These descriptions enable the technical integration of legacy systems. If needed, service implementations based on the Service Component Architecture (SCA) and the Business Process Execution Language (BPEL) can be generated.

The sixth chapter talks about the prominent approaches being used to move legacy (written using COBOL) systems to SOA systems: direct and indirect migration. Direct migration implies wrapping the current COBOL routines of a system with a software layer developed under a newer platform that can be used to offer Services. In contrast, indirect migration requires re-designing and re-implementing the COBOL routines' functionality using a newer platform as well. In this chapter, the authors have proposed a novel migration approach, which takes the best of the two previous approaches. To assess the advantages and disadvantages of these approaches, this chapter presents a case study from a government agency COBOL system that has been migrated to a service-based system using the those approaches. As a result of having these migration attempts, the authors have presented the trade-off between direct and indirect migration, the resulting service interfaces quality, and the migration costs.

The seventh chapter details about the SOA migration project (funded) which was aimed at developing an adaptable migration process model with an accompanying tool support based on MDA technologies. This process model, which combines reverse and forward engineering techniques, was applied in two different case studies on moving a monolithic software system to SOA and to a transformation-based language migration from COBOL to Java.

The eighth chapter describes the cloud-based software delivery model and how legacy software can be made ready to be hosted and delivered from online, on-demand, and off-premise clouds. Software as a Service (SaaS) is the futuristic software discovery, access, consumption, subscription, pricing, and management model. However, this transition from Software off-the-shelf to SaaS is not trivial as many issues (business, application, and technical) come into play. This chapter presents a stepwise procedure and a method to migrate non-SaaS applications to SaaS.

In the ninth chapter, the authors have focused on data migration from various data stores to cloud and vice versa. They have discussed various challenges associated with this reciprocal migration and proposed a simple yet powerful model whereby data can be migrated between various data stores, especially cloud data stores. The results clearly show an efficient way to move data from conventional relational databases to Google App Engines and how data residing in the Google App Engines can be stored on relational databases and vice versa. They provide a generalized architecture to store data in any cloud data store. The authors have used RDF/RDFS as an intermediate model in the migration process.

The tenth chapter describes the authors' experience with a contemporary rendition of the migration activity migrating a Web-based system to a SOA application on two different cloud software platforms, Hadoop and HBase. Using the case study as a running example, they have reviewed the information needed for a successful migration and examined the trade-offs between development/re-design effort and performance/scalability improvements. The two levels of re-design, towards Hadoop and HBase, have required notably different levels of effort. The authors have found that both redesigns led to substantial benefits in performance-improvement.

There are some uncertainties for web-based social and collaborative applications. For example, many Web-based applications cannot predict the number of connections that they may need to handle. As such, applications must either provision a higher number of servers in anticipation of more traffic, or be faced with a

degradation of the user experience when a large number of clients connect to the application. Cloud-based deployments can alleviate these issues by allowing the application's server base to auto scale based on the user-demand. A cloud deployment can also employ servers in different geographic locations in order to offer better latency and response times to its clients. Moving a collaborative application from using a single server to a cloud and then to a distributed cloud is not a trivial matter, however. The eleventh chapter describes the authors' experience with how such a transition can be performed along with the architectural changes that had to be implemented at the server and cloud level in order to create a distributed execution that resides in the cloud.

There are two different approaches for service-oriented systems design. SOAP and REST are the two leading architectural styles and this chapter has explained about the comparative study between them. SOAP is oriented towards behaviour whereas the REST is towards state. This chapter has proposed a new architectural style, based on a combination of the best characteristics of SOAP and REST, which the authors have designated as Structural Services. Unlike REST, resources are able to offer a variable set of operations, and unlike SOAP, services are allowed to have structure. This style uses structural interoperability, which includes structural compliance and conformance.

In this chapter, the authors have discussed various issues and challenges related to the adaptation of existing service-oriented systems to REST architecture. The authors have introduced an adaptation framework process model in the context of enterprise computing systems and technologies, such as Model Driven Engineering and SCA. Furthermore, they have discussed open challenges and considerations on how such an adaptation process to REST can be extended, in order to yield systems that best conform to the REST architectural style and the corresponding REST constraints.

In the final chapter, the authors have analysed two concepts: system evolutivity and adaptability. Both are complex processes and need tools for simplification and streamlining. Therefore the authors have proposed an execution environment, which provides a homogeneous service representation used to integrate: their functionalities, their life-cycle and management operations, and lifecycle related concerns, like deployment. Their approach includes two integration mechanisms: the technologies integration supported by wrappers and concerns integration supported by the run-times.

This book is all about the correct compilation of proven and potential approaches, tools, techniques and tips for smoothly and systematically modernizing and moving old software systems to service systems that can be easily deployed in and delivered from cloud environments. This is an informative and inspiring book authored by accomplished professors, practitioners and professionals for worldwide software developers, architects, consultants and experts who are assigned to ponder about the ways and means of legacy migration.