Chapter 1
Service-Driven Computing: Challenges and Trends

Raja Ramanathan
Independent Researcher, USA

ABSTRACT

Information technology is rapidly evolving to facilitate the design, development, and implementation of the next generation of architectural practices, tools, and techniques that will enable smart services and seamless enterprise integration. Service-Driven Computing involves the use of software services that conform to service architectural paradigms, such as Service-Oriented and Resource-Oriented Architectures, to drive computing solutions that enable building massively distributed software systems for this new generation of applications. Although services can promote agile, flexible, and extensible applications, service invocations can be subject to network latency, network failure, and distributed system failures. Moreover, service configurations are likely to change over time. This chapter explores the challenges in service-driven computing relating to 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. Along the way, contributions from researchers on potential solutions to these challenges are identified and discussed.

DOI: 10.4018/978-1-4666-6178-3.ch001

INTRODUCTION

Software architecture has evolved from simple monolithic concepts to complex, multi-tiered, distributed, and componentized abstractions. A *Service Oriented Architecture* (SOA) (Pasik, 1994) style involves the use of software services, which are loosely coupled, autonomous, distributable, composable, and platform-independent computational entities that enable applications to publish, discover, and invoke business services (services that fulfill business functionality) using standardized service contracts and protocols, SOAP compliant XML messaging, and business events. On the other hand, a *Resource Oriented Architecture* (ROA) style focuses on the resource, which is a directly-accessible software artifact supporting specific data as the key element of abstraction and emphasizing simplicity, scalability, and usability. *Representational State Transfer* (REST) (Fielding, 2000) is a set of architectural guidelines expressed as ROA that takes into account the constraints of the World Wide Web's standard HTTP messaging model to facilitate the design of RESTful services that are resource-oriented.

While Service-oriented Computing (SOC) (Georgakopoulos & Papazoglou, 2008) is closely aligned with SOA, Service-driven Computing is primarily implemented by software services that conform to any pertinent service architecture methodology such as SOA and ROA. *Service-driven Computing* (Ramanathan, 2013a) uses these software services to drive computing solutions in which new business and scientific services are created by assembling granular application services (atomic and composite services) that interact through standard messaging mechanisms. Applications and integration solutions are assembled through *Service Mediation* and *Service Composition* techniques. SOA and REST are currently the most widely used architectural styles for implementing service-driven applications and integration solutions that support the dynamics of the organization.

Based on a vendor agnostic architecture model, service-driven applications will enable an enterprise to evolve the infrastructure in alignment with the needs of the organization and provide the flexibility for vendor diversification in terms of platforms and products. Service-driven computing will help enterprises to achieve increased return on investment (ROI) and lower total cost of ownership (TCO) due to the agility and reusability inherent in the methodology. Through the use of the service-oriented middleware, service-driven computing has enabled the reduction in software development complexity and costs and has been able to simplify the provisioning, management, and monitoring of business and scientific applications and services.

The typical software lifecycle of a service-driven architectural approach comprises several phases including service modeling, service realization, composite assembly, service versioning, service deployment, monitoring, and governance. The service-driven approach enables enterprises to build massively distributed software systems by enabling the assembly of composite services dynamically in a programming language and operating system agnostic manner and through the use of standard architectural styles, artifacts, and protocols. It facilitates the building of complex business and scientific processes from existing coarse grained service components based on a building blocks model and helps to effectively align business and scientific application requirements with technology. This approach empowers the Information Technology team to deliver applications that support and meet the dynamic needs of the organization in an agile manner.

Web Services are currently the most widely used technology that utilizes the SOA and REST