Chapter 13

Achieving Dependable Composite Services through Two-Level Redundancy

Hailong Sun
Beihang University, China

Jin Zeng
China Software Testing Center, China

Huipeng Guo
Beihang University, China

Xudong Liu
Beihang University, China

Jinpeng Huai
Beihang University, China

ABSTRACT

Service composition is a widely accepted method to build service-oriented applications. However, due to the uncertainty of infrastructure environments, service performance and user requests, service composition faces a great challenge to guarantee the dependability of the corresponding composite services. In this chapter, we provide an insightful analysis of the dependability issue of composite services. And we present a solution based on two-level redundancy: component service redundancy and structural redundancy. With component service redundancy, we study how to determine the number of backup services and how to guarantee consistent dependability of a composite service. In addition, structural redundancy aims at further improving dependability at business process level through setting up backup execution paths.
INTRODUCTION

Service oriented computing is known as a new computing paradigm that utilizes services as fundamental elements for developing applications. A traditional way of software development is based on the “divide-and-conquer” manner, in which software is divided into several modules and modules are implemented separately. Instead, service oriented computing provides a novel approach to building new software applications through the reuse of existing services. A service is a self-describing entity that can be discovered and accessed through standard-based protocols. Thus software development with service oriented computing is more about integrating existing services to meet application requirements. Intuitively, software productivity can be much improved with service oriented computing technologies. The underlying technologies generally include SOA (Service Oriented Architecture) and Web services. Following the service-oriented architecture, Web service supports better interoperability, higher usability, and increased reusability compared to traditional software technologies such as RMI and CORBA. According to (Zhang, Zhang, & Cai, 2007), the lifecycle of an SOA solution consists of modeling, development, deployment, publishing, discovery, composition, collaboration, monitoring and management. Especially, service composition is widely considered as an effective method to support the development of business applications using loosely-coupled and distributed web services over the Internet. As a result, a software application built in this approach exists in the form of a composite service, which is composed of a business process structure and a set of component services. One of the challenges faced by service composition is how to make a system adaptable to rapid changing user requirements and runtime environments. One way to address this issue is late-binding, which means that only abstract services are specified in the process of modeling and development while concrete services are chosen in runtime. To be more specific, service oriented software is finally implemented and instantiated in runtime. In other words, the development and running of service oriented software can not be separated clearly like traditional software. Therefore, service composition incorporates both design time and runtime issues.

Due to the highly dynamic and uncontrolled Internet environments, composite service dependability is one of the most important challenges to deal with in the field of service composition. The dependability property of composite services is of great importance to users, which includes many critical factors, such as availability, reliability and so on. A service with higher availability promises more functional time and a more reliable service reduces the probability to fail when it is invoked. A key issue to service composition is that the dependability of a composite service may change dynamically over time. Users however, desire that a composite service delivers consistent dependability, which makes it possible for the users to obtain the expected results. This is particularly important when the application is mission-critical, e.g., a disaster management. When the dependability of a composite service degrades, users may severely suffer from the decreased performance.

The intrinsic dynamicity of composite services stems from the fact that a composite service is composed of many component services that potentially belong to different providers distributed over the open Internet. First, each component service is subject to a specific environment and various changing factors, such as varying system load and available bandwidth. Second, a more complicated situation is that multiple component services simultaneously fail to deliver the required dependability.

This chapter is devoted to discussing the dependability of composite services, which is a critical issue for service-oriented applications. Redundancy is a well-known method to obtain the needed dependability in distributed systems. According to the aforementioned analysis, we