Chapter 16

Dependability Assessment of Service-Oriented Architectures Using Fault Injection

Nik Looker
Durham University, UK

Malcolm Munro
Durham University, UK

ABSTRACT

Dependability assessment is an important aspect of any software system and shows the degree of trust and quality of service that is delivered by a system. Validation and verification techniques commonly employed to ensure that systems are fit for use attempt to remove all faults so that error conditions cannot occur but since it is not feasible to verify all states a system can achieve, it is not possible to completely test a system. Conversely, dependability assumes that failures may occur in a system and that mechanisms exist to mitigate any failures and thus provide a trustworthy system. This chapter discusses the different issues associated with dependability. The different techniques that can be used to assess dependability are discussed and are related to Service Orientated Architectures. A number of cases studies are used to show the practicality of the techniques used.

INTRODUCTION

Dependability (Avizienis, Laprie, Randell & Landwehr, 2004) is a discipline that provides an assessment of how much trust can be placed on a system to deliver a Quality of Service (QoS).

DOI: 10.4018/978-1-60960-493-6.ch016

Systems need not be fault free but should deliver their functionality when required. This is because it is virtually impossible to engineer a system that can be guaranteed to be fault free; so dependability assumes that faults exist in a system but mechanisms exist to either eliminate them or tolerate their presence. In either case a dependable system will perform its intended function. Validation and
verification techniques, on the other hand, attempt to determine that a system contains no faults. This is an important discipline and increases the overall reliability of a system but is difficult to achieve with current techniques. Dependability is a more realistic approach since it measures the reliance that can be placed upon a system rather than validating it against its specification and includes methods that increase this.

This paper provides a review of dependability and some useful fault injection based assessment techniques that can be applied to web services that implemented a Service-Oriented Architecture (SOA). We present a number of case studies, based on some of our previous work, to demonstrate how these assessment techniques can be applied.

BACKGROUND

Service Oriented Architectures

A service in economics and marketing terminology is defined by Boone and Kurtz (1988) as “…intangible tasks that satisfy both business and consumer needs”. This definition originated to describe activities in the service industry such as hotels, garages, barbers, etc. In general terms a service is not owned by the customer but is something that is utilized to complete a task. The advantage of utilizing a service to do this is that the customer does not have to design, maintain or run the service. This definition can readily be adapted to software services.

A service in software terms is an entity that communicates with other entities via messages (Cabrera, Kurt & Box, 2004). This definition does not specify that the services be networked or the method in which messages should be exchanged. Further it does not specify that the entities must perform tasks or satisfy a specified requirement but this is usually taken as implied.

Software services can be characterized as that they:

- Must communicate over a network.
- Must provide an interface that can be utilized by external systems to access functionality.
- Should be discoverable in someway so that external systems can utilize them.
- Should be loosely coupled which allows composed systems to be adaptable.

Services are often used to implement client/server architectures (Cabrera, Kurt, & Box, 2004), which are systems that are composed of a client that utilizes a service provided by a server. The client accesses the service on the server to perform some task. A service may also implement its functionality by making use of other services in the same way as a client.

At an abstract level all that is required to implement a set of services and the exchange of messages between them is a remote invocation method and an interface definition language that defines the interface. Unfortunately most distributed systems rarely run on identical hardware and are frequently required to communicate with legacy systems and other organizations hardware/software. To overcome this problem and allow service-based systems to be constructed middleware is used. Middleware (Vinoski, 2002) is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to communicate across a network. Middleware eliminates differences between machines in a heterogeneous environment by marshalling data and includes an agreed set of useful functions.

Distributed services must be organized into a system to be useful and a common architectural model used to accomplish this is a Service Oriented Architecture. A SOA is an architecture that represents software functionality as discoverable services on a network. Channabasavaiah, Holley, and Tuggle (2004) define a SOA as “an application architecture within which all functions are defined as independent services with well-defined invok-