Available online at www.sciencedirect.com
ScienceDirect
Simulation Modelling Practice and Theory 15 (2007) 492–502
SIMULATION MODELLING PRACTICE AND THEORY
www.elsevier.com/locate/simpat

Simulation of fault monitoring and detection of distributed services

E. Grishikashvili Pereira [a], R. Pereira [b,*]

[a] Department of Computing and Information Systems, Edge Hill University, St. Helen's Road, Ormskirk L39 4QP, UK
[b] School of Computing and Mathematical Sciences, Liverpool John Moores University, Byrom Street, Liverpool, L3 3AF, UK

Received 24 February 2006; received in revised form 31 July 2006; accepted 24 November 2006
Available online 22 December 2006

Abstract

Software development has evolved to incorporate reusable components. A parallel development has been the availability, through the world-wide-web, of data, transactions, and communications. These developments have led to the emergence of web-services. In this context, the self-healing attribute is of great relevance: How does an application learn that a remote service has become unavailable? As the time to replace a service may vary considerably, we consider the relationship between service replacement time, the frequency of failure monitoring and the percentage of failed service calls. Extensive simulation results are presented showing the relationship between these variables.

Keywords: Distributed systems; Middleware; Autonomic computing; Fault-recovery; Performance evaluation

1. Introduction

Distributed applications are complex to develop and manage due to the interaction of their various components, and heterogeneity of their implementation, topology, deployment and network requirements. Middleware technology facilitates the development and interoperation of distributed applications. Until recently, however, little attention was focused on developing the middleware services that will guarantee a long lifecycle for distributed services and moreover will enrich them with self-healing abilities.

In recent times, considerable work has been developed in service-oriented software [1] and autonomic computing [2] communities. One important aspect of distributed, service oriented software is component failure: As applications are composed of a collection of services, which are distributed across networks, how can we detect, in an efficient way, if a given service has failed? Systems that can automatically detect and recover from component failure are called self-healing systems. The authors have developed the on-demand service assembly and delivery (OSAD) model for self-healing systems [3], and an outline of the model is presented in this

* Corresponding author. Tel.: +44 151 231 2103; fax: +44 151 207 4594.
E-mail address: r.pereira@livjm.ac.uk (R. Pereira).

1569-190X/$ - see front matter
doi:10.1016/j.simpat.2006.11.012

paper. One of the main components of a self-healing system is the system manager, responsible, amongst other things, for detecting service failure and start the recovery process. The recovery process involves finding a suitable replacement service and binding it to the application dynamically. In previous papers, the authors presented two schemes for detection of service failure, the on-use detection and pre-emptive detection [4–6]. In this paper we consider the pre-emptive detection scheme, and present a discussion and experimental results relating the mean time to service replacement to the failure monitoring interval.

This paper is organised as follows: Section 2 provides a brief overview of self-healing systems and a discussion of related work. In Section 3 we present an outline of the on On-Demand Service Assembly and Delivery (OSAD) model. In the rest of the paper we evaluate the effectiveness of failure monitoring and detection for services over local and wide area networks: Section 4 presents the experimental set up, in Section 5 we present the results relating the following parameters: failure monitoring frequency, service replacement time, and failure rate. Finally, Section 6 contains our conclusions.

2. Self-healing systems

In order to achieve the property of self-healing, systems must be able to adapt their own behaviour in response to changes in their environment, such as resource variability, changing user needs, and system faults. Self-healing systems involves the following five major elements [7]:

1. Runtime monitoring of a given target, (The system itself or its system parts or others);
2. Exception Event detection;
3. Diagnosis, including identification of events and the right course of action;
4. Generating a plan of change such as architectural transformation during a software reconfiguration process;
5. Validation and enactment of a given change plan.

The monitoring and problem detection is one of the essential features of self-healing systems [1]. For instance, the recovery-oriented computing project takes the assumption that recovery from failure is fundamental to system availability, as failures cannot be predicted or eradicated in computer systems [12]. Various architectural models for self-healing systems, based on monitoring, problem detection and repair have been developed. The use of architectural models has been explored by Garlan et al. [8], where the architectural models are used for the runtime system's monitoring and reasoning. For instance, to understand what the running system is doing in high level terms, detect when architectural constraints are violated, and reason about repair actions at the architectural level.

Reilly et al. [9] developed an architecture and associated middleware services to support dynamic instrumentation to detect abnormal systems' states (events) and trigger and control a self-healing process thereby ensuring safety. In [10] it was agreed that it is essential for self-healing systems to have strong system monitoring abilities. The authors have presented work demonstrating the advantages of incorporating a monitoring scheme for detection of failed services in distributed service oriented applications [4,6].

3. The OSAD model

We have designed and implemented the OSAD model which identifies and implements the self-healing behaviour as the following set of activities:

1. Monitoring the application;
2. Detection of the failure/change planning;
3. Discovery of alternative component;
4. Implementing the changes/replacement of failed component with an alternative component.

The OSAD model was implemented in the JAVA language using JINI middleware. The lifecycle of self-healing behaviour in our model (see Figs. 1) (for further details see OSAD model [11] follows the above listed.