Chapter 29
Reliability Modeling and Assessment for Open Source Cloud Software: A Stochastic Approach

Yoshinobu Tamura
Yamaguchi University, Japan

Shigeru Yamada
Tottori University, Japan

ABSTRACT

Software development based on the Open Source Software (OSS) model is being increasingly accepted to stand up servers and applications. In particular, Cloud OSS is now attracting attention as the next generation of software products due to cost efficiencies and quick delivery. This chapter focuses on the software reliability modeling and assessment for Cloud computing infrastructure software, especially open source software, such as OpenStack and Eucalyptus. In this chapter, the authors introduce a new approach to the Jump diffusion process based on stochastic differential equations in order to consider the interesting aspect of the numbers of components and users in the reliability model. In addition, the authors consider the network traffic of the Cloud in the reliability modeling and integrate the reliability model with a threshold-based neural network approach that estimates network traffic. Actual software fault-count data are analyzed in order to show numerical examples of software reliability assessment. This chapter also illustrates how the proposed method of reliability analysis can assist in quality improvement in Cloud computing software.

DOI: 10.4018/978-1-4666-6178-3.ch029

INTRODUCTION

In the early days of computing, many software applications were produced in non-distributed development environments (such as Mainframes). In such environments, software applications were highly dependent on the health of the particular host system. For instance, applications were down when the host computer was down. Since the 1980s, personal computers have penetrated into our daily lives, taking us away from the dependence on conventional mainframe machines. Personal computers have gradually provided better price/performance ratios; moreover the personal computer revolution has enabled software development to occur more cost effectively. UNIX workstations and personal computers have helped to reduce the cost of development and execution of software applications.

Architecture methodologies such as client-server and component-based architectures facilitated the development of sophisticated applications using flexible communications and interfacing models. This led to the evolution of distributed computing environments that enabled scalability and efficiencies in software applications. The prevalence of service-oriented architecture enabled the development of applications as individual services that could be composed dynamically into business applications to satisfy required business functions.

However, *distributed service-driven computing* that involved the dynamic composition of distributed components, increased the complexity of the applications and introduced more points of dependencies and failures in the system. The increased system complexity made their testing and validation more difficult and had a direct impact on software reliability. Thus, it became significant to quantitatively assess the reliability of software systems in distributed environments to maintain application quality and reliability.

It has become more difficult for software developers to produce highly-reliable software systems efficiently, because of the diversified and complicated software requirements that are implemented in software products. Thus, it seems necessary to control the software development process in terms of reliability, cost, and delivery time. The typical software development process used in enterprises is the waterfall model (Yamada, 2011). This model is used in many software development projects, especially, in large-scale software development such as distributed development environments. In recent years, various other software development methodologies such as the agile development model, V-model, etc., are being increasingly used for developing distributed software. In the last phase of the development process, software testing phase is carried out to detect and fix software faults introduced by human work prior to its release for operational use. The software faults that cannot be detected and fixed remain in the released software system after the testing phase.

The proliferation of Internet access around the world in recent years has increased public awareness of Web-based online interactive applications. Software developed based on the Open Source Software (OSS) model is being increasingly accepted by enterprises to stand up servers and applications. In the OSS model, individual components are developed by different developers using a *distributed development paradigm*. Open Source projects such as GNU/Linux Operating System, Apache HTTP server, etc, attest to the successful software delivery in the distributed development model used in open source projects.

In particular, Cloud OSS is now attracting attention as the next-generation of software products due to cost efficiencies and quick delivery. Implementation of Cloud computing environments using open source software such as OpenStack, Eucalyptus, OpenNebula, OpenShift, etc., is be-