Chapter 18
A Methodology for Model-Based Reliability Estimation

Mohd Adham Isa
Universiti Teknologi Malaysia, Malaysia

Dayang Norhayati Abang Jawawi
Universiti Teknologi Malaysia, Malaysia

ABSTRACT

In recent years, reliability assessment is an essential process in system quality assessments. However, the best practice of software engineering for reliability analysis is not yet of its matured stage. The existing works are only capable to explicitly apply a small portion of reliability analysis in a standard software development process. In addition, an existing reliability assessment is based on an assumption provided by domain experts. This assumption is often exposed to errors. An effective reliability assessment should be based on reliability requirements that could be quantitatively estimated using metrics. The reliability requirements can be visualized using reliability model. However, existing reliability models are not expressive enough and do not provide consistence-modeling mechanism to allow developers to estimate reliability parameter values. Consequently, the reliability estimation using those parameters is usually oversimplified. With this situation, the inconsistency problem could happen between different estimation stages. In this chapter, a new Model-Based Reliability Estimation (MBRE) methodology is developed. The methodology consists of reliability model and reliability estimation model. The methodology provides a systematic way to estimate system reliability, emphasizing the reliability model for producing reliability parameters which will be used by the reliability estimation model. These models are built upon the timing properties, which is the primary input value for reliability assessment.

INTRODUCTION

System software has become vastly more complicated in recent years, and ranges from e-commerce to industrial embedded systems, which support both functional and non-functional requirements of the system. In fact, the satisfaction of non-functional requirements or software quality, such as perfor-

DOI: 10.4018/978-1-5225-3923-0.ch018

mance, reliability, security and adaptability, towards functional requirements are often treated as critical. Performance-sensitive systems are widely used to support various application domains, such as telecommunications, embedded systems, and industrial control systems. The unreliability of such systems is often related to underperformance of systems and could affect the whole system. Therefore, it is largely necessary to ensure promising software quality from as early as possible in the design phase to satisfy today's demanding standards.

Software qualities generally can be defined as a combination of quality attributes of a system in a certain accepted degree (IEEE 1061, 1992). According to ISO 25000 (ISO/IEC, 2005), software quality is defined as a set of qualities of a system that satisfies the given system's requirements'. In recent years, several researchers and standard organizations have proposed the software quality taxonomy in order to define exactly which software quality is justified as being important for the software systems (McCall *et al.*, 1977; Boehm, 1978; Dromey, 1995; ISO 25000, 2005).

Among software quality attributes, considerable significance has been given to reliability attributes (Kramer and Volker, 1997). Software reliability is described as the capability of the system/component/services to perform a given task under a specific condition and time frame (IEC 9126-1, 2001). The system operation is said to be reliable when continuously operating with minimal failure. Software reliability generally acknowledges the probability that a system may fail within a certain period. The failure elements can be traced from the system's error and fault information (Eusgeld *et al.*, 2008). Faults in the system will impose a certain error that will lead to a system failure if no specific approach is applied to handle this matter. A system failure might be due to faults in implementation, unreliable system resources or misplaced system tasks (Salfner and Lenk, 2010; Valis and Bartlett, 2010). Therefore, wherever the operation of the system is disturbed, the consequences may result in an enormous risk to either people or the environment. On the other hand, the risk can be greatly reduced if an adequate reliability analysis is carried out before deploying the system (Krishna and Mall, 2010).

To quantify the reliability of a system, most traditional efforts are likely to be more interested in evaluating reliability analysis during system runtime (Yang *et al.*, 2009) because of its accurate analysis. However, this approach suffers from a few problems, in particular, an increase in effort and the costs involved in rectifying the problems (if the problems are visible) (Singh, 2011; Zio, 2009). Another approach that has gained importance to the reliability analysis is a model-based approach (Balsamo *et al.*, 2004; Immonen and Niemelä, 2007). The model-based approach at design phase could assist in estimating system reliability. It could therefore aid in identifying critical parts in the system that could cause a system failure and thus, use those results to further assess the design decision process. Therefore, it is suggested that the software reliability analysis should be made at design phase rather than during the implementation phase.

In recent years, existing works have appeared, in order to facilitate reliability estimation through design models (Brosch *et al.*, 2011, 2010; Distefano *et al.*, 2011; Gokhale and Trivedi, 2006; Spyrou *et al.*, 2008; Yacoub and Ammar, 2002). Thus, by facilitating the prior design models, developers can dynamically identify problems from a reliability point of view, hence reducing the risk after deploying the system. However, to successfully utilize the capability of the design model for reliability estimation purposes, a specific methodology is needed to systematically manage the design model for estimation analysis (Krka *et al.*, 2009). The previously mentioned works are mostly dominant in an ad-hoc process and haphazard implementation. In this sense, the flow of the information among the models is always