Chapter 14

Complexity Analysis at Design Stages of Service Oriented Architectures as a Measure of Reliability Risks

Muhammad Sheikh Sadi
Curtin University of Technology, Australia

D. G. Myers
Curtin University of Technology, Australia

Cesar Ortega Sanchez
Curtin University of Technology, Australia

ABSTRACT

Tremendous growth in interest of Service oriented Architectures (SOA) triggers a substantial amount of research in its reliability assurances. To minimize the risks of these types of systems’ failure, it is a requirement to flag those components of SOA that are likely to have higher faults. Clearly, the degree of protection or prevention of faults mechanism is not same for all components. This chapter proposes the usage of metrics that are simply heuristics and are used to scan the system model and flag complex components where faults are more likely to take place. Thus the metric output is some priority or it is a measure of likelihood of faults in a component. This chapter then suggests the designers for possible changes in the design if there remains any risk(s) of degradation of desired functionalities.

DOI: 10.4018/978-1-60960-493-6.ch014

INTRODUCTION

Service Oriented Architecture (SOA), which is prompting a variable shift in the distributed computing history, is subsisted in modern computing environments and is at critical risks due to permanent and transient faults in computing structures (Lakhal, Kobayashi, & Yokota, 2006). Permanent faults such as node stuck-at-1/0, transistor open, shorted transistors, etc., arise during fabrication or result from aging, and destroy the intended function of the circuit (Timor, Mendelson, Birk,
Complexity Analysis at Design Stages of Service Oriented Architectures as a Measure of Reliability Risks

& Suri, 2008). Transient faults, in contrast, do not damage the chips physically but are catastrophic for desired functionalities of the system (Mukherjee, Emer, & Reinhardt, 2005), (F. Wang, 2008), (Iyer, Nakka, Kalbarczyk, & Mitra, 2005). Both of these faults are severe for those SOA where reliability is a great concern (Narayanan & Xie, 2006), (Tosun, 2005). For example, online banking transactions where a single bit change (1 → 0) in the most significant bit of the data storing register, may cause a huge difference in balance. Due to processor scaling, reduction in operation voltages, exponential growth of number of transistors in a single chip, increase in clock frequencies, and/or device shrinking, the rate of these faults are moving upwards day by day (Saggese, Wang, Kalbarczyk, Patel, & Iyer, 2005), (Crouzet, Collet, & Arlat, 2005).

Prior research to cope with transient faults (which in turn create soft errors) mostly focuses on post-design phases, such as circuit level solutions, logic level solutions, spatial redundancy, temporal redundancy, and/or error correction codes. Early detection and correction of such problems during the design phase is much more likely to be successful than detection once the system is operational (Cortellessa et al., 2005). Estimating reliability (or at least identifying failure-prone components) early in the life-cycle of a design is therefore preferable (Jurjens & Wagner, 2005), (A. Bondavalli, 2001). From a pure dependability viewpoint, complex components attract more attention of soft errors tolerant approaches than others do, since reliability of a system is correlated with the complexity of the system (Khoshgoftaar, 1996), (Yacoub & Ammar, 2002). To minimize the risks of system failure, it is a requirement to flag those components of SOA that are likely to have higher faults. Clearly, the degree of protection or prevention of faults mechanism is not same for all components. Hence, an approach is needed at the design stage to highlight those complex components and suggest the designers for possible changes in the design if there remains any risk of affecting desired functionalities.

This chapter flags complex components at early design stage and investigates how to encourage the designer to explore changes that could be made in the existing model. For example, how the complexities of the components could be minimized, or how these components could be replaced with alternatives and/or with less complex components are examined. The objective is to keep the functionality and other constraints of the system unaffected or to make a trade-off between them, with the goal to minimize the reliability risks. Case studies illustrate the effectiveness of the proposed approach in determining components’ complexity ranking and then lowering their complexities. The model is expressed in Unified Modeling Language (UML) since this allows the modeler to describe different views on a system, including the physical layer (Wood, Akehurst, Uzenkov, Howells, & McDonald-Maier, 2008), (L. Wang, Wong, & Xu, 2007).

EXISTING WORK ON SOFT ERRORS RISKS MINIMIZATION

Software based approaches to tolerate soft errors include redundant programs to detect (Mukherjee, Kontz, & Reinhardt, 2002), (Reinhardt & Mukherjee, 2000), (Rotenberg, 1999), (Smolens et al., 2004) and/or recover from the problem (Vijaykumar, Pomeranz, & Cheng, 2002), duplicating instructions (Oh, Shirvani, & McCluskey, 2002), (Reis, Chang, Vachharajani, Rangan, & August, 2005), task duplication (Xie, Li, Kandemir, Vijaykrishnan, & Irwin, 2004), dual use of super scalar data paths (Ray, Hoe, & Falsafi, 2001), and Error detection and Correction Codes (ECC) (Chen & Hsiao, 1984). Chip level Redundant Threading (CRT) (Mukherjee et al., 2002) used a load value queue such that redundant executions can always see an identical view of memory. Although the load value queue