A User-Friendly Dependability Evaluation Environment for System Designers

Ann T. Tai Herbert Hecht
SoHaR Incorporated
Beverly Hills, CA 90211
E-mail: {tai,herb}@sohar.com

Abstract

In order to permit users with little analytic background to evaluate dependability, modeling tools require a user-friendly front end. With this motivation, we have developed a software tool referred to as SDDS for "Software Dependability for Distributed Systems." In particular, we have designed and implemented a graphical user interface (GUI) based on a "user-language" — basic building blocks representing system components and/or subsystems. Therefore, the user can specify his/her dependability model at a high level. SDDS then translates the high-level specification into a representation that can be automatically solved by the underlying modeling engine SHARPE (from Professor Trivedi, Duke University). The translation and solution processes are transparent to the user. We describe the design of SDDS in terms of its "user language" and the translator that converts the high-level user input (model specification) to the low-level representation that SHARPE will accept.

1 Introduction

Dependability evaluation are crucial for system specification, design and maintenance. Moreover, fast turnaround time of dependability modeling process is very important for those activities. In particular, a system/software engineer should be able to quickly get ideas about how his current products (specification, design, etc.) satisfy the requirements and where modifications need to be incorporated. Dependability evaluation can be made efficient if user-friendly modeling tools are employed such that engineers can handle and control the evaluation process themselves, instead of turning over the problems to the reliability/quality-assurance personnel. With the motivation of developing a tool that can be used by system architects and designers without specialized mathematical background or experience in model construction and solution, we have been developing a dependability evaluation environment called SDDS (Software Dependability for Distributed Systems) which features a graphical user interface (GUI) that permits the user to specify and arrange the system elements at block-diagram level [1]. SDDS then translates the high-level specification into a representation that can be automatically solved by the underlying modeling engine SHARPE [2]. The translation and solution processes are transparent to the user.

Section 2 provides a brief review of the modeling engine SHARPE. Section 3 then presents the design of SDDS. In Section 4, we illustrate the graphical user interface that enables people with limited analytic background to use the tool, followed by Section 5 which describes how the building-block level user input is translated to the code that the underlying modeling engine can accept. The concluding section summarizes what we have accomplished and points to some area for our future work.

2 Modeling Engine

SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) developed by Sahner and Trivedi is a tool for analyzing hybrid, hierarchical models for a class of performance, dependability and combined performance/dependability (performability) models [3]. It provides a specification language and solution methods for the following model types: series-parallel reliability block diagrams, fault trees, reliability graphs, Markov chains (acyclic, irreducible cyclic and cyclic with absorbing states), semi-Markov chains (acyclic and irreducible cyclic), series-parallel directed (acyclic) graphs, product-form (closed) queueing networks, and generalized stochastic Petri nets. Furthermore, any hierarchical combination of above model types can be specified and solved.

3 Tool Design

We are aimed at making the powerful features of SHARPE accessible to non-modeling specialists. The prospective users are the system architects and engineers who design, develop and maintain software/hardware systems. The requirements for the interface assume that users have only minimal dependability modeling skills. The minimum background presumed is that they understand the concept of component reliability and system state (operational versus non-operational) and are familiar with block diagrams.

Accordingly, we define two types of block that facilitate the user to evaluate a design or a system as follows.