TANGRAM-II: A Performability Modeling Environment Tool *

Rosa M. L. R. Carmo¹, Luiz R. de Carvalho¹, Edmundo de Souza e Silva¹, Morganna C. Diniz¹ and Richard R. R. Muntz²

¹ Federal University of Rio de Janeiro
Computer Science Department, NCE, Coppe/Sistemas
Cx.P. 2324, Rio de Janeiro, RJ 20001-970, Brazil
² University of California at Los Angeles, Computer Science Department
Los Angeles, CA 90024, USA

Abstract. TANGRAM-II is a modeling environment tool developed for research and educational purposes that provides a flexible user interface to describe computer and communication system models. It has a sophisticated graphic interface based on the public domain software package TGIF (Tangram Graphic Interface Facility) and an object oriented description language. The tool is a second and significantly more sophisticated version than the original prototype developed during the TANGRAM project. The current version is implemented in C++ and C and has several solvers for transient and steady state analysis of performance and availability metrics.

1 Introduction

In the past 20 years, many modeling tools have been developed to aid the modeler in the design process of computer and communication systems. Many tools were tailored to specific application domains, such as queueing network models (e.g. [22, 25], see also [1] for a survey of many tools developed prior to 1988) and availability modeling (e.g. [2, 13]). Others allow the specification of a general class of models, such as the Petri net based tools, e.g. [8, 21, 7, 16], those based on formal description languages, e.g. [4, 18, 17], and those which adopted a user interface description language specially developed for the tool, e.g. [23, 5, 3]. The tools also vary in terms of the flexibility of the user interface, the measures that can be obtained, and the sophistication of the analytic and/or simulation techniques provided to the modeler.

Many issues must be addressed during the development of a modeling tool. For instance, the user interface should be tailored to the application domain with which the user is concerned. In other words, if the user is developing an availability model, then the tool should allow her to specify system components that can fail, interactions between components, repair policies, operational criteria, etc. Such an interface is provided by the SAVE package [13]. On the other hand,

* The work of R.M.L.R. Carmo, L.R. de Carvalho, E. de Souza e Silva and M.C. Diniz is partially supported by grants and fellowships from CNPq(Brazil).
it is desirable that many modeling paradigms can be supported by the tool, or the interface should allow the description of general classes of models.

Petri net interfaces are particular suited for describing general models but the price paid for this flexibility is that the specifications do not resemble the way the modeler thinks about her system. Several tools have been proposed based on this formalism as cited above or have constructs closely related to those of Petri-nets, such as UltraSAN [21] where models are specified using the so called stochastic activity networks.

There are tools based on other general settings such as the concept of balls and buckets employed by MARCA [23]. In that tool, the user describes the number of buckets in the system and a list of inter-bucket transitions that represent the movement of a ball from one bucket to another. The claim is that models from different application domains can be easily described in terms of balls and buckets. METFAC [5] is another example of a tool that allows the specification of a general class of models. METFAC uses production rules operating on the global system state variables to describe system behavior. Others have proposed the adoption of formal high level specification languages such as Estelle [4, 18] and LOTOS [17].

Aiming to provide a general user interface, and yet supporting the development of different application domains, by constructing specialized objects for each domain of interest, Berson et al developed an object oriented paradigm [3] which allows the specification of Markovian models in a symbolic high-level language. A prototype was developed in Prolog that facilitates the implementation of the user interface language, as well as the description of complex component behavior in symbolic form. Although the use of Prolog has many advantages as described in [3], portability is a problem, as well as storage requirements for large systems.

Another important issue that must be dealt with when implementing a modeling tool is the large space cardinalities of most real system models. This problem influences not only the generation phase of the state transition matrix for the solvers, but also their implementation.

The identification of special structures in the model is also a desirable feature that affects the choice of the proper solution technique. Yet another issue is related to the interaction between the interface and the solvers. Several measures require special information to be provided by the user. For instance, in availability modeling, the user must specify the conditions in which the system is considered operational. In performability modeling, reward rates must be specified for the states. If the model to be solved is non-markovian, then the interface has to provide more information than that required to solve Markovian models, as will be exemplified later.

The software package we describe deals with several of the issues mentioned above, and is based on the object oriented paradigm of [3]. The research is a continuation of the work done at UCLA and UFRJ. The tool is developed for research and educational purposes and combines a sophisticated user interface and new solution techniques for performance and availability analysis. In section