Data Analysis and Visualization within the Möbius Modeling Environment*

Tod Courtney, Shravan Gaonkar, Mark Griffith, Vinh Lam,
Michael McQuinn, Eric Rozier, and William H. Sanders
Department of Electrical and Computer Engineering and Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, U.S.A.
{tod,gaonkar,mgriffth,lam,mcquinn,ewdr,whs}@crhc.uiuc.edu
http://www.mobius.uiuc.edu

Abstract

The Möbius modeling environment is a multiple-formalism, multiple-solver framework for discrete-event system analysis. The basis of the framework is an abstract functional interface that defines the interactions between modules. New formalism and solver modules continue to be added to the tool. This paper describes recent additions to Möbius that can loosely be characterized as providing a variety of result analysis and visualization techniques.

1 Möbius Tool

Möbius is a discrete-event system modeling and analysis tool that has been widely used for the analysis of dependability and performability properties of systems. More recently, it has also been used for the analysis of system survivability and assurance [1, 2]. Möbius is based on a framework that supports multiple modeling formalisms as well as multiple solution techniques. One of the motivations for the Möbius framework was the need for a common modeling environment that could support the development and comparison of a wide variety of new modeling technologies and provide an ability to compare the techniques with established technologies.

The Möbius framework is based on an abstract functional interface (AFI) [3]. The AFI is implemented by a set of C++ base classes that represent common components of discrete-event models: state variables, actions, and models. *State variables* hold the state of the model, *actions* generate events that change the state of the model, and *models* represent a component in a system and contain sets of actions and state variables. New modules, derived from the AFI base classes, can be easily incorporated into the framework to expand the set of available functionalities.

The Möbius tool architecture is designed to be portable across multiple platforms. The front end of the architecture consists of a Java-based graphical user interface. The back end is written in C++ to attain efficient execution of the models and solution techniques. Möbius has been available for Windows and Linux systems for some time, and support for MacOS X has recently been added. Further details on the Möbius framework and tool can be obtained from www.mobius.uiuc.edu.

2 New Capabilities

The Möbius framework supports multiple solution techniques, such as discrete event simulation and numerical solvers based on a variety of transient and steady-state algorithms. Each solver has the capability to solve models and generate results for the defined reward measures. In past versions of Möbius, the analysis modules typically exported their results via one of a variety of custom text file formats. This approach was sufficient during the research and development phase of each of the solvers, and could support the basic analysis of models with relatively few reward measures. However, it was clear that better support for results sharing, analysis, and visualization was needed when working with large and complex modules of real-world systems with many reward measures, possibly solved at multiple points of time.

2.1 Results Database

The first step to improve the handling of results within Möbius was the development of a uniform method of storing and retrieving results. A database has been developed to store the results produced with the analysis engines within Möbius, along with relevant metadata from the model. The database schema is flexible and straightforward, making it possible to store results from multiple types of analysis techniques within the same table structure. The database metadata provides support for queries of solution results based on many aspects of the model, including submodel names, model versions, date, analysis technique, and model parameter values. The results database provides the common interface for results sharing between the modules in Möbius. External tools can also interface with the database to gain access to data produced by Möbius to perform more extensive analysis.

2.2 Results Visualization and Exportation

As a companion to the results database, a results visualization and exportation module has been incorporated within Möbius to allow users to automatically generate

graphs of the results generated by the analysis techniques. A graphical user interface is used to define data queries and allow users to navigate through the data within the database and select the data of interest for the plots. Plot query and configuration details can be archived so that the plots can be easily regenerated as updated solution results are generated. Plots can be exported as image files, and data contained within the plot can be exported as a standardized text file for easy importation with external tools. The module integrates the charting capabilities of an open source package called JFreeChart, providing a powerful and flexible plotting package that supports a wide range of chart styles.

2.3 Simulation Trace Visualization

Mobius now supports a framework for multiple simulation trace files. The original trace file has been improved with its output sanitized by removing unnecessary data. The trace file supports four levels of granularity to assist the modeler in debugging a model. In addition, Möbius supports the XML-based format proposed by Kemper et. al. [4]. Their tool, Message Sequence Chart Visualizer, reads the XML tracefile output and produces a rich visual display showing the dynamic behavior of complex models. It displays each submodel as a process, draws connections between processes when an action fires, and displays the updated state of the model. This visualizer is beneficial for debugging a model and understanding the complex interactions between models.

2.4 Design of Experiments

Models of complex systems often contain model parameters for important rates, probabilities, and initial state values. By varying the parameter values, the system modeler can study the behavior of the system under a wide range of system and environmental assumptions. However, exhaustive exploration of the parameter space of a large model is computationally expensive. Design of Experiments techniques provide information about the degree of sensitivity each output variable has for the various input parameters in the model. Design of Experiments makes it possible to find parameter values that optimize measured outputs of the system by running fewer experiments than required by less rigorous techniques.

In Möbius, it is the role of the study to define the parameter values for the various experiments to be solved, making it the natural place to incorporate Design of Experiments techniques. Using the Design of Experiments (DoE) study, the modeler identifies the subset of model parameters to vary, and the subset that are constant. Then, one of several techniques is selected to define a set of experiments. The experiments are solved using any of the available solution modules within Möbius, and the experiment results are stored in the results database. The DoE study then guides the user through the process of building a regression model of the response and using the regression model to predict response values at points in the parameter space other than the experimental points. The regression model is then analyzed to make sure it is a good fit based on what is known about the system. If the model is good, it can be used to produce graphs of the response surface and also predict new parameter values for subsequent experiments.

2.5 Connection Solver

The connection solver allows a modeler to define a multi-step solution process for the analysis of his or her system. Each solution step solves a model that represents a portion of the system of interest using an existing analysis technique from within Möbius. Results from each solution step are passed to subsequent solution steps via the results database.

A graphical interface is provided to allow the modeler to construct graphs that represent the connection solution process by specifying nodes with three distinct functions: solution technique, connection function, and database query. Conduits connect nodes in the graph and represent data passing between the nodes. Outputs from one solution step pass through connection function nodes to become inputs to subsequent solution steps. Connection function nodes allow results from multiple solution techniques to be combined using arbitrary mathematical and logical expressions to produce the input into a subsequent solution step. Database nodes allow results data to be queried directly from the database, making it possible to archive known stable results without the need to regenerate them within subsequent connection solutions.

3 Acknowledgments

We would like to acknowledge the initial development of many of the modules presented in this paper: the results database by A. Christensen, the connection solver by D. Daly, and the design of experiments study by P. Webster. We would also like to acknowledge the contributions to the Möbius tool by the former members of the Möbius group as well as others from groups throughout the world. Finally, we would like to thank Jenny Applequist for her editorial assistance.

References

[1] F. Stevens, T. Courtney, S. Singh, A. Agbaria, J. F. Meyer, W. H. Sanders, and P. Pal, "Model-based validation of an intrusion-tolerant information system," in *Proc. SRDS 2004*, Oct. 2004, pp. 184–194.

[2] S. Singh, A. Agbaria, F. Stevens, T. Courtney, J. F. Meyer, W. H. Sanders, and P. Pal, "Validation of a survivable publish-subscribe system," *Int. Sci. Journal of Computing*, to appear.

[3] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster, "The Möbius framework and its implementation," *IEEE Trans. on Soft. Eng.*, vol. 28, no. 10, pp. 956–969, Oct. 2002.

[4] P. Kemper and C. Tepper, "Visualizing the dynamic behaviour of proc/b models," in *Proc. of the Conference on Simulation and Visualization (SimVis)*, March 2005, pp. 63–74.