

DESUMA: A Tool Integrating GIDDES and UMDES

L. Ricker

Department of Mathematics & Computer Science
Mount Allison University,
Sackville, NB, Canada
Email: lricker@mta.ca

S. Lafortune, S. Genc

Department of Electrical Engineering
and Computer Science,
University of Michigan
Ann Arbor, MI 48109-2122, USA
Email: stephane,sgenc@eecs.umich.edu

Abstract—The key features of the software tool DESUMA for the study of discrete event systems modeled by finite-state automata are highlighted. DESUMA is the tool resulting from the integration of the UMDES library of routines for the study of discrete event systems (developed at the University of Michigan) within the graphical environment for visualizing discrete event systems (developed at Mount Allison University).

I. UMDES

The UMDES library of routines for the study of discrete event systems (DES) modeled by finite-state automata (FSA) has been under development at the University of Michigan since the mid 1990's under the leadership of the second author. It is publicly available for download [1]. The routines currently available in UMDES implement many of the operations of the theories of diagnosis and control developed for discrete event systems by several research groups over the last 20 years. The control operations implement the main algorithmic techniques of the theory of *supervisory control* of DES initiated by Ramadge & Wonham [2] for systems modeled by FSA and centered around the properties of controllability, observability, nonblockingness, normality, and coobservability. The diagnosis operations implement the main techniques of the so-called *Diagnoser Approach*, initiated at the University of Michigan in [3], and include routines for constructing and analyzing diagnoser automata (logical and stochastic). UMDES also includes basic functions pertaining to the manipulation and composition of FSA. The routines in UMDES are written in C and have a command-line interface. Inputs to these routines are either command-lines or text files formatted specifically for the performed operation. Similarly, the outputs from the routines.

II. GIDDES

A software tool for visualizing DES operations, called a Graphical Interface for the Design of Discrete Event Systems (GIDDES), has been under development at Mount Allison University since 2001 under the leadership of the first author. The project began as a means for undergraduate computer science students to acquire good object-oriented software design techniques. Java was selected as the programming language because of the platform-independent development possibilities. There were two parts to the project: the design of the “front end” (the graphical user interface) and the

design of data and file structures for the “back end” (the FSA and DES libraries). Implementation of the back end was relatively straightforward; students simply used existing algorithms from the literature. The front end presented more difficult challenges. It was clear from the outset that writing code to perform the layout of the automata was outside the scope of the project. A Java-based open source package Grappa [4] (a subset of GraphViz ported to Java) was incorporated into the project architecture. While reuse of code is often a desirable design feature, the layout code included some executable files that had only been compiled for four different platforms. This restriction imposed several minor design adjustments. The key design feature of GIDDES is the separation of the front end and the back end. This means that the graphical user interface can easily be integrated with a completely different back end (not necessarily written in Java) and allows the user to visualize the output from other DES libraries.

III. DESUMA

Over the last two years, the authors have been working on integrating UMDES with DESUMA in order to take advantage of the features of GIDDES for visualizing FSA and mitigate the limitations of the command-line interface of UMDES. The result is the new tool DESUMA, an acronym that integrates Discrete Event Systems, University of Michigan, and Mount Allison. DESUMA is publicly available for download [1]. Most of the routines of UMDES are available from the pull-down menus of DESUMA in a graphical environment identical to that of GIDDES. See Fig. 1 for a screen shot of DESUMA.

The software architecture of DESUMA follows the Model-View-Controller (MVC) architectural pattern, where the program is divided into three collaborating classes. The *model* deals with the underlying data (e.g., FSA). The *view* shows the relevant information in the model to the user (e.g., toolbars). Finally, the *controller* gathers information from the user and uses it to modify the model (e.g., marking states). This separation of duties makes it straightforward to interchange views while still using the same model and controller classes.

The integration of GIDDES and UMDES required some additional Java code to (a) make the appropriate system calls to the C executable library functions; (b) ensure the expected

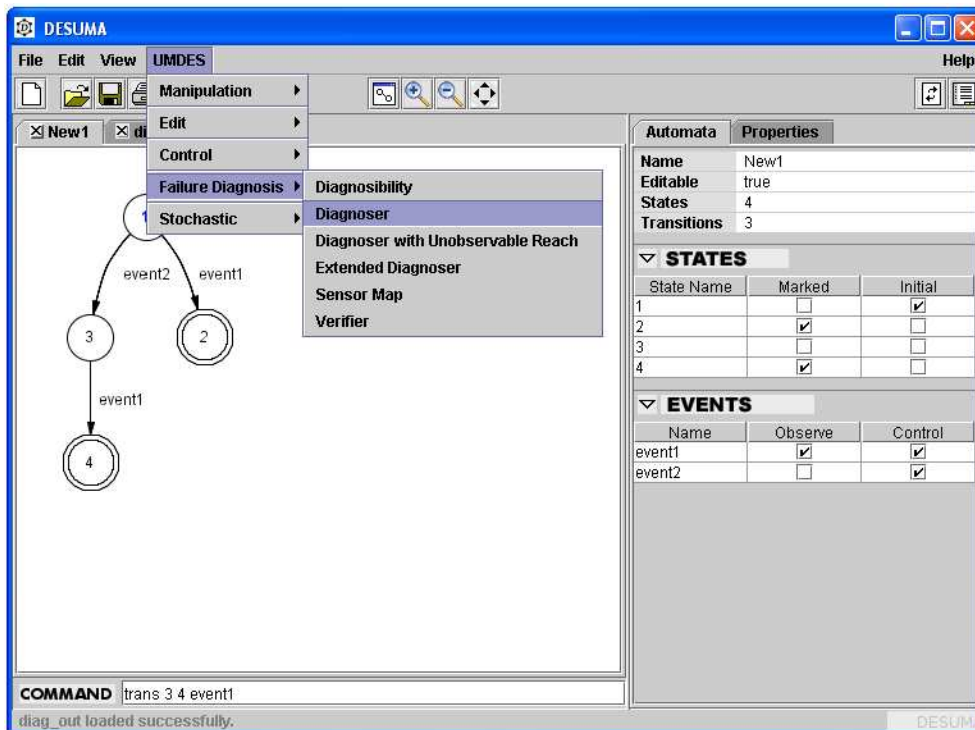


Fig. 1. Screen shot of DESUMA.

UMDES input and output files are passed correctly through the Java-based menu system of GIDDES; and (c) add the necessary UMDES commands to the GIDDES menu.

In a DESUMA session, users can open existing UMDES files (fsm format) or can create a new automaton using commands available in the graphical user interface. Some UMDES library functions require additional information, and in those cases, the user is prompted to supply the necessary details to a special pop-up window. At the end of a DESUMA session, automata are saved as UMDES files. As well, it is possible to export the graphical display to one of the following file formats: PNG, JPEG, GIF, PS, PS2 and XFIG. At the moment, we have restricted ourselves to exporting file types that are supported by Grappa and GraphViz.

The graphical environment of DESUMA is best-suited to visualize FSA with small state spaces. The ability to handle problems that result in very large state spaces is constrained by the available memory on the host computer. There is a significant memory overhead with DESUMA, and as a result, problems with large state spaces are often better solved within UMDES using the command line interface. However, for FSA that can be drawn efficiently, respecting the memory of the host computer, the graphical environment of DESUMA is very helpful and makes DESUMA a useful educational tool.

IV. FUTURE VERSIONS

The next generation of DESUMA features a new plugin architecture. Plugin architectures allow for applications that are modular, customizable, and easily extensible. The front

end of GIDDES acts as the host framework. One distinct advantage of plugins is that it is possible to create custom versions of an application without source code modifications. Previously, DESUMA required the intermediate step of writing Java code for UMDES in the GIDDES source code. A UMDES plugin will alleviate this difficulty and allow for the development of additional UMDES features without concern for conflicts with the source code of the host framework.

Users are encouraged to try DESUMA and send their comments to the authors.

ACKNOWLEDGMENTS

It is a pleasure to acknowledge the contributions of William Belanger, William Fowler and Vera Ranieri (Mount Allison) and Ameet Soni and Robert Szymanski (University of Michigan) in the development of DESUMA. Financial support from the University of Michigan, Mount Allison University, and NSERC is gratefully acknowledged.

REFERENCES

- [1] Lafortune, S.: UMDES-LIB software library. [Online]. Available: <http://www.eecs.umich.edu/umdes/toolboxes.html>
- [2] Ramadge, P.J. and Wonham, W.M.: The control of discrete-event systems. *Proc. IEEE* 77(1) (1989) 81–98
- [3] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K. and Teneketzis, D: Failure Diagnosis Using Discrete-Event Models. *IEEE Trans. on Control System Technology* 4(2) (1996) 105–124
- [4] Mocenigo, J.: Grappa, a Java Graph Package. [Online]. Available: <http://www.research.att.com/~john/Grappa/>