Tool Support for Parametric Analysis of Large Software Simulation Systems

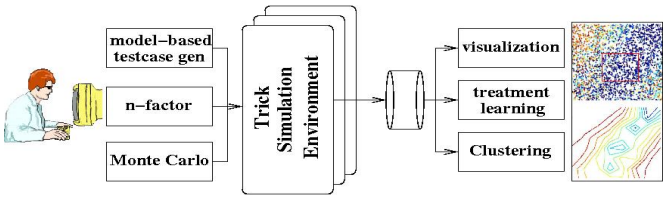
Johann Schumann*, Karen Gundy-Burlet†, Corina Păsăreanu‡, Tim Menzies§ and Tony Barrett¶
*RIACS/USRA, NASA Ames, Moffett Field, CA 94035, Email: Johann.M.Schumann@nasa.gov
†NASA Ames, Moffett Field, CA 94035, Email: Karen.Gundy-Burlet@nasa.gov
‡QSS Inc, NASA Ames, Moffett Field, CA 94035, Email: Corina.S.Pasareanu@nasa.gov
§Lane CS & EE, West Virginia University, Email: tim@menzies.us
¶Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, Email: barrett@jpl.nasa.gov

***Abstract*—The analysis of large and complex parameterized software systems, e.g., systems simulation in aerospace, is very complicated and time-consuming due to the large parameter space, and the complex, highly coupled nonlinear nature of the different system components. Thus, such systems are generally validated only in regions local to anticipated operating points rather than through characterization of the entire feasible operational envelope of the system. We have addressed the factors deterring such an analysis with a tool to support envelope assessment: we utilize a combination of advanced Monte Carlo generation with n-factor combinatorial parameter variations to limit the number of cases, but still explore important interactions in the parameter space in a systematic fashion. Additional test-cases, automatically generated from models (e.g., UML, Simulink, Stateflow) improve the coverage.**

The distributed test runs of the software system produce vast amounts of data, making manual analysis impossible. Our tool automatically analyzes the generated data through a combination of unsupervised Bayesian clustering techniques (AutoBayes) and supervised learning of critical parameter ranges using the treatment learner TAR3**. The tool has been developed around the Trick simulation environment, which is widely used within NASA. We will present this tool with a GN&C (Guidance, Navigation and Control) simulation of a small satellite system.**

I. Tool Architecture

The tool is designed to integrate with the Trick simulation environment [1] but also integrates with simulations built up in other environments (e.g. Simulink) . The user sets up the specifications of the desired parameter variations and their statistical properties and provides high level models (e.g., UML statecharts). Once the test-suite is generated, simulations are being executed by Trick, and the resulting data saved. Data analysis and visualization is controlled via a Matlab graphical user interface.



A. The Trick Simulation Environment

The Trick simulation development toolkit [1] has been developed for building dynamic, high fidelity robotics and space vehicle simulations at NASA. Trick uses an object oriented approach and automatic code generation to facilitate the construction of large and complex simulation systems, which can contain physical models (e.g., atmosphere or capsule aerodynamics), models of the hardware, and software systems. In order to achieve high performance, Trick can run on networked clusters of workstations or supercomputers.

B. Generating the Simulation and Test Cases

Large and complex simulation models have a multitude (often more than a hundred) of tunable parameters that need to be explored. A simple systematic exploration of all parameter values results in an exponentially large number of possible configurations. Worse yet, when dealing with simulations of physical systems, many parameters are real valued, thus making the configuration space unlimited. Even with appropriate discretization, guaranteeing coverage of the option space is a non-trivial problem. Our tool covers the parameter space using a combination of Monte Carlo generation, a n-factor combinatorial technique, and model-based automatic testcase generation. The Monte Carlo approach generates parameters for a simulation run by randomly selecting from user defined probability distributions, such as Gaussian or Uniform. The main drawback is a lack of any coverage guarantee, resulting in a need to run a large number of simulation runs to attain a given level of user confidence.

The n-factor combinatorial method makes a reasonable coverage guarantee while attempting to perform a minimal number of simulations runs [2], [3]. The underlying premise behind the combinatorial approach can be captured in the following three statements: (1) the simplest bugs are triggered by a single input parameter, (2) the next simplest bugs are triggered by an interaction of two input parameters, and (3) progressively more obscure (and rare) bugs involve interactions between more parameters. So errors can be grouped into families depending on how many parameters need specific settings to exercise the error. The n-factor combinatorial approach guarantees that all errors involving the specific setting of n or fewer parameters will be exercised by at least one test. Even for a large number of parameters v and parameter values d (continuous parameters are discretized), reasonable test cases can be generated very efficiently within a few seconds.

Problem d^v	#Tests (trivial)	#Tests (2-factor)
3^4	10^2	9
3^{13}	10^6	19
$4^1 \times 3^{39} \times 2^{35}$	10^{29}	29
10^{20}	10^{20}	216
3^{1000}	10^{477}	48

Pure combinatorial or statistical testing yields poor coverage in software components, which contain mode logic or state machines. Here, our tool uses a combination of model checking and symbolic execution techniques integrated into Java Pathfinder [4] to automatically generate specific test suites that guarantee a given code coverage (branch and path coverage). This tool can generate test cases from UML statecharts and Simulink/Stateflow models.

This combination of generation procedures allows the designer/engineer to easily and effectively carry out parametric evaluations of large simulation systems. The tool aims at a maximal reduction in the number of test cases while still retaining sufficient code coverage, in particular in highly structured code segments like mode logic or state machines.

C. Analysis and Visualization of Simulation Data

Each simulation run generates large sets of data (e.g., position, velocity, forces) for each point in time. Analysis of such high-dimensional data (often $d > 100$ and more than 10,000 runs) cannot be performed manually. Our tool provides support (a) to find structure in the data, and (b) to see, which parameters (or parameter ranges) caused a specific system behavior (root-cause analysis). We use AUTOBAYES [5] to automatically generate customized clustering algorithms from statistical specifications. Clustering is a well-known unsupervised learning method to find structure in large sets of data. Depending on the statistical properties of the simulation data (not all variables are Gaussian distributed), AUTOBAYES generates customized C/C++ code for the Matlab environment (mex-function), which estimates class parameters and class membership.

Root-cause analysis can be effectively done by the treatment learner TAR3, a weighted-class minimal contrast-set association rule learner. TAR3 [6] first splits the data into a "good"/"bad" category according to user-given criteria (e.g., distance from desired landing spot). Then, the treatment learner returns a set of treatments, which are conjunctions of linear constraints on the input variables. For inputs adhering to the constraints of the treatment, their effect with respect to the given criteria is calculated.

In general, the visualization of a high-dimensional data set is very difficult. Our tool uses the results of the clustering and treatment learning to guide the user toward "interesting" and "promising" ranges and envelope boundaries in the parameter space. Using the capabilities of Matlab, our tool provides a set of predefined, visualizing methods including scatter plots, likelihood plots, and histograms. The figure shows the expected variation in trajectory relative to these input parameters (tank pressure, center of gravity, moments of inertia, etc). This example is based on a computational model of the earth-based small-satellite simulator Hover Test Vehicle (HTV). The clusters in the figure were ranked according to their landing velocity and position, with colors ranging from Blue (best outcomes) to Red (worst outcomes).

II. CONCLUSIONS

This tool has been successfully used at NASA for the analysis of the GN&C system for reentry scenarios for the Orion capsule [7]. Although the tool has been developed around the NASA trick simulation environment, it can be easily adapted for other simulation or software execution environments.

ACKNOWLEDGMENTS

This research was sponsored in part by NASA under the ISRDS Contract NNA07BB97C (RIACS). Part of this work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA.

REFERENCES

[1] K. Vetter, "The trick user's guide, trick 2007.5 release," July 2007.

[2] D. Cohen, S. Dalal, J. Parelius, and G. Patton, "The combinatorial design approach to automatic test generation," *Software, IEEE*, vol. 13, no. 5, pp. 83–88, Sep 1996.

[3] I. S. Dunietz, W. K. Ehrlich, B. D. Szablak, C. L. Mallows, and A. Iannino, "Applying design of experiments to software testing: experience report," in *ICSE '97: Proceedings of the 19th international conference on Software engineering*, 1997, pp. 205–215.

[4] C. S. Păsăreanu, P. C. Mehlitz, D. H. Bushnell, K. Gundy-Burlet, M. Lowry, S. Person, and M. Pape, "Combining unit-level symbolic execution and system-level concrete execution for testing nasa software," in *Proc. of ISSTA*, 2008.

[5] B. Fischer and J. Schumann, "AutoBayes: A system for generating data analysis programs from statistical models," *J. Functional Programming*, vol. 13, no. 3, pp. 483–508, May 2003.

[6] Y. Hu, "Treatment learning," 2002, masters thesis, Unviersity of British Columbia, Department of Electrical and Computer Engineering. In preperation.

[7] K. Gundy-Burlet, J. Schumann, T. Menzies, and T. Barrett, "Parametric analysis of antares re-entry guidance algorithms using advanced test generation and data analysis," in *Proc. iSAIRAS 2008 (9th International Symposium on Artifical Intelligence, Robotics and Automation in Space)*, 2008.