Combinatorial Test Designs

Authored by: Brett Bush, PhD

Francisco Ortiz, PhD

25 March 2014

Revised 22 August 2018

The goal of the STAT COE is to assist in developing rigorous, defensible test strategies to more effectively quantify and characterize system performance and provide information that reduces risk. This and other COE products are available at www.AFIT.edu/STAT.

STAT Center of Excellence
2950 Hobson Way – Wright-Patterson AFB, OH 45433

Table of Contents

Revision 1, 22 Aug 2018: Formatting and minor typographical/grammatical edits.

Executive Summary

The Office of the Secretary of Defense (OSD), in the form of the Director, Operational Test and Evaluation (DOT&E) and Deputy Assistant Secretary of Defense, Developmental Test and Evaluation (DASD/DT&E), has been pushing for a more rigorous approach to test and evaluation (T&E) since 2009. Efficient and robust test design is needed more than ever in today's constrained fiscal environment. The right test design will determine how much testing is enough and also quantify how much risk is incurred in passing or failing an acquisition program in its T&E strategy. A well-planned design also identifies key resources and constraints for use in the T&E program. The classical approach has been to apply design of experiments (DOE) techniques to Department of Defense (DoD) acquisition programs in T&E. However, for deterministic systems, DOE may not be applicable because of the system's inability to produce any variation in the output with the same set of inputs. In such cases, we examine the use of combinatorial designs.

Keywords: STAT, rigor, scientific test, test and evaluation, deterministic, software

Introduction

Scientific Test and Analysis Techniques (STAT) is the application of the scientific method using mathematical and statistical techniques for planning, designing, executing, and analyzing experiments that address test objectives. While STAT has traditionally been applied to test physical systems, they can also be extended to test software-intensive systems. Performance measures from software tests tend to be deterministic, exhibiting the same outputs for specific settings of input factors. Due to this, many preferable design characteristics found in statistical designed experiments are no longer relevant and important. Additionally, the typical goal of DOE is to characterize or optimize the performance of a system. However, the goal in testing software is to find all the faults. Often, the key design characteristic for software tests is to obtain a high level of coverage of the input domain with as small a number of test runs as possible. Combinatorial designs, also known as factor covering arrays, are often employed to achieve this design characteristic. In the following sections, we provide an introduction to combinatorial designs and present several software tools readily available to create these designs.

Combinatorial Designs

In software testing, any particular setting of input factors could trigger a fault or error. Therefore, every possible combination of input factors would have to be investigated in order to ensure no faults or bugs exist. Consider a system with 10 input parameters, each with 5 different settings. Approximately 10 million (5^{10}) tests would have to be run in order to cover all possible combinations. This is obviously time consuming and impractical, not to mention costly. It has been estimated that more than 50% of

development time will be devoted to testing the software (Kuhn, Kacker, & Lei, 2010). Luckily, a study (Kuhn et al., 2009) at the National Institute for Standards and Technology (NIST) found that most faults are caused by the interactions between just a few parameters. Figure 1 shows the cumulative percentage of faults found versus the level of interactions between parameters for various types of systems. Over 80% of faults were found just looking at 3-way interactions, over 90% with 4-way interactions and virtually all with 6-way interactions.

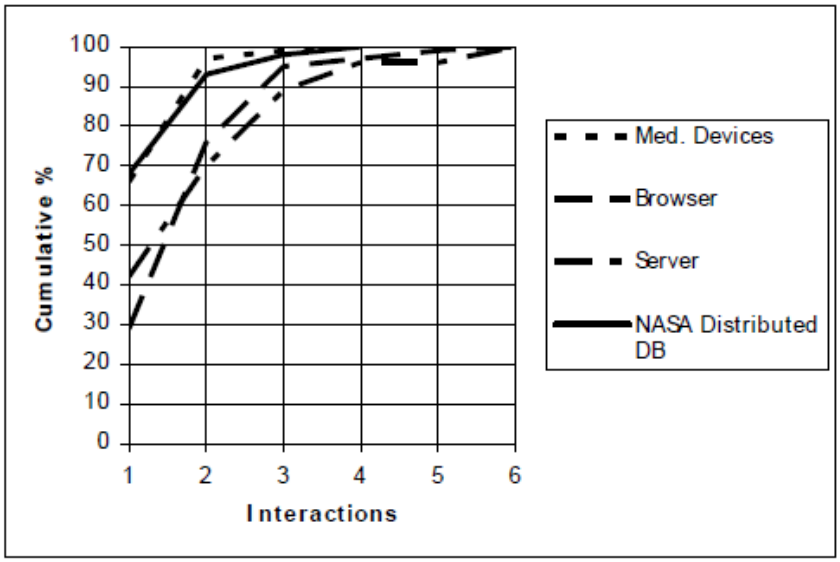

Figure 1: Cumulative percentage of faults found versus level of interaction (Kuhn et al., 2009)

Combinatorial designs attempt to maximize test coverage (i.e., the proportion of parameter combinations that appear in the test) with the minimum number of test cases. This is done by combining parameter values to cover all the t-way combinations. A covering array is the vehicle to perform this task. A covering array is a mathematical object that covers all t-way combinations of parameter values at least once (Kuhn, et al., 2009). These designs are very easy to use and very effective at finding faults/bugs. For the ten 5-value parameters mentioned earlier, all 2-way interactions could be tested in 49 runs; 3-way interactions in 307 runs; and 4-way interactions in 1,865 runs.

Software

The STAT COE does not endorse any particular software package, but instead encourages potential users to download and use different options and then evaluate their utility. Below, we list the tools provided by NIST as well as information and the location of other combinatorial design tools.

- Advanced Combinatorial Testing System (ACTS). This software is freely available through the NIST website (http://csrc.nist.gov/groups/SNS/acts/index.html). It can compute tests for 2-way through 6-way interactions.
- Combinatorial Coverage Measurement Tool. This software is also freely available through the NIST website (http://csrc.nist.gov/groups/SNS/acts/index.html). It can analyze existing tests for 2-way through 6-way interactions.
- www.pairwise.org. There are currently 44 different combinatorial design tools (to include ACTS and the combinatorial coverage measurement tool) on the www.pairwise.org website. We leave it as an exercise to the reader to click through the various tools and decide which may be of interest.

Example

Suppose we want to test Microsoft Word's ability to modify text with ten different options (Figure 2). Suppose that a 0 means the text feature is turned off while a 1 implies the text feature is turned on. These options are all binary inputs, so in order to test every possible combination, we would need 2^{10} (1,024) trials. From the research performed by Kuhn et al. (2009), we know that if we develop a covering array with strength-3 (meaning we look at all 3-way interactions), we will likely identify over 90% of the faults. For binary variables, a strength-3 covering array should contain all 8 (2^3) parameter value combinations for three selected parameters.

We can use one of the aforementioned software tools to develop a strength-3 covering array. Figure 3 shows the resulting 13-run test. You will notice in Figure 3 that for any three columns selected of the ten possible columns, all eight factor combinations are present (shown in the red, yellow, and green circles in the figure). So we have generated a covering array that includes all 3-way interactions. Observe that the use of this covering array allows us to test all 3-way combinations with only 13 trials. Compared to the original 1,024 trials, we have achieved over a 99% savings in trials.

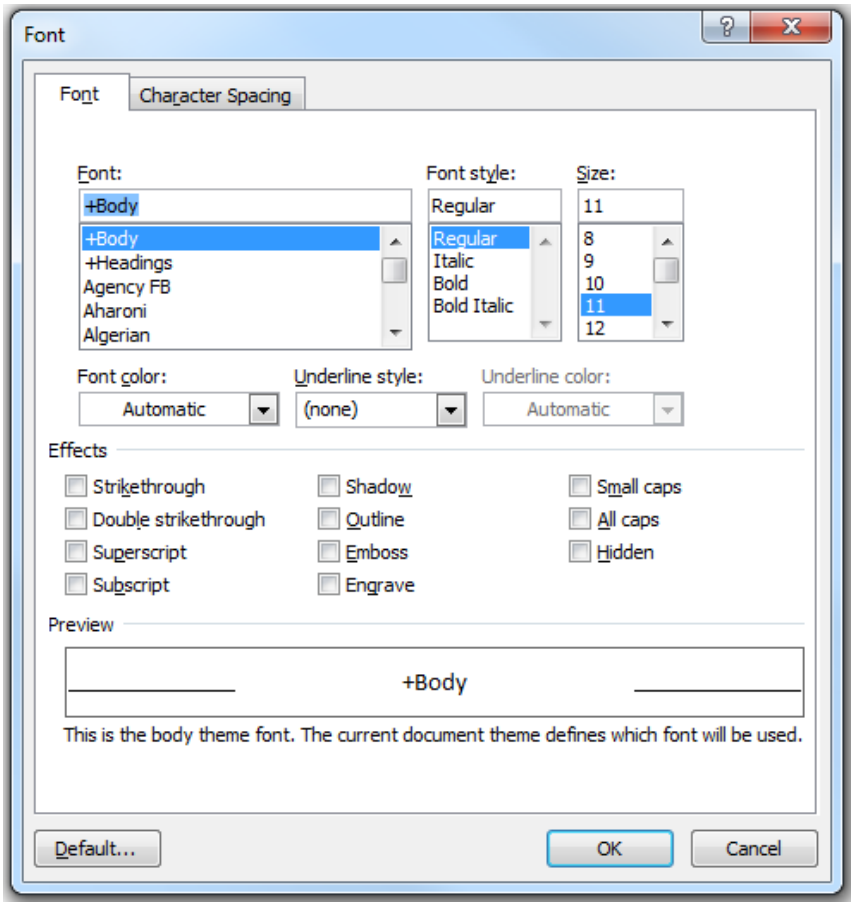

Figure 2: Example application for combinatorial test design, Microsoft Word font choices

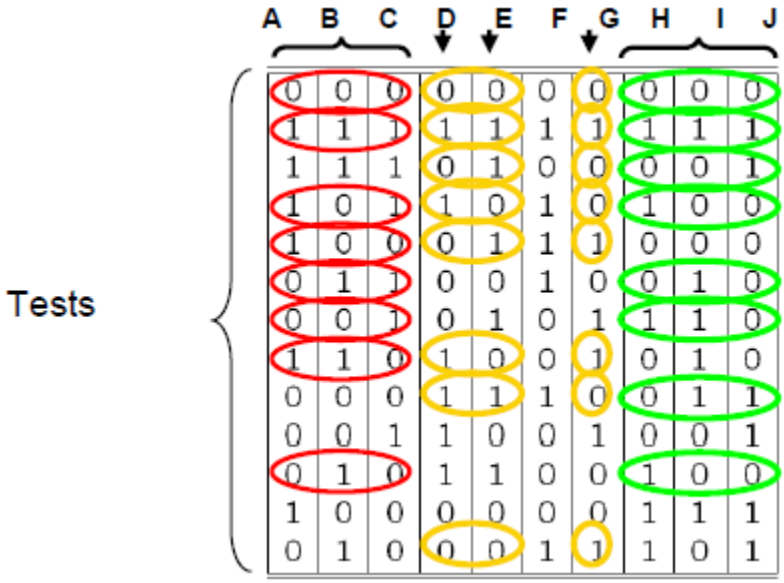

Figure 3: 13-run Strength-3 covering array contains all three-way interactions

ACTS Tutorial

We now step through usage of the ACTS software. ACTS has a Graphical User Interface (GUI) as well as a command line. It supports test generation for 2- to 6-way interactions. It handles multiple algorithms as well as input and relationship constraints. It also offers the ability to build a test set from scratch or add to an existing test set.

Let's examine the general layout of ACTS. The system view is a general tree structure that shows the components of each system (Figure 4). In the tree structure, each system is shown as a three level hierarchy. If a system has constraints and relationships, those will be shown at the same level as the parameter. On the right side of the system view, you will see two tabs: Test Result and Statistics. The Test Result tab shows a test set of the currently selected system, where each row represents a test run, and each column represents a parameter. Output parameters are also displayed as columns. The Statistics tab displays some statistical information about the test set. Notably, it includes a graph that plots the growth rate of the test coverage with respect to the tests (Figure 5).

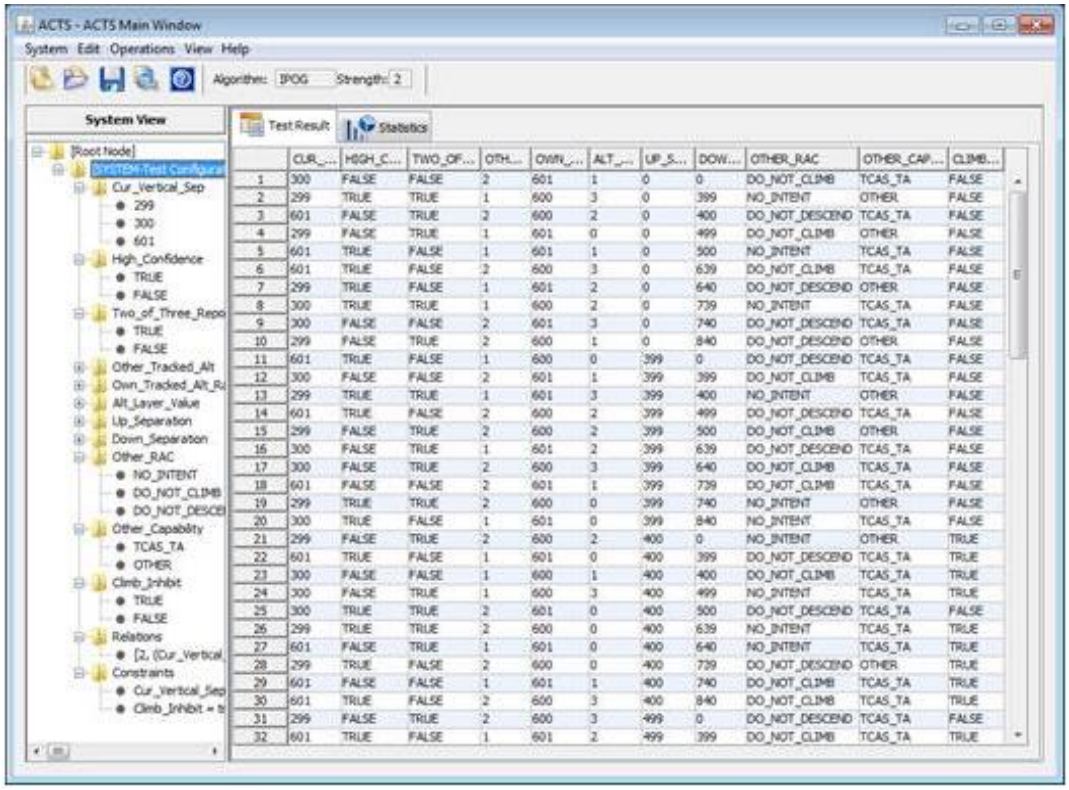

Figure 4: Advanced Combinatorial Testing System software main window

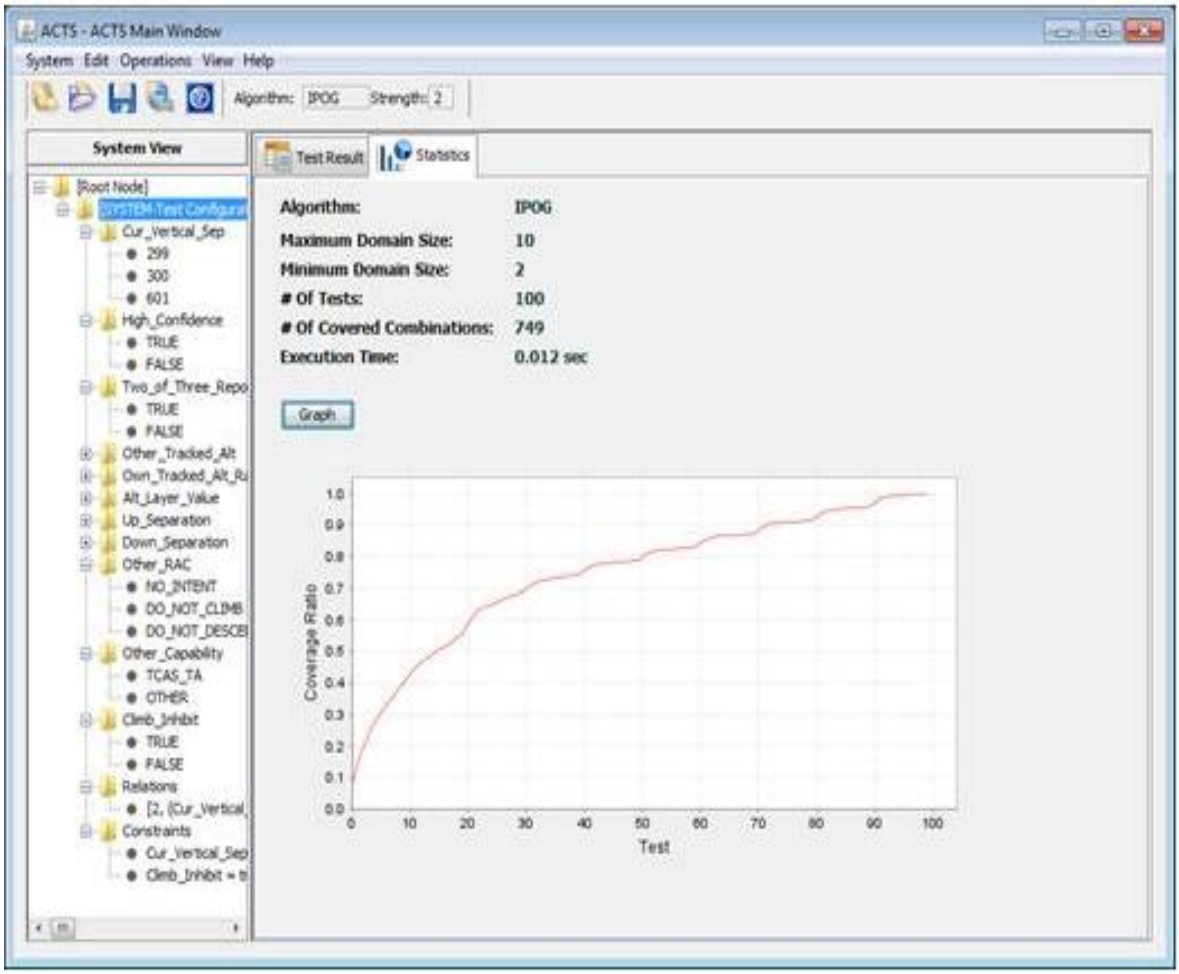

Figure 5: ACTS software shows coverage rate as test size increases

To create a new system, select from the top menu: *System* → *Menu*. The *New System* window contains a tabbed pane with 3 tabs: *Parameters, Relations,* and *Constraints* (Figure 6). The *Parameters* tab allows the user to specify the parameters and the values of those parameters in the new system (Figure 6).

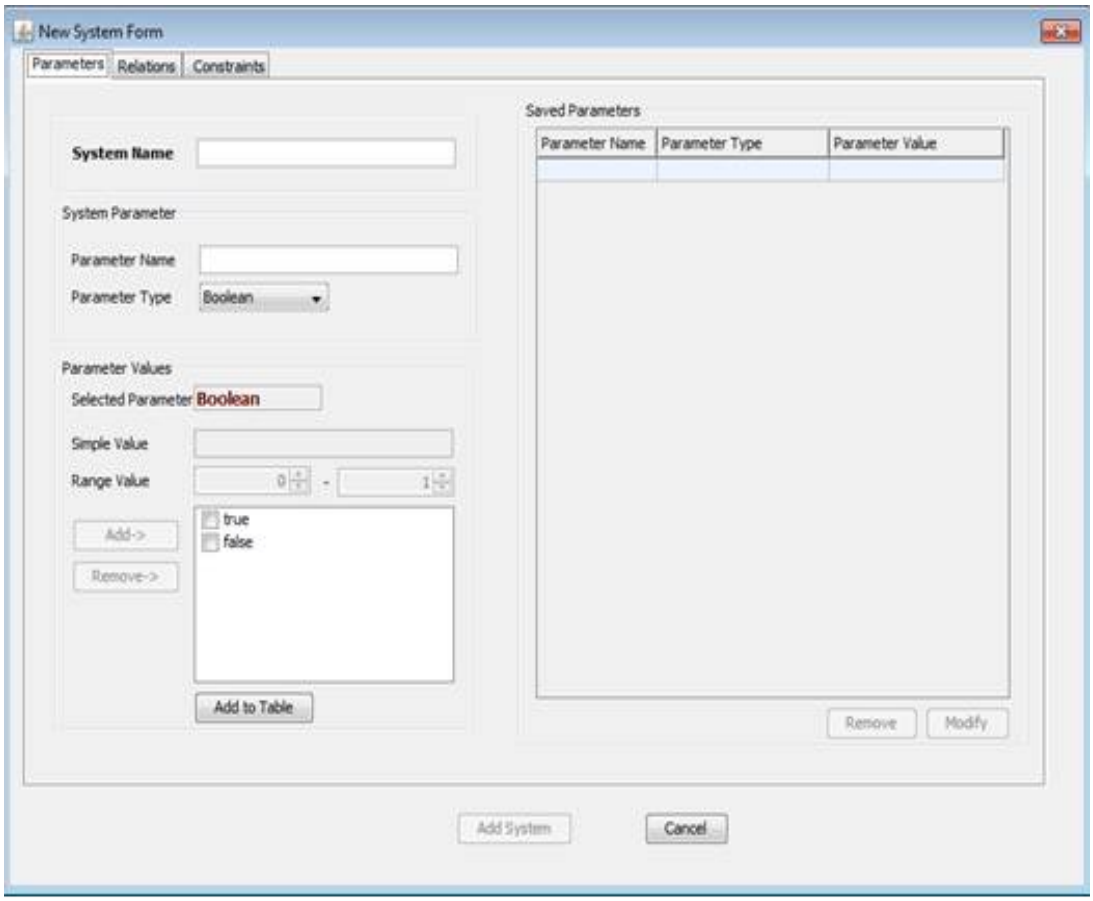

Figure 6: The *Parameters* tab to create a new system (ACTS software)

If we click on the *Relations* tab (Figure 7), the user will be allowed to create different parameters groups and cover those groups with different strengths. A default relation is automatically created that consists of all the parameters that have been specific in the *Parameters* tab with the default. Consider a system consisting of 10 parameters: P1, P2,…, P10. A default relation can be created that consists of all the parameters with strength 2. Additional relations can then be created if some parameters are believed to have a higher degree of interaction.

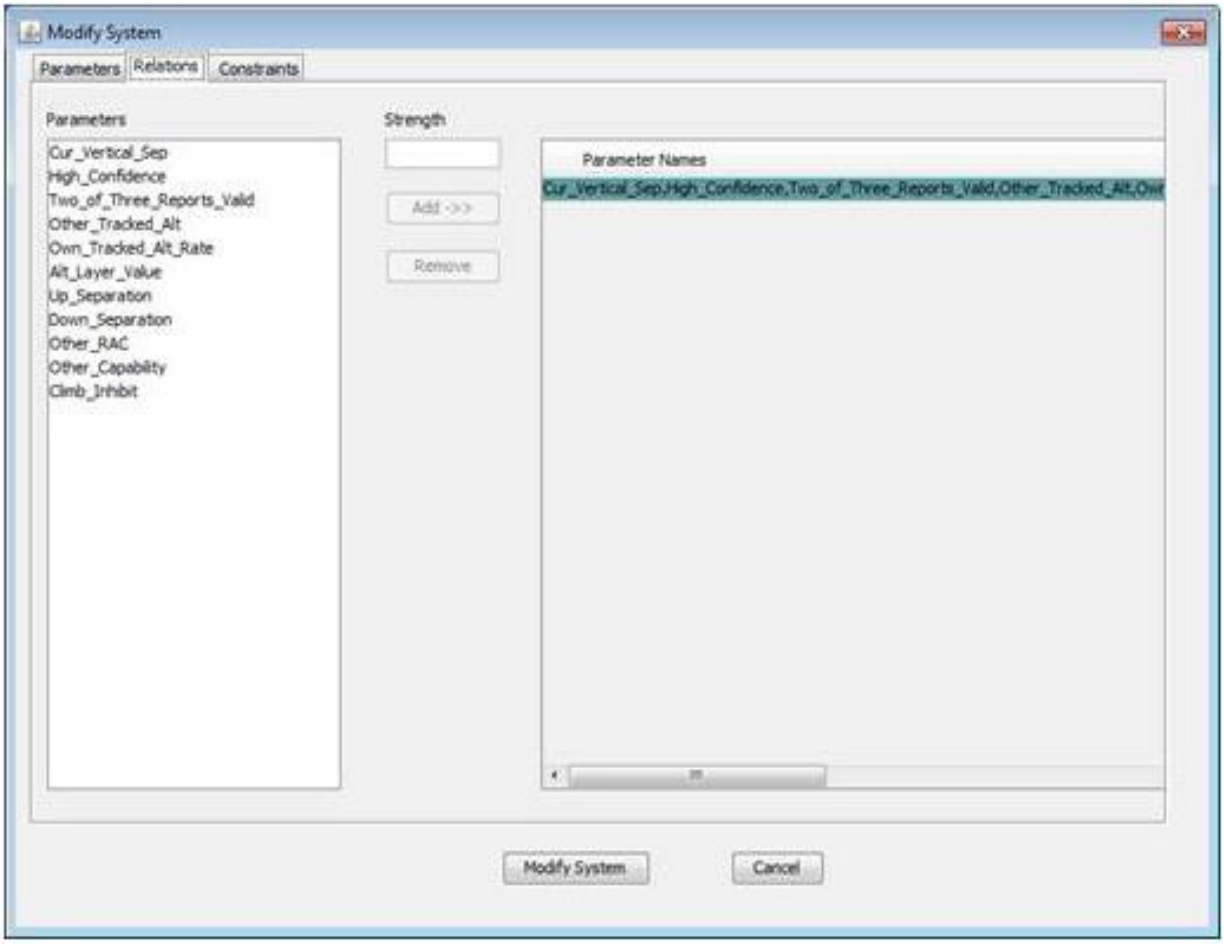

Figure 7: The *Relations* tab to create a new system (ACTS software)

If we click on the *Constraints* tab (Figure 8), the user will be able to specify constraints so that invalid combinations can be excluded from the resulting test set. For example, suppose that we want to conduct a test involving types of operating systems and types of browsers. For the purposes of our test, Internet Explorer has to be the browser when the operating system is Windows. We would create the following constraint rule: *(OS = "Windows") => (Browser = "IE")*.

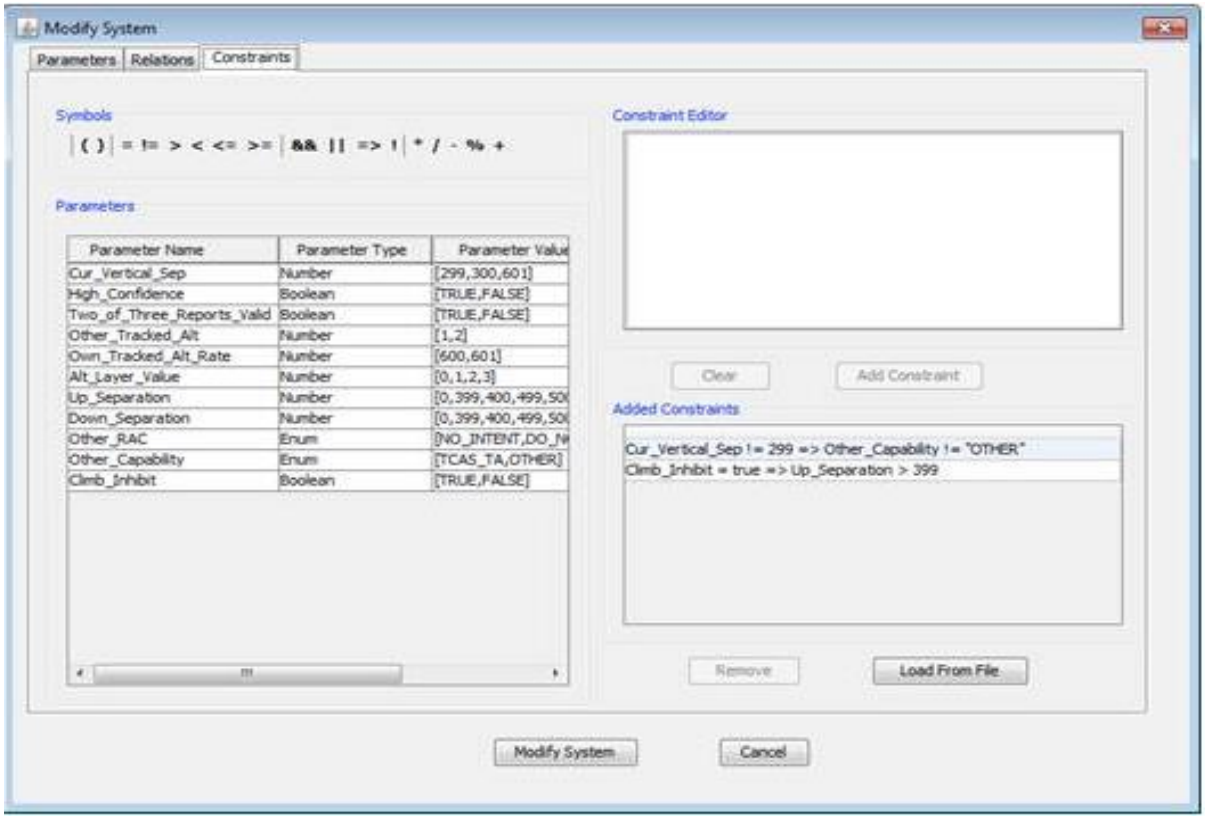

Figure 8: The *Constraints* tab to add constraints in a new system (ACTS software)

To build the test set, select the system in the *System View* and then select from the menu *Operations* → *Build* (Figure 9). Available options include the *Algorithm*. It is recommended that IPOG, IPOG-F, or IPOG-F2 be used for systems of moderate size. For larger systems, use IPOD-D and PaintBall. Note that relations and constraints can only be used if the IPOG algorithm is selected. The *Max Tries* box is used by the PaintBall algorithm and it specifies the number of candidates to be generated randomly at each step. The *Randomize Don't Care* box will replace all the "don't care" values (specified by an asterisk in the test set) with a random value. The *Ignore Constraints* box will ignore all constraints if checked. The *Strength* box specifies the default strength of the test set. The *Mode* box can be *Scratch* (in which the test set is generated entirely new) or *Extend* (which will build upon an existing test set). Finally, the *Progress* box displays the progress information in the console. We encourage the reader to explore the user guide associated with ACTS, available on the NIST website previously provided, to better understand all of the options available.

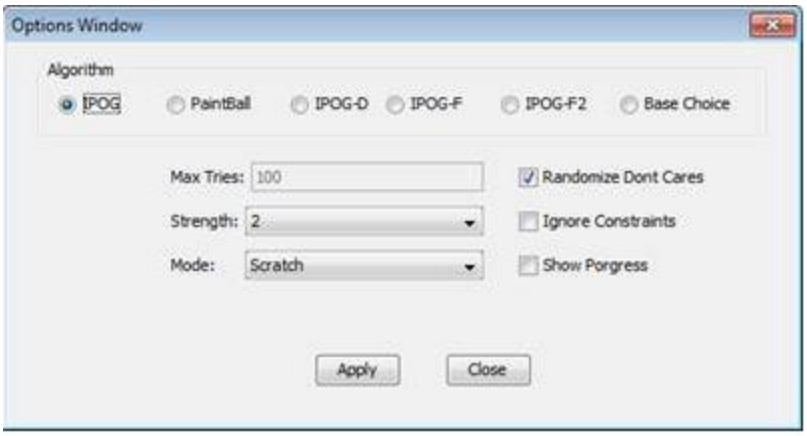

Figure 9: Building the test set in the ACTS Software

Conclusion

Scientific Test and Analysis Techniques (STAT) are traditionally applied to physical stochastic systems, but can also be applied to deterministic systems like software. Software testing is generally concerned with finding all faults/bugs in a program/system. The design space or input domain is defined by all possible settings of input factors and can be exceptionally large. For example, a system with 10 input parameters, each with 5 different settings is approximately 10 million (5^{10}) settings. Examining every combination of inputs settings is too time consuming and impractical. Based on previous research (Kuhn et al., 2010), a solid strategy would be to examine all 3-way or 4-way combinations of parameter values at least once. Combinatorial designs are a class of designed experiments that can create an efficient run matrix that covers all t-way combinations of inputs. This makes these designs ideal for software testing. For the ten 5-value parameters mentioned, earlier all 3-way interactions could be tested in just 307 runs. This paper provided a list of free software available that can be used to generate combinatorial designs. A brief example and tutorial was also provided. The STAT COE highly recommends the use of these tools and strategy when dealing with deterministic software systems.

References

Kuhn, Rick, et al. “Combinatorial Software Testing.” Computer (IEEE Computer Society), vol. 42, no. 7, 2009, pp. 94–96., doi:10.1109/mc.2009.253.

Kuhn, D.R, Kacker, Raghu N., Lei, Yu, “Advanced Combinatorial Test Methods for System Reliability”, *Reliability Society Annual Technical Report*, 2010.

Computer Security Division, et al. “Automated Combinatorial Testing for Software.” Automated Combinatorial Testing for Software, NIST, csrc.nist.gov/groups/SNS/acts/index.html.

"Combinatorial Test Case Generation." *Pairwise Testing*, www.pairwise.org/index.asp.