The goal of the STAT T&E 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.
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Summary</td>
<td>2</td>
</tr>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Combinatorial Designs</td>
<td>3</td>
</tr>
<tr>
<td>Software</td>
<td>4</td>
</tr>
<tr>
<td>Example</td>
<td>4</td>
</tr>
<tr>
<td>ACTS Tutorial</td>
<td>6</td>
</tr>
<tr>
<td>Conclusion</td>
<td>11</td>
</tr>
</tbody>
</table>
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 program. 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 test & evaluation. 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 will examine the use of the combinatorial designs.

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

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 deal with 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 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 test is to obtain a high level of coverage of the input domain with as small a number of tests as possible. Combinatorial designs also known as factor covering arrays are often employed to achieve to meet this design characteristic. In the following section we will provide an introduction to combinatorial designs and present some software tools readily available to create them.
Combinatorial Designs

In software testing, any particular setting of input factors could trigger a fault. Therefore, every possible combination of input factors would have to be investigated in order to ensure no faults or bugs. Consider a system with 10 input parameters, each with 5 different settings. Approximately $10^7$ 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 software development will be used for testing (Kuhn, Kacker, Lei 2010). Luckily, a NIST Study (Kuhn, Kacker, Lei, Hunter Aug 2009) found that most faults are caused by the interactions between a few parameters. Figure 1 shows the cumulative percentage of faults found versus the level of interactions between parameters for various 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](image)

Combinatorial designs attempt to maximize test coverage with the minimum number of test cases 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, Kacker, Lei, Hunter Aug 09). These designs are very easy to use, and very effective at finding fault/bugs. For the ten 5-value parameters mentioned earlier a 2-way interaction could be tested in 49 runs; 3-way interaction in 307 runs; 4-way interaction in 1,865 runs.
Software

The STAT T&E COE does not endorse any particular software package, but instead encourages potential users to download and use different options, and then evaluate their utility. We will 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 that already exist.

www.pairwise.org. There are currently thirty-nine different combinatorial design tools (to include ACTS and the combinatorial coverage measurement tool) on the www.pairwise.org website. We will 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). These 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, Kacker, Lei, and Hunter, we know that if we develop a covering array with strength-3 (meaning we look at all 3-way combinations), we will identify 90+ percent of the faults. We can use one of the aforementioned software tools to develop a 3-way covering array. Suppose that a 0 means the text feature is turned off while a 1 implies the text feature is on. You will notice in Figure 3 that any three chosen columns contain all eight possible values for 3 binary variables, 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 13 trials. Compared to the original 1,024 trials, we have achieved over a 99% savings in trials.
Figure 2.

Figure 3.
ACTS Tutorial

We will 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, that 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, 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).

![ACTS Main Window](image)

Figure 4.
Figure 5.
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, as well as the values of those parameters, in the new system (Figure 6).
Figure 7.

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.
If we click on the *Constraints* tab, 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: $(OS = \text{“Windows”}) \Rightarrow (Browser = \text{“IE”})$. 

![Figure 8.](image-url)
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.

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 10 input parameters, each with 5 different settings is approximately 10 million ($5^{10}$) settings. Examining every combination of inputs settings is obviously time consuming and impractical. Fortunately, a 2009 NIST Study (Kuhn, et. al., 2009) found that most faults are caused by the interactions between a few parameters. The study found that 3 way interactions find approximately 80% of faults and 4 way interactions find over 90% of faults. Therefore a solid strategy would be to examine all 3-way or 4-way combinations of parameter values at least once. Combinatorial designs are one group 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 interaction
could be tested in 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 T&E COE highly recommends the use of these tools and strategy when dealing
deterministic software systems.
References


