Chapter 14
Combinatorial Testing

Renée C. Bryce
Utah State University, USA

Yu Lei
University of Texas, Arlington, USA

D. Richard Kuhn
National Institute of Standards and Technology, USA

Raghu Kacker
National Institute of Standards and Technology, USA

ABSTRACT

Software systems today are complex and have many possible configurations. Products released with inadequate testing can cause bodily harm, result in large economic losses or security breaches, and affect the quality of day-to-day life. Software testers have limited time and budgets, frequently making it impossible to exhaustively test software. Testers often intuitively test for defects that they anticipate while less foreseen defects are overlooked. Combinatorial testing can complement their tests by systematically covering t-way interactions. Research in combinatorial testing includes two major areas (1) algorithms that generate combinatorial test suites and (2) applications of combinatorial testing. The authors review these two topics in this chapter.

I. INTRODUCTION

Software systems are complex and can incur exponential numbers of possible tests. Testing is expensive and trade-offs often exist to optimize the use of resources. Several systematic approaches to software testing have been proposed in the literature. Category partitioning is the base of all systematic approaches as finite values of parameters are identified for testing. Each of these finite parameter-values may be tested at least once, in specified combinations together, or in exhaustive combination. The simplest approach tests all values at least once. The most thorough approach exhaustively tests all parameter-value combinations. While testing only individual values may not be enough, exhaustive testing of all possible combinations is not always feasible. Combination strategies are a reasonable alternative that falls in between these two extremes.

DOI: 10.4018/978-1-60566-731-7.ch014

Table 1. Four parameters that have three possible settings each for an on-line store

Log-in Type	Member Status	Discount	Shipping
New member - not logged in	Guest	None	Standard (5-7 day)
New-member - logged in	Member	10% employee discount	Expedited (3-5 day)
Member - logged in	Employee	$5 off holiday discount	Overnight

Consider an on-line store that has four parameters of interest as shown in Table 1. There are three log-in types; three types of member status; three discount options; and three shipping options. Different end users may have different preferences and will likely use different combinations of these parameters. To exhaustively test all combinations of the four parameters that have 3 options each from Table 1 would require $3^4 = 81$ tests.

In this example, exhaustive testing requires 81 test cases, but pair-wise combinatorial testing uses only 9 test cases. Instead of testing every combination, all individual pairs of interactions are tested. The resulting test suite is shown in Table 2, and is contains only 9 tests. All pairs of combinations have been combined together at least once during the testing process. For instance, the first test from Table 2 covers the following pairs: (New member - not logged in, Guest), (New member - not logged in, $5 off holiday discount), (New member - not logged in, Standard (5-7 day)), (Guest, None), (Guest, Standard (5-7 day)), and (None, Standard (5-7 day)). The entire test suite covers every possible pairwise combination between components. This reduction in tests amplifies on larger systems - a system with 20 factors and 5 levels each would require 5^{20} = 95,367,431,640,625 exhaustive tests! Pairwise combinatorial testing for 5^{20} can be achieved in as few as 45 tests.

II. BACKGROUND

Combinatorial testing is simple to apply. As a specification-based technique, combinatorial testing requires no knowledge about the implementation under test. Note that the specification required by some forms of combinatorial testing is lightweight, as it only needs to identify a set of parameters and their possible values. This is in contrast with other testing techniques that require a complex operational model of the system under test. Finally, assuming that the parameters and

Table 2. A pair-wise combinatorial test suite

Test No.	Log-in Type	Member Status	Discount	Shipping
1	New member - not logged in	Guest	None	Standard (5-7 day)
2	New member - not logged in	Member	10% employee discount	Expedited (3-5 day)
3	New member - not logged in	Employee	$5 off holiday discount	Overnight
4	New-member - logged in	Guest	$5 off holiday discount	Expedited (3-5 day)
5	New-member - logged in	Member	None	Overnight
6	New-member - logged in	Employee	10% employee discount	Standard (5-7 day)
7	Member - logged in	Guest	10% employee discount	Overnight
8	Member - logged in	Member	$5 off holiday discount	Standard (5-7 day)
9	Member - logged in	Employee	None	Expedited (3-5 day)