A Unified Theory of Software Testing

Bret Pettichord
16 Feb 2003

This paper presents a theory, or model, for analyzing and understanding software test techniques. It starts by developing a theory for describing analytical test techniques, such as equivalence classes, pair-wise coverage and state modeling. It then develops the theory to cover other, more heuristic test techniques.

This theory states that all testing techniques are composed of a modeling schemata, one or more coverage objectives, a dispersal strategy and a theory of error.

All testing techniques address a fundamental problem of testing: vastly more tests are conceivable than you could possibly run. Each technique therefore consists of a method for modeling the software in a way that allows a coverage measure to be specified and achieved and a dispersal strategy that allows this to done efficiently. Testing theory has traditionally focused on the issue of coverage and the different ways to think of it (Kaner, 101 methods for code coverage). The obverse concept is dispersal, or a method for identifying and reducing tests that are "redundant".

Take, for example, the pair-wise coverage technique. This technique is often applicable when the interaction of several independent inputs or configuration parameters create more combinations than is practical to test. Suppose you have an application that supports each of the following components:

DATABASE
Oracle
DB2
SQLServer

WEB SERVER
IIS
Apache
Netscape

APP SERVER
WebSphere
WebLogic
Tomcat

If you test all possible combinations, you will have 27 test configurations: 3 x 3 x 3. The pair-wise technique states that you can reduce by only considering pair-wise combinations of the three parameters. Instead of 27 configurations, we only have to test 9. Here's a set of 9 test configurations that meet the pair-wise condition:

Oracle	IIS	WebSphere
Oracle	Apache	WebLogic
Oracle	Netscape	Tomcat
DB2	IIS	WebLogic
DB2	Apache	Tomcat
DB2	Netscape	WebSphere
SQLServer	IIS	Tomcat
SQLServer	Apache	WebSphere
SQLServer	Netscape	WebLogic

We came up with this matrix just by manual jiggering. Larger matrices often require the use of more systematic methods for ensuring pair-wise coverage. One popular technique is to use orthogonal arrays. There are also other algorithms that can sometimes produce smaller test sets. These are built into tools such as ALLPAIRS (Bach)and AETG (Bellcore).

A single mode error occurs based on the value of a single parameter. A dual-mode error occurs when two parameters must be set to particular values. The efficacy of the pair-wise technique is based on the general observation that most software errors are single-mode or dual-mode errors. In this case, the combination of any two types of software is considered sufficient to uncover interaction bugs.

Let's return to our theory of test techniques. The pair-wise technique is first of all based on a modeling of the software in terms of independent input or configuration values. (There are variations of the technique to handle cases where they are not completely independent.) These may themselves be equivalence classes -- a model that results from another test technique: equivalence class analysis. In any case, this model (independent inputs) allows particular coverage objectives to be articulated: pair-wise combination of input values, or indeed n-wise combinations. (In more complex situations with more than three input variables, 3-wise coverage can still be less than all-combinations.)

Implicit in the pair-wise coverage goal is a dispersal strategy: a method for eliminating tests that are "redundant." The redundancy is of course not certain: we never know quite which tests will uncover bugs. But the redundancy is with respect to the theory of error which underlies the technique. In sum:

Technique: Pair-wise Testing
Modeling Schemata: Enumerated independent inputs
Coverage Measure(s): Pair-wise, n-wise coverage
Dispersal Strategy: Orthogonal arrays or other pair-wise algorithms
Theory of Error: Errors are single- or dual-mode

The theory presented here states that all test techniques contain these four elements (modeling schemata, coverage measure, dispersal strategy, theory of error). Moreover, a test technique can be evaluated in several ways. A technique's *efficiency* is a measure of the degree to which its dispersal strategy allows it to meet its coverage goal. This is an internal characteristic and can often be determined analytically. A technique's *coherence* refers to the degree to which a theory of error is articulated and justifies the modeling schemata and coverage measures. This is a matter of critical analysis. A technique's *validity* is a measure of the degree to which its theory of error corresponds to the actually errors that appear in software in general. This is a matter of empirical observation. Thus, the assessment of test techniques requires analytical, critical and empirical thinking.

There has been some recent dispute and confusion regarding the differences and advantages of orthogonal arrays compared to other pair-wise algorithms. We'll review this debate as a way of illustrating the use of this theory and its terms of criticism. Orthogonal arrays which meet two conditions: pair-wise coverage of independent parameters and symmetry -- if one pair appears twice, all will appear twice. The symmetry condition is important for experiments that intend to use statistical methods to determine correlation among parameters and results.

There are two ways to analyze this dispute: (1) as two different dispersal strategies for the same technique and (2) as two different techniques. Considering them as different dispersal strategies, it can be clearly shown

that pair-wise algorithms are often more efficient than orthogonal arrays.

The analysis of them as different techniques is more difficult. Fundamentally, however, orthogonal arrays would require a different theory of error -- something beyond the dual-mode failures. I have not seen any such thing articulated and therefore consider orthogonal arrays as a separate technique to be incoherent. Perhaps someone who wishes to defend it as a separate technique can articulate a theory of error and perhaps even articulated a different coverage measure that takes advantage of the symmetry condition.

I provide this analysis not as a final word on pair-wise and orthogonal arrays, but as an illustration of how the theory and its terms of criticism can be used to aid the analysis of testing techniques.

I shall now move ahead with a review of several well-studied analytical testing techniques, using our theory.

Equivalence class analysis is a technique that sometimes goes under the name domain (or sub-domain) analysis. Different authors present the technique differently. Some focus on dividing input among valid and invalid domains. Others encourage a focus on finding inputs to reach all possible outputs. Indeed, this variation on presentation of what different authors apparently believe to the same technique was a motivation for the development of this theory. (Beizer, Jorgensen, Kaner, Gustafson, Myers)

The variation in presentation is probably partly due to the fact that with many programs the determination of whether a particular input is valid depends on other inputs. In my own mind, I am not quite sure whether equivalence class analysis really constitutes an analytical technique or rather a set of prototypical analyses. Myers, for example, only uses the technique to create valid and invalid sub-domains. Later authors (e.g. Gustafson, Jorgensen), however, present it as a technique for solving Myers' triangle problem (an application that Myers himself does not make): defining equivalence classes for each of the possible outputs (isosceles, scalene, and equilateral). Jorgensen justifies this extension: "Strongly typed languages eliminate the need for the consideration of invalid inputs.

Traditional equivalence testing is a produce of the time when languages such as FORTRAN and COBOL were dominant, hence this type of error was common." (p. 73) Beizer also makes reference to long list of "bug assumptions" to his presentation of domain analysis; these constitute what we call a theory of error. Interestingly, Beizer does not present "bug assumptions" for his other test techniques.

The literature thus presents a variety of examples under the rubric of "domain testing". My search for an analytical principle that defines the technique or distinguishes from others has been fruitless. Instead it has motivated me to define this theory of test techniques, which could be restated as all test techniques are methods for defining equivalence classes. Also, each author has used his own judgement to define a conception and method that works for him. They tend to treat this, an ancillary to the technique; I wish to focus on it and treat as a core element: testing is the process of articulating error hypotheses, using them to define coverage criteria and then dispersing tests across the defined domain.

Technique: Equivalence classes (domain testing)
Modeling Schemata: Define input domains based on validity or outputs Coverage Measure(s): Domain coverage
Dispersal Strategy: Select inputs from uncovered domains until covered
Theory of Error: Various

Boundary value analysis typically builds on equivalence classes. It begins by nullifying the initial assumption of equivalence classes: that any element of a domain is equally likely to uncover a bug. Instead, it says that items at the boundary are more likely to find other kinds of bugs (and just as likely to find the bugs that the equivalence class was designed for).

Technique: Boundary Value Analysis
Modeling Schemata: From Equivalence Classes
Coverage Measure: Boundary coverage: inner and outer boundaries
Dispersal Strategy: Pick boundary values
Theory of Error: Off by one errors, wrong comparators (e.g. > for >=)

Some other analytical techniques:

Technique: Decision Tables
Modeling Schemata: Decision Tables (Truth tables with actions)
Coverage Measure: One test case per column
Dispersal Strategy: One test case per column
Theory of Error: Logic errors

Technique: State Transition Diagrams
Modeling Schemata: Directed Graphs
Coverage Measure(s): Node, edge, path coverage metrics
Dispersal Strategy: Various algorithms for meeting stated coverage
goals
Theory of Error: various, often unspecified

I refer to these techniques as analytical because they all conform to some general rules:

- They each use formal modeling schemata that can be expressed in mathematical form.

- They each provide one or more coverage goals that can be defined mathematically, in terms of the modeling schemata.

- They each provide an algorithmic dispersal strategy.

These indeed are the characteristics that they highlight and that are often cited when they are claimed to be "scientific". They all however have some other characteristics, to which less focus is often made:

- They each use heuristic methods to create the test model which require the use of personal judgement.

- They each depend on a theory of error, whose validity is dependent on programming languages, operating environments, organizational dynamics...

Code coverage analysis is often cited as an analytical technique, but it actually takes a different form:

Technique: Code coverage analysis
Modeling Schemata: Directed graph generated from code
Coverage Measure: node, edge, path coverage

Dispersal Strategy: none
Theory of Error: you need execute code to find bugs in it

Unlike the other analytical techniques, code coverage does not require judgement and heuristic methods to create the model, but it also does not provide a dispersal strategy for efficient test creation. For these reasons, it is sometimes classified as an evaluation or assessment technique, rather than a technique for creating tests.

Note that like code coverage analysis, requirements traceability has the profile of an assessment technique, rather than one for test creation:

Technique: Requirements Traceability
Modeling Schemata: Identify all project requirements
Coverage Measure: Requirements coverage
Dispersal Strategy: None (Traditional usage provides one test per requirement, which provides low efficiency)
Theory of Error: Requirements misunderstood, forgotten

I'll state again a central observation of this theory: all test techniques require judgement and an implicit theory of error. Indeed understanding the theory of error is key to exercise of good judgement.

Myers makes a glancing mention of "error guessing" as a test technique and many people have classified exploratory testing as a method that depends on intuition. My claim, as a part of this theory, is that all testing amounts to error guessing.

A lesson I draw from this is that effective tester training teaches testers how to articulate error hypotheses and then design tests that effectively test it. It is this kind of activity, rather than the adherence to rote methods, that makes software testing scientific.

Indeed there has been a tendency to treat many of the traditional analytical techniques as foundational and the core of tester certification. The problem with this is that they codify methods for finding bugs that occurred decades ago! The use of techniques needs to be validated by reviewing the kinds of errors that are a concern for us today in the software that is being written today.

Indeed testing consists of creating and testing several types of models:

1. A model of system capability which describes the functions and operations of the system under test.

2. A model of user behavior which describes the goals and profiles of the users of the system.

3. A model of error which describes the kinds of faults that may exist in the system and how they may become manifest.

4. A model of the environment in which the system operates.

Good testing amounts to being able to create, articulate and test these various models.

I'll conclude this paper by reviewing the heuristic testing techniques developed by Bach.

Technique: Heuristic Testing
Modeling Schemata: Undefined
Coverage Measure: "How Good is This Test Plan" heuristics
Dispersal Strategy: Various heuristic lists ("test techniques”,” product elements", "quality criteria")
Theory of Error: Heuristic risk assessment

This technique, often called exploratory testing, works on a different model than the analytical techniques. It puts less emphasis on finding a formal model for testing and more on the judgement used to create such models. It also focuses on the need for error hypothesis (risk-based testing) as a driver for testing.

Acknowledgements

A presentation of these ideas was first made at WTST (Feb 2003), sponsored by the National Science Foundation. I thank Cem Kaner, James Bach, Richard Kopec and Lee Copeland for allowing me to sit in on their testing classes and appreciate their variety of approaches.