

# Narrow Spectrum Software Testing Addressing Complexity and Trust

**Rick Kuhn**, NIST

**M S Raunak**, Loyola University Maryland



*Focused, narrow spectrum tests can be one component of improved assurance for software products. Tests designed using knowledge of different failure types and fault characteristics should be more widely used.*

**S**oftware testing is a delicate combination of art and science. A great deal of sound theory has been developed, but in practice, this theory is often ignored. Most of the testing is ad hoc, especially for consumer-level software, where time to market can be the most critical factor in development. In this case, tests are often aimed primarily at demonstrating the specific functionality of the application being tested, such as showing that an inventory program accepts and saves records of new products received.

But many software functions are more generic, such as text search, set

examples include test catalogs for particular checks, such as in Marick.<sup>1</sup>

This focused approach, called *narrow spectrum software testing (NSST)*, should become a more common practice. For example, a string search on a database is a specific functionality that can be tested using a narrow focus to discover range-of-failure scenarios from unusual inputs. There are many reasons for employing NSST beyond the need for better software assurance. Much software testing is contracted out, with little to evaluate the thoroughness of assurance beyond the basic requirements of tracing and demonstrating functions (happy path testing).

Common Weakness Enumeration,<sup>2</sup> which seeks to describe software flaws generically such that they can be better understood and classified, with the ultimate goal of providing methods to ensure that particular weaknesses are not present in software. A more rigorous and systematic approach to this goal is the Bugs Framework,<sup>3</sup> which is being developed to provide a precise taxonomy of flaws, a sort of “periodic table” of software bugs. Knowledge in these data sets can be used to develop more systematic and focused tests for all types of flaws beyond security weaknesses.

What would such focused, narrow spectrum tests include, and how can their thoroughness of coverage be evaluated? When test coverage is measured, it typically refers to structural coverage (that is, code coverage), but if we are testing only a narrow range of functionality, it should not be expected that code is exercised outside of a few modules specific to the focus of the test, so other measures are needed. In particular, we need to define an input model for the specific functionality being tested and measure the degree to which it has been covered in tests. The input space model refers to the parameters and values tested and the definition of the equivalence classes used to select the test cases. Because rare combinations of input values can trigger failures, coverage measures for the input space should include the combination coverage achieved at each combination level (two- and three-way combinations, and so on).

To illustrate such an approach, we developed tests for full text search

**FOR ROUTINE FUNCTIONS OR COMMON PROBLEMS, WELL-DEVELOPED GENERIC TEST CASES CAN BE USED.**

membership checking, time and date management, and many others. Showing that the software works with more common inputs and configuration fails to test the system for unexpected combinations of inputs, which are more often the causes of failure. For routine functions or common problems, well-developed generic test cases can be used. This is common in security testing, where a variety of specialized methods have been developed to test for buffer overflow, memory leaks, and other common problems. Such focused test ideas go back to the early days of software testing research; other

Structural coverage may be used in some cases, but even this is often limited to the most basic (and minimally effective) metric of statement coverage. Focused, narrow spectrum tests can be one component of improved assurance that products are free of failures. Tests designed using the knowledge of different failure types and fault characteristics could be made widely available and used by the community.

A great deal of data on software problems, especially for security vulnerabilities, exists and can be used for understanding and organizing problem types. Examples include the

## IN THIS ISSUE

Hodges et al., the authors of “Physical Computing: A Key Element of Modern Computer Science Education,” consider how current computer science education may not address the needs of the global student population nor the latest developments in coding and data science. This brings forth the idea that computer science education should combine both the software and hardware aspects into a more holistic technique that mirrors the real world. The authors introduce a new education approach, termed *physical computing*, that addresses the need for a hybrid software and hardware methodology that is more hands on and device oriented in the classroom.

In “Blockchain: Can It Be Trusted?” Ahmed and Pathan explain that while blockchain supposedly provides security and privacy of data, known vulnerabilities have been identified.

Several benefits of blockchain are presented, and recent financial mishaps due to the application of blockchain are showcased. The authors do not argue against deploying blockchain but, rather, give the reader pause before large-scale blockchain adoption.

The last feature article in this issue is “Is Immersive Virtual Reality the Ultimate Interface for 3D Animators?” Lamberti et al. review the application of virtual reality to the field of computer animation, which is well known to be labor and skill intensive. This article investigates the impact of virtual reality animation systems on the performance of the animators by using qualitative and quantitative observations. The authors also look at the impact of these systems on interface usability and the quality of the content that is eventually produced.

– Jeffrey Voas, Editor in Chief

Digital Object Identifier 10.1109/MC.2020.2974138  
Date of current version: 9 April 2020

in a large, heavily used public database.<sup>4</sup> Our objective was to test the search functionality using not only the common search strings but also with uncommon combinations of letters, keywords, numbers, and special characters, which may cause the database-backed web application to depict a failure scenario. The other aspect of this test case selection was to model the input fragments and use systematic combinations of them to cover a large portion of the input space. The input model we developed for the text search included five parameters with both keywords and special characters, with 10, 4, 10, 4, and 10 enumerated values for the parameters. This is designated a  $4^2 10^3$  input space configuration (two parameters with four values

each and three parameters with 10 values). Covering all the possible input strings would then require 16,000 tests, but all the two-way combina-

coverage strengths (two way, three way, and so on) showed that 49 faults were discovered using two-way combinations, but no additional faults occurred

**WHEN NO NEW FAULTS ARE DISCOVERED USING  $(t + 2)$ -WAY TESTS, ADDITIONAL FAULTS ARE UNLIKELY TO BE FOUND.**

tions of the input fragments resulted in only 100 tests and all the three-way combinations produced 999 tests.


The key question in such testing is whether the tests have been sufficient. Comparing faults detected at different

using higher strengths of three and four way. Using higher-strength  $t$ -way tests provide a reasonable measure of test completeness. When no new faults are discovered using  $(t + 2)$ -way tests, additional faults are unlikely to

### DISCLAIMER

Products may be identified in this document; however, identification does not imply a recommendation or an endorsement by the NIST nor that the products identified are necessarily the best available for the purpose.

be found. The text search tests can be applied with little or no change to a variety of systems because text search is a common function. By using *t*-way factor combinations, we can show that the entire input space has been covered up to a suitable combination level.

**C**ombination coverage-based narrow-spectrum testing supplements basic structural coverage-based test selection. It provides a sound test engineering method with defensible, quantitative measures of test completeness. The approach is particularly useful for increasing trust in complex functionality of software systems. 

### REFERENCES

1. B. Marick, *The Craft of Software Testing*. Englewood Cliffs, NJ: Prentice Hall, 1994.
2. CWE, "Common Weakness Enumeration." Accessed on: Feb. 1, 2020.

## ABOUT THE AUTHORS

**RICK KUHN** is a computer scientist in the Computer Security Division at NIST. His research interests include software failures and vulnerabilities, testing and verification, and access control. Kuhn received an M.S. in computer science from the University of Maryland, College Park. He is *Computer's* Cybersecurity area editor and a Fellow of the IEEE. Contact him at [kuhn@nist.gov](mailto:kuhn@nist.gov).

**M S RAUNAK** is an associate professor and the chair of the Computer Science Department at Loyola University Maryland. His research interests include software verification and validation. In particular, he is interested in developing effective testing approaches for difficult-to-test software systems. Raunak received an M.S. and a Ph.D. from the University of Massachusetts Amherst. Contact him at [raunak@loyola.edu](mailto:raunak@loyola.edu).

- [Online]. Available: <https://cwe.mitre.org>
3. I. Bojanova, P. E. Black, Y. Yesha, and Y. Wu, "The bugs framework (BF): A structured approach to express bugs," in *Proc. 2016 IEEE Int. Conf. Software Quality, Reliability and Security (QRS)*, pp. 175–182. doi: 10.1109/QRS.2016.29. [Online]. Available: <https://www.nist.gov/publications/bugs-framework-bf-structured-approach-express-bugs>
  4. M S Raunak, D. R. Kuhn, and R. Kacker, "Combinatorial testing of

full text search in web applications," in *Proc. 2017 IEEE Int. Conf. Software Quality, Reliability and Security Companion (QRS-C)*, pp. 100–107. doi: 10.1109/QRS-C.2017.24.



IEEE COMPUTER SOCIETY

**DIGITAL LIBRARY**

Access all your IEEE Computer Society subscriptions at

**computer.org**

**/mysubscriptions**