Introducing Combinatorial Testing in a Large Organization

Jon D. Hagar and Thomas L. Wissink, Lockheed Martin

D. Richard Kuhn and Raghu N. Kacker, National Institute of Standards and Technology

A two-year study of eight pilot projects to introduce combinatorial testing in a large aerospace corporation found that the new methods were practical, significantly lowered development costs, and improved test coverage by 20 to 50 percent.

In 2009, when Lockheed Martin decided to explore the benefits of combinatorial testing methods as a way to reduce testing costs and maintain competitive processes, its decision was based on several developments. That same year, pilot projects using these methods at Eglin Air Force Base in Florida were showing promise in reducing testing costs. The US Department of Defense's (DoD's) Office of Test and Evaluation had recently endorsed the design of experiments (DOE) statistical testing approach,[1] some principles of which are in combinatorial testing. In the commercial realm, firms were reporting success with pairwise testing, a basic form of combinatorial testing that covers two-way factor combinations.

Together, these developments were sufficient motivation for Lockheed Martin to launch its own pilot projects. The goal was to make combinatorial testing available to its engineers and use it internally without mandates from either customers or management. To aid in this effort, the company developed a cooperative research and development agreement (CRADA) with the National Institute of Standards and Technology (NIST), which is one of NIST's mechanisms for conducting joint research with US industry. CRADAs also provide flexibility in structuring projects, assigning intellectual property rights, and protecting industry-related proprietary information and research results.

Lockheed Martin and NIST entered into the CRADA to better understand the applicability and effectiveness of a relatively new approach for software testing to improve the quality, safety, and reliability of US products and systems. A goal of particular interest was to better understand the challenges in introducing a new software-testing approach in a large US corporation.

Specific goals in introducing the method in Lockheed Martin projects were to

- evaluate the concept's viability;
- test process improvement in a variety of domains, including system, software, and hardware testing;
- make tests more effective in finding problems; and

- reduce testing cost, or at least the test life-cycle cost, by reducing errors found late in development or in the field.

Lockheed Martin entered into this investigation and improvement effort with the assumption that gathering enough sound evidence to introduce real change would take time and a series of efforts. These assumptions stemmed from both its process improvement experience and understanding of what motivates large organizations.

To assess the applicability of combinatorial testing, described in more detail in the sidebar "Understanding Combinatorial Testing," Lockheed Martin evaluated tools, generated sample test cases, and analyzed data from pilot projects. From this process, the company learned many valuable lessons about introducing combinatorial testing into a large corporate body.

TABLE 1. Tools used in the pilot projects.

Developer	Tool*	Application context
NIST/University of Texas at Arlington	ACTS	Covering array generation; constraint support
Air Academy Associates	SPC XL	Statistical analysis support
	DOE KISS	Support of simple design of experiments analysis
	DOE PRO XL	Design of experiments support
	DFSS Master	General analysis
Phadke & Associates	rdExpert	Historic test data analysis
Hexawise	Hexawise	Covering array generation; Web-based, group collaboration

*ACTS: Advanced Combinatorial Testing System; SPC: Statistical Process Control XL (an Excel plug-in); DOE: Design of Experiments; KISS: Keep It Simple Statistically; DFSS: Design for Six Sigma.

EARLY INVESTIGATIONS

Lockheed Martin's interest in combinational testing actually began in 2005, four years before it launched the pilot projects. At that stage, technical staff reviewed reports on the basic pairwise testing form of combinatorial testing from various industrial sources. The company then contracted with consultants to generate introductory training and informational materials, so that senior staff could get some exposure to the concepts. For experienced testers, Lockheed Martin also held an advanced class on combinatorial testing. However, although testers learned the principles of combinatorial testing, they only occasionally expressed interest in using it.

Once it decided on and funded a pilot evaluation, Lockheed Martin targeted one or two staff members per project to learn combinatorial test concepts and how to use the supporting tools. At this point, R&D had four main goals:

- conduct evaluation studies to determine which combinatorial test tools might fit company needs;
- conduct some corporate-funded case studies to evaluate the applicability of combinatorial testing for aspects of ongoing projects;
- generate an informational website to broaden combinatorial testing knowledge and promote skills development; and
- create a set of online training materials with exercises so that test staff could review and access the classes at any time.

Trade studies

Project staff in large companies are often skeptical about any technology they are not already using, because of the risks associated with using tools without a solid track record. Prototype evaluation studies can minimize the perception of risk by providing a way to assess a new concept without committing to its application. These studies are particularly useful if funding is outside of corporate project funding, as was the case in the combinatorial testing effort.

Lockheed Martin conducted several trade studies, including a comparison to historic F-16 design and test problems[2] and assessments of the method's applicability to visual display and flight testing, the support of vehicle and weapons configuration testing, and digital command system testing. The company selected candidate studies on the basis of test data availability and life-cycle stage.

Preliminary tool evaluation

Recognizing that effective combinatorial testing requires appropriate software tools, the company obtained a series of tools, compared their features, and applied them to real problems. It also created a website to provide a single destination for combinatorial test information and tools. Table 1 lists the tools evaluated and their application context.

PILOT TOOLS AND APPLICATIONS

Each pilot project required supporting tools to implement and support combinatorial testing. For the primary tool, Lockheed Martin chose the Advanced Combinatorial Testing System (ACTS; http://csrc.nist.gov/acts), jointly developed by NIST and the University of Texas at Arlington. The company supplemented ACTS as needed with the other tools listed in Table 1.

UNDERSTANDING COMBINATORIAL TESTING

System failures often result from the interaction of conditions that might be innocuous individually, so combinatorial testing can be effective for domains with many interacting parameters, such as aerospace applications. Research suggests that in some circumstances, covering four-way to six-way combinations can be nearly as effective as testing all possible combinations, often dramatically reducing the number of test cases.[1]

Because of these results, interest is growing in combinatorial testing, but many testers do not fully understand it. The method has its roots in design of experiments (DOE), a methodology for conducting controlled experiments in which a system is exercised for chosen test settings of various input variables, or *factors*. The corresponding values of one or more output variables, or *responses*, are measured to generate information for improving the performance of a class of similar systems. These methods were developed in the 1920s and 1930s to improve agricultural production, and later adapted for other industries.

Early testing with DOE revealed its weaknesses in testing software applications, which led to the use of covering arrays (CAs)—these are still the primary vehicle for implementing combinatorial testing in software systems.

DESIGN OF EXPERIMENTS METHOD

DOE's objective is to improve the mean response over replications. Japanese quality engineering guru, Genichi Taguchi,[2] promulgated a variation of these methods using orthogonal arrays (OAs), in which every factor-level combination occurs the same number of times. He used OAs in industrial experiments, aiming to determine the test settings at which the variation from uncontrolled factors was the lowest.[3,4]

The advent of computers and telecommunication systems underlined the importance of thoroughly testing software and hardware-software systems. Software engineers tried using OAs to include all pairs of test settings, but quickly realized the limitations of OA-based test suites. Often, an OA matching the required combinatorial test structure did not exist, and OA-based test suites tended to include invalid (nonexecutable) test cases.

COVERING ARRAYS

OA-based testing limitations led to the use of CAs, which can be constructed for any strength testing—unlike OAs, which are generally limited to strengths 2 and 3. In general, OA use is now relegated to statistical questions, while software testing relies more on covering arrays.

CAs have several advantages over OAs:

- They can be constructed for any combinatorial test structure of unequal test setting numbers.
- If an OA exists for a combinatorial test structure, a CA of the same number or fewer test cases is also possible.
- Test suites can exclude invalid combinations.

TABLE A. Nine tests that cover all two-way interactions (pairwise).

Test	Location	Text_size	Doc_size	Select_method
1	Start	Small	Small	Mouse
2	Start	Medium	Medium	Edit_menu
3	Start	Max	Max	Keyboard
4	Middle	Small	Max	Edit_menu
5	Middle	Medium	Small	Keyboard
6	Middle	Max	Medium	Mouse
7	End	Small	Medium	Keyboard
8	End	Medium	Max	Mouse
9	End	Max	Small	Edit_menu

A CA is basically a matrix that includes all t-way combinations of values for some specified interaction level t. A fixed-value CA, CA(N; t, k, v), is an $N \times k$ matrix of elements from a set of v symbols {0, 1, ..., (v – 1)} such that every set of t columns contains each possible t-tuple of elements at least once. The positive integer t is the *strength* of the covering array.

A mixed-value, or mixed-level, covering array is a generalization of the fixed-value CA that allows columns to have different numbers of distinct elements. Most software testing problems require mixed-level arrays because parameters might have different numbers of values or equivalence classes.

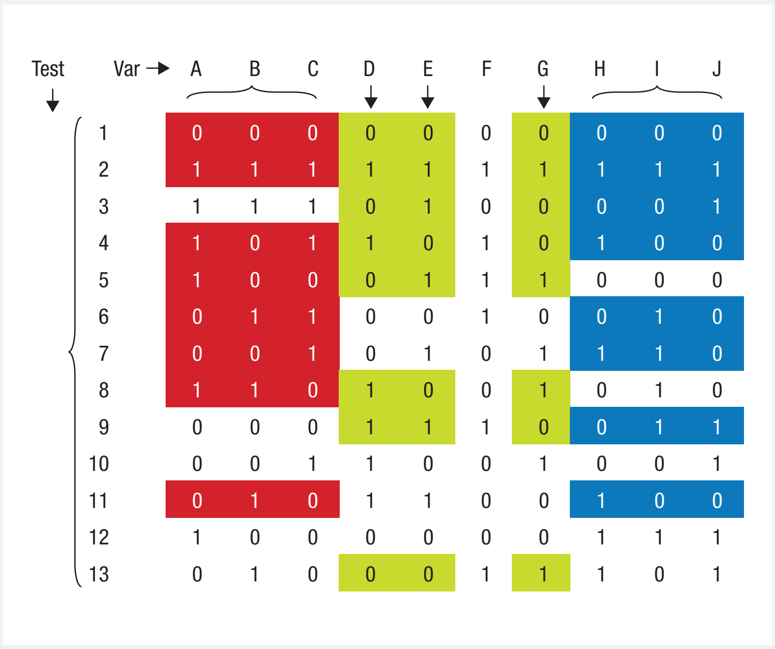
To illustrate a CA, consider testing a four-parameter function that inserts text into a document. The four parameters and representative values are `location: {start, middle, end}`, `text_size: {small, medium, max}`, `document_size: {small, medium, max}`, and `selection_method: {mouse, edit_menu, keyboard}`. Thus, 3^4 (81) possible combinations will exhaustively test these values. However, the nine tests in Table A can test all two-way interactions—that is, all possible value pairs occur at least once among the tests.

Reducing the test set from 81 to 9 is not that impressive, but a larger example illustrates a more dramatic decrease. Suppose the test is of a manufacturing automation system with 20 controls, each with 10 possible settings—a total of 10^{20} combinations. With a covering array, only 162 tests can cover all pairs of these values.

Figure A gives another example, this one of a test with a three-way covering array for 10 variables with two values each. As the figure shows, any three array columns contain all eight possible values for three binary variables. For example, taking columns A, B, and C, all eight possible three-way combinations (000, 001, 010, 011, 100, 101, 110, 111) occur somewhere in the three columns together. Any combination of three columns chosen in any order will also contain all eight possible values. Collectively, therefore, this test set will exercise all three-way combinations of input values in only 13 tests, as compared with 1,024 for exhaustive coverage.

Testers can generate similar arrays to cover combinations of more parameters. In general, the number of t-way combinatorial tests required is proportional to $v^t \log n$, for n parameters with v possible values each. It is best to limit the number of equivalence classes or discrete parameter values to about 10 or fewer, but applications with a larger number of parameters do not present a problem. In general, combinatorial testing will cover applications with up to a few hundred parameters, which is sufficient for most software testing problems.

Methods for constructing CAs fall into three categories: algebraic;[5] metaheuristic, such as simulated annealing and tabu search;[6,7] and greedy search.[8,9] For certain special combinatorial test structures, algebraic methods are extremely fast and can produce very compact CAs, but many testing problems do not fit the requirements of algorithms in this category. Metaheuristic methods have produced some of the smallest CAs, but they



Test ↓ / Var →	A	B	C	D	E	F	G	H	I	J
1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	0	1	0	0	0	0	1
4	1	0	1	1	0	1	0	1	0	0
5	1	0	0	0	1	1	1	0	0	0
6	0	1	1	0	0	1	0	0	1	0
7	0	0	1	0	1	0	1	1	1	0
8	1	1	0	1	0	0	1	0	1	0
9	0	0	0	1	1	1	0	0	1	1
10	0	0	1	1	0	0	1	0	0	1
11	0	1	0	1	1	0	0	1	0	0
12	1	0	0	0	0	0	0	1	1	1
13	0	1	0	0	0	1	1	1	0	1

Figure A. Testing with a three-way covering array. The tests must cover 10 binary variables with two values each, which would take 2^{10}, or 1,024, tests to cover exhaustively. However, testing with the three-way covering array takes only 13 tests (rows). Highlighted numbers correspond to the eight possible three-way combinations in each set of three columns.

UNDERSTANDING COMBINATORIAL TESTING (continued)

can be slow, particularly for large testing problems. Greedy methods are faster than metaheuristic methods and can handle arbitrary test structures, but they might not always produce the smallest CAs. However, greedy methods are also convenient for excluding or including certain combinations because of constraints among parameter values. Typically, they are the most widely used category in tools to support practical problems.

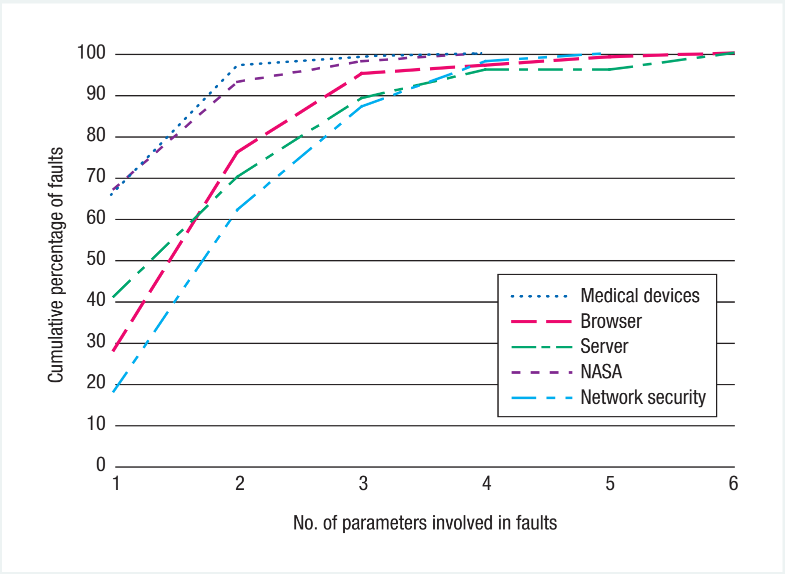


Figure B. Percentage of errors triggered by *t*-way interactions. Most faults tend to be triggered by a single value, with the fewest being triggered by six-way interaction.

EMPIRICAL BASIS

Most failures result from a single parameter or a two-way interaction between parameters, but what about the remaining faults? In a study of actual software faults in a variety of domains,[1] faults followed the Interaction Rule, which says that most faults are triggered by a single value or an interaction between two parameter values, and progressively fewer are triggered by three-, four-, five-, and six-way interactions.[10]

Figure B summarizes the results of this study. In the Web server application (Server), a single value caused roughly 40 percent of the failures, such as a file name exceeding a certain length. Another 30 percent were triggered by the interaction of two parameters. Thus, three or fewer parameters triggered a total of almost 90 percent of the failures in that application. Curves for the other applications have a similar shape, reaching 100 percent detection between four- and six-way interactions.

These results suggest that combinatorial testing, which exercises strong interaction combinations, can be an effective approach to high-integrity software assurance. The key insight in combinatorial testing is this: [1]

If we know from experience that t or fewer variables are involved in failures for a particular application type and we can test all t-way combinations of discrete variable values, then we can have reasonably high confidence in the application.

References

1. D.R. Kuhn, D.R. Wallace, and A. Gallo, "Software Fault Interactions and Implications for Software Testing," *IEEE Trans. Software Eng.*, vol. 30, no. 6, 2004, pp. 418–421.
2. G. Taguchi, *System of Experimental Design: Volumes 1 and 2*, Kraus Int'l, 1987.
3. R.N. Kacker, "Off-Line Quality Control, Parameter Design, and the Taguchi Method," *J. Quality Technology*, vol. 17, 1985, pp. 176–209.
4. M.S. Phadke, *Quality Engineering Using Robust Design*, Prentice Hall, 1989.
5. N. Sloane, "Covering Arrays and Intersecting Codes," *J. Combinatorial Designs*, vol. 1, no. 1, 1993, pp. 51–63.
6. M.B. Cohen, C.J. Colbourn, and A.C.H. Ling, "Constructing Strength Three Covering Arrays with Augmented Annealing," *Discrete Mathematics*, vol. 308, no. 13, 2003, pp. 2709–2722.
7. C.J. Colbourn, "Combinatorial Aspects of Covering Arrays," *Le Matematiche* (Catania), vol. 58, 2004, pp. 121–167.
8. Y. Lei and K.C. Tai, "In-Parameter-Order: A Test Generation Strategy for Pairwise Testing," *Proc. 3rd IEEE Int'l Symp. High-Assurance Systems Eng.*, 1998, pp. 254–261.
9. R. Bryce, C.J. Colbourn, and M.B. Cohen, "A Framework of Greedy Methods for Constructing Interaction Tests," *Proc. IEEE CS 27th Int'l Conf. Software Engineering* (ICSE 05), 2005, pp. 146–155.
10. D. Kuhn, R.N. Kacker, and Y. Lei, *Introduction to Combinatorial Testing*, CRC Press, 2013.

ACTS is a freely downloadable research tool for generating combinatorial *t*-way test suites based on covering arrays.[3] It has many features that made it ideal for the pilot projects:

- Testers can exclude test-setting combinations that are invalid according to user-specified constraints.
- Two test generation modes support building a test suite from scratch or extending an existing suite to save earlier tests.
- Variable-strength test suites are possible. For example, all factors could be covered with strength 2 and a subset of the factors (known to be interrelated) could be covered with higher strength 3.
- ACTS verifies if the user-supplied test suite covers all *t*-way combinations.
- The tool comes with a GUI, a command line interface, and an API.

In addition to tools, Lockheed Martin evaluated application areas in systems, software, hardware, and materials testing—areas that its traditional test processes were already addressing but often without a systematic scientific approach. It was interested in applying combinatorial testing to

- different system configurations, which could have a variety of user, hardware, electronic, and software options;
- software with a range of parameters and variables that might interact to create errors;
- configurations in a flight test scenario, which can have many use conditions and hardware configurations, such as different targets, weapon options, altitudes, flight paths, or sensors for targeting;
- support user interface testing in system software, which can entail many option combinations in menus, user displays, pages, and interactions; and
- test the hardware settings of complex systems, such as switch setting, sensor inputs, signals, and outputs.

Staff quickly suggested other areas for consideration, but the company kept its focus on small initiatives that would help develop a basic understanding of combinatorial testing and tools to support its implementation.

PILOT PROJECTS AND RESULTS

In all, Lockheed Martin evaluated 14 areas from the candidates in the trade study evaluation. From these, it selected eight of those with the best data availability and strongest management support for its pilot projects.

Flight vehicle mission effectiveness

The project staff created six combinatorial test cases in different mission effectiveness (ME) areas and worked with the existing test team using historic ME plans. The study compared combinatorial testing relative to test cases created from a statistical analysis tool. Results indicated no major difference between the combinatorial testing and statistical approach, but the test team felt that combinatorial tools were easier to use.

Flight vehicle engine failure modes

The project team considered failure combinations, failure accommodation, and fault-tolerance combinations under defined constraints. An analysis of historic plans revealed missed test cases and showed that combinatorial testing would yield 30 percent better coverage.

Flight vehicle engine upgrade flight and lab testing

The team generated six combinatorial test cases covering vehicle variations in various flight modes and conditions. The generated cases were statistically equal to existing team test plans.

F-16 ventral fin redesign flight test program

The F-16 ventral fin redesign effort[2] is a case study that demonstrates how combinatorial testing can be applied to design analysis problems, solutions, and evaluation. The team generated a set of combinatorial test studies, data sets, and reports that re-created the original analysis but with combinatorial testing instead of highly experienced expert judgment. The team found that combinatorial testing would have reduced the number of tests by roughly 20 percent.

Electronic warfare system testing

The team generated four combinatorial test cases from existing test plans. The idea was to show the validity of using a combinatorial test tool to analyze word-based test plans. Results demonstrated the feasibility of the rdExpert tool's features to analyze existing test plans while providing improvements by reducing the total number of test cases by 10 percent.

System test flight parameters

Parameters included navigation accuracy, electronic warfare performance, sensor information, and radar detection performance test points within system test flights.

Combinatorial testing revealed the possibility of generating effective test cases with diverse subsystem interactions.

Electromagnetic effects engineering

The team analyzed the existing test plan and generated new tests. They saw no coverage improvement with combinatorial testing, but they felt that the tool might have generated cases faster than was possible with the human analysis and judgment that was part of the existing test plan.

Digital command system testing

The system had 16 commands for performing various file operations in three file systems. Most commands had more than one parameter, and some parameters accepted up to three values. Also, some parameters accepted a continuous range of integer values, so the project team added constraints to limit impossible operations.

The team produced 26 test cases spanning 1,148 test combinations. The tests uncovered several major bugs that had gone undetected through unit testing.

EVALUATION OF PILOT PROJECTS

Beyond using only classic requirements-based testing, the combinatorial testing efforts demonstrated that the methodology had the potential for effective and affordable tests in software, hardware, system, and flight testing. Combinatorial testing was applicable when the test structure involved multiple options or parameters.

Culture change

Combinatorial testing requires some culture change for systems engineering and test teams. The pilot effort showed that organizations must incorporate the method into early test planning as well as provide training and management support. Lockheed Martin's initial estimate is that combinatorial testing and supporting technology can save up to 20 percent in test planning and design costs if used early on in a program and can increase test coverage by 20 to 50 percent.

The tool investigation and pilot studies were generally positive, and Lockheed Martin has already identified possible future efforts. All the combinatorial test tools investigated worked, and results were similar. Each tool had features that would make it viable given certain criteria. For example, if cost was the main issue, open source tools would be the best choice. Some tools had better test-plan analysis features; yet others would work better with distributed teams. Overall Lockheed Martin's tool studies revealed that combinatorial test tools are maturing nicely.

General observations

Staff from Lockheed Martin reviewed project problems and results and documented general observations as positive results or mixed.

Positive results came mainly from recognition of the problems that combinatorial testing revealed. An example is the F-16 project team's discovery that staff cost could have been lower. Many project staff members were generally impressed with the method's ability to enhance and supplement existing test knowledge and abilities.

After the pilot projects ended, several project teams continued using the ACTS tool and found software errors before product release. These positive results became data points for engineers who wanted more evidence of combinatorial testing's viability.

Mixed results centered on attitudes about cost versus improvement gains. Several projects that analyzed existing test plans did find weaknesses in test-case coverage but did not want to add tests because of the resulting increased test cost. After applying combinatorial testing, another project found at least one new error, but felt that combinatorial testing was not a vast improvement. However, they had no similar project for comparison.

Lessons learned

Testing teams tend to have fairly rigid processes, and introducing new ideas to change the testing mindset can prove difficult unless a champion takes on the task of promulgating combinatorial testing.

Because their expectations about improvement were sometimes unrealistic, teams often found it hard to see where and how to apply combinatorial testing to existing test efforts. Some teams found the number of tests had increased, and they had expected combinatorial testing to *reduce* that number. They failed to recognize that the original test set had to increase because it did not provide good coverage in the first place. In addition, the teams did not fully appreciate how combinatorial methods could help identify weaknesses in test sets that are intended to be thorough but are not.

ORGANIZATIONS CAN REAP COMBINATORIAL TESTING'S BENEFITS BY MEASURING *T*-WAY COVERAGE OF EXISTING TESTS AND SUPPLEMENTING AS NEEDED.

Lockheed Martin also learned lessons about the role of training in achieving method buy-in. Teams that expressed interest in using combinatorial testing failed to adopt it because of cost, skill, or time. The company now offers training and tools to help compensate for the lack of follow-through in some of these areas.

Lack of training also contributes to resistance in using tools to support new methods. Because many testers were not trained in combinatorial testing, they saw the supporting tools as foreign to their experience and thus resisted using them. Lockheed Martin now provides online training and familiarization to start overcoming this resistance.

Training is essential for internal technology transfer as well. The company established a website for engineers featuring a basic introduction to combinatorial testing and the Design of Experiments method, webinar materials, and white papers, as well as links to tools, external websites, training, and presentation packages. The company also enhanced online training materials with the knowledge gained from R&D efforts. Staff built small training modules as prototype classes with exercises on combinatorial testing and tool use and made them Web accessible.

INCREASING ACCEPTANCE

Results from the pilot projects suggest ways to broaden the acceptance of combinatorial testing. Approaches to integrating this method and supporting tools into existing practice must provide support materials that include training packages and data on cost and effectiveness. Tools must be easy to use and integrate readily with industry test infrastructures and frameworks, such as allowing an automatic import into model-based test tools.

Perhaps, most important, organizations need to adapt combinatorial testing and supporting tools to fit their existing procedures.

Improved support materials and guidance

To improve combinatorial testing support, Lockheed Martin is continuing to develop training support, including textbooks and classes, case studies, and industry sites with additional information. Further, the addition of combinatorial testing to standards such as ISO/IEC/IEEE 29119 or DoD standards will likely encourage the adoption of math-based methods into software testing from the beginning of a project's life cycle.

Internally, organizations must provide rigorous hands-on training. Lockheed Martin plans to enhance its online forum and provide more such training for staff. It also plans to encourage subject-matter experts to promote combinatorial testing on projects; educate management, and highlight projects that are already using combinatorial testing. In documenting our pilot project experience, we hope to illustrate how the measurement and evaluation of pilot projects can advance software engineering through the increased adoption of scientific test methods.

Adapting to existing procedures

It is not always practical to redesign an organization's testing procedures to use tests based on covering arrays. Testing procedures often develop over time, and employees have extensive experience with a particular approach. Units of the organization may be structured around established, documented test procedures, particularly in organizations that must test according to contractual requirements or standards.

Because much software assurance involves testing applications that have been modified to meet new specifications, the organization is likely to have an extensive library of legacy tests—tests that it can reuse to save time and money. These tests might not be based on covering arrays.

If creating new test suites is not an option, the organization can glean the advantages of combinatorial testing by measuring the combinatorial coverage of existing tests and then supplementing those with additional tests as needed. Building covering arrays for some specified level of *t* is one way to provide *t*-way coverage. However, many large test suites naturally cover a high percentage of *t*-way combinations. For example, tests developed for NASA spacecraft provided better than 90 percent pairwise coverage despite being developed without combinatorial testing tools.[4] If an existing test suite covers almost all *t*-way combinations for an appropriate level of *t*, the suite might be sufficient for the required assurance.

Determining the level of input or configuration-space coverage can also help in understanding the degree of risk that remains after testing. If testing has covered 90 to 100 percent of the relevant state space, the risk is likely to be smaller than it would be after testing that covers a much smaller portion of that space. These considerations led NIST to develop an approach that measures an existing test suite's combinatorial coverage and then automatically extends it to provide the desired coverage level. NIST has developed a sophisticated tool based on this approach, which also allows for constraints among variables.[5]

Early efforts to include combinatorial testing into the Lockheed Martin test culture have been largely positive. Projects within the company have taken advantage of the tools, website, and training. Individual engineers have been supportive and even enthusiastic. Traffic on the combinatorial testing website has grown, and interest in improving the prototype online training materials continues.

Existing projects have been somewhat slow to consider combinatorial testing, most likely because project staff are risk averse and thus wary of new ideas that are not mandated under a contract. Lockheed Martin expects that new contracts and improvement efforts will be more likely to use combinatorial testing.

The company supports such improvement efforts and plans to continue using the method, related test tools, training and websites, and management support. The work we have described reflects early efforts, and Lockheed Martin takes a long-term view of improvement. Engineers look forward to working with industry, government, tool vendors, researchers, and others to promote combinatorial testing. NIST is already incorporating lessons from this cooperative research effort into generic advanced measurement and testing research to promote improved quality, safety, and reliability in software and systems.

ABOUT THE AUTHORS

JON D. HAGAR is an independent software product integrity, verification, and validation testing consultant. While performing the work reported in this article, he was an engineer and manager of software testing at Lockheed Martin. Hagar received an MS in computer science with specialization in software engineering and testing from Colorado State University. He is the author of *Software Test Attacks to Break Mobile and Embedded Devices* (CRC Press, 2013) as well as a contributor to ISO/IEEE standards development. Contact him at embedded@ecentral.com.

D. RICHARD KUHN is a computer scientist in the Computer Security Division at the US National Institute of Standards and Technology (NIST). His research interests include combinatorial testing, empirical data on software failure, and access control systems. Kuhn received an MS in computer science from the University of Maryland College Park, and an MBA from the College of William & Mary. He is a senior member of IEEE. Contact him at kuhn@nist.gov.

RAGHU N. KACKER is a researcher in NIST's Applied and Computational Mathematics Division. His research interests include software testing and the evaluation of the uncertainty in outputs of computational models and physical measurements. Kacker received a PhD in statistics from Iowa State University. He is a Fellow of the American Statistical Association and American Society for Quality. Contact him at raghu.kacker@nist.gov.

THOMAS L. WISSINK is the director of integration, test, and evaluation at Lockheed Martin. His research interests include developing, testing, and managing software-intensive systems. Wissink received a BS in computer science from Florida Atlantic University. He is a Lockheed Martin Senior Fellow and a member of the National Defense Industrial Association. Contact him at tom.wissink@lmco.com.

DISCLAIMER

Identification of commercial products in this article is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products identified are necessarily the best available for the purpose.

REFERENCES

1. C. McQueary, "Using Design of Experiments for Operational Test and Evaluation," memo, Office of the Secretary of Defense, May 2009; www.dote.osd.mil/pub/policies/2009/200905UsingDoEforOTE_MOA.pdf.
2. A.M. Cunningham, J. Hagar, and R.J.J. Holman, "A System Analysis Study Comparing Reverse Engineered Combinatorial Testing to Expert Judgment," *Proc. 5th IEEE Int'l Conf. Software Testing, Verification and Validation* (ICST 12), 2012, pp. 630–635.
3. Y. Lei et al., "IPOG/IPOG-D: Efficient Test Generation for Multi-Way Combinatorial Testing," *Software Testing, Verification, and Reliability*, vol. 18, no. 3, 2008, pp. 125–148.
4. J.R. Maximoff et al., "A Method for Analyzing System State-Space Coverage within a *t*-Wise Testing Framework," *Proc. IEEE Int'l Systems Conf.* (SysCon 10), 2010, pp. 598–603.
5. D.R. Kuhn et al., "Combinatorial Coverage Measurement Concepts and Applications," *Proc. 2nd IEEE Int'l Workshop Combinatorial Testing* (IWTC 13), 2013, pp. 352–361.