

# Design Of Experiment For Software Testing

By [Madhav S. Phadke](#)

When DOE is used for software testing, there is a large amount of savings in testing time and cost. Various users in automotive, telecommunication and defense industries report big productivity improvements to their traditional testing methods. This success is due to two major important factors: 1. Focused attention on the usage of software, and 2. A mathematically sound way to span the entire operating domain with minimum set of test cases.

The job of a software tester is to attempt to break the system in every possible way so that all faults will be detected, which will therefore increase the likelihood of delivering fault-free software to the customer.

Let us set forth a simple scenario for software testing. Consider a function to be tested with four parameters: A, B, C, and D. These parameters could be the arguments of the command line entered from the terminal, the state of an interface, input from a connecting device, or the initial states of internal parameters. Suppose each parameter has three possible levels. This parameter-level table specifies the test domain consisting of 81 possible combinations of the test parameter levels.

Consider the example of testing copy machine functionality for test parameters and levels. Here, parameter A could be number of original papers and levels being 1 paper, 10 papers or 51 papers. Parameter B could be Duplex with levels being one-sided copy or two-sided etc. as shown in Table 1. The software has to perform well for all possible values of these parameters.

Test Parameters	Level 1	Level 2	Level 3
A. Number of Originals	1	10	51
B. Duplex	1 to 1	1 to 2	2 to 2
C. Collating	None	Yes	Stapled
D. Interrupt	None	Panel	Tray

A thorough analysis of the requirements document and understanding of the usage of the software is needed to prepare the parameter-level table and test plans. Knowledge of how the software is written is usually not necessary to prepare this table.

## Orthogonal Array-Based Test Cases

Table 2 shows the test plan using OA (Orthogonal Array) L9. It has nine rows and four columns. The rows correspond to test cases; the columns correspond to the test parameters. Thus, the first test case comprises Level 1 for each parameter, i.e., it represents the combination A1, B1, C1, D1. The second test case comprises combination A1, B2, C2, D2, etc.

Test Number	No. of Originals	Duplex	Collating	Interrupt
1	1	1 to 1	None	None
2	1	1 to 2	Yes	Panel
3	1	2 to 2	Stapled	Tray
4	10	1 to 1	Yes	Tray

5	10	1 to 2	Stapled	None
6	10	2 to 2	None	Panel
7	51	1 to 1	Stapled	Panel
8	51	1 to 2	None	Tray
9	51	2 to 2	Yes	None

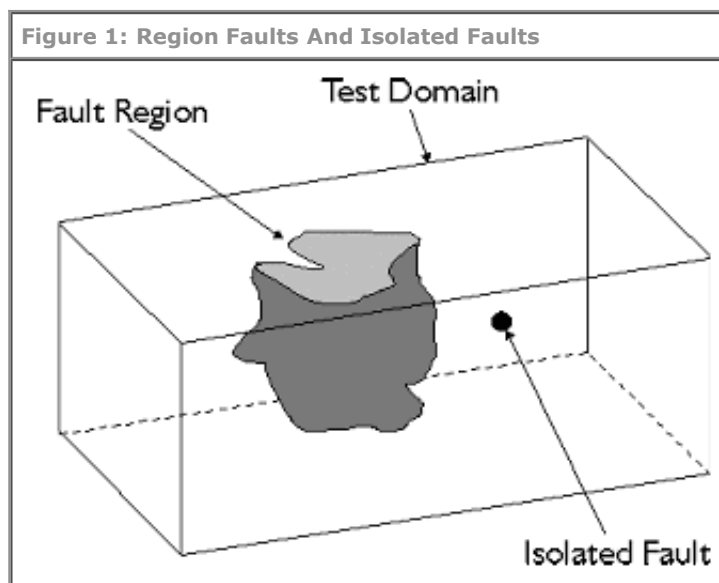
An orthogonal array has the balancing property that, for each pair of columns, all parameter-level combinations occur an equal number of times. In OA L9, there are nine parameter-level combinations for each pair of columns, and each combination occurs once. Taguchi [2] and Madhav S. Phadke [1] provide a comprehensive discussion of orthogonal arrays and their selection for specific applications.

By conducting the nine tests indicated by L9, we can accomplish the following:

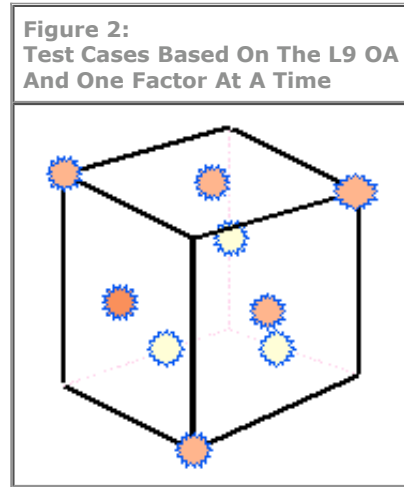
1. Detect and isolate all single-mode faults. A single-mode fault is a consistent problem with any level of any single parameter. For example, if all cases of factor A at Level A1 cause error condition, it is a single-mode failure. In this example, tests 1, 2, and 3 will show errors. By analyzing the information about which tests show error, one can identify which factor level causes the fault. In this example, by noting that tests 1, 2, and 3 cause an error, one can isolate A1 as the source of the fault. Such an isolation of fault is important to fix the fault.
2. Detect all double-mode faults. If there exists a consistent problem when specific levels of two parameters occur together, it is called a double-mode fault. Indeed, a double-mode fault is an indication of pairwise incompatibility or harmful interactions between two test parameters.
3. Multimode faults. OAs of strength 2 can assure the detection of only the single- and double-mode faults. However, many multimode faults are also detected by these tests. This can be understood by studying the interaction tables and the geometric view of the test cases presented in the next section.

### Geometric View of Test Cases

Software faults can be divided into two categories: region faults and isolated faults, as shown in Figure 1. Faults that occur for only one specific combination of test parameter levels, such as by a data-entry error, are called isolated faults. Assurance against isolated faults is not possible without testing all combinations. However, when the logic is faulty, there is a tendency for a region of the test domain to exhibit malfunction. Such faults are called region faults. Single-mode and double-mode faults are special cases of region faults. Orthogonal Array based testing is highly effective for the detection of region faults with a relatively small number of tests.



A geometric picture of the test cases can provide an insight into the benefits of OA-based test cases in the detection of region faults. To facilitate geometric visualization, take the example of software with only three test parameters A, B, and C. The test domain is cube-shaped and consists of 27 lattice points. Test cases based on the OA L9 and those based on the one-factor-at-a-time method are graphically displayed in Figure 2. The OA-based test cases are dispersed uniformly throughout the test domain.



### A Case Study

A comprehensive case study of orthogonal array based testing was published by AT&T in 1992 [3]. The study detailed regression testing of a PC (IBM format) and local area network-based electronic mail software called StarMail. Originally, a test plan was prepared that consisted of 1,500 tests that would take 18 weeks to conduct. However, because of development delays, the testing time had to be cut short to only eight weeks. The lead test engineer had, therefore, prepared an alternate plan involving 1,000 tests that would take two people eight weeks. But he was unsure about the quality of testing, i.e., its ability to detect all faults. To alleviate these problems, a test plan was prepared using the orthogonal array based Robust Testing method.

The Robust Testing method began with the division of the functions of the StarMail software into 18 categories, test parameters and levels for each category were found, and an OA to select test cases for each category was used. For example, for the copy/move function, the selected OA L27 was used to ensure all pairwise combinations of these parameters. Different parameter tables were prepared for remaining 17 categories.

In all, only 422 tests were needed to test the software. These tests identified 41 faults in the software, which were fixed, and the software was released. Two years of operation in the field generated no faults within the scope of testing, which indicated that relative to the scenarios encountered, the test plan was 100 percent effective in identifying faults. Had AT&T run the alternate test plan that involved 1,000 tests, the lead tester estimated they may have found only 32 of the 41 faults. Compared to the original testing plan, Robust Testing required 50 percent less testing effort, identified 28 percent more faults, and was more productive (number of faults detected per tester week) by a factor of 2.6.

### Regression Testing

Orthogonal array based testing is very useful in regression testing. After fixing the faults, one can easily prepare a new test plan using array rotation technique. A new regression test plan is depicted in Table 3, which is different from the test plan shown in Table 2.

Test Number	Duplex	Collating	Interrupt	No. of Originals
1	1 to 1	None	None	1
2	1 to 1	Yes	Panel	10
3	1 to 1	Stapled	Tray	51
4	1 to 2	None	Panel	51

5	1 to 2	Yes	Tray	1
6	1 to 2	Stapled	None	10
7	2 to 2	None	Tray	10
8	2 to 2	Yes	None	51
9	2 to 2	Stapled	Panel	1

A lot of time and cost savings are available during this phase.

### Conclusion

Use of orthogonal array based testing has demonstrated to produce superior test plans that improve testing productivity by a factor of 2. This method is found effective in testing the incremental work done in all stages of software development. These stages include writing requirements, selecting architecture, designing the system (functional breakdown), unit testing, platform testing, integration testing, prototype testing, and system testing.

### About The Author

Dr. Phadke pioneered the application and development of the Taguchi Method / Robust Design method in USA and is a recipient of the Taguchi Award, 1985. He has worked closely with Dr. Genichi Taguchi since 1980, and they have co-authored several articles, advancing the field of robust design.

Dr. Phadke is the author of the first engineering text book in English on Taguchi Method, [Quality Engineering Using Robust Design](#), published by Prentice Hall, 1989. The book, which is considered the most authoritative book in English on Robust Design, has been translated in three languages, German, Chinese, and Korean.

Dr. Phadke can be reached at [Madhav@PhadkeAssociates.com](mailto:Madhav@PhadkeAssociates.com), <http://www.PhadkeAssociates.com>.

### References

1. Phadke, M.S. "Quality Engineering Using Robust Design" Prentice Hall, Englewood Cliff, NJ. November 1989.
2. Taguchi, Genichi. "System of Experimental Design" Edited by Don Clausing. New York: UNIPUB/Krass International Publications, Volume 1 & 2, 1987.
3. Browmlie, Robert; James Prowse; Madhav S. Phadke. "Robust Testing of AT&T PMX/StarMAIL using OATS" AT&T Technical Journal, Volume 71, No. 3 May/June 1992, pp 41-47.
4. Phadke, M.S. "Planning Efficient Software Tests" CrossTalk Journal of Defense Software Engineering, October 1997, pp 11-15.

[Terms of Service](#). Copyright © 2000-2005 iSixSigma LLC. All rights reserved. Visit us at <http://www.iSixSigma.com>.

[Return To Previous Page](#)