# Utilizing Design of Experiments to Reduce IT System Testing Cost

**Kedar M. Phadke, Phadke Associates, Inc.**
**Madhav S. Phadke, Phadke Associates, Inc.**

**Abstract.** Software and system testing cost the commercial and defense industry hundreds of millions of dollars annually. In addition, conducting each set of tests takes multiple man-months, delaying time to market of key technologies. In this current economic environment, organizations are looking for ways to reduce the cost of testing and time to market while ensuring that defects are not passed on to the customer. At the same time, organizations are very reluctant to change their standard testing processes due to the heavy cost of field failures, regulatory concerns, and risk-averse culture.

This paper describes a comparative study undertaken to assess the benefits of using Orthogonal Arrays (OA) for generating test plans in IT systems in the financial services industry. The formal process used for the comparative study consisted of enlisting the support of senior management and conducting multiple side-by-side pilots to compare the cost and risk of OA based testing versus the Business as Usual (BAU) testing practices. Our customers ran 20 side-by-side studies to evaluate the effectiveness of OA based testing and realized an average reduction in total test effort by 41%. In addition, all defects detected by the BAU process were detected by the OA based testing process. Further, in 40% of the cases, the OA based testing process found more defects. The cost and schedule savings translated to tens of millions of dollars in labor and schedule.

The paper also discusses the pros and cons of OA testing versus other testing approaches, namely, pairwise testing, N-Way testing, and classical Design of Experiments (DoE).

Utilizing OAs for system and software testing will significantly reduce cost, schedule and risk. For the aerospace and defense industries, OA testing will help address the current environment of tighter budgets and schedules while ensuring end users promised performance. This process is being adopted by several top tier defense and aerospace system developers for software and system testing and its applications have demonstrated significant reduction in both program cost and risk.

## Overview of OA Testing Process

OAs are a mathematical tool that has been studied and utilized for centuries by mathematicians, scientists, and engineers for a variety of applications [1,2,3,4,5,6,11,12]. The most well known, Leonhard Euler, utilized OAs (also called Latin Squares) to cleverly arrange multiple ranks of military officers and for war games. One of the co-authors, Madhav Phadke, introduced the use of OAs for software testing while at AT&T Bell Laboratories in the 1980s, achieving great success for network and telecommunications system testing [7].

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 as given in Table 1. This parameter-level table specifies the test domain consisting of 81 possible combinations of the test parameter levels. (In the Robust Design literature, "factor" is often used in place of "parameter.")

| Test Parameter | Level 1 | Level 2 | Level 3 |
|---|---|---|---|
| A | $A_1$ | $A_2$ | $A_3$ |
| B | $B_1$ | $B_2$ | $B_3$ |
| C | $C_1$ | $C_2$ | $C_3$ |
| D | $D_1$ | $D_2$ | $D_3$ |

*Table 1: Test Parameters and Levels*

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.

Table 2 shows the OA 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 corresponds to the combination A1, B2, C2, D2, etc.

| Test Number | Test Parameter A | Test Parameter B | Test Parameter C | Test Parameter D |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 2 | 2 |
| 3 | 1 | 3 | 3 | 3 |
| 4 | 2 | 1 | 2 | 3 |
| 5 | 2 | 2 | 3 | 1 |
| 6 | 2 | 3 | 1 | 2 |
| 7 | 3 | 1 | 3 | 2 |
| 8 | 3 | 2 | 1 | 3 |
| 9 | 3 | 3 | 2 | 1 |

An OA 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 [8] and Madhav S. Phadke [9] provide a comprehensive discussion of OAs and their selection for specific applications. By conducting the nine tests indicated by L9, we can accomplish the following:

• ***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 fault. 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.

• ***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.

• ***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 by virtue of the fact that OA-based tests are uniformly distributed in the test domain.

Real software testing problems tend to have dozens of parameters with two to 15 potential values per test parameter, thus manually determining appropriate OAs is a challenge for most software test professionals. Commercial tools for generating OAs for specific problems can be very helpful for this task. The cases studies in this paper were all conducted using a commercial software tool, rdExpert™ Test Suite, for OA generation [10].

## Enterprise Mortgage IT System Validation

Several case studies have been conducted to validate the OA testing process for IT systems within the financial services industry. This section details one specific case study for a mortgage bank. The next section provides a summary of 20 similar studies conducted at 10 large financial services firms.

A major mortgage bank was revamping its enterprise IT system to better meet customer needs. The bank has several business processes geared towards different stages of mortgage processing. In the past, each business process had its own software systems, and the hand-over between processes were made manually. This caused delays in servicing customers and resulted in loss of business. For example, the bank had a system for accepting mortgage applications and a separate system for underwriting. This meant that once an application was accepted, it had to be manually input in the underwriting system, processed, and then a quote was manually input back into the application portal for the customer. In the new environment of customers demanding immediate feedback on mortgage applications, this was not fast enough. To address this customer need, the mortgage bank was developing an enterprise integration platform to automatically transmit data between the disparate systems and make the end-to-end process more efficient.

The bank had hired an outsourced provider to develop and test the integrated system. The scope of the system included all major business processes such as 1) loan application and approval, 2) payment acceptance and management, 3) data storage and access, and 4) corporate reporting and governance. At the time the bank began considering OA for testing, the outsourced team was seven months behind on delivery and 20% over budget.

The bank management culture was by tradition risk-averse so the teams were hesitant to change the current BAU testing approach. This risk-averse culture, coupled with the severe consequences of field failure and regulatory rules, made it even more difficult to change the process. To alleviate these concerns and at the same time assess the benefits of using OA for testing, the management decided to fund parallel teams to conduct testing of the application. One team would utilize the traditional BAU approach and the other team would utilize the OA testing approach. Both teams were tasked to complete the entire end-to-end process, including test case design, test scripting, test execution, defect analysis, and root cause identification. After completing the process, management would be able to evaluate the effects of using OA compared to BAU on cost, risk, and schedule.

The BAU process was an industry standard process and the key steps were as follows:

• Understand the most-likely customers.
• Understand the most-likely paths in processing customers' mortgage applications.
• Create several test scenarios that ensure all top-level requirements are covered, with majority focused on most-likely circumstances.
• Include crisis scenarios.

Table 3 provides a simplified view of the systems integration test planning scenarios. The highlighted values were considered most likely from a customer perspective.

| Application | Loan Type | Credit Verification | Payment | Payment Amount | Customer Data Access | Datacenter Status |
|---|---|---|---|---|---|---|
| Online | Home Equity | Experian | Check through mail | Correct amount | Online reports | All online |
| Phone | Non-Traditional | Equifax | Bank transfer via phone | Underpayment | Mailed reports | 1 Datacenter offline (Routine service) |
| Retail Center | Jumbo 1 | Transunion | Debit card via phone | Overpayment | Online and mailed reports | 2 Datacenters offiline (Critical) |
| Mail | Jumbo 2 | Internal* | Bank transfer via online portal | Permature repayment (Termination) | | |
| Partner Broker | Traditional | | Debit card via online portal | Final payment (Termination) | | |
| | | | Cash at retail center | | | |
| | | | Check at retail center | | | |
| | | | Third party transfer | | | |

*Table 3: Simplified View of Systems Integration Scenarios*

Once the team had determined the most likely customer scenarios and paths, they would generate test cases to validate those specific situations. They would then add variations to the scenarios to include less likely customer scenarios, crisis situations, and other test cases suggested by experts. The team utilizing the BAU test process generated 188 test scenarios to validate the enterprise system.

The OA team utilized a different approach for selecting test scenarios and the key steps were as follows:

- Understand the requirements domain (determine the parameter-level table).
- Peer review the test parameters and levels.
- Generate test conditions based on OA.
- Prioritize the test plan for most likely and most important customer scenarios.

Instead of working on the most likely scenarios up front, the OA team first compiled a thorough summary of the key parameters and levels. Table 4 in the appendix provides an abridged summary of the test parameters and levels. Please note that some details have been changed to preserve client confidentiality. Prioritization of the test was completed at the end, just before execution. The OA team generated 81 test scenarios to validate the enterprise system. Both teams generated test scripts, executed the test plans, and evaluated defects. Table 5 provides a summary of the results.

The side-by-side comparison clearly shows that the OA approach detected all the unique defects detected by the BAU process. Thus, both methods have the same effectiveness. The OA approach reduced the test planning effort from 480 hours to 145 hours, a 70% reduction. The test execution effort is the sum total of the effort needed for the environment set up, running the tests, and defect analysis. The BAU approach required 1,670 hours for test execution while the OA approach needed only 890 hours, which represents a 47% saving. At the average loaded hourly cost of $72, the cost for the BAU approach was $154,800 whereas the cost of the OA approach was only $74,520, representing a 52% cost reduction.

## Side-by-Side Studies at Multiple Financial Services Firms

Similar to the mortgage bank case study described above, 20 complete side-by-side studies were conducted at 10 large financial services firms in the U.S. and Europe.

| Test Plan | No of Tests | Test Planning Effort (hrs) | Test Execution Effort (hrs) | Unique Defects Found |
|---|---|---|---|---|
| **BAU (Business as Usual)** | 188 | 480 | 1670 | 12 |
| **OA (using rdExpert™ Software)** | 81 | 145 | 890 | 12 |
| **Savings Comparison** | **107** | **335** | **780** | **Same Defect Coverage** |

*Table 5: Summary of Results: Enterprise Application Validation*

| Factor Name | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | Level 6 | Level 7 | Level 8 |
|---|---|---|---|---|---|---|---|---|
| **Application** | Online | Partner Broker | Phone | Mail | Retail Center | | | |
| **Applicant History** | New Customer | Previously Approved but No Loan Issued | Previously Denied | Previous Customer | | | | |
| **Loan Type** | Jumbo 1 | Traditional | Jumbo 2 | Non-Traditional | Home Equity | | | |
| **Credit Verification** | Equifax | Experian | Internal* | Transunion | | | | |
| **Title Accreditation** | Title Partner 2 | Title Partner 3 | Title Partner 1 | | | | | |
| **Appraisal Partner** | Appraisal Partner 1 | Not Necessary (Override) | Appraisal Partner 2 | | | | | |
| **Payment** | Bank Transfer by Online Portal | Check through Mail | Debit Card by Online Portal | Cash at Retail Center | Check at Retail Center | Third Party Transfer | Debit Card by Phone | Bank Transfer by Phone |
| **Payment Amount** | Correct Amount | Premature Repayment (Termination) | Final Payment (Termination) | Overpayment | Underpayment | | | |
| **Customer Data Access** | Mailed Reports | Online and Mailed Reports | Online Reports | | | | | |
| **Broker Comissions** | Bank Transfer | Check through mail | | | | | | |
| **Datacenter Status** | All Online | 2 Datacenters Offline (Critical) | 1 Datacenter Offline (routine service) | | | | | |
| **Security Status** | No Issues | Major Security Issues (Critical) | Small Security Issues | | | | | |
| **Network Status** | No Issues | Portions of Network Offline (routine service) | | | | | | |
| **Business Reports** | Business Report 2 | Business Report 1 | Tax/Accounting Report 1 | Business Report 4 | Tax/Accounting Report 2 | Business Report 3 | | |
| **Governance** | Independent Private Entity | Government Entity | | | | | | |

*Table 4: Abridged List of Test Parameters and Values for Enterprise System Validation*

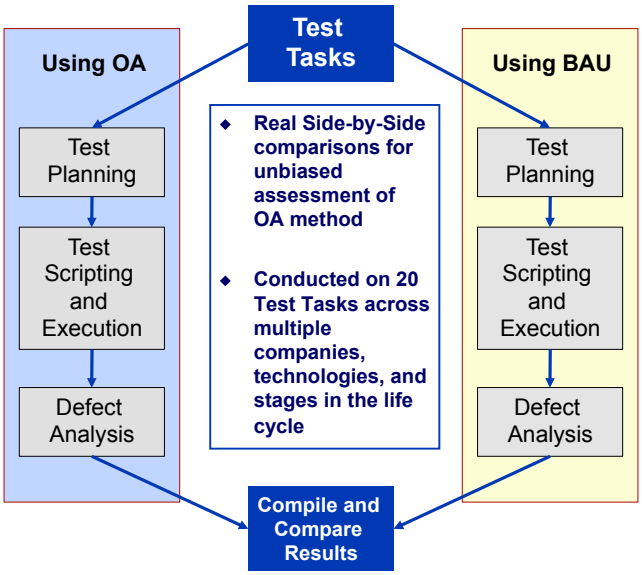
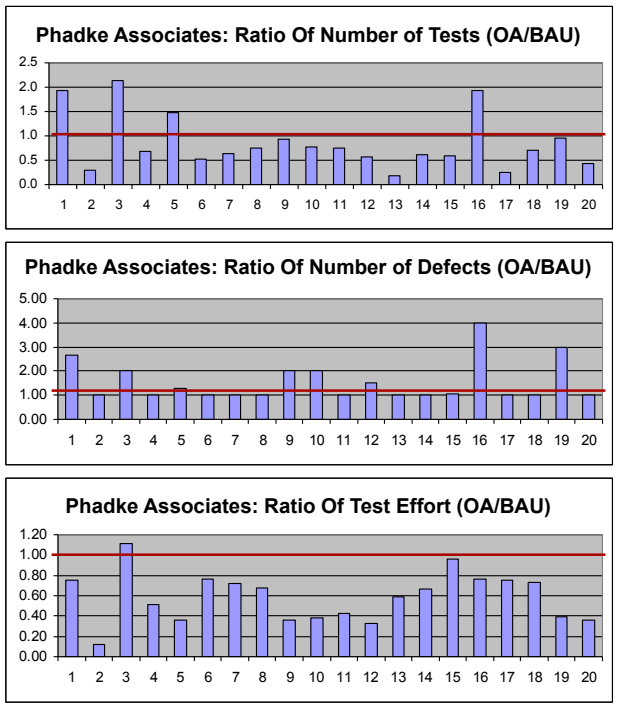



*Figure 1: Side-by-Side Process*

Four of the firms were investment banks, three were health/auto/property/liability insurance companies, one was a life insurance company, and two were retail/mortgage banks. All of the case studies included parallel teams so that management could get truly unbiased information to compare their BAU process versus the OA testing process. Figure 1 shows a flow diagram for the side-by-side process utilized by the companies. Note that the process is the end-to-end testing process including test planning, scripting, execution, and defect analysis.




*Graph 1: Result of Side-by-Side Studies*

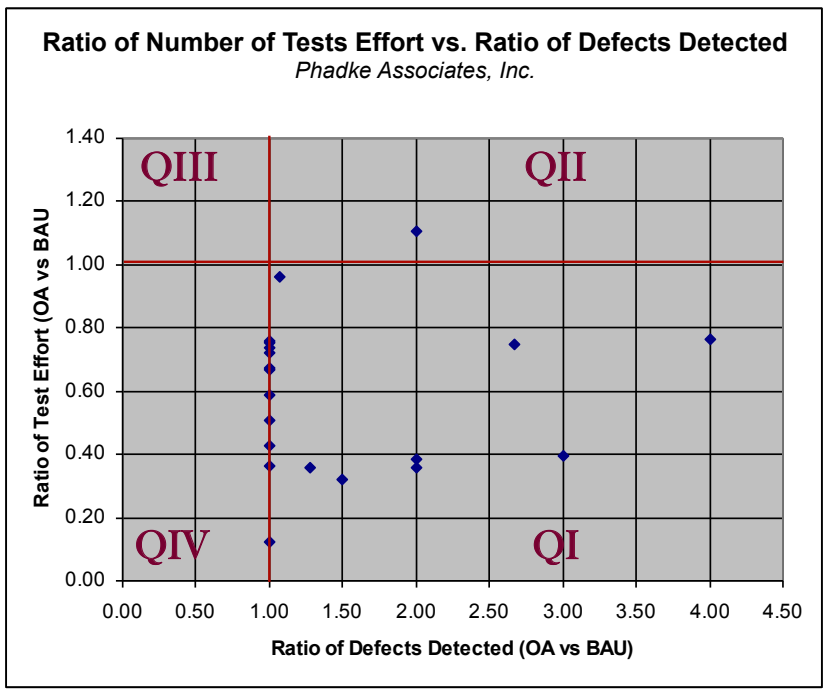
The objective of the side-by-side studies was to ascertain that, 1) OA tests do not increase risk (do not miss defects versus BAU), and 2) OA testing process does not increase total test effort. Graph 1 displays the ratios of test cases, defects detected, and total test effort for all 20 case studies.

The key findings for the management teams were that:

- All unique defects found by BAU were detected by OA testing process in each of the 20 cases.
- In 40% of the cases, OA tests detected more defects. Thus, in many cases, OA tests provided more risk reduction than BAU.
- Total test effort was reduced by 41% on average. This was a saving of tens of millions of dollars.
- In the four cases where the number of test cases or test effort increased, more defects were detected, so time was utilized productively.

In addition, the ratio of test effort (OA/BAU) is graphed versus the ratio of defects detected (OA/BAU) to further compare the effectiveness of OA testing versus BAU at the 10 Financial Institutions. Graph 2 displays the ratio of effort versus the ratio of defects detected. There are eight points in Quadrant I. These points represent the cases where the test effort of OA is less than or equal to BAU and the number of defects is greater than the number detected by BAU. This is the most desired quadrant to be in. Eleven points are the line bordering Quadrants I and IV. These points represent cases where both OA and BAU find the same number of faults. However, for these points the test effort for OA is less than the test effort for BAU. One point lies in Quadrant II. For this case, the OA required more test effort and OA found more defects. In other words, there was more test effort but the team found more defects. There are no test points in Quadrant III and Quadrant IV that are the least desirable quadrants to be in.

Thus the side-by-side case studies clearly demonstrated that the OA testing process is effective for significantly reducing the test effort and also simultaneously reducing the risk for IT testing in the financial services industry.




*Graph 2: Ratio of Test Effort vs. Ratio of Defects*

## Command and Control System

Similar to the financial services industry, the defense and aerospace industries are also facing significant pressure to reduce program cost and schedule. The OA testing process has been successfully applied by several teams in both government and industry. One of the more pressing challenges is effective testing of complex software intensive systems. The Defense Information Standards Agency (DISA) conducted a retrospective pilot of the OA testing process for one of their key modules of a software-based command and control system. The module was designed to retrieve and process data from multiple data sources and display the data in a composite picture. The data sources included human intelligence, measurement and signature intelligence, signals intelligence, Blue/Red Force data, friendly and hostile data from air, ground and sea, and several other sources. The contractor had developed a test plan to validate the module utilizing their BAU practices. Utilizing the OA testing process with the assistance of Phadke Associates, DISA was able to reduce the test plan size by more than 50% and estimated that the test planning effort could be reduced from 24 staff weeks to one staff week. The estimated savings of the reduced test plan and staff savings was $377,000. In addition, the analysis demonstrated that the original contractor test plan had over 340 test gaps and all gaps were eliminated by the more efficient OA testing process. This retrospective pilot demonstrated significant capability of the OA testing process to reduce test cost, risk, and schedule for defense software systems.

## Comparison With Other Test Planning Methods

During the piloting, the teams also compared OA testing with other common test planning methods, namely pairwise, N-Way, and classical DoE. Based on their findings they decided to conduct side-by-side testing pilots using only the OA testing process. Table 6 lists a summary of pros and cons identified for each method versus OA testing.

The teams realized that pairwise testing, a method to ensure that each pair is tested at least once, had the potential to reduce the number of test cases versus OA in some instances; however, the additional cost of defect analysis outweighed the potential reduction in execution cost. Since the test cases created by the pairwise method can be unbalanced, it requires significantly more time to isolate the root causes of defects. In addition, they found it challenging to effectively assess performance in terms of the statistical properties like mean and variance. In fact, to effectively conduct analysis of faults and results, the teams found they had to run several additional test cases. OAs distribute test cases uniformly in the multidimensional test domain [11], whereas pairwise test cases tend to be sparser in some regions than other regions. Consequently, pairwise tests can have less ability to detect faults compared to OAs.

N-Way is a test planning method that ensures that each N-Way combination of parameters is tested at least once. For example, users could specify all three-way combinations, in which case each triplet would be tested at least once in the test plan. One of the first challenges the teams discovered when examining the N-Way testing method was that the number of test cases, even for triplets, was significantly larger than their current BAU process, thus implying a significantly increased test execution cost. Also, similar to pairwise testing, the N-Way test cases are unbalanced and require significantly more time and effort to conduct defect analysis and assess performance. Due to these cost and schedule increases, this method was deemed financially prohibitive. Proponents of N-Way testing site the need to identify rare high order defects, such as five-way or six-way defects. Upon deeper analysis, teams have realized that the preferred approach to address this need is to use hierarchical test plans based on utilizing the broad knowledge of the system (or system-of-systems) architecture. This approach is more effective for detecting high order defects and also significantly more economical compared to the five-way or six-way test plan.

| Test Method | Pros | Cons |
|---|---|---|
| **Business as Usual** | Can be effective and efficient with highly skilled gurus and lots of time. (These are both rare commodities!) | Test plan effectiveness highly dependant on the individual. No consistency across organizations. |
| **Pairwise (not OA)** | Sometimes fewer tests than OA | Unbalanced test cases so debugging is challenging and performance assessments for continuous outputs even more challenging. Costs for test data analysis are much larger. |
| **N-Way (Greater than 2-way)** | Generally better coverage than OA and Pairwise | Significantly more tests so not affordable in today's economic environment. The cost is almost always much more than Business as Usual. Tests are unbalanced so same debugging and performance assessment challenges as Pairwise. |
| **Classical Design of Experiments** | Geared towards statistical modeling | Requires significant amount of staff training and expert guidance. Very difficult to cost effectively implement on a broad scale. Doesn't effectively address the multi-level designs that are necessary for system and software testing. |

*Table 6: Comparing other Test Planning Methods vs. OA testing*

Classical DoE is primarily aimed at model building that is not the objective for a majority of testing tasks, especially when it comes to software and IT testing. The methodology often emphasizes Resolution IV designs with repetitions that result in significantly more tests, thus making the method financially unaffordable, similar to N-Way testing. To combat the financial concerns, practitioners of classical DoE often recommend reducing the number of factors or restricting the number of levels of each factor; however, this technique increases the risk of missing faults and adds significantly more to the downstream program cost and risk. This is particularly challenging for software and IT testing problems that involve mixed level designs with numerous factors having more than two levels (often many more levels). For example, if you have five data types for a particular test parameter, you will have to restrict your test to only two of those data types. Another key challenge is the classical DoE concept of repetitions that are necessary for building confidence in the statistical models. For software and IT systems, repetitions add significant cost but very little additional technical information.

## Conclusion

The advantage of utilizing OAs for testing was demonstrated through 20 real end-to-end case studies where the OA process was run in parallel with the BAU process for IT testing at 10 large financial services institutions. OA-based testing resulted in a 41% reduction in total test effort (labor hours) and in all 20 cases, all defects detected by the BAU process were detected by the OA process. In 40% of the cases, the OA based testing process found more defects. The cost and schedule savings for these cases translated to tens of millions of dollars in labor and schedule.

The technical and managerial challenges for software and system testing in the defense and aerospace industry parallel those in the financial services industry in both scale and pressing need for "defect free" system delivery. Similar to the financial services industry, several defense and aerospace companies have piloted and are adopting the OA testing process and the rdExpert Test Suite Software. The results so far show that OA testing will help defense and aerospace industries meet the current challenge of tighter budgets and schedules while confidently delivering the end users promised performance.

## Disclaimer



## ABOUT THE AUTHORS

**Kedar M. Phadke** is Vice President of Phadke Associates, a global consultancy and software company specializing in statistical tools for improving testing and design productivity. Kedar has led numerous deployments for improving test and design effectiveness. He has a MS in Statistics, MS in Management, and a BS in Economics from the Wharton School, University of Pennsylvania.

**Kedar M. Phadke**
**Vice President**
**Phadke Associates, Inc.**
**1 Shawnee Court**
**Colts Neck, NJ 07722**
**E-mail: kedar@phadkeassociates.com**

**Dr. Madhav S. Phadke** is the Founder and President of Phadke Associates, Inc. He is an ASQ Fellow and the author of the first engineering textbook on Robust Design Methods in the U.S., "Quality Engineering Using Robust Design". He holds a Ph.D. in Mechanical Engineering and MS in Statistics from the University of Wisconsin - Madison, MS in Aerospace Engineering from the University of Rochester, and a BTech in Mechanical Engineering from the Indian Institute of Technology - Mumbai. Prior to founding Phadke Associates, Dr. Phadke was a manager in AT&T Bell Labs, a visiting scientist at the IBM Watson Research Center, and a Research Associate at the Army Math Research Center.

**Madhav S. Phadke, Ph.D.**
**President**
**Phadke Associates, Inc**
**1 Shawnee Court**
**Colts Neck, NJ 07722**
**E-mail: madhav@phadkeassociates.com**

## REFERENCES

1. Addelman, S., "Orthogonal Main Effect Plans for Asymmetrical Factorial Experiments," Technometrics, Vol. 4, 1962, pp. 21-46.
2. Kempthorne, O., The Design and Analysis of Experiments, Robert E. Krieger Publishing, New York, 1979.
3. Plackett, R.L. and J.P. Burman, "The Design of Optimal Multifunctional Experiments," Biometrika, Vol. 33, pp. 305-325.
4. Seiden, E., "On the Problem of Construction of Orthogonal Arrays," Annals of Mathematical Statistics, Vol. 25, 1954, pp. 151-156.
5. Rao, C.R., "Factorial Experiments Derivable from Combinatorial Arrangements of Arrays," Journal of Royal Statistical Society, Series B, Vol. 9, 1947, pp. 128-139.
6. Raghavrao, D., Construction of Combinatorial Problems in Design Experiments, John Wiley and Sons, New York, 1971.
7. Brownlie, Robert, James Prowse, and Madhav S. Phadke, "Robust Testing of AT&T PMX/StarMAIL Using OATS," AT&T Technical Journal, Vol. 71. No. 3, May/June 1992, pp. 41- 47.
8. Taguchi, Genichi, System of Experimental Design, Don Clausing, ed., UNIPUB/Kraus International Publications, New York, Vols.1, 2, 1987.
9. Phadke, Madhav S., Quality Engineering Using Robust Design, Prentice Hall, Englewood Cliffs, N.J., 1989.
10. rdExpert™ Test Suite Software. Published and distributed by Phadke Associates, Inc. <http://www.phadkeassociates.com>.
11. Phadke, Madhav S., "Planning Efficient Software Tests" Crosstalk: Journal of Defense Software Engineering, Published by the Software Technology Support Center, October 1997.
12. Phadke, Madhav S., "Robust Testing: A Process for Efficient Fault Detection and Isolation", Aerospace Testing Seminar, 2006.