Surviving the Top Ten Challenges of Software Test Automation

By Randall W. Rice, CSQA, CSTE
Rice Consulting Solutions, LLC

Abstract

For the past six years I have been surveying training and conference audiences with the question, "How many of your organizations own some type of automated capture/playback test tool?" Capture/playback tools capture or record the actions performed during a test session into software–like scripts that can be replayed against the same or updated version of software. The idea is to be able to repeat two or more tests identically and compare the results. A difference in test results may indicate the presence of a regression defect. Capture/playback tools own the largest market share of any test tool category and it is that category of tools that is the focus of this article.

In answering the question I pose to them, typically 80 to 90% of the audience will raise a hand to indicate their organization owns some type of automated test tool. The next question I ask is "How many of you who just raised your hand would consider automated test tools an integral part of your testing effort?" For that question, the typical response is about 10 to 20% of the first group. The main conclusion is that there is a large gap between the people who own automated test tools and the people who actually realize the benefits from test automation. Another observation is that the findings of this survey have not changed significantly over the past six years.

The observation that automated test tools account for a lot of "shelfware" is not new or insightful. In fact, the shelfware problem is shared with many types of software tools. What is useful is to examine the trouble spots in test automation, deal with them proactively, and perhaps mitigate the risks of tool abandonment.

Introduction

When faced with the dynamic world of automated test tools, many organizations make the best choice they can, try to make the tool work in their environment, and hope the rest of the organization will embrace the tool as well.

The purpose of this article is to outline the major challenges that I see most often in organizations struggling to make effective test automation a reality. The earlier these challenges are understood, the better prepared an organization will be to deal with them. Plus, the more an organization understands the issues of test automation, the less risk is seen in acquiring a tool in terms of time and money.

These challenges will be presented from the least to highest impact on the overall test automation effort. It is also important to understand that even organizations who have developed core competencies in working with test automation struggle at times with these challenges.

Although this article focuses on the challenges, there are many benefits that can be realized from test automation. To place the discussion in context, it is helpful to understand that tools are part of an effective test approach, but not the entire approach. You also need a workable process performed by trained and motivated people in a controlled environment. (Figure 1).

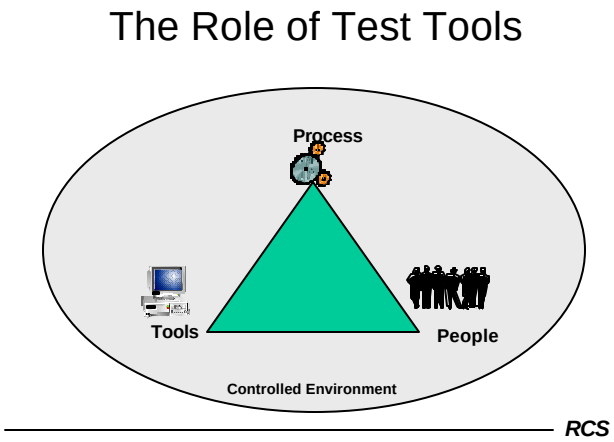


Figure 1

Challenge #10 - Lack of tool availability

This can be seen from two perspectives:

1) The tool is available, but you can't get the funding for it, or
2) There doesn't seem to be a tool on the market that does what you need or fits in your environment.

The first issue is based in your own management's understanding of testing and the priority given to testing in terms of funding. While the cost of automated test tools is high as compared to more common software packages, such as office automation products and software development suites, the anticipated value of the tools is seen in reduced time and greater precision in testing. The promise of test automation is increased productivity and accuracy, which is where the business case must be made. The cost of a single defect in most organizations can offset the price of one or more tool licenses. It's all a matter of where management chooses to spend the money – in defect detection or post-production rework, which is many times more costly than early defect detection.

The issue of not having tools available in a particular environment is more troublesome. Although there is now automated tool support in most environments, this does not mean the support in every environment is great. In older environments, tool support is very limited.

Solution Strategies

If funding is the issue:

- Measure the current cost of defects, especially in post-implementation rework. Use this information to help build a case for faster and more reliable testing using tools.
- Show the value of automated test tools for other groups besides testers, such as the value of developers using the tools.
- Consider the value and feasibility of low-cost or free tools. You can find low cost tools at www.riceconsulting.com/cheaptools.htm.

If getting a good technical fit is the issue:

- Network with other testers to find information about lesser-known test tools. Online Quality Assurance (QA) forums are also good places to query people about tools in lesser-supported environments.

- Try to find tools that will work between platforms. This will likely require the use of PC-based emulators as opposed to host-based tools.

- Investigate the possibility of building your own tools, or at least achieving a level of test automation using common scripting commands and comparison programs.

Challenge #9 - Lack of tool compatibility and interoperability

The issue of tool incompatibility and lack of interoperability is seen in organizations that have diverse technologies and applications. The desire is to be able to automate tests that bridge applications and platforms. This is a big challenge because most automated test tools have proprietary scripting languages and approaches, and for the most part, competing vendor tools do not interoperate.

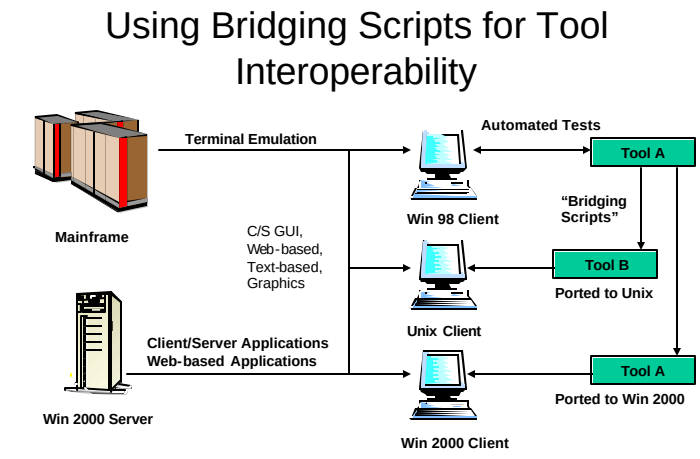
A related challenge is to conduct identical tests on a number of different platforms. This speaks to the issue of tool compatibility to various computing platforms and the ability to share scripts between tools and platforms.

Solution Strategies

- Select tools that have cross-platform capability to the greatest extent possible.

- Consider writing shell scripts and bridging scripts, perhaps in non-proprietary scripting languages, such as Tcl. See Figure 2.

- Evaluate critically whether the ability to perform cross-platform testing is a firm requirement.



Source: Integration and Interoperability Testing Course, Rice Consulting Solutions, LLC

Figure 2

Challenge #8 – Testware maintenance issues and a lack of configuration management processes

The one single issue that has been associated with test automation since the very beginning is finding an effective way to keep the automated testware (scripts, cases, etc.) current with the applications being tested. Even before Rapid Application Development (RAD) became popular for software development, people had difficulty keeping automated test scripts in sync with the applications they tested. Of course, the faster the applications change, the greater this challenge becomes.

Some tools handle rapid change better than others. Tools that are object-based as opposed to using relative positioning (row, column) are designed to recognize objects and how they behave as opposed to where they appear. This is good, however there are times when even the object-based tools may need to be used in a way that requires relative positioning. Relative positioning is a bad place to be in test scripting in terms of maintenance. A minor interface change can ripple through many scripts and require extensive updates.

Test automation is "software testing software." This means that items created using the automated test tools should be subject to the same level of control as any other software asset.

When software configuration management (SCM) is not in place for automated testing, the discipline is missing to work with the tools. Without SCM for automated test tools:

- Effort is duplicated because different people may each be building similar test scripts
- Reuse is not realized because people are all creating test scripts for single-use purposes.
- Existing automated test scripts are at risk of corruption if they are modified without the knowledge of the original author.

What's required for effective SCM for test automation:

- A workable process that everyone using the tool can understand and follow,
- A tool to manage the ownership, versions and organization of the automated test scripts, and
- A person to own the SCM process and ensure that people are following it.

Many of the popular automated test tools have integrated test case and test script management applications. However, you still need the process and the people to make the SCM effort work. You can also build your own test management tool using a database and basic file organization to group related tests into suites.

A related issue in this challenge is keeping up with changes to applications that are under test. This has been one of the biggest challenges in test automation since its inception. The degree of difficulty in dealing with application changes in automated testware depends on the tool and the technologies involved. In the object-based world, the more robust tools can be configured to ignore user interface changes as long as the objects still behave the same. However, if the tool uses row and column positioning, then each application change will require a change (or many changes) to the automated test scripts.

In character-based applications, such as mainframe CICS applications, all of the changes that impact the user interface will most likely require maintenance to the automated test scripts that test those interfaces.

Solution Strategies

- During your tool search, consider the people and processes that will be required to manage the automated test cases and test scripts.
- If you are in the object-based environment (such as Graphical User Interfaces), look for tools that accommodate changes to the user interface gracefully. These tools cost more than tools that do not offer such flexibility. However, many people have found that the added cost is small compared to the cost on continued manual maintenance of the testware.
- Design your test cases and test scripts to be modular. Instead of using one script to perform multiple functions, break the script into separate functions (Figure 3).

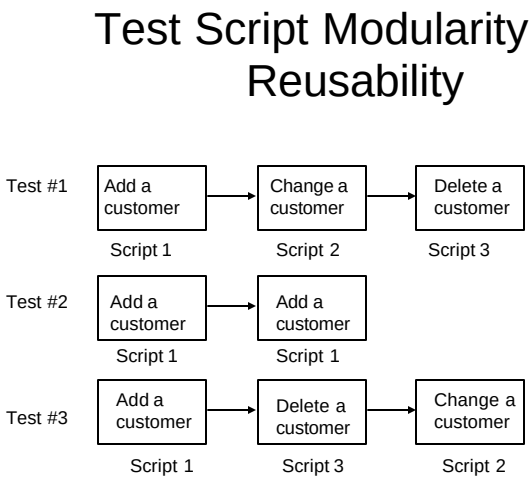


Figure 3

- Design your test cases and test scripts to be generic in terms of process and repeatable in terms of data. Read test data from a separate source. In other words, keep the testware free of test data so that when you do have to change the process or the data, you only have to maintain one item in one place. See Figure 4.

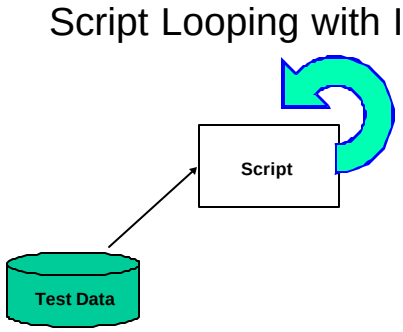


Figure 4

- Consider automated test scripts as part of an application's configuration set.
- Involve the prospective test automation SCM person in evaluating test tools and their respective test management offerings.
- Investigate the use of existing SCM tools currently owned by your organization.
- Trace your automated test scripts to functional requirements and defects.

Challenge #7 - Lack of a basic test process or understanding of what to test

Most automated test tools do not tell you what to test. Even the tools that have test case generation features do so at a user interface level and not at the functional requirement level.

If you don't know which tests are the most important and which tests are the most applicable for automation, the tool will only help perform a bad test faster. This is a universal principle that I often illustrate with the example of a power tool.

Let's say that I want to build a bookcase. I try cutting the wood with a handsaw, but it's far too slow and laborious. So, I decide to go to the hardware store and buy a power saw. Upon the purchase of whatever tool I can afford, looks good, or the salesperson convinces me of, I go home and start cutting wood. However, if I don't have bookcase plans or a very good understanding of how to build a bookcase, the saw will just help me make my mistakes faster! To be successful, I will need to first learn enough woodworking skills to understand not only the "what" and "when", but the "why" and "how" of building the bookcase. Then, I'm ready to use the tool effectively.

The tool vendors can train you in how to use the tool with all of its functionality, but the burden is on you to examine your own applications and determine which functions should be tested and to what extent.

Solution Strategies

- Create a set of evaluation criteria for functions that you will want to consider when using the automated test tool. These criteria may include:
    - Repeatability of tests
    - Criticality/Risk of applications
    - Simplicity of operation
    - Ease of automation
    - Level of documentation of the function (requirements, etc.)
- Examine your existing set of test cases and test scripts to see which ones are most applicable for test automation.
- Examine your current testing process and determine where it needs to be adjusted for using automated test tools.
- Be prepared to make changes in the current ways you perform testing.
- Involve people that will be using the tool to help design the automated testing process.
- Train people in basic test planning skills.

Challenge #6 - Lack of tool ownership and acceptance

In this challenge, the tool is not applied or is ignored. This is often the result of someone's good intention of buying a tool to make life easier, but the rest of the people don't use it.

Some of the reasons for lack of tool ownership and acceptance include:

- Difficulty in using the tool
- Not enough time to learn the tool and perform their normal level of work
- Lack of tool training
- Lack of management support for using the tool
- Lack of tool support, either internally or from the vendor
- Tool obsolescence

Solution Strategies

- Don't cut the tool training. Training doesn't guarantee success, but without it you are at risk of tool abandonment.
- Have someone in your organization in the role of a "tool smith." This person's job is to be the resident expert on the tools used for testing.
- Management needs to emphasize that the tool effort is important to them and tool usage is a required part of the testing process.

Challenge #5 - Inadequate tool training

We discussed the training issue in relation to Challenge #6, but this challenge carries its own set of concerns.

Some of the key issues for this challenge include:

- Skipping the vendor's training. The main motivation for this is lack of time and/or money. You'll spend more of both without the training!
- Not getting the right training, due to the incorrect selection of topics. For example, some tool users will need to learn in detail the tool's test scripting language, while other users will need to learn only the basic tool functionality.
- Inability to apply the training to your environment. This is where you learn to use the tool on the vendor's canned example, but have difficulty getting the tool to work on your own applications.
- Trying to learn by self-study. Yes, it can be done, but it takes time and dedication. More often than not, people tend to spend time with only the basic functions and not have the benefit of learning the lesser-known and perhaps more powerful features of the tool.
- Not enough time for training. This goes along with the "dive right in" approach often seen in Information Technology groups. When time is scarce, people tend to gravitate

toward the easy and basic functions at the expense of not learning the more difficult but more powerful ones.

Solution Strategies

- Include money in the tool proposal for training at least a core group of people.
- Match people to the most applicable training topics.
- Have tool training performed by the vendor at your location using some of your own applications as exercises.
- Find a skilled local consultant experienced with the tool to sit with your team for about 3 to 4 weeks to help get you started in creating automated tests. It is very important that your team does most of the work to accomplish the transfer of knowledge!

Challenge #4 - Incomplete coverage of test types

As you profile your tests and defect types, you will often find a wide variety of types of tests that need to be performed. These include tests for:

- Correctness
- Reliability
- Security
- Performance
- Usability
- Interoperability
- Compatibility
- Data Conversion

Although the tool may be very adept at automating many of these tests, there may be test types that the tool simply can't support. In fact, most organizations are very happy with a coverage level of 80% of their existing test case libraries.

Solution Strategies

- During tool evaluation, prioritize which test types are the most critical to your success and judge the candidate tools on those criteria.
- Understand the tools and their tradeoffs. You may need to use a multi-tool solution to get higher levels of test type coverage. For example, you will need to combine the capture/playback tool with a load test tool to cover your performance test cases.
- Manage expectations by reminding people that 100% test type coverage is not likely. However, by automating 80% of the tests, you have time to deal with the rest manually.

Challenge #3 - Lack of management support

Management support is needed in designing and deploying test processes that will support the effective use of test tools, reinforcing the role and use of automated test tools in the organization, and allowing time for tools to be integrated in the testing process.

Without management support, the entire test automation effort is at risk. If management doesn't clearly and consistently show their support of test automation, people will be less

inclined to show interest in using the tools. This is a major concern, especially considering that the learning curve of some tools requires dedication to overcome.

Perhaps the greatest challenge seen in management support is balancing high expectations of the tool benefits against the time, effort and discipline it takes to implement the tool. Management may become impatient about the lack of tool progress and shift their support to other initiatives.

The pressure is on the people who made the business case for the tools to show progress in a given timeframe. The problem is there are many unforeseen things that can delay or derail a tool initiative. In reality, if people fully knew all of the future problems with any given effort, they would be very reluctant to proceed. There is a place for optimism in acquiring tools. However, a heavy dose of realism is also needed to keep expectations in line with what is achievable.

Solution Strategies

- Communicate that it takes time and planning to build a firm foundation of people, processes and the right tools.
- When making the case to management for acquiring test tools, present the challenges as well as the benefits.
- Reinforce to management that they carry a great deal of influence in how people will accept automated test tools.
- Keep management informed of tool progress and issues that arise.

Challenge #2 - Inadequate test team organization

Most test organizations learn that automated testing is a new world in terms of how tests are designed and maintained. Most tests require more than just capture/playback. The tool user must also be able to work with the tool's scripting language to accurately replay the test session. It helps if the tool user is comfortable working with coding languages. However, if the tool user is not suited to coding, there is a risk that the tool will not be used.

Solution Strategies

- Add a person to the test team who is a "test scriptwriter." This person should be comfortable in working with code and be able to take the basic test that has been designed by a test analyst and convert it into an automated script. The difference between a traditional test team structure and one that has a scriptwriter role is shown in figures 5 and 6.
- Start simple with basic scripting concepts and add complexity later.

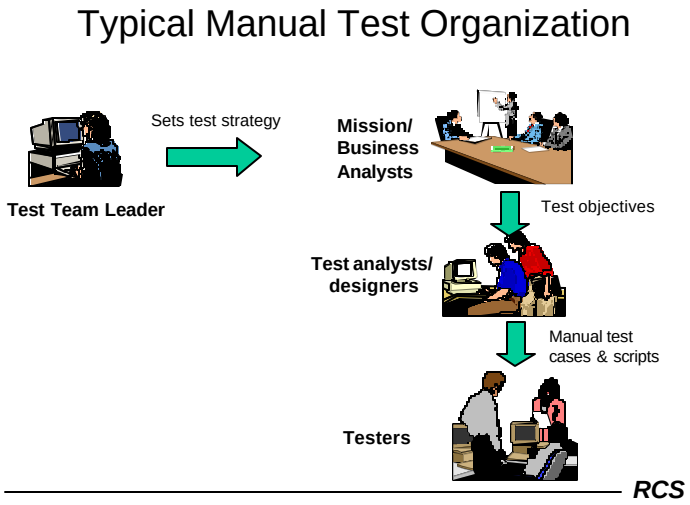


Figure 5

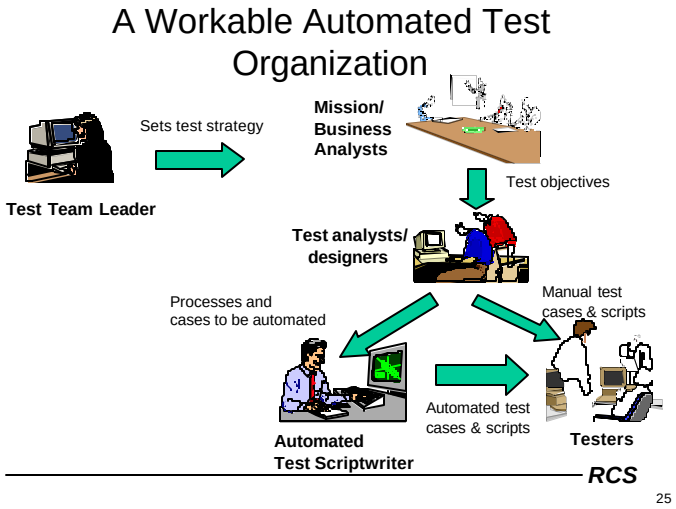


Figure 6

Challenge #1 - Buying the wrong tool

This is listed as the number one challenge in test automation because no matter what kind of process or organization you have, if the tool is not a good technical or business fit, people will not be able to apply it.

We know that a good process and organization are also essential for test automation. However, if the tool won't function at a basic level, people using the tool will simply give up trying to use it.

Unfortunately, too few people do adequate research before buying a test tool. Adequate research includes defining a set of tool requirements based on what the intended users of the tool need to accomplish, a set of evaluation criteria by which candidate tools will be judged, and the experience of other people who have used the tools under consideration.

Solution Strategies

- Take time to define the tool requirements in terms of technology, process, applications, people skills and organization.

- Involve potential users in the definition of tool requirements and evaluation criteria.

- Build an evaluation scorecard to compare the performance of the tools against a common set of criteria. Rank the criteria in terms of relative importance to the organization. A sample scorecard is shown in the following pages.

- Perform a "proof of concept" (POC) as opposed to an evaluation. In a POC the vendor often sends their technical team to your site to automate tests using your applications in your environment. Usually, a POC takes about a day to perform. The planning of the POC should be based on the evaluation scorecard. Testers should identify and define the most critical and most common tests they currently perform manually. These tests are often the ones that consume the most time and are the ones that offer the highest payback in test automation. A sample POC planning worksheet is shown on the following page, along with a coverage matrix to map POC items to test tool requirements (shown below).

SAMPLE PROOF OF CONCEPT
COVERAGE MATRIX

Test / Req.	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**
1.		X	X	X	X	X	X	X	X		
2.								X			
3.	X	X	X	X	X	X	X				
4.		X							X		
5.	X	X	X	X	X	X	X		X	X	
6.											X
7.										X	
8.											
9.	X										
10.	X						X				
11.											
12.				X							
13.				X	X						
14.					X						
15.				X							
16.											
17.	X	X	X	X	X	X	X	X	X		
18.	X	X	X	X	X	X	X		X		
19.	N/A										
20.	N/A										
21.	N/A										

SAMPLE PROOF OF CONCEPT
TEST TO BE CONDUCTED

PROJECT: Automated Test Tool Evaluation

TEST: Produce AP Checks	**TEST ID#: 1**

TEST OBJECTIVE: Validate Tool Requirements 3,5,9,10

TEST CONTROLS

1. Mary Jones will set up initial test data on the AS/400
2. A date is entered for the batch run
3. The job is executed.
4. Output (data files and reports) is verified.

TEST INPUT

- Vendor data
- Voucher data
- Discount terms

TEST OUTPUT

AP checks

- Checks are created for all vouchers that meet pay date criteria.
- Discounts are applied correctly.
- Checks are printed correctly (check amt., voucher #, discount voucher #'s, invoice amts., comments, etc. printed on the checks correctly).

Modified AP data (vouchers)

- Voucher data updated correctly (paid vouchers are updated correctly, held vouchers are not updated as paid, vouchers for held vendors are not updated as paid, etc.)
- Related data (e.g., journal entries made to the appropriate accounts.)
- Other reports (e.g., check stub overflow report)

TEST PROCEDURES

1. Enter vouchers to be paid based on invoices, date to pay and account terms. (variations of these variables will constitute the test cases.)

2. Run the AP Checks batch job.

3. Verify that processing was performed correctly. This is currently done by a visual inspection of the processed checks and data. For the purpose of the proof of concept, the before and after data need to be compared and the differences reconciled as being either valid or invalid.

Sample Test Tool Weighted Scorecard

#	Criteria	Weight	Tool A	Tool B	Tool C
1.	Record tests in a test procedure and allow the user (tester or developer) to replay the test session on a modified software module.	5			
2.	Support character-based interfaces as well as graphical user interfaces and web-based interfaces	5			
3.	Compare the results of two test sessions and identify differences between the test results	5			
4.	Allow test procedures (scripts) to be modified	5			
5.	Operate on a PC running Windows 2000	5			
6.	Track defects, produce defect reports, and perform defect analysis	3			
7.	Manage and control test procedures and test cases	5			
8.	Test interfaces with vendor software residing on PCs	3			
9.	Compare data files on the AS/400	3			
10.	Allow testers to access the test tool remotely via dial-in connection	3			
11.	Support both 80 column and 132 column screen displays	5			
12.	Support color and reverse highlighting	3			
13.	Support multiple messages scrolled on the same message line	3			
14.	Mask comparisons of unwanted display information	3			
15.	Record tests in a test procedure and allow the user (tester or developer) to replay the test session on a modified software module.	5			
16.	Host is the AS/400 platform	3			
17.	Emulator is Client Access	5			
18.	Current development language is RPG 3	5			
19.	Future direction is to convert to ILE	3			
20.	Future development languages include JAVA	3			
21.	Supports applications written in COBOL and SYNON	5			
22.	Client is Pentium III, 850 mhz., running Windows 2000.	3			

From: Integration and Interoperability Testing Workshop – Rice Consulting Solutions, LLC

Summary

These ten challenges are certainly not the only ones seen in test automation, but they are very common and have been the cause for many test automation projects to fail.

Successful software test automation is possible if fundamental issues are addressed and managed. Success depends on multiple factors that require the coordination of efforts between various groups in an organization. Automated software testing is truly a different way of testing and requires adjustments to current test methods and organizational structures. However, the payback from test automation can far outweigh the costs.

Bio

Randall W. Rice is a leading author, speaker and consultant in the field of software testing and software quality. Rice, a Certified Software Quality Analyst (CSQA) and Certified Software Test Engineer (CSTE) has worked with organizations worldwide to improve the quality of their information systems and automate their testing processes.

Rice is a regular speaker at international conferences on software testing and is also publisher of *The Software Quality Advisor*. He is co-author with William E. Perry of the book, *Surviving the Top Ten Challenges of Software Testing*, published by Dorset House Publishing Co.

Mr. Rice has over 25 years experience building and testing mission-critical projects in a variety of environments, including defense and private sector projects.

Contact Information

Randall W. Rice
Rice Consulting Solutions, LLC
P.O. Box 891284
Oklahoma City, OK 73189
Voice 405-793-7449
Fax 405-793-7454
E-mail: rrice@ricecosulting.com
Internet: www.riceconsulting.com

Additional Reading

Surviving The Top Ten Challenges of Software Testing by William E. Perry and Randall W. Rice, published March 1998 by Dorset House Publishing.

Testing Dirty Systems by William E. Perry and Randall W. Rice, to be published later this year (2003) by Dorset House Publishing.

Software Test Automation by Mark Fewster and Dorothy Graham, published May 2000 by Addison Wesley Longman

Automated Software Testing by Elfriede Dustin, John Paul and Jeff Rashka, published June 1999 by Addison Wesley Longman