RBCS
TIME TESTED.
TESTING IMPROVED.
www.RBCS-US.com

A Case Study in Successful Risk-Based Testing at CA

Introduction

This article presents a case study of a risk-based testing pilot project at CA, the world's leading independent IT management software company.

The development team chosen for this pilot is responsible for a widely-used mainframe software product called CA SYSVIEW® Performance Management, an intuitive tool for proactive management and real-time monitoring of z/OS environments. By analyzing a vast array of performance metrics, CA SYSVIEW can help organizations identify and resolve problems quickly. Companies are highly dependent on the reliability of their mainframe systems. If the mainframe doesn't run, the company stops. Mainframe workloads also are growing considerably as companies' businesses grow and as they continually seek to leverage data and applications in new ways.
At the same time, these companies are losing their experienced mainframe workforce, largely to retirement. This makes the quality of their mainframe management tools even more important to them.

CA piloted risk-based testing as part of our larger effort to ensure the quality of the solutions we deliver. The pilot consisted of six main activities:

- Training key stakeholders on risk-based testing
- Holding a quality risk analysis session
- Analyzing and refining the quality risk analysis
- Aligning the testing with the quality risks
- Guiding the testing based on risks
- Assessing benefits and lessons

This article addresses each of these areas – as well as some of the broader issues associated with risk-based testing.

What is Risk-Based Testing?

Generally speaking, risk is the possibility of a negative or undesirable outcome or event. Testing is concerned with two main types of risks:

- Product or quality risks, which are problems that can potentially affect the quality of the product itself, such as a defect that could cause a system to crash during normal operation.
- Project or planning risks, which are problems that can potentially affect overall project success, such as a staffing shortage that could delay completion of a deliverable.

Of course, not all risks are equal and there are a number of ways to classify the different levels of risk. The simplest is to look at two factors:

- The likelihood of the problem occurring, which depends primarily on technical considerations, such as the programming languages used and the constraints of a given computing platform.
- The impact of the problem should it occur, which depends primarily on business considerations, such as the financial impact of system downtime or the amount of lost staff productivity.

Risk-based testing is guided by the level of risk associated with items identified during analysis. Although risk can guide testing in various ways, there are three common ones.

First, during all test activities, test teams *allocate effort* to each *quality risk* item based on the relative level of risk. Test managers and analysts align the rigor and extensiveness of test techniques with the level of risk. They carry out test activities in risk order, starting with the most important risks. They also work with the project team to prioritize resolution of discovered defects based on the level of risk.

Second, test managers *implement control steps* for all significant identified *project risks*. A control step is either a mitigation (something done in advance to reduce the likelihood and/or impact of a risk) or a contingency (something you are prepared to do if the risk becomes an event to reduce the impact of the event). The higher the level of risk, the more thoroughly that project risk is controlled. These project risks must include risks related to testing itself, since problems during test execution can reduce test scope and thereby result in quality risks.

Third, test managers and test analysts *report test results and project status* in terms of residual risks. Which tests have been run and which haven't? Which have passed? Which have failed? Which defects have not yet been fixed or retested? How do the tests and defects relate back to the risks?

In other words, with risk-based testing, risk management is an ongoing event. The above three responses to risk occur throughout the project lifecycle. Quality risks are mitigated by running tests, and project risks are mitigated by controls. Risks and risk levels are periodically re-evaluated based on new information, and, if necessary, priorities, allocation of effort, and project controls are modified.

Why Adopt Risk-Based Testing?

All testing faces two serious challenges. First, the set of possible test cases is infinite. So, if test coverage is measured by dividing the number of tests run by the number that could have been run, test coverage is always zero percent ($n/\infty = 0$). In fact, testers always select a relatively small subset of tests from the set of tests that they could possibly run – so they have to be very smart about that selection. Selection based on risk makes the most sense both in terms of product quality and project success.

Second, because projects cannot take an infinite amount of time, all testing is "time-boxed" but the time-box is not a fixed size. Changes in upstream task durations for the project often compress the time-box for subsequent testing. The risk-based prioritization of tests directly addresses this challenge. By prioritizing tests according both to likelihood and impact, testers give themselves *the best possible chance of discovering the worst possible problems.* Also, at any given time during the test execution period, the tests that have been run are more important than the tests that *have not yet been run*. This allows test managers to make risk-based test

"triage" decisions to meet inflexible project deadlines – thereby minimizing the risks associated with any reduction in the scope of testing.

An assessment of CA's testing processes indicated that, like many test organizations, we faced challenges regarding both coverage and time constraint. The adoption of risk-based testing therefore seemed a wise choice.

Training Key Stakeholders

The first step in our pilot project was a one-day workshop on risk-based testing that covered the following topics:

- The basic principles of and rationale for risk-based testing
- Understanding categories of quality risks (functionality, performance, reliability, usability, etc.)
- How to perform a quality risk analysis and align testing with risk levels
- How to document quality risks
- How to monitor quality risks during test execution and report risk-based test results

In addition to the formal presentation, we had an open discussion and a two-hour hands-on exercise based on a hypothetical project, enabling attendees to explore the real-world issues presented by risk-based testing.

The Quality Risk Analysis Session

The quality risk analysis session consisted of two sub-sessions. During the first, the participants identified as many quality risk items as possible, using brainstorming techniques. We listed the main quality risk categories on three whiteboards. The participants wrote each risk item on a sticky note and posted it under the appropriate category. This sub-session lasted about three hours and identified more than 100 risk items.

We also identified 11 project risks (e.g., "The number and timing of QA bug discovery delays the release date.") and three miscellaneous issues (e.g. "Have all previous release fixes been merged into the code base?").

During the second sub-session, participants worked as a team to assess the likelihood and impact of each risk item. We also identified and eliminated risk items that were duplicates of other items.

To make it easy for the team to assess the likelihood of all risk items, we used the rating scale shown in Table 1. We had quick inter-rater agreement on the likelihood ratings for almost all of the quality risk items using this scale.

Likelihood	Rating	Comments
Very likely	1	Almost certain to happen
Likely	2	More likely to happen than not to happen
Somewhat likely	3	About even odds of happening
Unlikely	4	Less likely to happen than not to happen
Very unlikely	5	Almost certain not to happen

Table 1: Likelihood Rating Scale for Risk Items

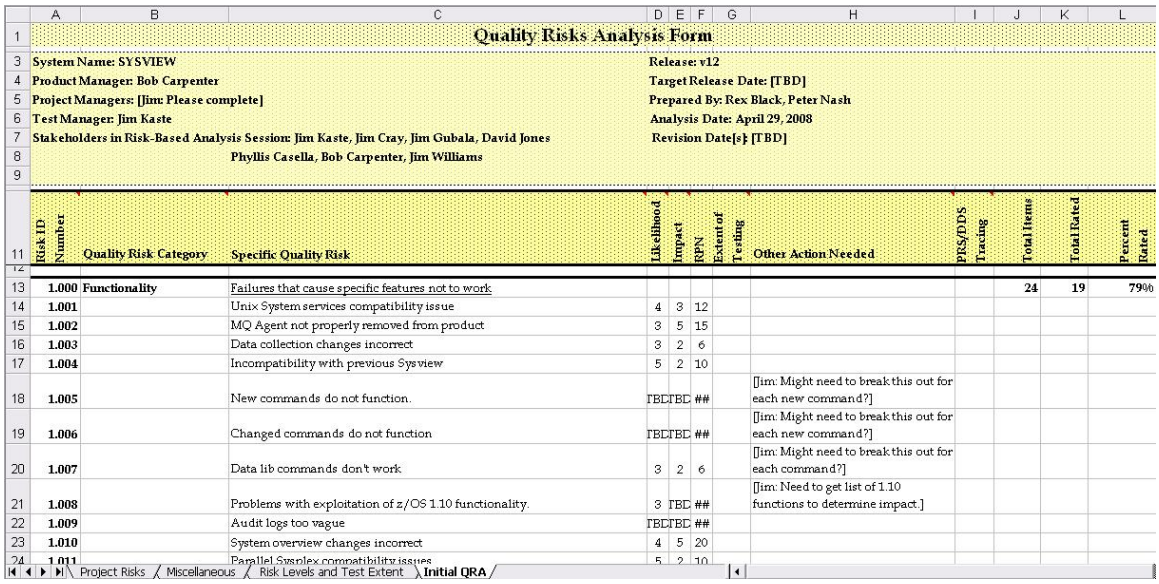
The participants struggled with the assessment of impact. Initially, we used the scale shown in Table 2. Serious debates occurred among participants about the different distinctions, particularly between the "must-fix now" and the "must-fix schedule" impacts. This slowed the process considerably.

Impact	Rating	Comments
Must-fix now	1	Top priority, "come in on Sunday" type of issue
Must-fix schedule	2	Schedule for attention and resolution as quickly as possible
Should fix	3	A major irritant but might not receive attention until other issues are addressed
Good-to-fix	4	An irritant for some number of customers that will cause some concern unless resolved
Don't fix	5	No or limited value to fixing this problem.

Table 2: Initial Impact Rating Scale for Risk Items

At the end of quality-risk analysis session, we had identified 92 non-duplicate quality risk items. Of those, the team had successfully rated the impact and likelihood for about 40%. We then asked one team member to assign tentative risk levels to the remaining risk items, subject to the approval of the team.

Figure 1 shows a portion of the quality risk analysis document at the end of that quality risk analysis session.



Quality Risks Analysis Form

System Name: SYSVIEW | Release: v12
Product Manager: Bob Carpenter | Target Release Date: [TBD]
Project Managers: [Jim: Please complete] | Prepared By: Rex Black, Peter Nash
Test Manager: Jim Kaste | Analysis Date: April 29, 2008
Stakeholders in Risk-Based Analysis Session: Jim Kaste, Jim Cray, Jim Gubala, David Jones, Phyllis Casella, Bob Carpenter, Jim Williams | Revision Date[s]: [TBD]

Risk ID Number	Quality Risk Category	Specific Quality Risk	Likelihood	Impact	RPN	Extent of Testing	Other Action Needed	PRS/DDS Tracing	Total Items	Total Rated	Percent Rated
1.000	Functionality	Failures that cause specific features not to work							24	19	79%
1.001		Unix System services compatibility issue	4	3	12						
1.002		MQ Agent not properly removed from product	3	5	15						
1.003		Data collection changes incorrect	3	2	6						
1.004		Incompatibility with previous Sysview	5	2	10						
1.005		New commands do not function.	TBD	TBD	##		[Jim: Might need to break this out for each new command?]				
1.006		Changed commands do not function	TBD	TBD	##		[Jim: Might need to break this out for each new command?]				
1.007		Data lib commands don't work	3	2	6		[Jim: Might need to break this out for each command?]				
1.008		Problems with exploitation of z/OS 1.10 functionality	3	TBD	##		[Jim: Need to get list of 1.10 functions to determine impact.]				
1.009		Audit logs too vague	TBD	TBD	##						
1.010		System overview changes incorrect	4	5	20						
1.011		Parallel Sysplex compatibility issues	5	2	10						

Project Risks | Miscellaneous | Risk Levels and Test Extent | Initial QRA

Figure 1: Quality Risk Analysis Worksheet at End of Risk Analysis Session

The team members successfully rated the remaining unrated items. In some cases, this required splitting a given item into two in order to assign an appropriate impact to each aspect of the risk. At the end of this process, we had 104 fully rated quality risk items.

Analyzing and Refining the Quality Risk Analysis

With the likelihood and impact rated for all risk items, we were able to calculate the risk priority number for each item by multiplying the likelihood and impact. Since both likelihood and impact were rated on a five-point scale, risk priority numbers ranged from 1 to 25 – with 1 being the most risky and 25 the least risky.

One potential problem with quality risk analysis is a "clumping" of risk ratings. This can occur when teams consistently skew the impact of risk items by basing their ratings on worst-case outcomes. It also can occur when teams use a scale with poorly defined distinctions.

To check for this, we created a histogram of our risk priority numbers, as shown in Figure 2.

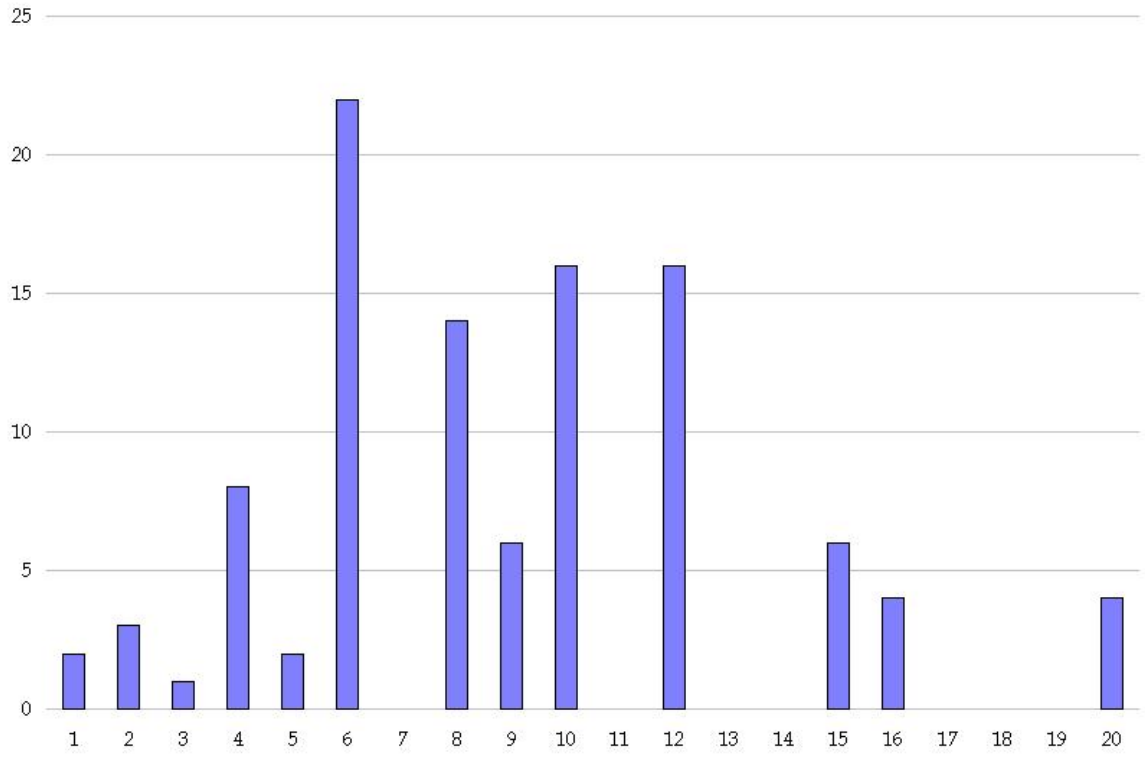


Figure 2: Risk Priority Number Histogram after Initial Assessment of Risk Levels

Note that some of the "dead zones" exist because there are no two integers between 1 and 5 that when multiplied together yield 7, 11, 13, 14, 17, 18, 19, 21, 22, 23, or 24. We didn't have a single risk item with a risk priority number of 25 because we had no very unlikely risks that we would not fix.

We did, however, see a strong skewing towards the left side of the histogram, with many risk items rated with a value of six. To check for the underlying cause, we looked at the number of risk items with each possible likelihood and impact rating, as shown in Table 3.

Likelihood	Count	Impact	Count
1	5	1	10
2	9	2	52
3	25	3	32
4	39	4	8
5	26	5	2

Table 3: Distribution of Likelihood and Impact Ratings

Looking at the left half of Table 3, the likelihood ratings at first look skewed. But CA SYSVIEW is a mature, well-established product with a maintainable, solid code base

and a rock-solid development team. For a newer product, such a distribution would likely be wishful thinking.

The right half of Table 3, on the other hand, may be more problematic. Half of the impact ratings are 2, which means "Schedule for attention and resolution as quickly as possible."

To address this clumping, we adjusted the distinction between an impact rating of two and an impact rating of three as shown in Table 4. This achieved a much better distribution of impact – as is evident by comparing Figure 3 to Figure 2.

Impact	Rating	Comments
Must-fix now	1	Top priority, "come in on Sunday" type of issue
Must-fix no workaround	2	Loss of important functionality with no workaround, so schedule for attention and resolution as quickly as possible
Must-fix w/workaround	3	Loss of important functionality but with a workaround, so schedule for attention as impact 1 and 2 issues are resolved
Good-to-fix	4	An irritant for some number of customers that will cause some concern unless resolved
Limited value to fix	5	No or limited value to fixing this problem.

Table 4: Adjusted Impact Ratings to Achieve More Precision

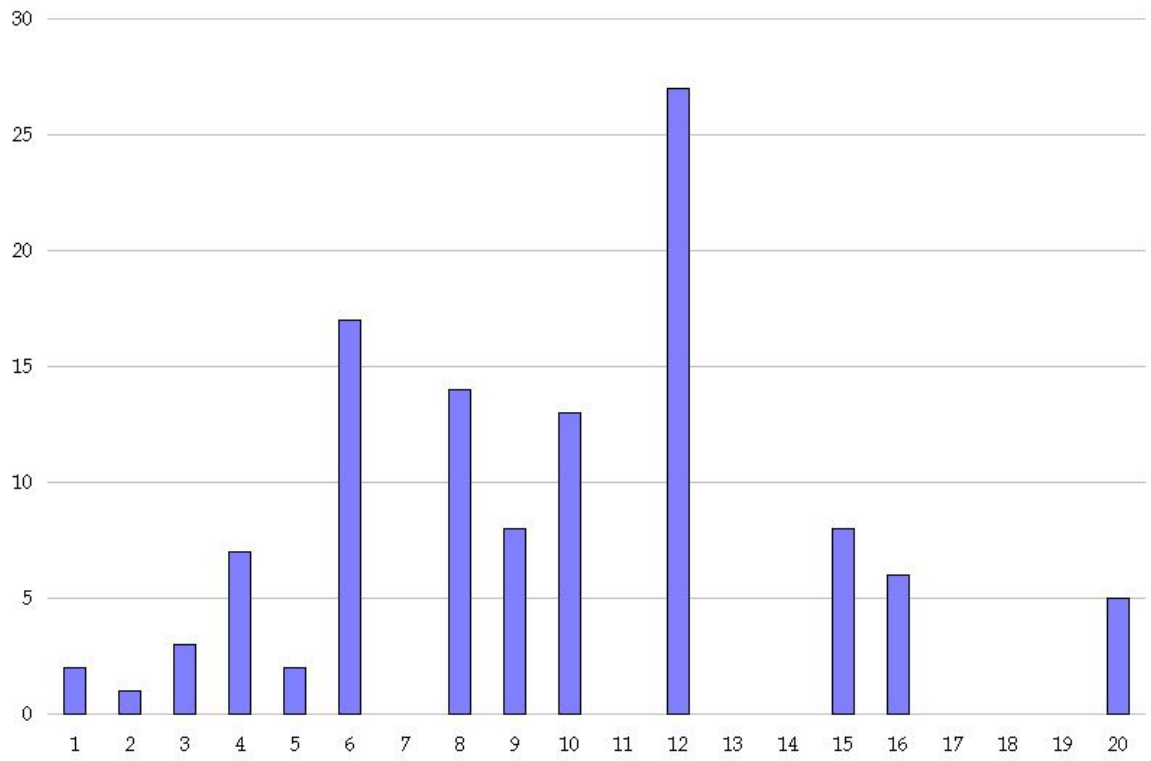


Figure 3: Risk Priority Number Histogram with Adjusted Assessment of Impact

Aligning the Testing and Quality Risks

With our initial risk analysis complete, we then set out to align testing with risk. This required us to:

1) allocate testing effort based on the level of risk
2) map risk items to specifications
3) map risk items to test cases
4) prioritize test cases based on risk levels of each risk item

One way to allocate effort is to establish a mapping between effort and risk priority numbers. Based on discussions with the key stakeholders, we devised the mapping shown in Table 5. Cross-referencing this table back to Figure 3, it is evident that all risk items receive at least cursory testing.

RPN Range	Extent of Testing	Comments
1 – 12	Extensive	Run a large number of tests that are both broad and deep, exercising combinations and variations of interesting conditions
13 – 16	Broad	Run a medium number of tests that exercise many different interesting conditions
17 – 20	Cursory	Run a small number of tests that sample the most interesting conditions
21 – 25	Opportunity	Leverage other tests or activities to run a test or two of an interesting condition, but only if it involves a very small investment of time and effort and only if the opportunity presents itself

Table 5: Mapping the Risk Priority Number to Testing Effort

To ensure that we could quickly update risk items based on requirements changes that might occur during the project, we mapped each risk item to the Product Requirements Specification (PRS) and Detail Design Specification (DDS) elements. Figure 4 shows 10 quality risk items in the category of serviceability, with corresponding testing effort and mapping to the PRS and DDS documents.

Risk ID Number	Quality Risk Category	Specific Quality Risk	Likelihood	Impact	RPN	Extent of Testing	Other Action Needed	PRS/DDS Tracing	Total Items	Total Rated	Percent Rated
5.000	Serviceablity	Problems with deploying the system, updating the system, etc.							10	10	100%
5.001		Data Com r12 doesn't support service DB	2	3	6	Ext		PRS 2.1.3			
5.002		Internal product tracing capabilities insufficient to resolve new function code bugs	3	4	12	Ext		PRS 4.2			
5.003		Report Writer has installation or packaging errors	1	3	3	Ext		PRS 2.1.6			
5.004		GMI will not install	4	2	8	Ext		DDS Vantage 2.2.3.1			
5.005		Can not apply GMI fixes easily	3	3	9	Ext		DDS Vantage 3.2			
5.006		Out of box implementation does not work	5	1	5	Ext		PRS 4.2			
5.007		Migration from previous releases do not work	3	2	6	Ext		PRS 4.2			
5.008		SMP/E installation does not work	1	1	1	Ext		PRS 4.2			
5.009		Share Queues use or service does not work	4	2	8	Ext		DDS IMS 4.3			
5.010		Migration from previous release is error prone	5	3	15	Broad		PRS 4.2			

Figure 4: Serviceability Risks with Testing Effort and PRS/DDS Mapping

Guiding the Project Based on Risks

Once we had determined how much testing effort and what level of priority each risk item merited, we then could associate each risk with the appropriate test activities. Where a single test related to more than one risk item, the test was assigned the highest of the associated risk priority ratings. This allowed us to establish an ideal-case prioritization and sequencing of tests based on risk.

The use of risk-based priority ratings to establish a testing plan was quite different from previous approaches. In the past, we typically assigned tasks based on the expertise and availability of staff resources. Sometimes, this meant that someone with a lot of expertise might be given a large number of important tasks. The result was that this person would perform some very important tasks until very late in the

project. With our risk-based system, this did not happen. All of our high-priority tests were executed early, and all low-priority tests were executed later.

Another thing that changed was how we prioritized the bugs found during testing. Although we already had standards for determining the severity of an issue, we were now able to incorporate risk into prioritization as well. This resulted in issues opened at a higher severity, which helped ensure that we didn't begin Beta testing with any open issues that mapped to risk items with a high rating.

Assessing Benefits and Lessons

Our experience with this pilot project highlighted several key benefits of risk based testing:

- **The ability to allocate test effort intelligently within constraints.** Theoretically, testing teams would like to test every aspect of their software exhaustively. This is simply not possible. Risk-based testing enabled the team to selectively eliminate testing tasks that would not negatively affect customers or the project – so that the team could focus its efforts on issues of real importance.
- **The ability to "find the scary stuff first."** Risk-based testing delivered a lot of value in terms of issue discovery. It enabled the team to pinpoint potentially serious problems that required remediation early in testing. This gave developers plenty of time to make changes without adversely affecting the overall project timeline.
- **The ability to respond flexibly to reductions in available test time and resources.** As often happens in the real world, the team lost a resource during the project. Having prioritized risks made it easier to re-assign tasks to others intelligently. It also made it easier to determine which tasks to eliminate in order to meet the target delivery date.
- **The ability to optimize quality.** Ultimately, the entire point of testing is to deliver products of superior quality to customers. With risk-based testing, we ensured that the product we delivered to our customers had been thoroughly tested for issues that might significantly affect their satisfaction.

We also learned several important lessons about risk-based testing and risk analysis. One important lesson was to include business users – and, in the future, potentially even customers – in the risk analysis process. Technical staff tend to think about risk impact primarily in technical terms, such as outages or functional annoyances. But people from the business side were able to offer better insight into what kinds of issues would be most problematic for them in terms of personal productivity or business process failures. This insight is invaluable for accurately assigning risk priority numbers to testing tasks.

As software developers are called upon to deliver increasingly sophisticated and complex products within tighter and tighter resource constraints, prioritization of testing tasks will become increasingly important both for achieving product quality and for meeting project deadlines. Risk-based testing planning is therefore an essential discipline – and one that testing teams should start adopt as soon as they can.

Acknowledgement

The authors would like to thank the members of the SYSVIEW team who participated in the risk analysis process for their time and diligence, especially Bob Carpenter, Product Line Manager; Phyllis Casella, Engineering Project Manager; Jim Cray, Director Software Engineering; Jim Gubala, Quality Assurance Engineer; David Jones, Manager Sustaining Engineering; Jim Kaste, Manager Quality Assurance; Jim Williams, Principal Product Manager; Rob Steiskal, Sr. Product Marketing Manager.

Portions of this article are excerpted from Rex Black's forthcoming book, *Advanced Software Testing Volume 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager*, due out October, 2008 from Rocky Nook.

Author Bios

Rex Black is President of RBCS (www.rbcs-us.com), a worldwide leader in testing services, including consulting, outsourcing, assessment, and training.

Peter Nash, Engineering Program Manager at CA, has more than 20 years of software industry experience.

Ken Young, Vice President of Process and Quality at CA, has more than 20 years of experience managing software development projects.