Quality Risk Analysis

By Rex Black, President, RBCS, Inc.

Testing any real-world system is potentially an infinite task. Of this infinite set of possible tests, test managers need to focus on the most significant risks to system quality. These are the potential failures that are likely to occur in real-world use or would cost a lot if they did occur. This article describes practical ways to analyze the risks to system quality, providing guidance along the way to achieving effective and efficient testing.

Key words: **Risk, Quality, Testing, Analysis**

INTRODUCTION

For any real-world system, there are an infinite number of tests that one can conceivably run. How does one choose the best possible subset? A smart approach would be to pick those tests that address the most important risks.

Testing reduces risks to system quality. It helps identify areas of the system that work properly (that is, the tests pass). It also helps identify opportunities to make the system better (that is, the tests fail, detecting bugs). So smart test managers should ask, "How do we test the most critical areas and find the most critical bugs?"

This article will explain how to apply the powerful techniques of risk analysis and risk management. The author provides five techniques for analyzing quality risks. Using these techniques, one can make informed, objective decisions about where to focus testing resources. By reassessing risk at major milestones during the project, one can continually optimize the effectiveness and efficiency of testing throughout the system development life cycle. The author illustrates the ideas using three case studies.

Why Risk Matters—and What It Has to Do with Testing

Any significant project involves risk. Risk increases with complexity, the number of participants, effort, budget, and duration. Capers Jones (1995) cites probabilities of software project failure ranging from 2 percent to 85 percent. He identifies inadequate testing as one of the four leading causes of failure, along with poor estimation, planning, and project tracking.

Many managers are familiar with project risk management. For example, they manage risks such as loss of key personnel and late deliverables from vendors. Classical risk management says to mitigate these risks using both proactive means (such as cross-training of staff) and reactive means (such as redundant component sourcing). One can mitigate risks to system quality through proactive means like reviews and through reactive means like testing.

The author prefers J. M. Juran's definition of quality: fitness for use. Quality is the presence of that which satisfies users, customers, and other stakeholders, and the absence of that which dissatisfies them. A quality system is fit for the users' purposes, provides the needed features, and contains few, if any, important bugs (Juran 1988).

A risk is a *potential* undesirable outcome. So, a risk to the quality of the system is any potential problem that could cause the system to fail to meet reasonable expectations of quality (Black 2003).

When one starts testing a system, he or she evaluates the quality of the system in terms of particular functions, characteristics, and behaviors. While the presence of some bugs is expected, knowing exactly which activities will find those bugs is the challenge.

As testing continues, some of the bugs are discovered. These known bugs are no longer risks; they are now *actual* undesirable outcomes. It is possible to fix these bugs and improve the quality of the system. The overall risk is reduced.

When running the tests, one also discovers where the bugs are not. The tests that pass show that the system works as expected under the tested conditions. Again, the overall risk is reduced.

The more thoroughly one tests the system, the more of the remaining bugs he or she finds. The more bugs one finds, the more bugs one can fix. The more thoroughly one tests the system, the more known-good, tested conditions one identifies. So, the more one tests, the lower the risk to the quality of the system. But the risk will never be reduced to zero, because there's always one more test one could run.

Which Quality Risks Should We Worry About?

Since it is not possible to test everything, it is necessary to pick a subset of the overall set of tests to be run. Quality risks analysis can help one focus the test effort.

While the phrase *quality risk analysis* may sound forbidding, mystifying, and complex, the underlying ideas—and the techniques—need not be. Simply put, quality risk analysis is a process for identifying, analyzing, and prioritizing categories of potential quality problems (that is, bugs) in one's systems.

Because quality risks are potential problems—with probabilities between zero and 100 percent—one factor that influences the relative importance of a risk is the likelihood of the undesirable outcome. In other words, is it likely that these kinds of bugs exist in the system and is it likely that users will encounter these kinds of bugs if they exist?

Another factor that influences the relative importance of a risk is the impact of that undesirable outcome on the users, customers, or other stakeholders. In other words, if these kinds of bugs exist in the system and users encounter them, how will the symptoms of the bugs affect the users? The impact of bugs can range from trivial to catastrophic on some systems, even deadly.

For example, suppose one is developing an online banking application. This application allows users to log into their bank accounts over the Internet. They can then pay bills, transfer funds between accounts, and download statements.

For such an application, security is clearly a major quality characteristic. In the area of security, risks to system quality include the possibility that criminal hackers gain unauthorized access to other customers' accounts or that hackers intercept account information in transit between the data center and customers' PCs.

As the steady stream of Internet hacking exploits point out, the likelihood of such problems are all too high. When they do occur, such problems have serious impacts on customers, businesses, and other stakeholders. Clearly, to have confidence in such a system, one would want to develop and test in such a way as to reduce these risks.

QUALITY RISK ANALYSIS TECHNIQUES

There are a variety of risk analysis techniques available these days. Each has its strengths and weakness, depending on the needs and practices of the project team. So, selecting the right technique is essential.

All of these techniques involve identifying and prioritizing the risks to the quality of the system under test. Typically, the risks are grouped or organized by major risk categories, such as functionality, performance, security, and so forth. A cross-functional team of project stakeholders usually identifies the risks. Rather than rely only on the stakeholders' opinions and recollections, the analysis should also draw upon historical bug and field failure data from past projects, requirements and design specifications, sales figures, market research, and anecdotal information from customers, competitors, or clients.

Once the risks are identified, each risk is assigned a level—a measure of its degree of importance. In a later section, the author discusses the factors that affect the level of risk and how one can combine these factors to come up with an aggregate measure of risk, as well as how to use the aggregate measure of risk to decide on the level of effort to expend on testing. Following are five techniques for analyzing risk (see also Table 1):

1. *Informal quality risk analysis* techniques do not entail much beyond what is described in this section so far. Such techniques provide an easy way to get started in quality risk analysis. One is likely to miss some important areas of risk, especially during early risk analyses, but even informal techniques make it possible to achieve a better degree of test focus and coverage than one can achieve without any risk analysis at all.
2. For *ISO 9126 quality risks analysis* techniques, one should use the quality characteristics and subcharacteristics described in the ISO 9126 system quality standard as the categories for the risk analysis. The six main characteristics of quality are:
    - Functionality
    - Reliability

- Usability
- Efficiency
- Maintainability
- Portability

In each category are two or more subcharacteristics. ISO 9126 provides a more structured approach and reduces the likelihood of missing some major risk elements. It also tends to add a bit in terms of time and paperwork.

3. *Cost of exposure* quality risk analysis techniques focus on the following question: What are the expected losses associated with various risks, and how much should one spend to reduce those risks? An expected loss is the product of the probability of the loss multiplied by the cost of the loss. Such techniques allow the project management team to make a hard-nosed, economic decision about testing. For this technique to work well, however, the risk analysis team must be able to accurately estimate the probabilities and costs of various losses.
4. *Failure mode and effect analysis* goes beyond discussing risks. Using this technique, the risk analysis team tries to identify the different ways the system could fail and all the possible effects those failures would have on customers, clients, the business, society, and so forth. This technique is quite detailed. It can produce finely calibrated testing, but it also can produce a ton of paperwork and lots of invested time.
5. *Hazard analysis* techniques are like failure mode and effect analysis, but done backward. One starts first with the effects—the hazards—and tries to work backward to causes. Along the way, the likelihood of those causes should become clear. In some cases, though, there are many causes of different kinds of bad behavior, so this technique tends to work best with systems that do only a few things.

Technique	Informal (Black 2002)	ISO 9126 (ISO 2000)	Cost of exposure (Black 2003)	Failure mode/ effect analysis (Stamatis 1995)	Hazard analysis (Craig and Jaskiel 2002)
In a nutshell	Rely on history, experience, and checklists	Follow industry-standard quality characteristics	Estimate costs of testing versus not testing	Identify the potential bugs and the effects on stakeholders	Analyze causes of hazards (sources of risk)
Strengths	Easy, lightweight, flexible	Predefined, thorough, common	Balanced, pecuniary, traditional	Precise, meticulous, general	Exact, cautious, systematic
Weaknesses	Participant-dependent, gappy, imprecise	Potentially over-broad, over-regimented	Data-intensive, exclusively monetary	Lengthy, document-heavy, hard-to-learn	Easily overwhelmed by complexity
Consider on ___ projects	Low-risk or agile	Standards-compliant	Financial or actuarial	High-risk or conservative	Medical or embedded
Avoid on ___ projects	Safety-critical or regulated	Very unusual or structure-intolerant	Safety- or mission-critical	Chaotic, fast-changing, or prototyping	Unpredictable or complex

Table 1 A comparison of risk analysis techniques

Case Studies: Quality Risks Analyses

To provide an idea of the documentation and information produced by some of these techniques, look at two case studies, Figure 1 and Figure 2. These figures show excerpts from the two documents the author and his associates developed for these projects.

Figure 1 shows a portion of the document produced during an informal quality risk analysis for an Internet appliance. Such devices allow users to send and receive e-mails, browse the Web, purchase items at e-commerce Web sites, and so on.

Figure 2 shows a portion of the document produced during a formal quality risk analysis, following the failure mode and effect analysis technique. The author analyzed the risks for a utility that ensures file security on PCs by forcing the complete erasure of all digital information related to a file upon deletion.

Quality Risk Category	Specific Failure Mode(s)	Priority
Functionality	Client won't boot	1
	Client mail fails	1
	User data mishandled/lost	1
	Client browse fails	1
	Client software update fails	1
	Client user interface fails	1
	Device login fails	1
	Multiuser client logins fail	1
	Mail server loses mail	1
	Update server fails/unreliable/drops updates	1
	Preferences/state server fails/unreliable/loses state information	1/2
	Web server fails/unreliable/drops updates	1
	Scheduler/logging fails	2
	Sending mailer incompatibilities	1
	SPAM filtering ineffective	1/2
	Attachment refusal rude/silent	1
	RTF support fails/incomplete	1
	Basic/premium support differentiation fails	1
	Wrong number handling invalid	
	Area code changes mishandled	
	Reconnect time-out fails	
Reliability	Client crashes frequently	1
	Server crashes frequently (w/o failover working)	1
	Net connection fails	1
Performance	Slow e-mail uploads/downloads	2
	Slow updates	2
	Slow Web access	2
	"Jumpy" action or intermittent slowdowns	2

Figure 1 Informal quality risks analysis for an Internet appliance

	A	B	C	D	E	F	G	H	I	J	K	L
1	Failure Mode and Effects Analysis (Quality Risks Analysis) Form--RPN Sorted											
7												
8	Inital FMEA--RPN Sorted											
9	System Function or Feature	Potential Failure Mode(s)- Quality Risk(s)	Potential Effect(s) of Failure	Critical?	Severity	Potential Cause(s) of Failure	Priority	Detection Method(s)	Detection	Risk Pri No	Recommended Action	Who/ When?
10	Shreds Swap Files	Fails to Shred	Security Breach	Y	1	Program Error	1	Test; Debug Trace; Code Review	1	1	Test; Debug Tracing; Code Review	
11	Shreds Swap Files	Shreds Excessively	Data Loss	Y	1	Program Error	1	Test; Debug Trace; Code Review	1	1	Test; Debug Tracing; Code Review	
12	Compression Compatibility	Damages Data	Data Loss	Y	1	Program Error	1	Test	2	2	Test	
13	Compression Compatibility	Hangs System	Data Loss	Y	1	Program Error	1	Test	2	2	Test	
14	Compression Compatibility	Shreds Improperly	Data Loss	Y	1	Program Error	1	Test	2	2	Test	
15	Internet Files Recognition	Recognizes Incorrectly	Data Loss	Y	1	Program Error	1	Test; Debug Trace; Code Review	2	2	Test; Rules Validation	
16	Network Compatibility	Shreds Network	Data Loss	Y	1	Program Error	1	Test	2	2	Test Selected/ Improve Network Coverage	
17	Removes File Name	Damages FS	Data Loss	Y	1	Program Error	1	Test; Debug Trace; Code Review	2	2	Test; Debug Tracing; Code Review	

Figure 2 Failure mode and effects analysis for a PC file security utility

Quality Risks Analysis Process

Whichever technique one chooses, he or she can follow a similar process for quality risk analysis.

1. Identify the quality risk analysis team.
2. Select a technique.
3. Identify the quality risks. Prioritize the risks. Select appropriate risk mitigation actions. (One may include not only testing at various stages, but also reviews of requirements, design, and code, programming techniques such as array bounds checking, and so forth).
4. If the risk analysis team identified problems in requirements, design, code, or other project documents, models, or deliverables during the analysis, report those problems for resolution.
5. Review, revise, and finalize the quality risk analysis document.
6. Check the quality risk analysis document into the project library under change control.

7. At regular intervals (for example, major project milestones such as the completion of the requirements, design, and implementation phases, and test readiness and exit reviews) and as new information becomes available (for example, completion of a test cycle), review and update the risk analysis, adding new items and reassessing the level of risk for the items.

The author's preference is to carry out step 3 as a single meeting, using brainstorming or similar elicitation techniques. However, he has performed this step as a series of conversations, even one-on-one, between the person charged with the risk analysis and the quality risk analysis team.

In addition to selecting the right technique, it is necessary to select the right participants in order to getting the most out of the quality risk analysis process. One needs a cross-functional team that represents the interests of all stakeholders in testing and quality. Without representation of all interests, it is possible to miss some key risks, as well as over and under estimate some levels of risk.

Who are these key stakeholders? As a general rule, there are two groups. First, there are those who understand the needs and interests of the customers and/or users, or who have immediate contact with the testing process. (Customers and users themselves can be involved, too, when available.) With their insight into business considerations, they can help assess the impact of potential problems. Second, there are those who have insight into the technical details of the system. With their insight into technical considerations, they can help assess the likelihood of potential problems.

In addition to the information-gathering nature of the process, the consensus-building aspects are also critical. Both business-focused and technically focused participants can and should help prioritize the risks and select mitigation strategies.

Case Studies: Quality Risk Analysis Processes

For the Internet appliance, the author conducted the quality risk analysis as a series of interviews and discussions with the key managers and technical leaders in the organization. He had drafts of the marketing requirements and design specifications, as well as a shared high-level vision of the system among the stakeholders. The risk analysis

discussion built a solid consensus around where the author and his associates would focus the test effort.

The quality risk analysis process had two interesting side effects. First, it revealed that the vision of the system was evolving and not entirely consistent, especially at the detail level. Agreeing on what was at risk, quality-wise, helped promote a detailed consensus on what it would mean for the system to have quality. Second, the risks analysis (along with later test design efforts) highlighted a number of problems in the requirements and design documents. These problems were resolved, preventing bugs from entering the testing process.

For the file security application, the author's client had no written specifications. As with the Internet appliance, there was a shared vision of the system to be built. Fortunately, the author was able to get all of the stakeholders to commit to spending an afternoon in a quality risk analysis meeting. The resulting document served as a guide for testing focus and as a first step in the test design process.

In addition to creating a useful document, there were two interesting side effects. First, the development team decided to take steps to proactively prevent bugs. Inspired by the quality risk analysis session, development managers and technical leads implemented quality risk mitigation strategies such as robust design and code inspections. Second, the risk analysis document became the *de facto* requirements document. Since quality risk analyses tend to focus on what not to do, they express requirements in the negative.

Focusing Testing Resources Using Aggregate Levels of Quality Risks

Quality risk analysis should identify both the risks themselves and their relative levels of risk. To determine the levels of risk, likelihood and impact are typical factors to consider. The author likes to rate both of these on a five-point scale: very high, high, medium, low, and very low. Failure mode and effect analysis uses three factors --severity, priority, and likelihood of detection -- rated on a scale from 1 to 10.

To translate the two (or three) factors into a single, aggregate risk level, one can use the quality risk analysis team's collective judgment, a formula, or a table. For the Internet appliance case study, the author relied on collective judgment and decided the

appropriate aggregate level of risk. For the file security case study, he followed the failure mode and effect analysis standard of multiplying the ratings for the three factors to obtain a risk priority number from 1 to 1000, which was then divided into ranges representing levels of aggregate risk. To use a table, first construct a look-up table such as the one shown in Table 2, then use it to assign the aggregate level of risk based on the two factors for each risk item. Given the aggregate risk levels, one then uses them to determine the extent of the testing he or she wants to perform. For example, one might adopt the rules shown in Table 3.

		Impact				
		Very high	*High*	*Medium*	*Low*	*Very low*
Likelihood	*Very high*	Very high	Very high	High	High	Medium
	High	Very high	High	High	Medium	Medium
	Medium	High	High	Medium	Low	Low
	Low	High	Medium	Low	Low	Very low
	Very low	Medium	Medium	Low	Very low	Very low

Table 2 A look-up table for aggregating two risk factors

In Table 3, note that one makes no commitment for testing very low and low risks areas. For medium, high, and very high risks, he or she commits increasing amounts of time and effort to mitigate the related risks through testing. Where are the appropriate dividing lines between each category? This is again a question for the stakeholders. What level of testing do the stakeholders need to do to feel comfortable that each risk is adequately mitigated?

The answer to this question must be realistic. In the absence of any constraints, one might like to do extensive testing against most, if not all, risks. However, as mentioned earlier, this is not possible. So, the question each participant in the risk analysis must ask is, "What would I be willing to spend—in terms of schedule time, project budget, and possibly even foregone features—to mitigate this risk through testing?"

Aggregate Risk	Extent of Testing	Comments
Very low	None	Only report bugs observed in these risk areas.
Low	Opportunistic	Leverage other tests or activities to run a test or two of an interesting condition in the related risk area, but only if it involves a very small investment of time and effort and only if the opportunity presents itself.
Medium	Cursory	Run a small number of tests that sample the most interesting conditions in the related risk areas.
High	Broad	Run a medium number of tests that exercise many different interesting conditions in the related risk areas.
Very high	Extensive	Run a large number of tests that are both broad and deep, where deep tests exercise many combinations and variations of interesting conditions.

Table 3 Mapping relative quality risks to the extent of testing

Case Studies: Relative Quality Risks as a Guide

For the Internet appliance project, the author followed rules like those in Table 3. Risk served as the guide for high-level design of test suites and tools all the way down to the low-level design and implementation of test cases and data. Given that the author was working on a medium-sized project following a waterfall/V-model system development life cycle, this degree of structure was appropriate.

On the file security application project, the approach was more informal. Across the project team, everyone understood that the test team would give the highest risk areas the most attention, the medium risk areas less attention, and the lowest risk areas perhaps no attention at all. Given that the author was working on a small project following a relatively agile system development life cycle, this kind of informality was appropriate.

RISK MITIGATION AND MANAGEMENT THROUGHOUT THE LIFE CYCLE

So far, readers have seen how the quality risk analysis process and resulting document can serve as a foundation for the design of tests. However, it can also serve as a guide for

the quality professional throughout the project. Let's look at seven ways one's analysis can help mitigate and manage quality risk.

First, why wait for testing to mitigate the important quality risks? As shown in the file security case study (see Figure 2), quality risk mitigation actions can include various quality assurance tasks. Requirements, design, and code reviews have proven helpful on many of the projects the author worked on. Adoption of and adherence to coding standards can help programmers avoid dangerous constructs. Defensive programming techniques such as pair programming, test-first programming, and stepping through the code with a debugger can help detect mistakes during code development. The more important the quality risk, the smarter it is to include multiple quality assurance tasks focused on it, at various stages of the life cycle, especially the early stages.

Second, as one designs and develops tests, he or she can use the quality risk analysis document to ensure completeness. Assume, for the moment, that he or she is following the approach shown in Table 3. In this case, each test case the test team developed should map to one or more medium-, high-, or very high-risk areas. Each medium-, high-, or very high-risk risk area should map to one or more test cases. The higher the relative risk, the more tests for that risk area. If one wants to know for sure that the testers are covering all the risks appropriately, ask them to keep track of this mapping, sometimes called a *traceability matrix* (see, for example, Craig and Jaskiel 2002).

Third, as one prepares to run the tests, he or she can use this traceability from risks to test cases to pick which tests to run first. A good rule of thumb for prioritizing test case execution is: Look for the scary bugs before looking for the trivial bugs. So, again following the approach shown in Table 3, the tests associated with the very high risks should precede those associated with high risks, which should precede those associated with medium risks.

Fourth, as one begins to run the tests, he or she should gather actual test results and find real—not potential—bugs. The kinds and number of bugs one finds in each risk area should provide feedback on the accuracy of the risk analysis. If there are a lot of bugs in unexpected places, perhaps the technical risk—that is, the likelihood of bugs related to some risk areas—is higher than previously thought. If there are dangerous bugs where

one thought there would be trivial ones, perhaps the business risk—that is, the impact of bugs related to some risk areas—is higher than originally thought? If one can trace the test results and bugs back to the risks from the first day of test execution, one can adjust risk analysis and testing focus based on what he or she learns (Black 2002).

Fifth, as one continues to run tests, he or she might find less time for testing than needed. Project team turnover, unanticipated specification changes, longer-than-expected testing periods, late deliverables from programmers and vendors: all of these events can put the squeeze on testing. By tracing the relative levels of risk associated with particular test cases, one can intelligently trim the amount of testing to be done in a way that minimizes the increase in risk.

Sixth, as one reaches the end of a development project, he or she sometimes finds the need to make one more change, add one last feature, or fix one last bug. Some amount of retesting (that is, regression testing) is usually required to make sure one hasn't broken anything that worked before (that is, haven't caused regression in the system). To select a regression test set, one can—and probably should—analyze the changes being made to determine the likelihood of regression bugs associated with particular changes, additions, or bug fixes. However, one can—and probably should—also use the quality risk analysis to pick the most important tests.

To cover the seventh and last benefit, notice that the author mentioned *traceability* many times in this section. The best way to capture and support all the benefits described here is to create a traceability database. A high-level entity-relationship diagram for such a database is shown in Figure 3. Notice how, in this diagram, one can relate quality risks to test cases, test cases to test results and bugs, and test results and bugs back to quality risks. This makes it possible to report test results not only in terms of the quantity of test cases (for example, run, passed, and failed) and the number of bugs (for example, found, fixed, and remaining)—which after all are fairly abstract quantities—but also in terms of the risks that one has mitigated through testing and where the risks remain high.

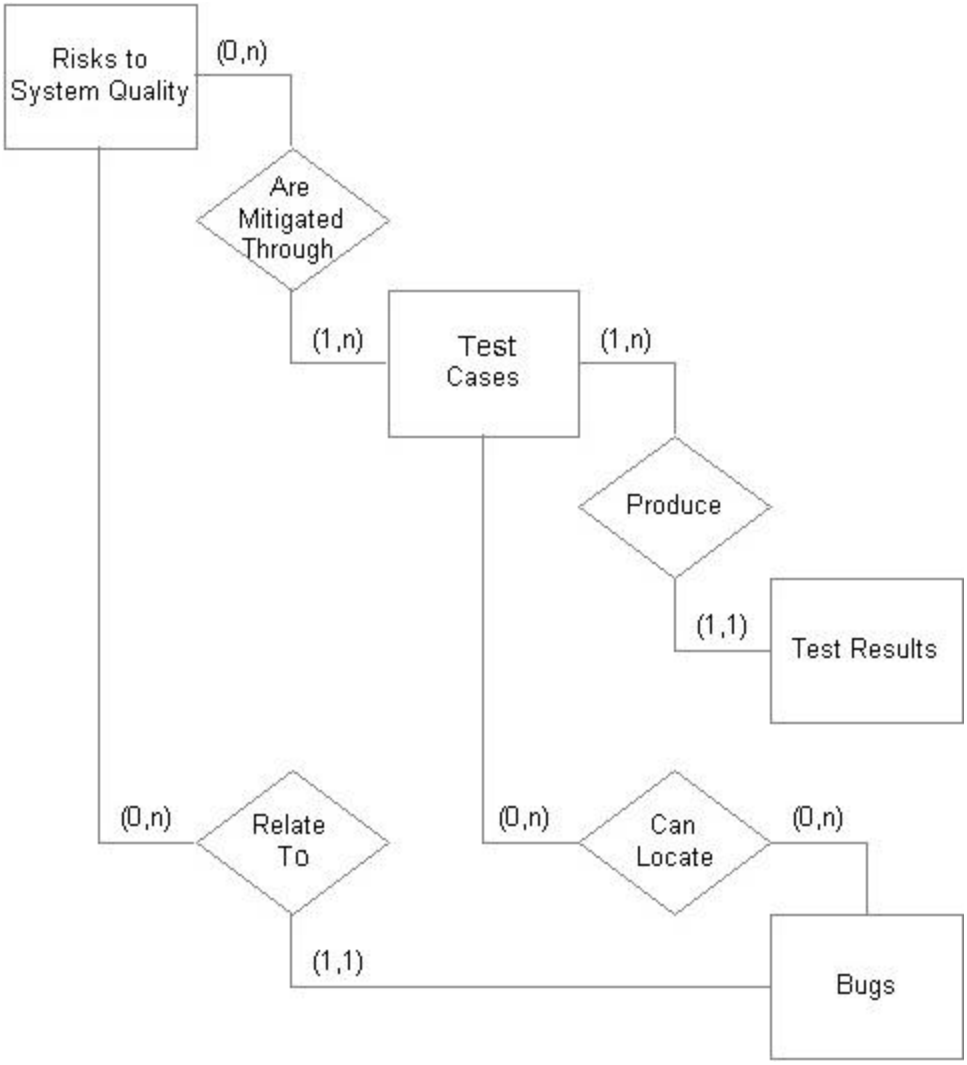

Figure 3 An entity-relationship diagram for relating risks, tests, results, and bugs

A Cautionary Case Study

There was one engagement where the quality risk analysis process broke down. The author's client had extensive written specifications. There was also a core of about eight people who had a shared vision of where the system was to go. However, all of the stakeholders were "too busy" to spend time on risk analysis. They asked that the author and his associates perform the risk analysis.

They did so, in part by interviewing people in person, on the phone, and via e-mail, but mostly by analysis of the requirements and design specifications. They then circulated the

quality risk analysis document for comments. Few stakeholders commented, because few read the document. Nevertheless, with this analysis being their only guide on where to focus, they developed tests and got ready to run them.

At that point, schedule and budget pressures caused the project managers to reduce the breadth and depth of testing. Those reductions did not follow the risk analysis at all. The project management team had no commitment to the risk analysis. They were not aware of what it could tell them. So, they arbitrarily made decisions about which tests to drop and which to keep based on their limited understanding of relative levels of quality risk. Based on this experience, the author has concluded that involving the right stakeholders and ensuring the right level of stakeholder participation is more important to the success of the quality risk analysis process than the availability of extensive documentation.

CONCLUSION

In this article, the author explained how and why one could use risk analysis as the foundation for testing and other quality assurance tasks. He discussed how to extend one's thinking about traditional project risk management to include the risks to system quality. He covered the factors of likelihood (technical risk) and impact (business risk) of potential problems that establish relative levels of quality risks.

Having laid out the general concepts, the author moved on to the mechanics. He examined five quality risk analysis techniques: informal; ISO 9126; cost of exposure; failure mode and effect analysis; and hazard analysis. He covered a general-purpose, adaptable process for performing a quality risk analysis.

Finally, the author looked at the many ways one can apply quality risk analysis. Quality risk analysis provides a foundation for smart test design and development. However, that's only a start. The author also showed how, throughout the project life cycle, one could continue to use the quality risks analysis to reassess, manage, and mitigate the risks to system quality.

REFERENCES

Black, R. 2003. *Critical testing processes*. Boston: Addison-Wesley.

Black, R. 2002. *Managing the testing process*, second edition. New York: John Wiley and Sons.

Craig, Rick, and Stefan Jaskiel. 2002. *Systematic software testing*. New York: Artech House.

ISO. 2000. *ISO 9126-1:2000 Information technology: Software product quality.* Geneva, Switzerland: International Organization for Standardization.

Jones, T. Capers. 1995. *Estimating software costs.* New York: McGraw-Hill.

Juran, J. M. 1988. *Juran on planning for quality.* New York: Free Press.

Stamatis, D. H. 1995. *Failure mode and effect analysis*. Milwaukee: ASQ Quality Press.

BIOGRAPHY

A 20-year software and systems engineering veteran, Rex Black is President and Principal Consultant of RBCS, Inc., a leader in software, hardware, and systems testing. For over a decade, RBCS has served its worldwide clientele with training, consulting, staffing, and on-site, off-site, and off-shore project execution expertise. RBCS' dozens of clients span sixteen countries on four continents, including Adobe (India), ASB Bank, Bank One, Cisco, Comverse, Dell, the US Department of Defense, Hitachi, NDS, Reef, and Schlumberger. His popular first book, *Managing the Testing Process*, now in its second edition, has sold over 17,000 copies around the world, including Japanese, Chinese, and Indian releases. His latest book, *Critical Testing Processes*, published at the end of 2003, has already sold about 2,000 copies, with Japanese and Russian translations on the way. Rex has also written numerous articles, along with presenting papers, seminars, and speeches at various US and international conferences.