Early and Effective: The Perks Of Risk-Based Testing

You Can Seize Control Of Your Testing Process With Better Use Of Resources

By Robin Goldsmith

Testing is our primary means of controlling system and software risks. Every testing authority explicitly states that testing should be in proportion to the degree of risk involved: The greater the risk, the more testing is needed. For example, the IEEE SWEBOK software engineering body-of-knowledge section on testing states, "Testing must be driven based on risk; i.e., testing can also be seen as a risk management strategy." (Version 1.0, May 2001, page 75)

Conducted correctly, risk-based testing enables better use of limited time and resources to ensure that the most important work is completed and that any tests that are skipped or shortchanged are of lesser import.

Planning Your Process

To determine which test cases are the most deserving of your available time and resources, your testing process should be planned and designed, largely through identifying, prioritizing and addressing potential risks.

To be confident in your testing process, you must think systematically about what needs to be demonstrated, rather than depend on the busywork paper-pushing that many seem to confuse with planning. Value is maximized by economically and appropriately documenting risks so they can be remembered, shared, refined and reused. Write no more than is useful—and no less.

Risk exposure represents the combination of impact (damage, if the risk does occur) multiplied by likelihood (that the risk will indeed occur). Much of risk literature involves variations on a useful but mechanical method for quantifying risk: assigning respective values to impact and likelihood, such as 1 = Low, 2 = Medium, and 3 = High, and then multiplying to produce a risk exposure score of 1 (very low) through 9 (very high). Figure 1 (page 26) shows the respective exposures graphically: Low = 1 box, Medium = 4 boxes and High = 9 boxes. As with most images, these graphic depictions can enhance understandability.

Robin F. Goldsmith, JD, is President of Needham, Mass.–based consultancy Go Pro Management, Inc. A prominent speaker and author of "Discovering REAL Business Requirements for Software Project Success" (Artech House, 2004), he advises and trains systems and business professionals. He can be reached at robin@gopromanagement.com.

However, pictures can be misleading: Some testers may adjust the scale as a form of gamesmanship. For instance, values of 1 = Low, 3 = Medium and 5 = High yield risk exposure scores of 1 (very low) through 25 (very high). As Figure 2 (page 27) suggests, an exposure of 25 boxes can seem far greater than an exposure of nine boxes, although a maximum 25 score on a 1–25 scale actually represents a risk identical to a maximum 9 score on a 1–9 scale.

Virtually everyone agrees on the concept of basing testing on risk impact and likelihood. However, testers continually encounter difficulties in effectively applying risk-based testing in practice. In explanation, many testing authorities imply that the way to identify and prioritize risks is obvious, and that too often, overly simplistic examples make a task seem deceptively easy—until you try to put the techniques in practice with a real system.

An even more significant hidden risk to the development and testing process occurs when testers are unaware of the possibility that they may not be adequately identifying and prioritizing risk-based tests. Risks that aren't identified can't be controlled, and risks that are identified too late can be overly difficult and expensive to mitigate. The complacency arising from unwitting overreliance on flawed processes can be a thornier problem than the flaws in the processes themselves.

To more fully gain the advantages of risk-based testing, testers and others involved in the development process must learn to thoroughly identify and reliably prioritize potential risks.

Picking Priorities

Accurately identifying potential risks is the first, most important and most difficult step in controlling risk through testing or other means.

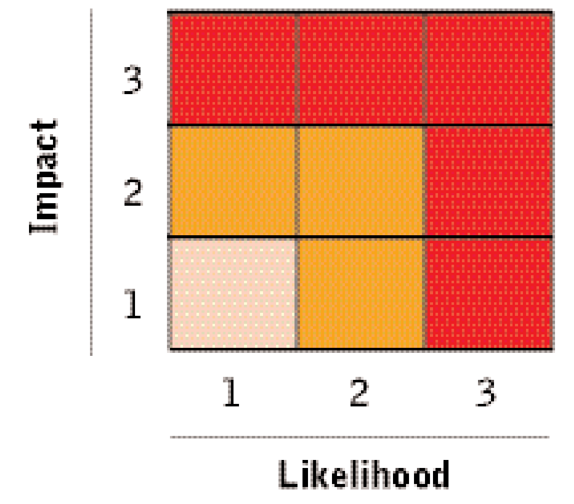
Risk literature is full of checklists of risks that commonly afflict various situations. While these lists can serve as helpful reminders of things to look for, they can also create a number of usually unrecognized problems. And, though they're intended to guide thinking, checklists can easily become a substitute for thoughtfulness. Many checklist users miss things because they overrely on their lists, uncritically assuming they address everything they need to cover, or failing to understand how to apply lists to their own situation.

Again, evaluating risk identification effectiveness to avoid mindless oversights is at least as important as having defined procedures, guidelines and checklists to follow in identifying potential risks.

Moreover, most checklists help with identifying potential risks, but not with prioritizing those risks. Thus, if you've found a checklist helpful, it may be worth the effort to expand the list items to aid prioritization. For example, a typical risk checklist item might be "Interfaces with other systems," which invites possibly unreliable subjective interpretations. For a more objective

Photograph by A. Navrin

FIG. 1: RISK PRIORITIZATION



analysis of interface risks, the item could be rewritten as:

Low risk = 0 interfaces with other systems
Medium risk = 1–2 interfaces with other systems
High risk = 3 or more interfaces with other systems

Checklists have a tendency to inadvertently intermix perspectives, which can seriously interfere with the prioritization process. Apples-versus-oranges confusion can be reduced by consciously conceptualizing risk identification perspectives and systematically approaching each perspective individually. Many published risk checklists reflect a dominant perspective that could be refined to facilitate apples-versus-apples comparisons.

The bulk of the literature on risk comes from a management perspective. People with engineering and operations orientations, including testers and non-engineering folks such as many business users, tend to concentrate on risks from a product/technical perspective. In part because generalized lists are unlikely to address an individual situation's specifics, business effects comprise the perspective that is most commonly neglected—and which, ironically, carries the most significant risks of all.

Business Effect Risks

The common failure to sufficiently recognize, understand and appreciate the importance of business effect risks virtually guarantees that all other risk-control efforts (such as testing) will be less effective.

Therefore, the first and most important test-planning and design risk-identification step is to determine the effects on your business if the system/software doesn't work as needed. Note that the issue is *what* the business requires, not product/system/software requirements that dictate *how* a presumed product or system solution is intended to work. Business effect risks answer the question, "So what if such and such happens—or doesn't?"

Checklists can inadvertently mix perspectives, which can interfere with the prioritization process.

Not only are we unaccustomed to thinking in business terms, even when we do, we may have difficulty thinking about effects. Thus, despite conscious direction to identify business *effect* risks, people tend to instead identify actions and events that are presumably *causes* of some damaging effects, though those effects are generally not articulated.

Intermingling risk causes and effects, which is a common checklist approach, also inhibits meaningful prioritization, because comparisons must be made among comparables. Since risk is an effect, it's essential that impact/likelihood prioritization be focused entirely on potential risk effects. On the other hand, mitigation approaches are based on those effects' causes, which in turn must be clearly understood.

Business effects are the most important of all risks because they provide a context for analyzing all risks. Without such a context, it's easy to miss the most important management and techni cal risks, and divert your attention to more visible but perhaps less important areas.

Management Risks

Much of the risk literature deals with management risks, such as lack of resources and time, long duration, large projects and faulty development processes, including poor estimation, inadequate testing and requirements changes. Since 9/11, articles on risk have tended to deal with security breaches, although in the aftermath of Hurricane Katrina, the emphasis has shifted somewhat to the risks of natural disasters.

The most common published checklists come from a management risk perspective, and almost all such lists unwittingly intermingle causes and effects. Attempting to use these checklists for risk-based testing poses several additional pitfalls.

First, except for a limited set of specifics such as security and disaster recovery, testing may not be a relevant response to mitigate many management risks. For instance, an understaffed project with an overly aggressive deadline will undoubtedly encounter quality issues, partly because testing has probably already been squeezed. More testing in general will help, but such general management risks don't help you determine *which* testing to increase.

Second, risk involves statistical probability. However, for most organizations, many of the typical management-risk checklist's items aren't risks, but certainties. Practically every project is underestimated, usually by a large percentage that grows as requirements appear to

change, always in ways that necessitate additional time and resources. Incorrect estimates result in allocating insufficient or inappropriate time and resources, which regularly leads to deliveries that are late (or nonexistent), over budget, and wrong.

Inadequate testing is both a symptom and a cause of the predictable problems of typical development practices. Squeezing testing is a common management response to underestimated projects that are running late, but being late also stresses development and produces still more errors, which the squeezed testing is in turn incapable of catching. By definition, such faulty processes will certainly continue to create the conditions that produce problems. Traditional risk-based testing is highly unlikely to be the means for ensuring the time and resources necessary to break this cycle.

Third, testers tend to approach matters from the bottom up, often mainly in terms of executable test cases, and checklists seldom recognize the interrelationships of various management risks. This often quashes awareness about the ways that testing could in fact help mitigate the conditions cited on management risk checklists.

For example, the main reason for inadequate budgets, resources and schedules is inadequate task definition and estimation, which stems mainly from inadequate understanding of the work to be done, which is generally caused by inadequate product/system design, which is mainly due to inadequate definition of the real business requirements.

Failure to adequately discover the real business requirements is often caused by a lack of awareness of their nature and of the vital need to identify them in detail. Instead, conventional wisdom concentrates on the system/software/functional requirements, which actually comprise a high-level design of the product/system that is *presumed* to be what is needed to satisfy the *presumed* but usually undefined real business requirements. Table 1 summarizes these distinctions. Focusing solely on any downstream issue such as high-level design addresses symptoms rather than causes.

By recognizing how these management risks are interrelated and largely stem from inadequate requirements, appropriate testing can be a powerful aid to mitigation through detecting requirements and design issues early, when they're easiest and cheapest to fix.

FIG. 2: RISK PRIORITIZATION GAMESMANSHIP

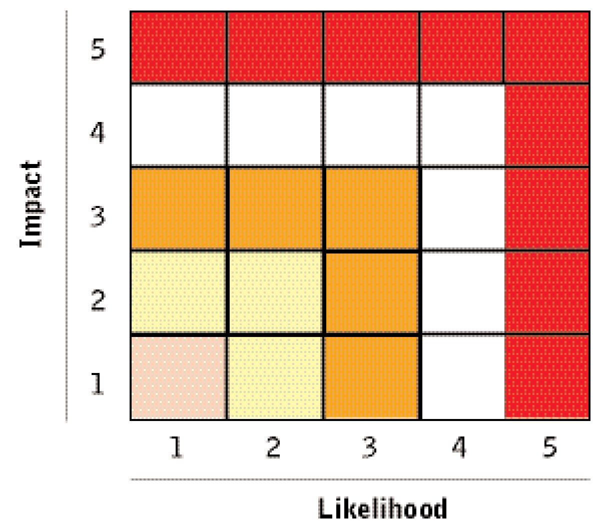


TABLE 1: TWO TYPES OF REQUIREMENTS

Business/User Requirements	Product/System/Software Requirements
Business/user perspective and language, conceptual.	*Human-defined* product or system perspective.
Exists within the business environment and thus must be discovered.	One of the possible ways (high-level design) to accomplish presumed business requirements.
When delivered, accomplished or satisfied, provides value by serving business objectives, typically solving expected problems, meeting challenges or taking opportunities.	Often phrased in terms of the external functions that the product/system is expected to perform; thus also called *functional specifications/requirements.*

Reviews

Reviews are the method used for "testing" or evaluating the adequacy of earlier development deliverables, such as requirements and designs. Many organizations consider reviews to be quality assurance rather than testing. Regardless, while certainly helpful, conventional requirements and design reviews tend to be far weaker than is usually recognized.

Some organizations don't review their requirements and designs at all; those that do ordinarily use only one or two weaker-than-realized techniques. Most conventional reviews miss many of the most important issues because they erroneously focus on product/system/software requirements' high-level design, rather than on real business requirements that provide value. Also, conventional reviews often concentrate mainly on matters of form, such as clarity and testability, rather than substance. And too often, the rest of the organization disregards such format-oriented review results as trivial nitpicking.

In contrast, Proactive Testing (see "Mindset and Methodology" sidebar, page 30) can break this vicious cycle, with more than 21 ways to evaluate the adequacy of business requirements, and more than 15 ways to evaluate designs. Many of these are more powerful, content-oriented techniques that also spot the overlooked and incorrect requirements and designs that format-oriented reviews tend to miss.

Overlooked and incorrect business requirements often relate to significant business effect risks. Business and management who appreciate the importance of these issues can look to Proactive Testing to spot red flags before they become predictable problems.

Moving Beyond Reactive Testing

Testers tend to come at risks from the perspective of the product or system being developed. Conventional risk-based testing, as espoused by essentially every testing authority, analyzes a system's design to identify risks related to the system's functionality/features (which are sometimes called *requirements risks*) and/or the system's technical structure (its components and how they're put together), and then tests the higher risks more frequently.

Such risk identification may be done individually or in a group. Ordinarily, the system's functionality is defined based on user interface design, where each menu or GUI button choice is deemed a separate feature to analyze for risk. To identify structural risks, organizations often have a wider variety of technical design artifacts. For example, architecture and network diagrams, database structures and object models each may be reviewed to identify different types of component risks.

While sometimes guided by checklists, product/technical risk reviewers are often simply directed to apply their experienced judgment to identify those features and components that will have a significant impact if they go wrong, as well as those that seem likely to go wrong.

For each test-design specification to be tested, a set of test-cases is identified that will prove that the feature, function or capability really works.

Product/technical risk checklists generally prompt attention to things that are new or changed, complex or large, or used frequently. They also focus on interfaces, dependencies and historical problems. While definitely good advice, such checklists are often difficult to apply to particular systems, especially with regard to functionality.

This conventional risk-identification approach is necessary and important, but is generally less effective than realized because it's reactive. One reason for this reactivity is that, much to their frustration, testers often don't come on board until the end of the development process, after problems have already been incorporated in the code. And, in a side effect of such reactive late involvement, risk identification can become overwhelming, with masses of details dumped on the testers all at once. Without effective ways to get a handle on the whole thing, it's easy to get mired in detail and lose sight of bigger issues.

In addition, the greatest weakness of most conventional testing is that risk identification merely reacts to and therefore tends to be limited to whatever's in the design or code. Being guided by the design—or, even worse, the code— can lead risk identification away from recognizing errors and omissions.

The Proactive Plus

In addition to more effective reviews of requirements and design, Proactive Testing includes but goes beyond conventional testing's reactive product risk identification. Taking business and management as well as product perspectives, Proactive Testing uses specific techniques that help identify many important risks that reactive approaches miss. Moreover, Proactive Testing addresses risks earlier and at multiple points in the development process, scaled to need.

Consequently, whereas conventional reactive testing enables more frequent testing of higher risks, Proactive Testing enables testing higher risks not just more often, but earlier, when problems are easier to fix.

Proactive Testing can employ the approach suggested by IEEE Std. 829-1998 to organize the test planning and design process. When applying this proven project-management practice, testing is systematically broken down into smaller, more manageable pieces. At each point in the process, risks are identified and prioritized to guide a successively narrower focus on the most important areas to test.

This structure provides a way of thinking, not a mandate for generating paperwork, as some people (mis)interpret the Standard. However, it's important to write information down so it can be remembered, refined and reused. In Proactive Testing, you write as much as is helpful—no more, but no less.

The earliest and most important Proactive Testing risk analysis occurs as part of master test planning, which is performed most effectively as soon as you have a high-level system design.

The master test plan is the project plan for the testing project, which is a subproject within the overall development project. Each key risk to the system being developed is addressed in a detailed test plan—for each unit, integration, system and special (such as load or security) test. Detailed test plans dealing with the highest risks should be tested more often and earlier in the life cycle.

Because Proactive Testing identifies big risks early, test plans dealing with the higher risks can be tested not only more often, but earlier in the life cycle.

Powerful Proactive Testing techniques identify the biggest risks, including up to 75 percent of the showstoppers that conventional reactive testing ordinarily overlooks because typical development processes have failed to address these critical risks in the system design. Moreover, Proactive Testing identifies the risks early enough to let testing drive development. That is, while it's important to merely catch showstoppers prior to production, Proactive Testing can increase value dramatically by catching the showstoppers in time to prevent writing affected related code that otherwise must be thrown out and rewritten.

The value of spotting these ordinarily overlooked risks is greatest in conjunction with high-level design, but remains significant throughout development. Payback from preventing a showstopper risk is always valuable, even if it's not found until the day before the show would have been stopped.

While checklists can help spot generic types of risks that afflict many systems, Proactive Testing also emphasizes understanding the specifics in order to reveal the system's unique big risks, which are perhaps the most commonly overlooked. The written risk findings can become a checklist for future risk analyses, but beware that overreliance on checklists can prevent the meaningful, content-based risk identification that Proactive Testing uses to spot so many otherwise overlooked risks.

It's especially valuable to involve a broad range of participants in identifying these big risks, since Proactive Testing facilitation prompts each different view to identify risks that other participants would probably miss. Experiencing the discovery of big risks wins instant advocates for early, Proactive Testing involvement. In addition, a group representing a broad mix of involved stakeholders provides the most effective risk prioritization.

Scaling and Carrying Through

Each of these big-risk areas can be addressed in a detailed test plan, which should be just elaborate enough to be helpful; no more, but no less. The higher the risk involved, the more analysis will be needed. Lower-risk areas can be given respectively less attention, even if only to the point of documenting the conscious decision not to analyze them further.

Because Proactive Testing identifies these big risks early, detailed test plans dealing with higher risks can be tested not only more often, but earlier in the life cycle.

To do this, development must structure and schedule its work to create the higher-risk components earlier so they can be tested earlier. One powerful way to do this is to define the sequence of builds based at least in part on Proactive Testing risk prioritization.

Introducing New

yKAP® 2.4

Your Kind Attention Please

The Antidote for Over-Priced, Under-Featured, Hard-to-Use Bug-Tracking Software

yKAP 2.4

The next generation Web-based Defects and Issues Tracking Software

- Superior Functionality.
- Powerful Performance.
- Easy Customization.
- Effortless Collaboration.
- Unbeatable Value Pricing

Compare yKAP 2.4 and Save Up to 45% Off Your Purchase Price! Visit, download & save:

www.yKAPbugTracker.com

DCom solutions

MINDSET AND METHODOLOGY

Proactive Testing is as much a mindset as a methodology for risk-based software test planning, design and management. Positioning testing within an overall quality and business value context, Proactive Testing emphasizes WIIFMs (What's In It For Me) that win over users, managers and developers. Rather than the traditional adversarial perception of testing as an obstacle to delivery, Proactive Testing can save you aggravation while helping you deliver better systems cheaper and quicker.

The Proactive Testing life cycle embraces true agility to minimize wasted effort by enabling the development process to build more right in the first place. At the same time, by intertwining more thorough and effective tests of each development deliverable at multiple key points, it can economically detect more errors.

Proactive Testing uses higher-payback test-planning and design methods that augment conventional testing techniques to find numerous test conditions commonly overlooked by traditional testing methods, allowing more effective scaled risk prioritization and repeated refocusing on the most important tests. Moreover, anticipation can promote reuse and let testing drive development to prevent problems or catch them earlier, when they're easier to fix.

By maintaining a real process perspective, Proactive offers a fresh take on conventional User Acceptance Testing (UAT). Rather than a subset of technical testing (unit, integration, system/special), UAT should be kept independent. With truly proactive, user-centered planning/design of UAT up-front for execution at the end, Proactive Testing can help improve requirements and designs while providing a more thorough and effective double-check of the developed system.

Finally, Proactive Testing accepts responsibility for results, which includes meaningful measurement and active management.

Each detailed test plan identifies the set of test design specifications that, taken together, give assurance that the subject of the detailed test plan works. Each test-design specification describes the set of test cases (inputs/conditions and expected results) that must be conducted to ensure confidence that a feature, function or capability works.

Just as Proactive Testing methods identify numerous otherwise-overlooked big-risk areas, additional special techniques successively identify many risks in ordinarily overlooked features, functions and capabilities. Conventional methods that fail to identify these risks can't include them in risk prioritization or mitigation.

The fully identified set of test design specifications is then prioritized according to the risks they address. Those dealing with the highest risks are tested earlier and more often. In addition, by structuring test-design specifications to be reusable, Proactive Testing offers an additional way to increase the amount of testing in the time available.

For each test-design specification to be tested, a set of test cases is identified that will prove that the feature, function or capability works. The test cases are ranked based on the risks they address, with the highest-risk test cases executed earlier and more often.

Prioritization Alternatives

It's common to use the well-known impact-multiplied-by-likelihood high/medium/low ratings risk-prioritization approach described above. However, this method frequently takes too much work and is too difficult to achieve consensus on, especially with a broadly representative group of stakeholders.

A further concern with the technique is that rating each risk individually (as does the commonly used mandatory/desirable/ideal differentiation technique) can actually prevent effective prioritization because each risk viewed in isolation tends to be rated high.

By definition, prioritization demands distinctions among risk choices, so it's advisable to use a risk prioritization technique that produces a ranking. As a technique, ranking raises two further issues.

First, for meaningful ranking, your selections must be limited to a manageable number, typically no more than 15 and preferably closer to seven (a range that numerous scientific studies have found to be the parameters of the typical human attention span).

Second, since rank alone doesn't convey relative differences in importance, it's more valuable to use a risk analysis technique that enables ranking based on quantified importance. Such methods can be gross or precise; formal or informal.

The simplest technique is to gain group consensus on a relatively small subset of risks that are very important relative to the fuller set of all identified risks. The subset of highest risks can then be ranked, or ranking can be skipped if it's certain that all the subset members will be addressed.

You can also try a more formal technique: the *100-point must system.* Each participant is given 100 points that he must allocate among the risk choices in proportion to importance. All 100 points must be allocated, and at least one point must be allocated to each risk choice. Wide discrepancies are then discussed and adjusted as appropriate, followed by arithmetic averaging of respective risks.

Managing the Process

Effective risk-based testing doesn't end with risk identification and prioritization.

Effective risk-based testing doesn't end with risk identification and prioritization.

Tests of the highest risks must be planned, designed and executed. As time and resources allow, lower risks must also be tested.

Most importantly, testing effectiveness must be monitored and measured, issues must be detected in a timely manner, and appropriate actions taken.

Ultimately, the time and cost of testing must be weighed against its benefits. Did the testing catch problems that would have made a significant impact? Would the problems have been found without risk identification and analysis? Would the costs of those problems' occurrence exceed the costs of detecting them? What kinds of damage occurred in spite of risk-based testing? And what additional/different risk analysis and/or testing would have prevented the damage? ☒