Inside Scoop on Risk-Based Testing

Robin F. Goldsmith, JD

President, Go Pro Management, Inc.

robin@gopromanagement.com

Abstract

Prioritizing tests based on risk is an essential aspect of all testing rather than a separate testing technique. Why then do so many people mistakenly think of it as something separate to be used only when your testing back is against the wall? Moreover, many find that conventional descriptions of how to assess and prioritize risk don't work in the real world.

This article reveals why typical risk approaches fall short and ways to make them actually work. It describes the conventional view of risk-based testing and why it's wrong, suggests how to take advantage of the fact that all testing is risk-based, and compares the value of various ways to assess risk for testing.

Biography

Robin F. Goldsmith, JD *is one of the few with legitimate credentials in both QA/testing and requirements/business analysis/value, as well as project and process management/measurement/improvement. Uniquely positioned to help you get REAL value right results right, he works directly with project teams assisting, coaching, and training business, systems, and project professionals on risk-based Proactive Software Quality Assurance™ and Proactive Testing™, REAL requirements, project management and leadership, REAL ROI™, metrics, outsourcing, and process improvement. Go Pro Management, Inc. President and SQGNE past President and current Vice President, Robin is author of the Artech House book* Discovering REAL Business Requirements for Software Project Success *and the forthcoming* Cut Creep—Write Right Agile User Stories and Acceptance Tests.

Conventional View of Risk-Based Testing

Testing is the main means of controlling software risk. Consequently, identifying and prioritizing risk is an important element of determining what to test and how much to test respective elements.

Seemingly related to this, m term "Risk-Based Testing" is widely-known and used. However, the term seems to be widely misused. I realized this thanks to repeated experiences with the following Dictated Project Constraints exercise in my *Managing Testing Projects* full-day seminar.

> You are the Test Manager for a major system with a scheduled 6 months total project duration. Your Test Group is to do the System Test in months 5 and 6. You have the following concerns:
>
> 1. Two months of testing by your small group seems way too little.
> 2. You suspect the developers will be late getting the code to Test.
> 3. Previous projects had quality problems when testing was squeezed.
>
> *Identify:*
>
> *1. The points you would raise to convince management to give you what you need.*
>
> *2. Management's likely responses.*
>
> *3 Why?*

Participants typically cited the following points to convince management to give them what they need:

- Inadequate amount of testing in past led to trouble.
  - Need more testing.
- Two months testing is too little.
  - Need more test time.
- Test group is too small to do needed testing.
  - Need more testers.
- Many errors are in requirements and designs.
  - Involve testers earlier to make sure requirements are testable and known.
- Make Development deliver code in time to test it, preferably in multiple smaller and mainly earlier releases.
- Cut the functionality to what we can test in given time.

Participants uniformly agreed on management's likely response:

- **No!**
  - You're not getting more time or testers to enable doing more testing.
  - We're not changing development to involve testers earlier or to make Development deliver code earlier to it can be tested more thoroughly.
  - We can't cut functionality. It's all needed.
- Suck it up and do a better job of testing with the time and resources you've got.

Many testers in different classes and from different organizations all seem to have similar perceptions of testing's likely requests and management's likely responses to them. The testers tend not to have much insight into why management responds as it does except agreeing that management doesn't understand, or apparently value, testing. They also agree management should understand and value testing, and by implication should grant testing's requests.

At about this point in the exercise's discussion, since they're getting no satisfaction from management, *as a last resort* participants say something like,

- "Then, we'll do risk-based testing, just test the highest risks, not the rest."

Reiterating the Conventional View of Risk-Based Testing

This common response to the seminar exercise demonstrates the common view of risk-based testing, that it:

- I is a special technique which somehow is different from other testing
- Occurs only in conjunction with explicit risk analysis
- Generally is used infrequently and as a last resort when other approaches have been exhausted.

I contend that this definition not only is wrong but indicates additional issues that limit testing's effectiveness. That is, risk-based testing is not special or different from other testing, need not be done explicitly, and should not be considered a last resort.

All Testing Is Risk-Based

Rather than being special or different, basing testing on risk is inherent in all testing. That's because testing is our main means of reducing risk of systems and software. There's a simple relationship: the more risk, the more testing is needed.

Thus, whether or not recognized, conscious, or explicit: **all testing is risk-based!**

When we decide what to test, what to test first/next, how much to test it, and what and how much to retest, we're deciding based on the risks we perceive. The question is how adequately we've identified and analyzed the risks.

Too often, the answer is that we haven't identified and analyzed the risks very well. That shortcoming is exacerbated when the act of identifying and analyzing risks is not recognized, conscious, or explicit. Then it's much more likely that we overlook and misunderstand risks that testing could help reduce.

Testing's value is primarily driven by risk-based determination of what and how much to test. Only then is it worthwhile to address how best to perform relevant tests within budget, time, skill, and resource constraints.

Classical Risk Management

All risk management involves six steps:

1. Identify potential risks
2. Prioritize the risks, qualitative & quantitative
3. Assess the causes
4. Accept risk or define a mitigation strategy
   a. Avoid doing the risky thing
   b. Control the risk's likelihood and/or impact
   c. Transfer some or all of the risk
5. Define a contingency plan
6. **Implement** the strategy, monitor and control

How well potential risks are identified in Step 1 determines how effectively risks can be managed. Consequently, most risk management training emphasizes ways to identify potential risks; and most risk management weaknesses relate to missing or misunderstanding potential risks.

Testing is concerned with mitigating a risk's impact or likelihood by demonstrating whether the risk comes true in certain circumstances which affect the risk's impact or likelihood. When testing reveals a risk will come true, then further actions are needed to prevent the risk from occurring in production. Such actions include recoding, often in response to redesign.

Steps 1-5 are planning activities. Planning is necessary but not sufficient to manage risk. Thus, after planning, the next most important determinant of risk management effectiveness is how well the planning is implemented. Implementation includes monitoring and controlling performance of the planned risk management activities. In turn, that includes recognizing the need for and making relevant adjustments.

It's common to keep risk-related information in a "Risk Register." Whether or not explicitly designated as a risk register, it's valuable to capture the set of identified risks and analysis of them. Keeping this information together in identifiable form aids answering questions that may arise regarding the risks; and it provides a convenient way to guide and track risk-related activities, especially testing.

Elements of Risk

Risk, or technically *Risk Exposure*, is the interaction of risk's two elements: *impact* if the problem occurs *times* the *likelihood* that it will occur. It's common to portray the interaction of these elements graphically in a Risk Matrix, such as shown in Figure 1.

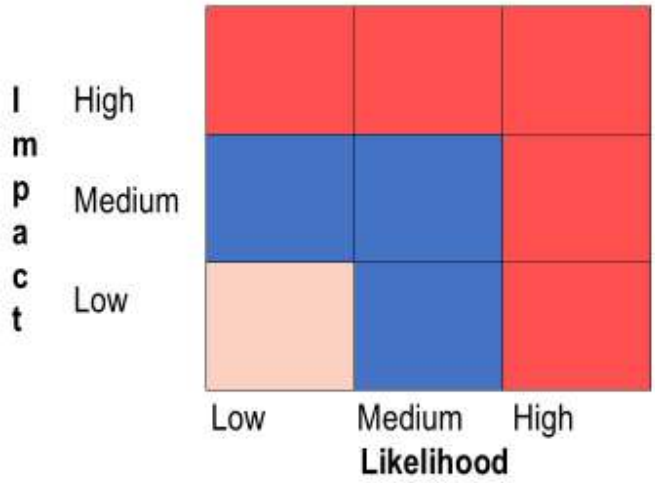

Figure 1. Risk Matrix

Impact increases the more severe damage is, which often is a function of the number of people affected. Sometimes an impact may not be inherently severe but can cause severe consequences. The software industry tends to pay special attention to impact regarding the cost, time, effort, and ability to fix a problem that has occurred. The availability of workarounds can reduce impact.

Likelihood of a problem increases with size and complexity. Technology which is new to the world is more likely to have problems than technology which has been around long enough for many of its problems to have been detected and fixed. Even more mature technology is still more likely to have problems when it is new to an organization which has not yet had a chance to become familiar with the technology. Products and people which have had problems previously are more likely to have (often similar or related) problems. Workers who lack suitable skills, motivation, and methods are more likely to have problems.

Software testing traditionally examines the risks of product *features* and *components*. Features typically tie to key functions the product performs. Feature risk sometimes is referred to as "requirements risk" and frequently is identified as a system;s menu

choices. Components relate to how the software is built, that is the technical pieces implementing the functionality. The impact times likelihood of each identified feature and component risk ordinarily is determined in order to prioritize the risks so *the highest risks can be tested most*.

Qualitative vs. Quantitative Ratings

Qualitatively rating (typically as high, medium, or low) each individual risk's impact and likelihood is probably the most common risk analysis technique. It's also common to depict these ratings graphically in a Risk Matrix as shown in Figure 1. The graphical matrix can aid communication of findings but does not help the fundamental difficulty of determining degrees of impacts and likelihood.

Many organizations maintain checklists of common risks, which often are based on some of the many risk checklists found on the Internet. Such checklists can aid identifying risks that otherwide might be overlooked. However, over-relying on checklists can interfere with identifying risks specific to one's project.

Moreover, such checklists seldom provide good guidance for assessing impact and likelihood of those risks. Instead, risk analysis usually relies on subjective judgment by individuals with varying levels of relevant knowledge. In additionr, the reliability of such judgments can be further limited when done implicitly, frequently even unconsciously.

In general, *quantitative* measurements are considered preferable to qualitative ones. Quantitative measures are in numbers, as opposed to seemingly less precise qualitative measures such as high, medium, and low. Would you agree that Figure 2 seems more precise than Figure 1?

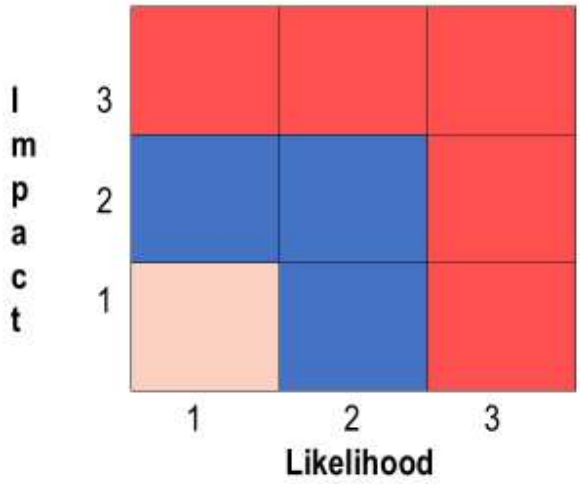

Figure 2. Quantitative Risk Matrix

Note that for the most part, the distinction between qualitative and quantitative risk ratings is fairly illusory. It's illusory because the typical approach to quantification is to make a qualitative judgment and then "quantify" it by assigning a score for each qualitative value. Typical scoring is 1 for low, 2 for medium, and 3 for high impact or likelihood.

Risk exposure, or simply risk, is calculated by multiplying the impact score times the likelihood score. For this, quantifying has an actual advantage, because it provides an easily expressed and understood measure of risk exposure. Thus, multiplying ratings of 1-3 gives a range of risk exposure scores from 1 through 9. A score of 9 is clearly higher risk than a score of 1, 2, 3, 4, or 6.

In contrast, describing the same degrees of risk exposure qualitatively is more cumbersome and less-readily understood. It's harder to recognize the relative risk exposures when stated as low-low, low-medium, low-high, medium-medium, medium-high, and high-high.

Additional Risk Matrix Considerations

As shown in Figure 3, it's common to shade the risk matrix boxes to indicate risk exposure. Typically, the high-high box in the upper right-hand corner is shaded red. The low-low box in the lower left-hand corner is shaded green. The remaining boxes in

the middle of the diagram are shaded yellow. A common variation would also shade the high-medium boxes red and the low-medium boxes green.

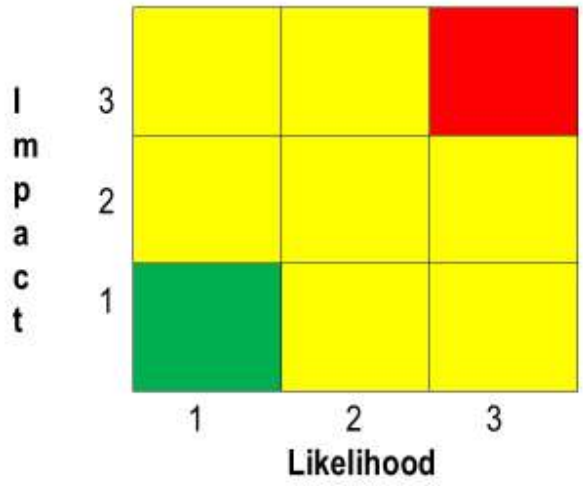

Figure 3. Highlighting Risk Exposure

Such a simple visual picture of risk is very appealing and can be useful. However, caution, it is subject to disingenuous manipulation. Consider what happens to the diagram when the scoring is changed to low = 1, medium = 3, and high = 5. Now the diagram as shown in Figure 4 is a 25-box five-by-five matrix, which graphically implies that a high-high score of 25 is higher risk than an actually identical high-high risk score of 9 on a three-by-three matrix. The effect is even greater when a ten-by-ten scoring matrix is used.

ffffffffffffffff

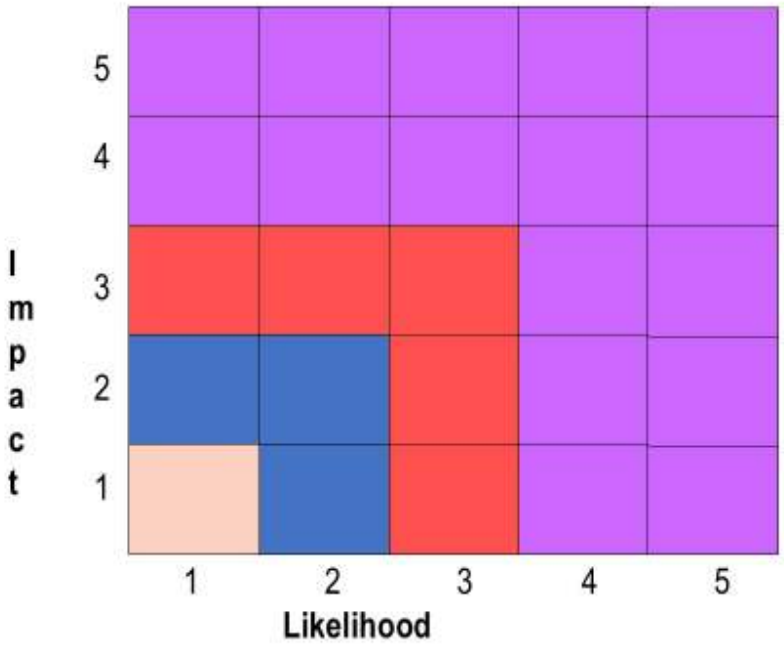

Figure 4. Varying Risk Matrix Appearance

As an aside, a useful but also possibly deceptive scoring approach is to make the scores non-linear. Usually this is done to increase emphasis of very bad outcomes. For instance, the scores might be low = 1, medium = 3, and high = 10 or even 100. In this way, a truly unacceptable riskxz, such as global thermonuclear war, would easily stand out visibly from less severe outcomes, even when likelihood is low.

Another common questionable use of these methods is to represent the truly-artificial "score" as a percentage, rather than arbitrary "points." Especially when using a ten-by-ten scoring matrix with possible score of 100, depicting risk exposure as a percentage makes it seem more accurate, when in fact such "percentages" are not real. In contrast, some sophisticated organizations, such as insurance companies and airlines, truly use quantitative and percentage measures o risk, which are calculated reliably based on large amounts of relevant data. Regardless, people tend to place greater confidence in risk representations where numbers are assigned as opposed to qualitative scores. It can be valuable to efine objective characteristics indicating the respective high, medium, and low levels of impact and likelihood. Such objective scoring helps reduce variability of judgments among raters and thereby enhances risk analysis reliability and repeatability. For example:

Low	= cosmetic,	interfaces with 0 other systems
Medium	= impedes working,	interfaces with 1-2 other systems
High	= won't work,	interfaces with 3 or more other systems

Rating vs. Ranking

Prioritization is based on analyzing the amount of risk exposure per risk item. The most common method used for determining amount of risk is *rating* and multiplying the impact times likelihood. As discussed above, rating is inherently unreliable when it's done subjectively without objective guidelines. Even when done objectively, though, rating has a number of additional issues limiting its usefulness.

Looking at each choice one-at-a-time takes a lot of time and effort. Even with objective criteria, ratings are often inconsistent and not repeatable, in part because people's evaluations tend to shift based on items they've already evaluated. That is, context matters; evaluating in a different sequence can produce different ratings for the same items.

A bigger issue with rating multiple items is that it often fails to indicate priority. It's common for every item, except perhaps one or two, to be rated high impact-high likelihood. Prioritization requires a way to say an item is higher risk than the remaining items, and so forth; but there's no way to distinguish respective priority sequence of items with equal risk ratings.

Effective prioritization requires forcing distinctions, which is what *ranking* does. Note, Agile's "relative sizing" is essentially a ranking technique. When estimating the "size" of work needed to implement a user story, Agile defines size with respect to the size of another, presumably known, user story. Agile tends not to estimate risk separately but could do so using the same relative approach.

However, ranking also has issues. Continually taking multiple items into account tends to seem harder and take longer than rating each item one-at-a-time. However,, that seeming difference in effort evaporates when one recognizes how much frankly unnecessary additional effort is needed to prioritize equally-rated items.

For ranking to be practical, the number of choices to be ranked needs to be limited to people's span of attention, often referred to as "the magic number seven plus or minus two." If there are too many choices to be ranked, they need to be grouped to enable comparison.

Other Confusions Regarding Risk

Most of what is written and taught about risk deals with *project management risk*. That mainly involves lack of suitable time and resources. While these indeed are relevant to performing testing and may relate to the total amount of testing needed, they do not identify specifically what to test or distinguish priorities for testing more and earlier.

It's common for organizations to rely on checklists to identify risk items.

However, checklists tend to focus on *causes* and triggers of risk occurrences. Causes are what can be addressed to mitigate risks, such as by testing, but are not a suitable

basis for determining risk priorities. Rather, priorities must be based on comparing *effects*—impact times likelihood risk exposure. Very often those identifying risk causes fail to then determine their respective effects which are needed for meaningful prioritization.

Reactive vs. Proactive Risk Analysis

Typical testing risk analysis is *reactive* and relatively weak. Reactive risk analysis tends to come late in the development process, usually reacting to code which already has been written. In the limited time usually available for testing, testers

- Create as many test cases as they can think of
- Prioritize those test cases based on risks they address
- Run the highest-priority tests first and more until they run out of time.

Such traditional reactive testing does catch defects, largely because the development process makes so many; but it also usually misses many of those defects. Reactive testing is expensive, in part because effort often is expended creating test cases that end up not being run. The bigger expense of reactive testing relates to the effects of *risks that it misses* and must be addressed later, when they're harder and costlier to fix.

Instead of starting with time-consuming creation of test cases, each of which deals with a small risk, *proactive testing* looks first at identifying and prioritizing *large risks*. Special techniques which are beyond the scope of this article help identify large risks that ordinarily are overlooked. For the highest priority large risks, similar techniques are applied to identify and prioritize *medium-sized risks*, again including many that ordinarily would be overlooked.

In turn, for the highest-prioritymedium-sized risks, similar techniques are applied to identify *small risks. Small risks are addressed by individual test cases*. Thus, proactive testing only spends the most time-consuming part—creating test cases—on ones that will be run and address the truly highest priority risks. Moreover, this approach prioritizes far more effectively because it considers many ordinarily-overlooked risks.

Proactive Testing's benefits increase further because proactive risk analysis can be performed before a line of code is written. As such, development can avoid many of the design errors that typically produce defects. Moreover, proactive testing enables shifting the sequence of development to build the highest risk pieces first, so they can be tested to catch remaining fewer problems earlier, when they're easier and cheaper to fix. Many of those prevented and early-caught defects otherwise would have necessitated especially time- and effort-consuming redesign and recoding. The amount of avoided work becomes even larger when prevented and early-detected defects also would have affected other pieces of code.

Instead of writing related code and then having to redesign and rewrite it when risks come true, related code isn't written until Proactive Testing risk analysis has shaken out

the design. In these ways, by *letting testing drive development*, risk-based Proactive Testing enables delivering better software quicker and cheaper.