Risk-Based Test Reporting

Paul Gerrard. 14 February 2002.

Suppose we have done a risk analysis and all tests for all test stages are related to a risk. We can obviously say at the start of system test execution say, that none of the test objectives have been met. Because test objectives all relate to a single risk, we can therefore say:

At the start of test execution, we presume that all the risks to be addressed by this phase of testing still exist.

That is, all known product risks are outstanding. With this assumption, we are saying that the system is "guilty until proven innocent" or put another way, the system is entirely unacceptable. This is obvious perhaps, but why is this important?

On the first day of testing, we can say, "we have run zero tests, here are the outstanding risks of release". As we progress through the test plan, one by one, risks are cleared as all the tests that address each risk are passed. Halfway through the test plan, the tester can say, "we have run some tests, these risks have been addressed (we have evidence), here are the outstanding risks of release." Suppose testing continues, but the testers run out of time before the test plan is completed. The go live date approaches, and management want to judge whether the system is acceptable. Although the testing has not finished, the tester can say, "we have run some tests, these risks have been addressed (we have evidence), here are the outstanding risks of release." The tester can present exactly the same message throughout the test phase, except the proportion of risks addressed to those outstanding increases over time.

How does this help? Throughout the test execution phase, management always have enough information to make the release decision. Either management will decide to release with known risks, or choose not to release until the known risks (the outstanding risks that are unacceptable) are addressed.

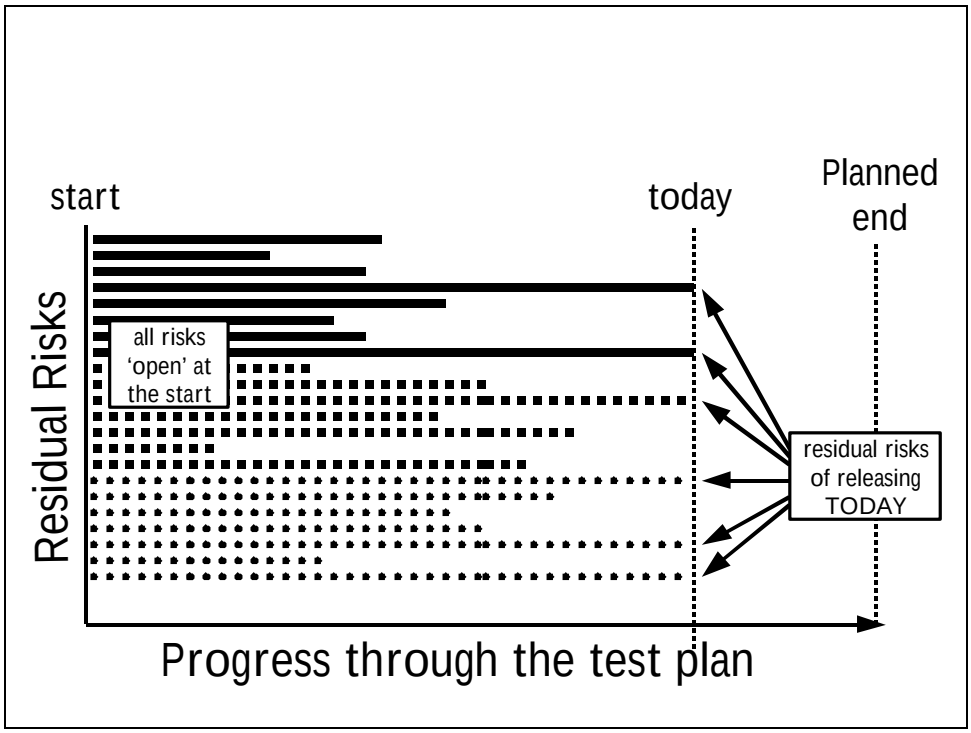

Figure 1. Risk based Reporting.

A risk-based test approach means that cutting testing short does not preclude a rational decision from being made. It just makes the decision to release less likely.

How might you report risk-based test execution progress?

In Figure 1, you can see a diagram representing progress through the test plan, but in the form of risks addressed over time. Along the vertical axis, we have the known risks of release, to be addressed by the testing. On the first day of testing all risks are outstanding. As the test progresses along the horizontal axis, you can see risks being eliminated, as more and more tests complete successfully. At any time during the test, you can see the current risk of release. "Today", there are six risks remaining. If the risks represented by the solid line are 'critical' risks, then it is clear that the system is still not yet acceptable. The tests not yet executed or not yet passed that block acceptance are clearly the high priority tests.

If we identified many risks in our project, and in a large, complex project you might identify between 60-80 risks, these risks are likely to be addressed across all the development and test stages. So, the horizontal scale might include all stages of testing, not just system or acceptance testing. As the project proceeds, the risks that are down to the developers to address through unit and integration testing are as visible as those in acceptance. The value of reporting against all risks in this way is that the developers, system and acceptance testers all see clearly the risks for which they are responsible. Management too, has visibility of the risks and can see risks being closed as test evidence is produced. If test evidence is not produced, risks will remain open. If you have problems in your organization with developers doing testing badly or not at all, this form of reporting might encourage them to test (or test better) and produce the required information to management.

In many projects, the development activities are defined in terms of coding tasks. Sometimes the tasks are described as "code and test module XXX". Well, we know what developers really like to do, don't we? Coding gets 90% of the effort and testing is squeezed yet again. To avoid this, we always recommend that code and test activities in project plans be defined as separate tasks (sometimes planned and performed by different people), especially at the component level. For critical components, we can document the test objectives derived directly from the risk assessment in component test plans. We can reference those risks on test reports sent to senior management. If managers pay attention to these risks, developers might pay more attention to component testing.

Consider what might happen if, during a test stage, a regression test detects a fault. Because the test fails, the risk that this test partially addresses becomes open again. The risk-based test report may show risks being closed and then re-opened because regression faults are occurring. The report provides a clear indication that things are going wrong – bug fixes or enhancements are causing problems. The report brings these anomalies directly to the attention of management.

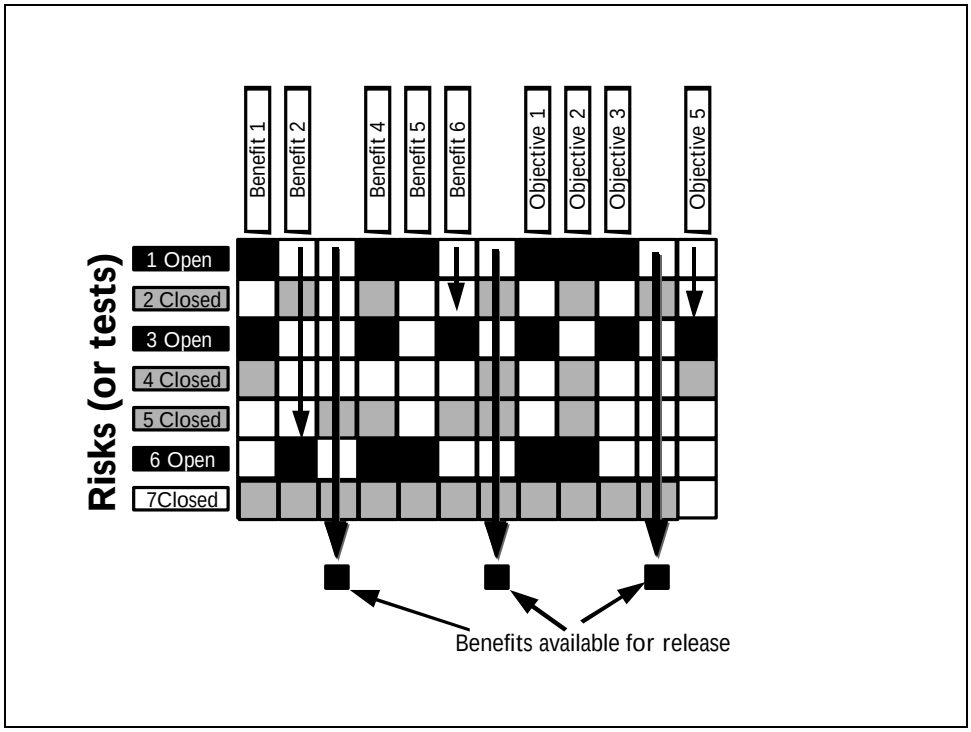

Figure 2. Benefit/objectives based test reporting.

In figure 2 we present a matrix of risks that block the cardinal objectives and benefits of the project. Reading along the top row of the matrix, the first risk that is open blocks benefits 1, 4 and 5 and blocks Objectives 1, 2 and 3. The second risk did block benefits 2, 4 and 7 and objectives 2 and 4. Since the risk is now closed because all the relevant tests were passed the benefits 3 and 7 and objective 4 became unblocked and 'fall through'. You can imagine this diagram to be a little like the KerPlunk™ game, where marbles in a cylinder are held up by needles that pass horizontally through holes in the cylinder. As needles are withdrawn, marbles fall through. This is what we aim to do in testing: to focus on the tests that address the risks that block benefits and the cardinal objectives. If a benefit is blocked, we can see straightaway which risks block it and therefore which tests we need to prioritize to unblock the benefit.

Stakeholders are most interested in the benefits that are available and the objectives that can be achieved. The benefit-based test reports present this clearly. Project management are most interested in the risks that block the benefits and objectives. The benefits-based test reports focus attention on the blocking risks so that the project manager can push harder to get the tests that matter through.

One final point: If testers present risk and benefits based test reports, the pressure on testers is simply to execute the outstanding tests that provide information on risk. The pressure on developers is to fix the faults that block the tests and the risks of most concern. Testers need not worry so much about justifying doing more testing, completing the test plan or downgrading "high severity" incidents to get through the acceptance criteria. The case for completing testing is always self-evident: has enough test evidence been produced to satisfy the stakeholders' need to deem the risks of most concern closed? The information required by stakeholders to make a release decision with confidence might only be completely available when testing is completed. Otherwise, they have to take the known risks of release.

How good is our testing? Our testing is good if we present good test evidence. Rather than getting so excited about the number of faults we find, our performance as testers is judged on how clear is the test evidence that we produce. If we can provide evidence to stakeholders for them to make a decision at an acceptable cost and we can squeeze this effort into the time we are given, we are doing a good testing job. This is a different way of thinking about testing. The definition of good testing changes from one based on faults found to one based on the quality of information provided.

Most managers like the risk-based approach to testing because it gives them more visibility of the test process. Risk-based testing gives management a big incentive to let you take out a few days early in the project to conduct the risk assessment. All testers have been arguing that we should be involved in projects earlier. If managers want to see risk-based test reporting they must let testers get involved earlier. This must be a good thing for testers and our projects.

paulg@evolutif.co.uk , www.evolutif.co.uk