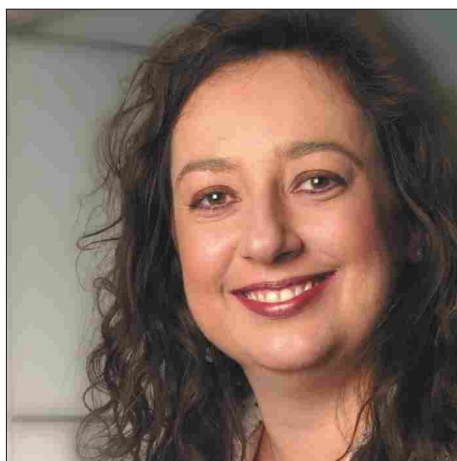


# Are we there yet?

How risk-based testing can help in making the go/no-go decision

Every parent will readily identify with the “are we there yet?” question that bored children ask continually as we battle through traffic to our favourite theme park. Whether we are evaluating the choices we can take to shortcut the route to our destination, or evaluating the facts that will determine a go/no decision in a software development project, we must surely agree that the final decision is never an easy one to make.



**Sarah Saltzman**  
is technology support  
manager at Compuware

## Who is right?

There are many influencers involved in determining whether an application is ready to be launched into the production environment. Each person you talk to will have a different opinion on the status quo. Plus the relationships we build and maintain across different business departments can also have a significant impact on the way we work. A business analyst is primarily concerned with understanding how the requirements for the business have been implemented, but does not necessarily want to know how it is manifested in code.

A programmer will want to know that the module he or she developed or modified has not introduced functional errors in the software, or adversely impacted other aspects of the application. The programme manager is concerned with the timely implementation of the project, that the software meets the needs of the stakeholders and, ultimately, that the application will scale to meet the anticipated numbers of users and delivers the business benefit that the project set out to achieve.

The test manager is more concerned with ensuring that adequate time has been allocated to ensure that test scripts can be run and that they are covering the typically 'buggy' areas of code that prior experience has proven to expose weak spots in the

software. Each person has a view that is equally valid, so it can fall upon the shoulders of the testing group to make sense of the differing requirements of the business, development and testers alike, to give the programme manager a true feel for the state of the application when the go/no go decision is made.

The nature of the relationship between the testing department and the other departments could impact how the decision is reached. When faced with this kind of decision it is never easy to be truly objective, because human nature dictates that there can be a personal influence on any decision. It is easier to make a decision based on facts gathered as the project moves through the various phases of its life-cycle. It is important to take account of the facts that provide a valuable insight into the quality of the application being developed, in terms of requirements covered or not yet covered, together with an analysis of the amount of code that has been covered by thorough testing.

But this will not be sufficient to satisfy the demands of the business. There has to be a fundamental reason why an application was developed in the first place. There must be a value attributed to the cost of the project that results in a positive impact on the business. Conversely there is a risk associated with the failure of the application, and therefore a cost attributable to it too.

Given that timescales are invariably squeezed by the time the testing phase commences, how can the testing manager ensure that the critical parts of the application are tested sufficiently to meet the needs of the business and cover the risks so that the application can be deployed with confidence?

At the end of the day, can we ensure that when faced with the inevitable 'Are we there yet?' question, we are able to give an objective answer?

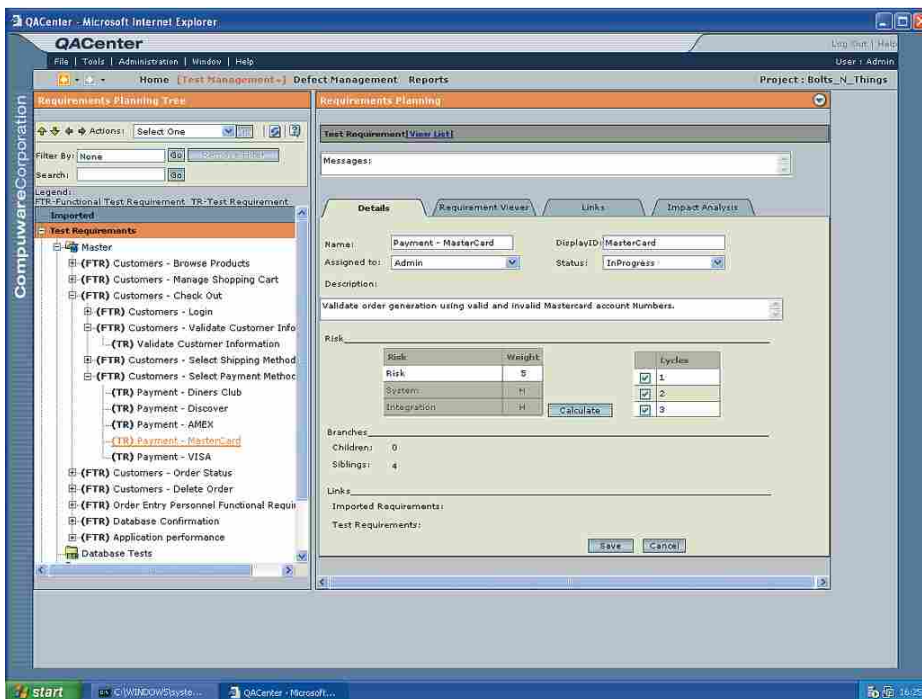


Fig. 1 Applying risk weightings to test requirements

### Risk Based Testing (RBT)

RBT provides an objective approach to the dilemma by adding a process for incorporating business requirements into the testing mix. This ensures that the business and IT requirements are considered together 'in the round' relative to each other. RBT provides a structured, unbiased opinion on where the testing priorities truly lie.

Let's consider an example. Due to a partnership with a new back-office credit card processing provider a retailer has had to make significant changes to the payment processing modules in the on-line order processing system. Each of the payment methods presents a potentially critical failure point that should be tested, but the testing team has only five days to complete the 140 test cases needed to validate the entire checkout process, something that typically takes eight days.

A decision must be made whether to delay the release of the application and risk loss of revenue - and potentially lose face with the new partner - or blindly choose a few tests to run and deploy 'with fingers

crossed'. Neither option is appealing, but with RBT a more predictable outcome can be determined. RBT ensures that the most important functions of an application are tested first, regardless of how limited the testing time frame may be.

### Evaluate the risks

Let's go back to our example. When we evaluate the risks based upon feedback from the sales department we can establish that the top three payment methods are Visa, Mastercard and American Express. The remaining payment methods constitute less than 5% of remaining payment methods. While reviewing the testing requirements the test manager can apply weightings to reflect the increased risk of failure should the Visa payment method fail. Additionally, the testing department may know that the Mastercard payment process has historically been hounded with problems in previous releases of the software, and indeed may have defect trend reports to support this point of view. Taking this into account, an increased weighting can be applied to recognise the focus that

this area should require when testing. Moreover, this can provide very useful feedback to the development team in terms of where quality initiatives in improving code should be centred for future releases.

RBT allows the testing team to focus their efforts on testing the right things from an IT and business perspective. This approach provides a more balanced view of the application by calling out the critical areas to be tested based upon the business goals and the cost of failure for each area to be tested.

### Test, prioritising your efforts against the relative risk of failure

In Fig. 1 you will see that a weighting has been applied to the Mastercard payment method, resulting in a high priority being applied to this particular test. When it comes to running the high priority tests, we can be assured that this particular test case will be included automatically in the test suite that is generated.

The programme manager will have confidence in the knowledge that the most important aspects of the software, in terms

of business and technology risk, are being correctly prioritised in the test plan. Using information gathered through application usage, requirements and test history, the test manager now understands which tests are a priority and can commit the necessary resources to complete the testing in time.

### Metrics and Measures

As the project progresses metrics and measures become important in determining the answer to the inevitable 'Are we there yet?' question. Fig. 2 gives a clear understanding of the nature of the project. We can see that all of the high and medium priority tests have been executed, but we have some issues outstanding with defects raised with development. A small percentage of tests with a low priority have not been executed. Drilling down the nature of the failure and relating it back to the original testing requirement will enable us to determine the criticality of the outstanding issues and determine objectively if we are able to deploy at this stage.

Defect trend reports can give a useful indication of how the testing effort is matching gaps in quality. Running the high priority tests early in the test cycle means that we are able to identify the highest priority defects sooner. In clearly communicating the issue back to the development team, in addition to the risk that this issue exposes to the business, we are enabling the development team to prioritise their efforts in fixing the business-critical bugs as quickly as possible. This means that once the fixes come back to the testing team in cycle 2, they can be readily flushed out and re-tested prior to deployment.

Furthermore, once a project has completed, an in-depth analysis of the kind of bugs raised, linked to their relative priority to the business objectives, allows development managers to leverage this

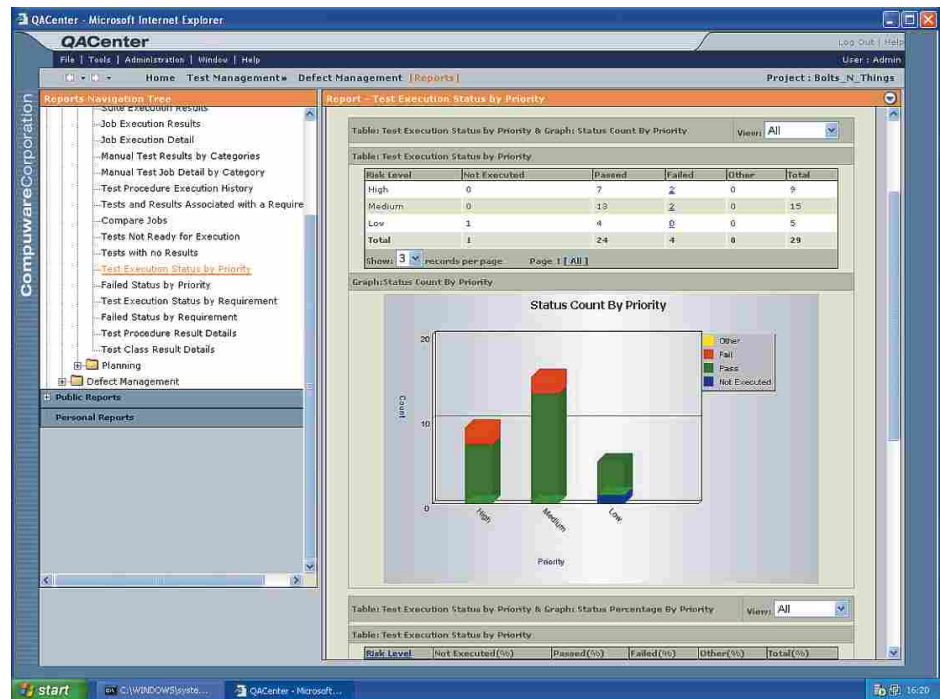


Fig. 2 Reviewing the progress of the testing effort

information and focus any quality initiatives they are rolling out on areas of code that have a direct impact on the business bottom line.

To conclude, working collaboratively with the business and IT gives the test manager the opportunity to improve his testing strategy objectively and clearly. Adopting a risk-based approach towards software testing enables the priorities in the testing strategy to be visibly and rationally understood.

This all boils down through quantifiable metrics and measures to providing a truly objective and more definitive answer to the "Are we there yet?" question ■

sarah.saltzman@uk.compuware.com

**Subscribe to**  
Professional Tester Magazine