The Five Major Challenges to Risk-Based Testing

Stuart Reid

Risk-based testing (RBT) has been around in various forms for over 20 years, and accepted as a mainstream approach for over half that time, now being an integral part of popular certification schemes, such as ISTQB, and the basis of the new ISO/IEC/IEEE 29119 Software Testing standards,. We all use risk on a day-to-day basis in our daily lives (e.g. 'should I walk the extra 50 metres to the pedestrian crossing or save time and cross here?') and similarly many businesses are based on the management of risk, perhaps most obviously those working in finance, such as banks and insurance companies. Despite this, and the fact that RBT is not a complex approach in theory, it is still rare to see RBT being applied as successfully as it could be. This paper initially introduces the basic RBT concepts and then highlights the main obstacles to effectively implementing RBT and suggests means of addressing them in practice.

RBT in a nutshell

Before considering the obstacles, let's first briefly describe the principles behind RBT. Risk analysis is used to identify and score risks, so that the perceived risks in the delivered system (and to the development of this system) can be prioritized and categorized. The prioritization is used to determine the order of testing (higher priority risks are addressed earlier), while the category of risk is used to decide the most appropriate forms of testing to perform (e.g. which test phases, test techniques, and test completion criteria to use). A valuable side-effect of using this approach is that at any point the risk of delivery can be simply communicated as the outstanding (untested) risks.

The evolving business situation and the results of performing testing against perceived risks will naturally change the risk landscape over time thus requiring RBT to become an ongoing activity. It is best practice to involve as wide a range of stakeholders as possible in these activities to ensure that as many risks as possible are considered and their respective treatments (or not – we may decide to not address some risks) are agreed.

RBT Challenge 1 – RBT should work at all test levels

Many test managers are initially introduced to RBT as testers and use it to prioritize and target their testing within a certain test phase, often system testing. The use of RBT by individual testers to manage their own testing is certainly a valid application of the approach, but its potential at the level of deciding test strategy for the complete project is even more powerful. At this level RBT is used to decide and justify which test phases to use (or not) and to define test completion criteria for each of these phases, thus addressing higher level risks.

The use of RBT at even higher levels should not be ignored. It is also worthwhile identifying risks that apply across the whole programme or organization and determining means of mitigating these risks via testing. Such an approach can lead to the mitigation of risks through the definition of an organizational test policy and an organizational test strategy that will define guidelines for testing across the whole organization. One obvious way of ensuring RBT is used at all levels is to mandate

its use in the organizational test policy, or if there is no test policy, include requirements for the use of RBT in the organizational test strategy.

RBT Challenge 2 – RBT should address testing across the whole life cycle

As a test manager, when writing the Project Test Plan there is a temptation to only include those testing activities that you control directly. The Project Test Plan, however, needs to address all the testing performed across the whole life cycle whether it is carried out by testers or developers (or any other stakeholders). After all, if the test manager doesn't take responsibility for testing, who else will? A common occurrence is that testing activities early in the life cycle, such as reviews of requirements and designs, which we know are extremely efficient in terms of defect detection and prevention, are not planned and executed effectively, if at all. A second is that the rigour of developer testing (typically limited to unit testing) is decided by the developers alone, which may lead to lower quality code being passed into later testing phases. We need to ensure that the planned testing is targeted at mitigating risks as early as possible in the life cycle, even when that testing may not actually be performed by members of the testing team.

RBT Challenge 3 – RBT should not be a 'stand alone' means of managing risk

Although RBT is a valid and valuable approach to managing testing, it becomes far more powerful when it is integrated with the risk management performed by the project manager and the developers. In this way, we can share a common understanding of the risks to the project (ideally documented in a shared risk register), how these risks interact (e.g. poor development leads to higher testing costs), and how they should be mitigated. This allows us to ensure risks are mitigated in the most efficient manner by those that are best placed to do it; often prevention by developers is far more efficient than later detection by testers.

If introducing RBT into an organization, be sure to take advantage of any other risk-based approaches to project management and development that are already there. Not only is it far better to end up with an integrated approach to risk management, but getting buy-in to risk management from all the relevant stakeholders is time consuming and if they are already managing risk in other areas it is far easier than starting from scratch.

RBT Challenge 4 – RBT requires a 'professional' level of test maturity to work effectively

RBT will not work if those attempting to use it do not possess a high enough level of test maturity. In order to be able to select the testing that is most appropriate for a given risk, then the tester or test manager must know the range of testing options that are available to them and how these options relate to the different risk types. This requires a practical level of familiarity with test case design techniques, the effectiveness of each at detecting different types of defects (and so mitigating risks), and in which test phases they are most effective. This knowledge is the bedrock of the professional software tester. In an industry where many testing practitioners find it difficult to name more than one test case design technique (let alone apply it), it is not surprising that many struggle to apply RBT effectively. One specific group of testers that seem to have particular difficulty applying RBT are those that limit themselves to 'requirements testing', assuming all requirements are equal, and performing no test prioritization. Typically these testers are not aware which test case design techniques they are using (if any), or that there are options open to them to vary the

level of rigour of their testing for different levels of risk. This is one area for improvement where many testers could cost-effectively spend some time.

RBT Challenge 5 – RBT should not only address risks in the deliverable product

Most new users of RBT tend to concentrate on the risks in the deliverable product (e.g. the risk that the accounting system miscalculates profits or the risk that the web-based system discloses customer details), however for RBT to work effectively we also need to consider the risks to the performance of testing on the project itself. This category includes risks such as the late delivery of code from developers and the lack of testing resources available to the test manager. For RBT to work most effectively both product and project risks (and their mitigation) need to be considered together as they can have an immediate effect on each other, and the optimal balance needs to be achieved. For instance, if there is a project risk that the available time for testing is shortened it is not satisfactory to simply reduce the amount of testing as this will typically result in fewer defects being detected and will nearly always result in a consequential product risk of a 'buggy' deliverable. In this case any mitigation needs to strike a balance between the two interacting risks to achieve an outcome that is acceptable to all stakeholders, and the ability to successfully identify and implement such compromises is the sign of a true professional test manager.

Conclusions

RBT is known to be best practice for today's professional testers, but, despite the simple concept, many testers struggle to apply it in an effective manner. This is often due to the scope of RBT as applied being too narrow, by it not encompassing the higher levels of testing, such as in the organizational test strategy, and by it not being used as the basis for test planning that covers the whole life cycle. RBT can also fail to fulfil expectations when it is not integrated into the wider risk management practices within an organization, or when RBT is only used to address risks in the deliverable software rather than also considering the risks to the testing itself. These challenges can largely be addressed by the industry changing its perspective on RBT and widening its view. Probably the biggest challenge to the effective use of RBT is the lack of maturity of test practitioners, however, this should be seen as an opportunity for testers to ensure that they acquire the full 'testing toolset' to allow them to effectively mitigate the risks with the right testing options.