BEST PRACTICE
Risks Associated with Software Testing

Testing is inherently a risk-based activity. Most companies would not pay to add testing to the cost of the project if there was not a cost associated with the risk of failure. Exhaustive testing is impractical for most applications under development. Exceptions include applications that support very high risk processes, such as air traffic control, nuclear station operations, defense systems, and so on. The project team must design a test strategy that utilizes a balance of testing techniques to cover a representative sample of the system in order to minimize risk while still delivering the application to production in a timely manner.

It is the test manager's responsibility to determine how to apply the test methodology to achieve the greatest level of confidence in the application under development. Risk is a major driver in the test planning activity. When conducting risk analysis, two major components are taken into consideration:

- The probability that the negative event will occur.
- The potential loss or impact associated with the event.

The test manager must determine the appropriate amount of testing to perform based upon the risks associated with the application. These risks can arise from the newness and reliability of the technology being used, the nature of the application, or from the priority of the business functions under test. The amount of testing that should be performed is directly related to the amount of risk involved.

Understanding the risk-based nature of testing is also the key to dealing with the chronic problem of inadequate test resources. Risk must be used as the basis for allocating the test time that is available, and for helping to make the selection of what to test and how to allocate resources.

The test manager is also responsible for identification of potential risks that might impact testing. Some of the primary testing risks include:

- **Not Enough Training/Lack of Test Competency**
  The majority of IT personnel have not been formally trained in testing, and only about half of full-time independent testing personnel have been trained in testing techniques. This causes a great deal of misunderstanding and misapplication of testing techniques.

- **Us versus Them Mentality**
  This common problem arises when developers and testers are on opposite sides of the testing issue. Each feels that it is “out to get” the other. Often, the political infighting takes up energy, sidetracks the project, and accomplishes little except to negatively impact relationships.

- **Lack of Test Tools**
  IT management may have the attitude that test tools are a luxury. Manual testing can be an overwhelming task. Although more than just tools are needed, trying to test effectively without tools is like trying to dig a trench with a spoon.

- **Lack of Management Understanding and Support of Testing**
  Support for testing must come from the top, otherwise staff will not take the job seriously and testers’ morale will suffer. Management support goes beyond financial provisions; management must also make the tough calls to deliver the software on time with defects or take a little longer and do the job right.

- **Lack of Customer and User Involvement**
  Users and customers may be shut out of the testing process, or perhaps they don’t want to be involved. Users and customers play one of the most critical roles in testing; making sure the software works from a business perspective.

- **Not Enough Schedule or Budget for Testing**
  This is a common complaint. The challenge is to prioritize the plan to test the right things in the given time.

- **Over Reliance on Independent Testers**
  Sometimes called the “throw it over the wall” syndrome, developers know that independent testers will check their work, so they focus on coding and let the testers do the testing. Unfortunately, this results in higher defect levels and longer testing times.

- **Rapid Change**
  In some technologies, especially Rapid Application Development (RAD), the software is created and modified faster than the testers can test it. This highlights the need for automation, but also for version and release management.

- **Testers are in A Lose-Lose Situation**
  On the one hand, if the testers report too many defects, they are blamed for delaying the project. Conversely, if the testers do not find the critical defects, they are blamed for poor quality.

- **Having to Say "No"**
  Having to say, "No, the software is not ready for production," is the single toughest dilemma for testers. Nobody on the project likes to hear that and frequently testers succumb to the pressures of schedule and cost.

- **Test Environment**
  The work environment is not conducive to effective and efficient testing.

- **New technology**
  The new focus on client/server, Intranet, and Internet applications has introduced even more risk to the test process. These multi-tiered systems are more complex than traditional mainframe systems, and therefore have a higher level of risk associated with their testing and implementation. Security, performance, availability, complex component integration, and a team new to the technology are just a few of these new risk factors.

- **New developmental processes**
  Along with these new delivery platforms come new methods for software development. Project teams have moved away from traditional "waterfall" methods, and follow a much more iterative approach to development and delivery. An integral part of this iterative approach is that testing activities are more integrated across the development cycle than before. Traditional system testing is not just conducted at the end of development, but may be conducted multiple times throughout the project for major components, and then completed once all development is complete.

Based on the risk assessment of the risks associated with the test process, changes to the test process may be needed. For example, if the testers lack training in the test process, that training should be provided prior to testing. If the training cannot occur then extra supervision would be required. If inadequate test resources and schedule time is not available the number of tests may need to be reduced.

Risk analysis will determine the risks associated with the software being tested and performing the test process. Risk analysis should determine the magnitude of the risks, and prioritize them in importance. Test planning will then consider those risks in developing the Test Plan.

References

Guide – CSTE Common Body Of Knowledge, V6.1