



Reducing Cost and Duration of the Test Phase

Last Updated: June, 2010



Introduction

Reducing cost and duration of a test phase is of utmost importance to stakeholders. Needless to mention overshooting the estimated cost and duration is a cause for stakeholder's dissatisfaction, who is always keen on deploying a system without delay.

From a cost perspective, stakeholders and project managers can sometimes be surprised by how much a test phase can cost. Take for instance, if substandard software is delivered to the test team, the cost can increase due to defect raising, fixing and retesting; causing either an extended release date or high costs for staff overtime. On top of this, late running development, poor estimations or critical defects can further delay the release date.

This whitepaper outlines some of the methods available to reduce the duration and cost of the test phase, whilst maintaining or improving the software quality.

The Importance of Planning

Additional time spent on planning phase with substantial involvement of the core team, can pay rich dividends as the project unfolds. It is important to include the test manager in the process from start who could also acts as a reviewer of the plan. Assumptions, constraints, items in or out of scope, entry or exit criteria can be identified and agreed upfront and changes can be managed appropriately as the project progresses.

Estimates need to be created, based on historic data and with reference to the Business Analysts and Developers. If it seems that the required testing will exceed the planned dates then a risk based or prioritized approach can be adopted early on and with the stakeholder's approval. Often this is missed and projects end up in the test phase, demanding extra test resources or putting extreme pressure on the few resources already available.

It is also necessary to highlight in the test execution, run rates and expected defect rates per day or week. Sometimes, test estimates neglect the time required for raising defects or for re-running the tests. Once these rates are established, it is essential to ensure that there are enough fixed resources allocated to turn around the expected defects in a timely manner.

There are several projects that the development teams have to take care of. Many a time the developers are assigned another project withdrawing them from the one they were initially working on. This causes a defect backlog on the project. This defect backlog can have undesired effect on the test duration with days being wasted waiting for critical defects to be fixed and management resources being taken out of action for the defect triage.

AppLabs.com

Partnering with the Development Team

Successful software delivery requires development and testing teams to be independent yet collaborative. Testers may be included in requirement reviews, requirement changes and module demos, partly so they can add value but also to ensure they are creating the right test scripts. If they are kept in the dark then they will have to undertake significant script rework and may raise invalid defects upon delivery. This will add significant value to the duration and cost of testing.

It is important to know if the development team is aware of the automated tools that the testers intend to use and better still, how the tools work. If the developers are unaware then they may constantly be changing object properties, causing a large amount of script maintenance and troubleshooting for the automated testers. Nothing is more frustrating than defect fixes, causing significant software regression, late in the test phase. There may be a temptation to rush fixes back into the test environment but a little extra time for the development team to review fixes and consider the wider impact can prevent a project going off the rails. Fixers should be measured on fix quality and not just fix turnaround.

Test Scripting

It is vital to ensure that scripts are produced in a prioritized order. There is no point going down to the nth degree on a fairly unimportant feature rather than having scripts in place for some of the major functions. It is best to write high level one-line tests for each aspect of the requirement and then prioritize before adding detailed steps to each script. If you run out of script creation time then at least you will have covered the most important ones and the least important scripts can be run against the high level scripts if required. Similarly, wherever possible, scripts should be executed on priority basis.

Time could be set aside for the test environment to be clean and ready prior to the delivery of software. The test environment could be separate from the development or fixing environments and fix delivery guidelines should be established within the test plan.

If possible, it is good to run more than one test environment, especially if system tailoring options are required to be at different settings for other tests. Also, if one environment goes down or if a close of business cycle is being run on one environment then testing can continue on another.



If there are a large number of test resources then any downtime can most likely have a huge cost.

It is also worth asking whether the servers and desktop machines have a high enough specification. A small investment in the server could prevent systems hanging for a large number of people. You might also be surprised that a RAM upgrade in the automated testing machines can make a considerable improvement to the number of script runs.

Other Considerations

Alternative testing – In a large project there will almost certainly be some modules ready for testing before others, so rather than plan a very strict waterfall with everything being delivered to the test team on mass; it can pay dividends if it is arranged to take some modules early, followed by running a regression test of these modules during the final phase.

Onshore/offshore model – Existing test resources can be supplemented with offshore resources and this can reduce the test duration.

Risk management – Ensure that test risks are included with the overall project risks at the outset. Each risk should have an owner, a mitigant and an action date. Risks will need to be reviewed regularly.

Progress tracking – Test creation schedules and test execution schedules should be created upfront and then progress tracked against them daily or weekly. If targeted levels are not being reached then measures can be put in place to resolve the issue before it is too late.

Summary and Conclusion

In order to make improvements in the test duration; cost, proper planning, high quality estimates, prioritized test preparation and a good relationship with the development team are paramount. Many of the topics covered above are basic project management and test management techniques but it is surprisingly common for projects to ignore some or all of them. Certain items such as test estimation techniques, resource planning and test automation are worthy of much further guidance. These are areas that require significant research and experience. Testing vendors like AppLabs assist businesses by successfully applying all of the above-mentioned methods and many more. Such process improvements can yield major cost savings and provide higher quality software on targeted release dates.