Tips for planning tests

You have specified your Tests and organized them using Test Sets or using the Test Repository. Now, you want to know how to address properly the planning phase of your testing, right?

You may start by creating a Test Plan, then create the Tests, and then add those Tests to the Test Plan.

Planning your testing is also dependent on the methodology and process you're adopting.

Below are some planning examples. Feel free to adapt these to your own internal process in a way that best fits your team.

Planning, from an "aim" perspective

Instead of creating one Test Plan for your release, you may create multiple Test Plans to track different Tests.

This may be useful if you want to have clear visibility of how certain groups of Tests are progressing.

You may create different Test Plans:

- for Tests related to new features
- for regression testing (i.e., for Tests related with features implemented in previous versions)
- for security-related Tests
- for performance-related Tests
- for compliance-related Tests
- for non-functional Tests

Please note

This approach is independent of the methodology being used. You can use the tips described next to extend your methodology and gain additional visibility over certain Tests.

Planning, from a "methodology" perspective

Agile

Scrum

If you're following Agile, particularly Scrum, you'll have sprints of one or more weeks. Each sprint represents one iteration until at the end, you have a shippable product. Testing can occur hopefully during the period of the sprint or more closely to the end of it. In any case, you can have a Test Plan per sprint to track the results of the Tests you want to execute, including the ones that validate the features implemented in that sprint. Besides this, you may also want to have a Test Plan for regression testing to ensure you're not breaking anything in that sprint.

Please note

A sprint is a more manageable way of dividing the complexity of an entire release. A sprint has a short, well-defined time period.

Kanban

In Kanban, you're limiting the amount of WIP issues in a given state, namely, the ones that are in testing. Because you're not dividing your release into several periods of time, as you do with Scrum, you can manage it as if the period would embrace the time frame of the release.

Thus, you can have a Test Plan to track the Tests related with the features being addressed on the Kanban board. You may complement it with additional Test Plans, namely, for tracking the regression testing. It may be a good approach to have the Test Plan with regression tests, or smoke tests, as an independent Test Plan, so it is more apparent if regression is occurring or not.

Waterfall

In this scenario, testing is done as a separate phase at the end of development. During testing, bugs may be found, which will be directed back to requirements analysis/development. You wait for changes/fixes to be made, and then test it once again.

Having a Test Plan assigned to the version currently being developed as a way to group the results from multiple testing cycles may be enough.

Iterative Waterfall

In this approach, a release is somehow split in several internal mini-releases, with subsets of the features implemented. It is assumed that some of these features may not be complete/stable/finished.

Each intermediate release may have one specific Test Plan as a way to track the Tests and their results. The Test Executions made in the context of each of these Test Plans can also be associated with a more broader Test Plan, so their results would also get reflected in a broader sense.

In sum, you can simply create one Test Plan for tracking all the Tests you want to validate in that version along with additional Test Plans, one per iteration /intermediate release. Then, you need to ensure that all Test Executions created in the context of your intermediate release's Test Plan are also reflected in the "master" Test Plan.