How ALM Enhances the Value of Your Test Team

Application lifecycle management (ALM) provides beginning-to-end management of your software development project by providing tools to assist with handling communication, process, and data.

The product team is composed of a cross-section of departments, and members of the test team are key participants in any successful development project. Testers perform a number of important functions, including product verification of requirements, quality level assessment, standards validation, and feedback about process change.

As with any software development team, the test team benefits when automation can be used to help eliminate human error and reduce repetitive tasks. Although test tools have come a long way in supporting automation, the value of testing can be significantly improved when the testing function is integrated with ALM. For example, ensuring requirements have been addressed properly implies that there must be a proper level of integration between requirements management and test case management.

ALM spans team boundaries, allowing every team to provide more value as an active part of the entire project process. Combining an ALM tool with appropriate process enables clear communication among development teams.

Some key elements of most modern ALM tools can provide even better benefits. Sharing a common data repository, management UI, process engine, and management philosophy provides value to every team, and any improvements are made to both process and tool infrastructure, making it much easier to justify improvements.

The Importance of Test Case Management

The test team is usually responsible for creating a repository of test cases that verifies a product's functionality. It must be able to run and record the results of test cases against the product in its various operating environments. Ultimately, it is the recommendation of the test team that determines whether a product is ready to be released.

It used to be that most of a test team’s resources were consumed in actual functional testing. Now there is more focus on test automation, allowing more testing resources to be used for test case development, automation, and process improvement.

There are a number of primary and secondary test case categories when attempting to marry testing with configuration management. Summarized in table 1, these test case categories show that the test process touches many parts of the lifecycle.

Primary test categories	
Black Box	Verify product requirements.
White Box	Verify the design (a.k.a. software requirements).
Change	Verify a change package (i.e. a logical change to the software product).
Bug	Reproduce defects and verify that the known defects are fixed.
Sanity	Ensure basic sanity of a deliverable and its platform/infrastructure.
Stress	Identify response of a system operated near the edge or outside of the product requirement's envelope.
Beta	Identify defects resulting from incomplete requirements, unforeseen use cases, and imperfect testing.
Secondary test categories	
Regression	Re-application of test cases to ensure all features still work (e.g. black box).
Feature	Run a subset of black box testing focused on a particular feature.
Performance	Run a subset of black box testing dealing with real-time issues.
Environment	Run a subset of black box testing identifying the effect of a product on the environment (and vice versa).
Validation	Run a subset of black box testing targeted to meeting specific, required standards.
Unit	Run a mix of black box and white box testing at a card, module or subsystem level.
Alpha	Initial beta testing is typically performed using an internal test site.

Table 1: Test case categories

For the most part, the primary test categories are used to introduce new test cases into the repository, while the secondary test categories repurpose (or reorganize) the existing test cases to provide more targeted test coverage. These categories help to identify the scope of the test team function. There is plenty of data that needs to be collected, created, captured, organized, and summarized.

ALM Helps Organize Your Work

Modern ALM tools provide many of the capabilities not only to keep test cases organized, but also to retrieve, deliver, and collect test cases for a single task or purpose. These tools also have capabilities that make it easy to compare sets of test cases (or other artifacts) across builds and releases. This, in turn, provides more precise metrics that allow tracking of product quality as well as development team and process quality. For example, when the test case failure rate increases, it's nice to know that the primary factor was the introduction of new features that still needs some work.

In a silo, a test team can perform testing on a function quite well. But expose that silo to "the rest of the farm" and suddenly management functions better. It has up-to-date test status information it needs to make a go/no-go decision. But having the ongoing, live feedback leading up to that decision, management has the ability to take action to preclude a no-go possibility. In my experience, all functions of ALM are improved by the continuous availability of test information.

If you have been through multiple application development lifecycles, you'll undoubtedly agree that testing is not a standalone function. In an ALM environment, each ALM function contributes to the advancement of the testing process. In return, each ALM function benefits from these testing efforts. Consider the following ways that the various test categories interact with the wider range of ALM functions.

The Importance of Requirements Management

Requirements traceability is critical to any successful project. It is important to know that for every requirement, there is a valid set of test cases, and every test case verifies a product requirement. The classical requirements traceability matrix is a sparse matrix, which helps to visually assert this traceability. In practice, with thousands of test cases run against a product, the visual benefit is really only useful in teaching about traceability. The ALM tool that spans both requirements management and test case management should supply the ability to query traceability for any subset of the requirements specification, as well as for any subset of the test case collection.

How Configuration Management Supports Build and Release Management

More than supporting requirements traceability, it is important to track the history of actually running each test case against each build and release. The shared data repository afforded by a modern ALM tool should provide easy query access to identify which test cases have successfully passed and which have failed during a given stream of release development.

Requirements typically change over time, and the test suite must change accordingly along with the software being tested. Configuration management is used to ensure that the right revisions of requirements, test cases, and software come together to test the builds leading up to a release.

Generally, test cases are either added to or removed from test case configurations. But there are many complex tests that require version and change control as development of the tests themselves evolve. This version and change control capability is already provided by modern ALM tools for software and their requirements.

As well, each configuration has optional components and variants that are part of the software and build configurations. Testing must take these options and variants into consideration. This is data that is tracked in any modern ALM tool and can be used to aggregate test results to assess the product quality for various components and variants.

Continuous integration requires a certain level of automated testing. The build and test functions of an ALM tool can help ensure that productive testing is automatically performed on each system build. This requires identification of the correct environment data setup followed by running of automated tests (which should not be too time consuming).

Typically, the data associated with a test case needs a few modifications to provide higher level benefits. A simple data field that rates the automation level and running time of a test case or test suite can be used to help select the appropriate set of automated tests to be run against each build.

ALM tools can show the progression over time of test case automation. These tools can estimate expected test times—not only for automated test runs, but also for any test session, based on which test cases are selected and which features need to be tested.

Problem Tracking Is a Must-Have

Problem tracking (also known as defect and issue tracking) is another ALM function tightly integrated with test case management. In practice, each problem should have one or more test cases that are used to verify the existence of the problem in existing builds and to ensure the problem has been eliminated in more recent builds. Such test cases should be derived from the problem description as long as the problem has been described that it is easy to reproduce. Linking the test case to a problem report helps to ensure that all problems have sufficient test case coverage. Such a link also helps to automate the running of a test suite whose purpose is solely to ensure problem reports have been properly fixed.

The ALM tool should support queries such as “When was this problem fixed?” “Is this problem present in build *xyz*?” and so on. Ideally, a successful running of the test case should automatically advance the state of the problem to the fixed state.

How Feature Tracking and Project Management Fit into ALM

A suite of tests is created based on a feature description document or requirements analysis document. It is important that test cases relate directly to the feature it has been created to test. This allows easy generation of feature-based test runs.

Success rates can be fed back into the ALM tool to help identify the actual completion level of the feature, at least from a functional perspective. Similarly, there are white box test suites derived from design documents that help test the design of an API, a module, or even a subsystem. Similar techniques can be used to help assess design-level completeness.

This feedback helps improve project management accuracy. If a project manager asks an inexperienced developer about completion status for a task, the response is something like, “Eighty percent complete with 20 percent left to do.” It has been my experience that the last 20 percent takes more than 20 percent of the estimated time. Tracking of test case feedback versus developer feedback also provides valuable feedback to the software developer.

Project management is critical to test team planning. When PM is integrated across all ALM functions, resource allocation and schedule bottlenecks become easier to manage more effectively.

ALM Benefits Test Team Value

An ALM-wide approach provides commonality of tools and processes, as well as product-wide visibility of what is really going on throughout the software development lifecycle.

A good ALM tool capability tracks test sessions, allowing simple metrics to be developed for improving the test team's productivity over time. Some ALM tools have code search capabilities that can be used by the test team for searching through test cases. Finally, most ALM tools can organize and report hierarchically. This capability is especially helpful across a changing landscape of new product releases, variant products, and evolving test case content.

The intangible value that results from increased team-wide communication of all ALM information is not to be underestimated. The test team feels more a part of the product team. The rest of the product team is awakened to the various roles of the test team. And as they become more aware, there are efforts to improve the process. Developers see the benefit of formally cataloguing their own tests and passing them on to the test team, both for black box and white box testing. Configuration management personnel identify the need for developers to annotate software updates as a result of tests that have changed along with test results. The entire team is educated on how to define problem reports so that each problem is easily tested.

It is important to identify ALM tools that provide the capabilities allowing easy and ongoing process customization, shared and extensible data schema, and customization of the UI on a role-specific basis. Tools must be responsive, add noticeable value to each user, and support ALM-wide automation. Once these basics are covered, there is no end to the advances that can be achieved to improve the value of the test team and the entire product team.