Test management best practices
How to better handle your software testing efforts

Chip Davis
November 07, 2006

An important part of software quality is the process of testing and validating the software. The purpose of this article is to introduce concepts about, and provide general best practices in the field of, test management. Test management is the practice of organizing and controlling the process and artifacts required for the testing effort. The article discusses how IBM® Rational® ClearQuest®, IBM® Rational® ClearCase®, and IBM® Rational® Requisite Pro® can improve your testing.

Introduction

Purpose
Few people can argue against the need for improved quality in software development. Users of technology that utilizes software have come to expect various faults and flaws and, especially in the world of personal computers, we consider frequent problems to be completely normal and expected. However, as software development matures, we are beginning to better understand how to achieve a necessary improvement in quality. The purpose of this article is to introduce concepts about, and provide general best practices in the field of, test management.

What is test management?
An important part of software quality is the process of testing and validating the software. Test management is the practice of organizing and controlling the process and artifacts required for the testing effort. Traditional tools used for test management include:

- Pen and paper
- Word processors
- Spreadsheets

Larger testing efforts may use home-grown software test management solutions, usually built on spreadsheets or databases, or commercial test management applications such as IBM® Rational® ClearQuest® Test Manager or Mercury TestDirector.

The general goal of test management is to allow teams to plan, develop, execute, and assess all testing activities within the overall software development effort. This includes coordinating efforts of
all those involved in the testing effort, tracking dependencies and relationships among test assets and, most importantly, defining, measuring, and tracking quality goals.

**Aspects of test management**

Test management can be broken into different phases: organization, planning, authoring, execution, and reporting. These are described in more detail below.

**Test artifact and resource organization** is a clearly necessary part of test management. This requires organizing and maintaining an inventory of items to test, along with the various things used to perform the testing. This addresses how teams track dependencies and relationships among test assets. The most common types of test assets that need to be managed are:

- Test scripts
- Test data
- Test software
- Test hardware

**Test planning** is the overall set of tasks that address the questions of why, what, where, and when to test. The reason why a given test is created is called a test motivator (for example, a specific requirement must be validated). What should be tested is broken down into many test cases for a project. Where to test is answered by determining and documenting the needed software and hardware configurations. When to test is resolved by tracking iterations (or cycles, or time period) to the testing.

**Test authoring** is a process of capturing the specific steps required to complete a given test. This addresses the question of how something will be tested. This is where somewhat abstract test cases are developed into more detailed test steps, which in turn will become test scripts (either manual or automated).

**Test execution** entails running the tests by assembling sequences of test scripts into a suite of tests. This is a continuation of answering the question of how something will be tested (more specifically, how the testing will be conducted).

**Test reporting** is how the various results of the testing effort are analyzed and communicated. This is used to determine the current status of project testing, as well as the overall level of quality of the application or system.

The testing effort will produce a great deal of information. From this information, metrics can be extracted that define, measure, and track quality goals for the project. These **quality metrics** then need to be passed to whatever communication mechanism is used for the rest of the project metrics.

A very common type of data produced by testing, one which is often a source for quality metrics, is **defects**. Defects are not static, but change over time. In addition, multiple defects are often related to one another. Effective **defect tracking** is crucial to both testing and development teams.
Other factors in test management

In addition to the software and hardware test artifacts and resources, the testing team has to be managed. Test management must coordinate the efforts of all project team members involved in the testing effort. This requires controlling user security and permissions for testing members and artifacts. For projects that span more than one site or team (which is rapidly becoming the norm), this also includes organizing site and team coordination.

The particular testing process for a project will have an obvious bearing on test management. For an iterative project, test management will have to provide the foundation and guide the effort to plan, execute, and evaluate testing iteratively. Following this, the testing strategy will also have to follow the test management framework.

Related software development disciplines

While all disciplines in software development have an association to the testing discipline, there are several that have particularly important connections to testing:

- Requirements management
- Change management
- Configuration management

Requirements management is a precursor to the bulk of the testing effort, providing a significant amount of test motivation and validation needs. A project's particular requirements management process can have a huge impact on the test management process. One analogy for this relationship is that of a relay race, where the first runner represents requirements management and the next runner, receiving a baton hand off, represents test management. IBM® Rational® RequisitePro® is a tool for finding, documenting, organizing, and tracking requirements. More information can be found on the IBM® developerWorks® RequisitePro resource page.

Change management affects all parts of software development, but the tracked changes most relevant to the testing effort are defects. Defects are frequently the main communication channel between testing and development. Counts and metrics derived from defects are also often used as measures of quality. ClearQuest is a powerful, highly configurable tool for managing numerous types of changes and activities throughout the software development cycle. More information can be found on the IBM® developerWorks® ClearQuest resource page.

Configuration management is important to test management because tracking which builds to test at what time is crucial to testing. Configuration management controls the builds as well as the environments that are tracked by test management for test execution. IBM® Rational® ClearCase® is the leading configuration management tool. More information can be found on the IBM® developerWorks® Clearcase resource page.

Test management challenges

One way to sum up the objectives of test management is answering the following questions:
• Why should I test?
• What should I test?
• Where do I test?
• When do I test?
• How do I conduct the tests?

While this may seem straightforward enough at a high level, there are many obstacles that frequently arise in typical software development. These challenges are described below.

**Not enough time to test**

Except for certain specialized or highly mission-critical applications, very few software projects include sufficient time in the development lifecycle to achieve a high level of quality. Very often, the almost inevitable delays in a software project get assigned to the already short "testing cycle". Even the best projects are very likely to have difficult time constraints on testing tasks. The effects of this obstacle on test management are constantly changing priorities and shifting tasks, as well as reduced data for test results and quality metrics.

**Not enough resources to test**

In addition to the shortages in time, there is quite often a difficulty getting the right resources available to perform required testing activities. Resources may be shared on other tasks or other projects. While hardware resources for testing can add delays and difficulties, a shortage of human resources can be even more difficult to resolve. The effects of this obstacle on test management are roughly the same as those for time shortages.

**Testing teams are not always in one place**

More often these days, testing resources might be available but not at the same geographic location. Leveraging talent around the globe to reduce costs has become commonplace, but this introduces considerable technical obstacles. How do teams on another continent share artifacts and stay coordinated without delays and discord affecting the overall project? How can a project maximize efficiency with geographically distributed development?

**Difficulties with requirements**

While there are many testing strategies, validating requirements is typically the primary, highest priority testing that needs to be completed. To do this requires complete, unambiguous, and testable requirements. Less-than-perfect requirements management can lead to more profound issues in the testing effort. Using a tool such as RequisitePro can significantly help improve requirements management and facilitate the development of good requirements.

For test management to be effective, there must be seamless access to the latest changing system and business requirements. This access must be not only to the wording of the requirements, but also to the priority, status, and other attributes. In addition, this requires the utmost coordination and communication between the teams developing the requirements and the teams performing the testing. This communication must go in both directions to ensure quality.
**Keeping in synch with development**

Another team coordination that is required to allow for software quality is between testers and developers. Aside from critical defects, it is almost a tradition in software development that the testing team's work is only the tester's concern. However, it is essential for everyone, especially the developers, to understand both the current level of quality and what has and has not yet been tested.

For testing teams to use their precious time efficiently, they have to keep up with constant changes in code, builds, and environments. Test management must identify precisely what build to test, as well as the proper environments in which to test. Testing the wrong builds (or functions) results in wasted time, and can severely impact the project schedule. Testers must also know what defects are already known, and should therefore not be re-tested, and which are expected to be fixed. Testers must then communicate the defects found, along with sufficient information to facilitate resolution, back to the developers.

**Reporting the right information**

A testing effort is only useful if it can convey testing status and some measures of quality for the project. Generating reports is simple enough, but presenting the right information (at the right time, to all the appropriate people) can be trickier than it seems for several reasons:

- If there is **too little information**, then the project stakeholders will not fully understand the issues affecting quality, in addition to the reduced perceived value of the testing team.
- If there is **too much information**, then the meaning and impact of key information becomes obscured.
- There are often **technical hurdles** that can impede how to share information to different roles in different locations.

Another consideration in reporting results is exactly how the information is arranged, and in what formats (that is, the information can be tool based, browser-based, or in documents). Project stakeholders' understanding of the testing and quality information will be reduced if there are technical or other restrictions limiting the arrangement or format of reports. Data should be presented in a clear and logical design that conveys the appropriate meaning, not in a layout constrained by tools or technology. It is therefore essential for test management to consider the need for flexibility and capability in providing a wide range of reporting formats.

**What are the quality metrics?**

One of the primary goals of a testing team is to assess and determine quality, but how exactly do you measure quality? There are many means of doing this, and it varies for the type of system or application as well as the specifics of the development project. Any such quality metrics need to be clear and unambiguous to avoid being misinterpreted. More importantly, metrics must be feasible to capture and store, otherwise they might not be worth the cost or could be incomplete or inaccurate.

**Test management recommendations**

The following are general recommendations that can improve software test management.
Start test management activities early

While this may seem like the most obvious suggestion, few software projects truly apply this concept. It is easy and not uncommon to begin identification of test resources at an early stage. However, many test analysis activities (such as the identification of critical, high-priority test cases) can and should begin as soon as possible. As soon as use cases are developed enough to have a flow of events, then test procedures can be derived. If a project is not utilizing use case requirements, then tests can still be derived from the validation of initial requirements. Developing tests as soon as possible helps alleviate the inevitably forthcoming time constraints.

Test iteratively

Software testing should be an iterative process, one that produces valuable testing artifacts and results early on in the overall project lifecycle. This follows the first recommendation of starting the testing process early: an iterative testing process forces early attention to test management. Test management guides this by organizing the various artifacts and resources to iterations. This risk-based approach helps ensure that changes, delays, and other unforeseen obstacles that may come up in the project timeline can be dealt with in the most effective manner.

Reuse test artifacts

Reusing test artifacts within a project, or across projects, can greatly improve the efficiency of a testing team. This can greatly relieve the pressure of limited time and limited resources. Reusable artifacts include not only test automation objects, but also test procedures and other planning information. In order to reuse artifacts efficiently, test management must do a good job of organizing and delineating the various testing-related information used for a given project. Reuse always requires some forethought when creating artifacts, and this principle can be applied in test management generally.

Utilize requirements-based testing

Testing can be broken down into two general approaches:

- Validating that something does what it is supposed to do
- Trying to find out what can cause something to break

While the latter exploratory testing is important to discover hard-to-predict scenarios and situations that lead to errors, validating the base requirements is perhaps the most critical assessment a testing team performs.

Requirements-based testing is the primary way of validating an application or system, and it applies to both traditional and use case requirements. Requirements-based testing tends to be less subjective than exploratory testing, and it can provide other benefits as well. Other parts of the software development team may question or even condemn results from exploratory testing, but they cannot dispute carefully developed tests that directly validate requirements. Another advantage is that it can be easier to calculate the testing effort required (as opposed to exploratory testing, which is often only bounded by available time).
It can also provide various statistics that may be useful quality metrics, such as test coverage. Deriving test cases from requirements, and more importantly tracking those relationships as things change, can be a complex task that should be handled through tooling. RequisitePro and the test management capabilities in ClearQuest provide an out-of-the-box solution that addresses this need.

The constraint to this process is that it depends on good system requirements and a sound requirements management plan to be highly effective. Exploratory testing, on the other hand, can be more ad hoc. It relies less on other parts of the software development team, and this can sometimes lead to testing efforts focusing less on the important tasks of validating requirements. While a superior testing effort should include a mix of different approaches, requirements-based testing should not be ignored.

**Leverage remote testing resources**

To help alleviate resource shortages, or to simply maximize the utilization of personnel, you should leverage whatever resources you can, wherever they are located. These days resources are likely to exist in multiple geographic locations, often on different continents. This requires careful and effective coordination to make the most of the far-flung testers and other people involved with test management. There can be considerable technical challenges for this to be efficient, and therefore proper tooling is needed. The test management capabilities in ClearQuest with MultiSite is a tool that simplifies the complexities of geographically distributed test coordination.

Should you utilize a Web client or automatically replicated data? These are two solutions available that make collaboration with remote practitioners possible. The former is simple and relatively easy, but there is still a potential constraint of network latency, especially if accessed across the globe. For remote access by a limited number of people or with limited functionality, this is a good solution. However, for situations where a number of people in different locations make up an overall virtual testing team, you will need to have data copied on local servers to maximize the speed at which they can work. This also means that you will need an easy and seamless way to automatically synchronize the data across each location. This is where ClearQuest MultiSite can be essential for test management.

**Defining and enforcing a flexible testing process**

A good, repeatable process can help you understand a project's current status and, by being more predictable, where it's headed. However, different projects will have different specific needs for the testing effort, so a test management process that automates workflows needs to be flexible and customizable. The process should be repeatable (to provide predictability), but more importantly, it must allow for improvements. It has to be easy enough to make revisions, including adjustments during the course of an iterative project, so that it can be optimized through changing needs.

Defining a process with workflows to guide team members doesn't do much good if it can't be enforced in any way. How strongly it needs to be enforced will vary with different organizations and projects. Software projects in many organizations now have a need to comply with various regulations, such as SOX and HIPPA. Some have a need for auditability of changes, project
history, and other strict compliance validation such as e-signatures. Whether your project’s test management requires strict process enforcement or a more casual approach, you need a mechanism for defining and enforcing something. One such test management tool that provides all of these capabilities is the test management capabilities in ClearQuest.

**Coordinate and integrate with the rest of development**

Software testing has traditionally been kept highly separated from the rest of development. Part of this comes from the valid need to keep the assessment unbiased and increase the odds of finding defects that development may have missed. This need is especially apparent with acceptance testing, where the best testers are ones who are blind to the design and implementation factors. However, this specific need only represents one of many aspects of software testing, and it should not create the barrier and impediments to developing quality software that it usually winds up doing.

Software testing must be integrated with the other parts of software development, especially disciplines such as requirements management and change management. This includes vital collaboration between the different process roles and activities, maximum communication of important information, and integrated tooling to support this. Without this coordination, quality will be reduced from missed or misunderstood requirements, untested code, missed defects, and a lack of information about the actual software quality level.

**Communicate status**

An effort is only as valuable as it is perceived to be, and how it is perceived depends on what is communicated to the stakeholders. Good test management must provide complete and proper reporting of all pertinent information. Ongoing real-time status, measurement of goals, and results should be made available, in all the appropriate formats, to all relevant team members on the software development project.

Reporting should also be more than just traditional static documents. Given the constant changes going on, it is necessary to have easily updatable output in a variety of formats to properly communicate information. All of this will enable different project roles to make the right decisions on how to react to changes as the project progresses.

Information from the different software disciplines is not entirely separated. This article has already mentioned the important relationships between test management and other disciplines such as requirements, change and configuration management, and development. It is therefore crucial that the outputs coming from test management can be easily combined with other project data. Current technology makes possible consolidated views combining all project metrics into a dashboard so that the overall project health can be determined. Tools also make it possible to clearly show and assess the relationships between test, development, and other project artifacts.

**Focus on goals and results**

Decide on quality goals for the project and determine how they might be effectively and accurately measured. Test management is where the goals, the metrics used to measure such goals, and
how the data for them will be collected are defined. Many tasks in testing may not have obvious completion criteria. Defining specific outputs and measures of ongoing progress and changes will more accurately define the activities and tasks of the testing effort. Keeping specific goals and metrics for testing in mind not only helps track status and results, but also avoids the last-second scramble to pull together necessary reports.

Storing test management results in a single, common repository or database will ensure that they can be analyzed and used more easily. This also facilitates version control of artifacts (including results), which will prevent problems with out-of-date or invalid information. All of this will help project members to understand the progress made, and to make decisions based on the results of the testing effort.

Automate to save time

There is a lot to test management, and its many tasks can be very time consuming. To help save time, tools can be used to automate, or at least partially automate, many tasks. While simple tools like word processors and spreadsheets provide great flexibility, specialized test automation tools are much more focused and provide a greater time-saving benefit. Tasks that benefit greatly from automation include:

- Tracking the relationship of testing to requirements and other test motivators
- Test case organization and reuse
- Documentation and organization of test configurations
- Planning and coordination of test execution across multiple builds and applications
- Calculating test coverage
- Various reporting tasks

Proper tooling and automation of the right tasks in test management will greatly improve its value and benefits.

Summary

An important step to improving software quality is advancing test management practices beyond old-fashioned, document-based methods. Test management encompasses various functions, including planning, authoring, executing, and reporting on tests, as well as addressing how testing fits in and integrates with the rest of the software development effort. There are a number of daunting and inevitable challenges to test management, such as scarcity in time and resources, testing teams located in remote geographic locations, problems linking testing with requirements and development, and reporting the right information.

The good news is that there are a number of best practices that can help prevail over these challenges. Starting test activities early and iteratively, focusing on goals and results, and coordinating and integrating with the rest of development will keep the testing effort from becoming an afterthought to the software. Maximizing the reuse of test artifacts, leveraging remote testing resources, defining and enforcing a flexible testing process, and automating will all help overcome resource constraints.
One tool that can implement these practices is the test management capabilities in ClearQuest. It directly addresses many specific technical needs, for example: working with offshore teams through ClearQuest MultiSite. It also provides a flexible framework for creating the right test management solution for any project or organization's needs.

© Copyright IBM Corporation 2006
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)