

QMTTest: Quality Through Testing



Overview

Code that has not been tested adequately generally does not work. Yet, many applications are deployed without adequate testing, often with catastrophic results. It is much more costly to find defects at the end of the release cycle than at the beginning. By making it easy to develop tests, and execute those tests to validate the application, QMTest makes it easy to find problems earlier, rather than later.

Product managers understand the value of testing. Inadequate testing results not from a lack of awareness, but from a lack of tool support. Without proper tools, creating tests is difficult; executing tests and interpreting the results is nearly impossible.

Major software vendors who use professional tools for revision control, build management, and code comprehension often still rely on fragile command scripts to perform testing. The scripts in use within a single organization tend to have widely varying interfaces and produce output in different formats.

The few tools that do exist often focus on a particular application domain (such as the testing of graphical user interfaces) or enforce particular testing paradigms that may not be appropriate for all organizations. Integrating domain-specific tools to form a comprehensive testing plan results in additional costs. Tools that constrain the way in which tests are developed, or require that all tests output results in the same format, impose restrictions that get in the way of testing; they force users to focus on the tool rather than on the testing.

QMTest provides a cost-effective general purpose testing solution that allows an organization to implement a robust, easy-to-use testing program tailored to its needs.

Advantages

■ POWERFUL

QMTest can be extended to handle any application domain and any test format. QMTest works with existing testsuites, no matter how they work or how they are stored. QMTest's open and pluggable architecture supports a wide variety of applications.

■ INTUITIVE

QMTTest can be installed in minutes. All QMTTest operations, including test creation and execution, can be performed from QMTTest's web-based user interface.

■ APPLICABLE

QMTTest is useful to managers, software engineers, quality assurance engineers, and testsuite designers.

■ PORTABLE

QMTTest runs on all major platforms. QMTTest can test software running on platforms ranging from embedded systems to supercomputers.

■ SUPPORTED

CodeSourcery provides a variety of services for QMTTest users including training and customer support, the implementation of custom testsuites, and integration with other tools.

Powerful

QMTTest can serve as the core infrastructure in any testing program. QMTTest can test everything from graphical user interfaces for Windows applications to database query language implementations on mainframes. QMTTest can be used by developers to run a few hundred tests before checking in code, or by Quality Assurance (QA) engineers to run hundreds of thousands of tests distributed over a large testing farm. QMTTest's power comes from its pluggable component architecture; nearly every aspect of QMTTest's behavior can be extended and customized by writing small amounts of code in the popular Python programming language.

Extensibility

■ DOMAIN

QMTTest can be extended to test any application domain. Extensions that embed domain-specific data allow tests and results to be presented in a manner that matches the domain.

■ STORAGE

QMTTest's default test database format can be modified, extended, or replaced to allow tests to be stored in a format that is suitable for the application domain.

■ REPORTING

QMTTest allows tests to store any relevant information in reports. QMTTest's reporting format is based on the industry standard Extensible Markup Language (XML) to facilitate the generation of custom reports and graphical output.

■ EXECUTION

As shipped, QMTTest can run tests in serial on a single machine, or in parallel on a testing farm of possibly heterogeneous machines. For very large test loads, or for specialized situations, QMTTest can use a custom test execution engine to obtain optimum performance.

Domain

Every application domain needs to be tested in a specific way. For example, when testing a programming language compiler, the natural inputs are source files, and a test succeeds if the code generated by the compiler executes as required by the language specification. For a graphical user interface, on the other hand, the natural inputs would be mouse clicks and keystrokes in particular locations, and the test would succeed if the display was updated appropriately.

QMTTest is designed to accommodate any application domain. QMTTest can use a customized “test class” that describes the inputs to a test, how to execute the test, and how to determine whether the test passed or failed. For example, to test compilers, you would create a test class that runs the compiler, executes the generated code, and examines the output of both the compiler and the generated code to determine success or failure. Similarly, to test a graphical user interface, you would create a test class that simulates the user’s manipulation of the interfaces and checks the resulting display.

Storage

Just as the execution of tests varies with the application domain, so does the natural storage format. For example, a natural format for a compiler test is a source file with special comments that indicate what output should be expected.

QMTTest supports the use of custom “test databases” to solve exactly this problem. A test database can create test instances from any storage format, or even generate them on the fly. By using a customized test database, you free your engineers from focusing on QMTTest and allow them to focus on the testing itself. For example, the compiler developer need only save a new source file to her disk to create a new test; the graphical user interface developer need only record his interactions with the interface.

Custom test databases also protect your investment in existing tests. A custom database and custom test class can be used so that you do not have to manually modify any of your existing tests to use them with QMTTest. Using these techniques allows complete integration with QMTTest; you can use QMTTest’s graphical interface to manipulate your existing tests and even to create new tests.

Reporting

Often, it is not enough to know whether a test passed or failed. If it failed, you want to know why it failed. If it passed, you might want to know how long it took to run, or how much memory your application used. QMTest allows you to embed additional information in test results so that you can record this information. The information is stored in an XML format so that you can extract it later with any tool capable of processing XML, including web browsers and report generators.

Customized report generators can display this information in the form of web pages, spreadsheets, or slide presentations. These easy to understand, graphical displays allow managers to identify problems in the application and direct resources to solve those problems.

Execution

For many applications, running tests in serial on a single processor is sufficient. If you have multiple processors, on one or more machines, QMTest can automatically take advantage of that additional power to run your tests in parallel. But, if you are testing a scientific application on a massively parallel machine, or if you are performing performance testing for a distributed database, you may have specialized requirements for test scheduling, execution, and distribution.

In these situations, QMTest can use a customized “test execution engine” to execute your tests. The test execution engine can use feedback mechanisms, such as dynamic load balancing, to allocate tests among machines. For server throughput testing, it could provide information to tests about network topology so that clients could run closer to, or farther from, servers.

Intuitive

QMTTest is as easy to use as it is powerful. QMTTest is simple enough that it is just as useful in a one-person development project as it is when developing a large application for the enterprise. Users can take immediate advantage of QMTTest without any customization. QMTTest uses a web-based graphical interface, but still allows power users to edit and execute tests from within their editors or from the command line. QMTTest comes with a variety of useful test classes, suitable for general purpose use.

Simplicity

■ INTERFACE

QMTTest uses a graphical, web-based interface to provide a uniform look and feel on any platform. New tests, and entire testsuites, can be created by filling in a few simple forms.

■ FUNCTIONALITY

You do not need to create domain-specific test classes, or special purpose test databases, to take advantage of QMTTest. The test classes and database that are provided with QMTTest are suitable for use in a wide variety of application domains.

■ INTEGRATION

Extensions to QMTTest integrate seamlessly. New test classes and databases work just like the test classes and databases that come with QMTTest.

■ ENCAPSULATION

Users who work with one part of QMTTest do not need to understand other parts of the system. Testing experts can design test classes without concerning themselves with the test storage format. Users can create tests without knowing how tests are stored or how the test class is implemented.

Interface

QMTTest is easy to use from the moment you download it. QMTTest is available in prepackaged form for a variety of popular operating

systems. Once installed, users create new tests and testsuites using a web browser. QMTest comes with an integrated web server; no server configuration is required. Every form has context-sensitive help that explains how to use the form, and links that automatically display the online manual. Tests can be executed from directly within the user interface.

Using this interface, users can execute just one test, or entire testsuites. By saving the results of one test run, users can create a “master” results file. QMTest will then display results relative to the master information. For example, QMTest will specifically flag tests that passed in the master results file, but failed in the current environment. This feature means that users no longer have to remember which tests are “supposed to fail.”

Because some users prefer not to use graphical interfaces, QMTest allows users to create and execute tests manually. Power users can even create programs that automatically create new tests.

Functionality

Although you can extend QMTest to target any application domain, many users of QMTest will be able to use the default components provided without any modification. QMTest ships with a variety of useful test classes, including a test class that runs an external program and checks its exit code. Using this test class, users can immediately use existing testing scripts and special-purpose programs with QMTest.

QMTest’s default database format uses XML. The use of XML facilitates the storage of the database in any popular source control system. Because XML is a text format, users can manipulate the database via a text editor, or through scripts and other programs. For example, you can easily create a shell script that creates new tests and adds them to the database.

QMTest comes with a powerful execution engine that can automatically execute tests in parallel across multiple machines. Users (or administrators) produce a network description that indicates which machines are available, and what operating systems are running on those machines. QMTest can then automatically divide the tests that are to be run among these machines.

Integration

Extensions to QMTest are treated just like core components of QMTest. QMTest's graphical user interface automatically displays forms for the creation of tests using customized test classes, and automatically stores and retrieves tests using customized test databases. In fact, all of QMTest's default test classes, databases, and execution engines are implemented using the same technology that is used to implement extensions.

Encapsulation

QMTest is designed so that you only have to understand the parts of QMTest that you are using. A user who is creating a new test fills in a form without needing to understand how the test is stored or executed. Therefore, once the initial investment in a customized test class or database has been made, there are little or no recurring costs. Return on investment occurs quickly as users are able to take advantage of the power afforded them by the customized test class.

Applicable

QMTTest is useful to people working in a variety of different capacities, including:

- MANAGERS

Managers can more accurately predict milestone dates and understand how to allocate resources.

- SOFTWARE ENGINEERS

Software engineers can write better code and test more comprehensively.

- QA ENGINEERS

Quality assurance engineers can write more tests, write better tests, and provide better feedback to developers.

- TESTSUITE IMPLEMENTERS

Those who create testsuites can provide greater value to their customers by building testsuites that are easier to use.

Managers

Software systems are fragile; changes in one part of the code often affect other parts. When the code does not build, or does not run correctly, the entire development team may be unable to make progress. By making it easier for developers to test changes, organizations can reduce the time other developers spend sitting idle. By encouraging the development team to use QMTTest, managers produce better quality software with the same resources.

Managing a software project is a difficult task. One of the constant questions is whether or not the software will be ready to ship by the scheduled date. Nothing is worse than realizing that there is a critical problem with the product just as the freeze date nears.

QMTTest makes it easy for managers to keep track of the overall progress of their development projects by viewing reports and charts that indicate how many tests are failing, which failures are new, and what categories of tests have the most failures. This information makes it possible to allocate resources to the parts of

the code that have the most problems, and to provide realistic scheduling information to other parts of the organization.

Software Engineers

Software engineers know how hard it is to build software that works perfectly the first time. Even the best engineers are not perfect. Nothing is more satisfying than writing code that “just works.” By testing their code carefully, and writing new tests as they go, engineers can improve the reliability of the code they write.

Of course, engineers do not like to spend a lot of time and effort writing tests, or running them. QMTest makes both creating and executing tests easy. The parallel execution feature of QMTest allows developers to make use of unloaded machines, leaving their workstations free for development.

QA Engineers

Using QMTest lets quality engineers focus on testing applications, not on the mechanics of getting tests written and executed. QA engineers will particularly appreciate QMTest’s flexibility. When working on applications with unique features (like audio output or a distributed execution model), engineers are not limited to fragile shell scripts. Instead, the flexibility of QMTest allows quality engineers to build domain-specific test classes that target the application in question.

By using custom test classes, QA engineers can embed additional data in test results that helps to pinpoint problems, or provide information about program performance in addition to correctness. QMTest’s command-line interface makes it easy to set up nightly test runs.

Testsuite Implementers

There are now commercially available test suites for every major programming language, for the Hypertext Markup Language (HTML), for the Lightweight Directory Access Protocol (LDAP), and even various parts of the POSIX specification. Of course, testsuites are more effective when they are easy to use and when the results are easy to interpret.

The implementers of these testsuites do not wish to limit the use of their products by restricting their use to particular platforms, or by requiring customers to buy additional tools. They want to focus on the application domain in which they are experts, not the details involved in constructing a portable test execution package.

By packaging testsuites with QMTest, implementers can provide greater value to their customers while concentrating on what they do best. Customers appreciate the functionality provided by QMTest and the fact that that QMTest provides a uniform interface across testsuites.

Portable

The technology used to develop QMTest was selected because it is portable to all major software development platforms, including Windows, Mac OS X, GNU/Linux, and other UNIX like systems. The web-based interface works with Internet Explorer, Netscape, and other popular browsers. Storage formats have been carefully designed to avoid portability problems when moving tests from one platform to another.

QMTest can also be used to test software for embedded systems or other special purpose hardware. QMTest can perform remote operations to execute programs on the embedded hardware and report the results on the host system. Similar techniques can be used to execute tests on mainframes, or other machines where QMTest may not be available.

Supported

Setting up testing infrastructure correctly is of critical importance. It is not always easy to figure out what to test, or how to test it. Creating domain-specific tests, custom test databases, and domain-specific or organization-specific results analysis tools can take a substantial amount of effort; effort which might be better devoted directly to product development.

CodeSourcery works with its customers to install, integrate, and customize QMTest. CodeSourcery can help its customers create test suites that provide complete coverage of their applications and take full advantage of QMTest.

CodeSourcery's customer support, including upgrades to new versions of QMTest, is useful even to organizations that use QMTest without any extensions.

CodeSourcery also provides training sessions to show users how to take full advantage of QMTest.

Contacting CodeSourcery

To obtain additional information about QMTest, to purchase services related to QMTest, or to obtain a demonstration version of QMTest, please contact CodeSourcery in one of the following ways:

By telephone: (916) 791-8304

By email: info@codesourcery.com

On the web: <http://www.codesourcery.com>