Software Testing – A Time for Standard Models

Introduction

Software technology has always been in need of testing. This is due to a number of particular aspects, including the fact that (a) the number of usage scenarios tends to be high, (b) the user's actions are sometimes unexpected and hard to predict, and (c) more then in other cases, the software developer tends to have a tunnel vision and omit from his or her code all the complexities of points (a) and (b).

Software testing has evolved from spontaneous "kicking the tires" to well thought methodologies and to various technologies that support the activities of the tester. This led to increased specialization, with whole teams, departments or even outside service providers dedicated to the single goal of insuring a high quality of the software. Testing has graduated from a sideshow to a major component of software technology, and even to a specialized industry.

This evolution of software testing created, among other things, a specialized vocabulary and a number of concepts, which are universally accepted. When one speaks of unit testing, system testing, acceptance testing or performance testing, most people involved in the software development process understand the terms. This proves that the industry is already moving to some incipient forms of standardization. However, we believe that this is only a first step, and further standards may greatly benefit not only the people directly involved in testing, but also everybody connected with software development and even the software user community.

As with many other standard models, the most immediate and obvious advantage of a testing model standard would be the sharing capability. This means that various parties involved in testing activities will be able to share information, by utilizing software-testing tools that could communicate between themselves. The whole testing process would become more efficient as separate steps could be seamlessly integrated. A tool which supports unit testing will be able to pass information to a tool which supports integration testing and the results would be analyzed by a third tool which analyzes test coverage or test results.

Testing communities

As standards are meant to help different parties communicate between themselves and work together, it is important to identify which are these parties and what are their roles in the testing process. It is also important to notice that all these parties have slightly different needs for testing.

- System analysts and architects collect requirements and design the applications or software products at a functional and structural level. They are aware of the use cases involved and of the activities that could be performed. They have an interest that all such cases and activities are properly tested to assure the functional

integrity of the application. They may be also interested in performance testing for the validation of their major architectural decisions.

- Software developers are interested in unit testing various components that they create, as well as in the subsequent integration testing. They are mainly concerned that the software is bug-free and delivers the expected outputs.

- QA teams are exclusively focused on all aspects of the software quality and may plan, design and run a variety of tests. They may also analyze the test results and make sure that the software passes certain preset standards of quality.

- End users, the final beneficiaries of the software, may perform a number of functionality and performance tests, in what is commonly called acceptance testing. The tests results help them to accept a software application or to select between a number of software products offered by different vendors.

- Security experts may perform specific testing to insure that no unauthorized people have access to protected resources.

- Auditors may perform functionality tests to insure that the application conform to the users' standards and accurately reports information.

Given the current outsourcing trends, it is possible that some of the communities defined above are not limited to one company, but span a number of vendors offering various services.

As such communities have different testing goals, they may employ different testing tools. A number of testing tools were developed and are offered by software vendors, while some others are developed and used in-house.

Testing tools

We consider here a number of categories of testing tools, classified by different services they offer. It is however possible that some tools offer multiple services, although it is rare that one tool covers all needs.

- Unit test generators may create unit test cases, which are very specific and based on the internal knowledge of the application code. One may, for instance, generate the test case for a particular computation in a program.

- Some tools allow the users to capture test cases based on particular actions performed by a person in a runtime environment. The user executes certain operations and his or her actions are automatically recorded and transformed in a test case.

- A number of modern UML tools allow the users to automatically create test cases based on requirements or on use cases and action diagrams.

- There are tools specialized in automatic execution of test cases. They may run a very large number of cases in a batch mode and report the results.

- Some tools offer management of the test cases and of the testing process in general. They may also analyze the results and offer sophisticated summary reports.

Benchmarks

Benchmarks are standardized collections of test cases, which could be used to measure the performance of a software product. Since the test cases are the same for various software products, the results may be used to measure the performance of one against another.

The need for cooperation

Since various communities and various tools are involved in the same process (although in different steps of the process), it is natural that they need to exchange information. The QA team, for example, may want to include the tests used by developers during unit testing. If the user acceptance tests failed, the developers would like to have access to the test cases that produced the failures. These are just some of the scenarios and we can easily imagine much more.

We can talk about two types of testing cooperation:

- Vertical cooperation occurs between various parties that are part of the same process of building, releasing and using the software. In the classic waterfall approach to software development, the test cases need to pass from one team to another through all the development phases. The capacity to share test cases is not guaranteed, especially when outside vendors participate and use their own tools.

- Horizontal cooperation occurs when a different organization intends to run the same tests as another organization. This happens, for instance, when one company sells the software to another company and intends to include the test cases as one of the assets. It may very well happen that the buyer has already standardized on another testing tool, in which case the transfer of the test cases is very difficult.

A test model standard would facilitate the sharing and cooperation in all these cases.

Elements of a test model standard

A standard may emerge in an organization like OMG as a result of multiple requirements and multiple proposals submitted by multiple parties. We can, however, list some of the

possible features of such a standard. In principle, the standard model should be able to describe in a unique format…

- Unit test cases – expressed as inputs and expected outputs.

- System and integration test cases – expressed as a series of user actions followed by expected system responses.

- Code Coverage Models to cover predictive and actual code coverage attained
    - Predictive Coverage Models provide a useful simulation of the test run before it happens

- Definition of a "Base" or "Smoke Test" Package – Given the economics of testing, a strong need is to be able to run a basic suite of tests – frequently – to ensure basic level of functionality. Standardizing on how this is defined, managed and maintained would help test case creators fashioning these test suites efficiently

- Design for Testability – The ability to assess a System Design (including Use Cases etc.) for how well it can be tested is a useful yardstick. While the ultimate goal has to be end-user satisfaction, the internal health of an architecture is often assessed by how well and easily it can be tested. A standard means of measuring this could be an effective gauge for Test Professionals

- The result of each test case, including not only data, but also performance or other aspects.

- Optional pointers or references to the actual code involved in a test, at any level of detail (system, subsystem, program, class, method, line of code).

- Optional pointers to other models from which the test cases are derived (for instance to the actions in an action diagram).

- Consolidation of test results (percentage of fail or success by various categories)

Relationships to other standard models

As software testing is a measure and a description of the quality of software at a given moment in time, it is natural to link a test model to any other models describing the software. Such links are already mentioned above, and they may include references to:

- Nodes in the parse tree of a program (ASTM)
- Syntax constructs or various software assets (KDM)
- Use case, actions (UML)

- Business rules (SBVR)

Scenarios

- The teams involved in the same development process could pass test cases from one to another.

- One tool is use to generate test cases, another to run them, yet another to analyze the results. The best of breed tool could be used for each case.

- Benchmarks could be easily transferred to various environments and shared between different technologies.

- A company can change both the tools for testing (in case of internal testing) or the testing service vendors, not being locked in a proprietary format.

OMG as the best venue for developing a Testing Model

The expertise and experience in building a framework for a test model could be found in OMG. The OMG is an open membership, not-for-profit consortium dedicated to producing and maintaining specifications for interoperable enterprise applications. The OMG membership roster includes many of the most successful an innovative companies in the computer industry, as well as those at the forefront of using technology to gain a competitive edge in their business. All have made the commitment to actively participate in shaping the future of enterprise, Internet, real-time and embedded systems.

OMG's modeling standards, including the Unified Modeling Language™ (UML®) and Model Driven Architecture® (MDA®), enable powerful visual design, execution and maintenance of software and other processes, including IT Systems Modeling and Business Process Management. OMG's middleware standards and profiles are based on the Common Object Request Broker Architecture (CORBA®) and support a wide variety of industries.

CONTRIBUTORS (so far)

Michael Oara – Relativity Technologies, Inc.
Jaideep Mirchandani - Relativity Technologies, Inc.