Ensuring robust, rapid-fire releases—the QA program at Sakonnet

By: Alarik Myrin, Chief Technology Officer, Sakonnet Technology, LLC

Automated testing is only one dimension of Sakonnet Technology's overall quality assurance program, which is vital for providing clients with five robust new releases each year of Xenon, Sakonnet's Java-based energy trading and risk application.

Introduction

A strong quality assurance (QA) program is essential for the long-term viability of any complex enterprise software company, and Sakonnet's is the foundation on which the entire company is built. While it might be possible for large software projects to survive for a short time skimping on the effort dedicated to quality, such projects are said to be running a "technical deficit," which over time can accumulate an overwhelming "technical debt." When this happens, it becomes impossible to evolve the software to meet new requirements or to pursue new opportunities.

Indeed, at Sakonnet we seek to turn the QA program from one of simply avoiding this kind of downside, into delivering exceptional value-added to our clients. By way of background, our main application, called Xenon®, is a Java-based multi-tier system used by banks and energy companies for capturing and processing wholesale energy market trades, and measuring resulting market risk. A single source code base is used by all our corporate clients. The QA program around Xenon, coupled with our hosted or ASP solution model, enables us to deliver robust upgrades of Xenon into production at a much faster pace than would otherwise be feasible. As a result, our end-users including traders and risk managers, can reap the value of each upgrade's new functionality sooner. Sakonnet now makes five Xenon upgrades available for production use per year, i.e. roughly one every two months, versus a typical industry level of once per year.

QA program goals; costs of late defect detection

Of course defects arise at each stage of any software development process; the question is how best to minimize and address them at a sustainable cost. An effective quality program should accomplish three main goals: a) minimize introduction of defects b) identify any defects shortly after they are introduced; and c) provide confidence that all material defects—and nearly all defects—have been identified and resolved.

A key point is that the cost of a defect skyrockets as the time increases between when it is introduced and when it is addressed. The sooner a defect is detected, the fewer people are needed to resolve the issue. The following table, drawn from industry studies, illustrates this point:

Table: Estimated average costs of fixing defects, based on when they are introduced and detected (in multiples of lowest-cost fix)

	When detected:				
When introduced:	Require-ments	Archit-ecture	Constr-uction	System test	Post-release
Requirements	1	3	5-10	10	10-100
Architecture	-	1	10	15	25-100
Construction	-	-	1	10	10-25

(Source: Code Complete, 2nd Edition, by Steve McConnell. Microsoft Press, 2004. pp. 29.)

For many software organizations, "quality" can be synonymous with "testing." While testing is a crucial part of a first-rate quality program, it is hardly the only part. In fact, if quality is not considered until the system test phase of a project, the costs arising from defects are probably ten times higher than they need to be.

The four dimensions of Sakonnet's quality program

Bearing this in mind, Sakonnet's QA program has four key dimensions: 1) company approach, 2) quality inspection, 3) testing, and 4) process tools.

1. Company approach

At Sakonnet, the company's approach is to focus on *quality as a business imperative*. As a result, quality is taken into consideration in all the company's major activities, including strategy, investments, resource allocation, staff selection and continuing training. For example, we make heavy investments in proprietary and commercial software and in the hardware needed to conduct testing processes, and now have 35 separate testing environments for current and development versions of Xenon. Also, while everyone at Sakonnet is involved in the quality program, 17% of staff is dedicated full-time to it as members of our QA team.

2. Quality Inspection

The next dimension of the Sakonnet quality program involves formal and informal quality inspections of every artifact (or piece of work-product) created during the software lifecycle, from requirements documents, to design documents, to test cases, to code. The goal here is to identify defects before they can cause us much damage.

In particular, code inspections—the act of reading code to look for problems—are quite different from testing, in that they can find quality problems unrelated to functionality, such as code readability problems or scalability concerns. They can also suggest extra tests that might not have been obvious from the functional requirements (so-called "white-box" tests). Inspections have been demonstrated to uncover both more total defects than testing, as well as more defects per hour. They can also be performed much earlier in the development process.

Outside of code inspections, the QA team inspects for quality all Sakonnet's business requirements documents, system specifications, and test cases.

3. Testing

Testing represents the third dimension of Sakonnet's quality process. As a mature testing organization, Sakonnet has a wealth of documented test cases that produce repeatable results. Sakonnet records the outcome of any given test run, and repeats the tests frequently, to make sure that the entire system continues to behave as expected after incremental changes are applied by the developers. The need for highly-frequent testing, and the fact of Xenon's ever-expanding functionality, means that testing must be as automated as possible, otherwise we would be swamped by the cost and burden of manual testing.

The first key aspect of testing, is for developers to practice test-driven-development, a typical part of "agile" development methods. It entails their writing "unit tests" (a test of a given process, e.g. a numerical calculation, that will have a certain output for given inputs) for each "method" (a particular sequence of functional logic in the source code) *before* they sit down to write the actual method. The tests initially fail, but they provide immediate feedback to the developer (as the method is written) about whether there are any bugs lurking in the method. Currently Sakonnet has 4,000 unit tests in place that run every hour, and a failure in any one of them causes our development staff to be immediately notified so that the problem can be addressed before the next hourly build of a development version of the software. Testing and feedback on this scale represents an astounding advance on conventional methods of software development.

On top of this developer approach, Sakonnet uses automated testing applications. One is from Mercury, called QuickTest Pro. An end-user's experience of Xenon is simulated. A Sakonnet QA engineer writes a script of actions the user would take. The application can then create mouse clicks and key strokes just as if a user were actually on the system, and also check that the results displayed on the screens are correct. These tests are run every night, and certain members of the QA team analyze their results every morning.

Automatic load and performance (L&P) tests written by Sakonnet are executed continuously on the most critical and intensive operations such as the End-of-Day

batch job (which generates all the formal reports on the day's trading activity and results), so that if any developer makes a code change which alters the results, or materially impacts the performance of these operations, the developer can be alerted within hours, rather than months later during the final system test executed before shipping a new release to clients. To make the tests as meaningful as possible, the QA team applies them to clients' actual trade portfolios, some of which are amongst the largest in the energy industry (under strict controls on handling such confidential information).

Finally, an exciting new automated testing technique named FIT (which stands for Framework for Integrated Tests) is being rolled out across the entire company. This method was originally created by Ward Cunningham. FIT turns the requirements artifacts produced by our business analysts into the basis for automated tests. The way it works is that business analysts place requirements into tabular models that serve as the data model for tests written by developers. The tables themselves (which could be written in Excel) serve as the input and expected output for the tests. By combining the two most challenging aspects of software development—communicating complex ideas and testing complex functionality —FIT promises huge gains in overall productivity.

In addition to these automated tests, the QA team at Sakonnet performs extensive manual exploratory testing, and creates and documents new test cases to expand the scope of our test coverage.

4. Process Tools

The final dimension of our quality landscape is process tools (foremost, a proprietary application called QUBA), put in place to track defects and issues through resolution. These tools are integrated with our overall development process tools (e.g. Subversion, an open source version control system) and produce many useful quality metrics that are used to refine the quality process and track the coverage of our testing efforts.

Conclusion

Sakonnet is dedicated to substantive, frequent, and (last but not least) *robust* new releases. Our focus on quality produces immediate benefits for our clients, since they start using the new functionality far more quickly. Our thorough and sophisticated QA program will continue to incorporate further technical advances allowing more automation and faster feedback on defects. In this way we can be sure of scaling up our software offerings to clients well into the future.