



December 20, 2005

The Software Quality Lifecycle

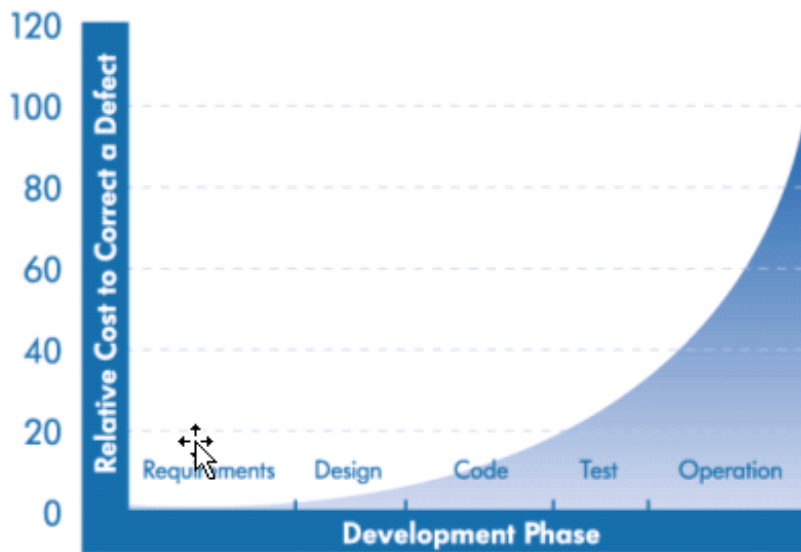
by Yochi Slonim

Missed deployment and delivery dates, budget overruns, failure to comply with industry regulations, interrupted workflows and frustrated customers are the natural byproducts of application flaws. The current approach to resolving application problems--and ensuring software quality throughout the entire useful life span of the application--is simply not getting the job done. The time has come for a sea change in how we manage problem resolution, and ensure quality for complex, composite new applications, as well as aging legacy systems.

In our online, wired world, the digital and physical worlds are so intertwined that almost nothing can happen unless the relevant applications are up and running. You would think that as software applications become indispensable, they would likewise become more and more dependable. Instead, the rising tide of systems complexity and interdependency are making enterprise application problems inevitable, intractable, and more elusive. Last year we saw Lufthansa cancel the flights of over 3000 booked passengers due to a computer fault in the check-in systems. We've seen insufficient testing lead to the failure of Nike's i2 demand forecasting in 2000, costing the company more than a forecasted \$100 million in sales. We've seen a failure to test for specific conditions contribute to the August 2003 blackout that affected the Northeastern United States and Canada. And beginning in June of last year bugs in connections between Hewlett-Packard's legacy order-entry system and SAP systems caused a backlog of customers orders ultimately costing the company \$40 million in lost revenue (*CIO*, 11/15/05). In fact, software errors, or bugs, are so prevalent and so detrimental that, according to a study published in 2002 by America's National Institute of Standards and Technology (NIST) software bugs cost the American economy \$60 billion a year or about 0.6 percent of its gross domestic product.

The Current Zeitgeist: Testing = Quality

Researchers from the Carnegie Mellon University estimate that programmers make between 100 and 150 errors per 1000 lines of code. A plethora of similar research has clearly established the legitimacy of Quality Assurance as a fundamental lifecycle process. Gone are the days when the length of the quality assurance phase was axiomatically the number of days between the end of coding and the release date. There are still some laggard organizations that task the QA team to "make-up" time lost in development, but the zeitgeist in software quality is still rooted in the concept that quality can be "tested in" to software applications. There are some solid economic roots to that thinking. In his software classic *Software Engineering Economics* (Prentice Hall, 1981), Barry Boehm published some, at the time, eye-opening information to demonstrate how the cost for removing a software defect grows exponentially with the phase of the development lifecycle in which it is discovered; see Figure 1.



Forward-looking organizations realize that the new paradigm for lifecycle application quality is to build quality assurance into every phase of the development lifecycle. There's no doubt that applying more rigor to the definition of business requirement and investing more in design reviews will result in software applications that are more aligned to the business

Figure 1.

needs (one dimension of total software quality). But many organizations are still locked in struggle to ensure the software works as designed--and performs as expected--after deployment. To address that challenge, they are optimizing their Quality Assurance Business Systems.

Optimizing the Quality Assurance Business System

The goal of a QA Business System is to integrate people, process and technology in an optimal organizational structure to deliver a very straightforward result--to help the development process deliver quality software faster, at a total lower cost.

Leading companies are making organizational decisions to create separate QA organizations; they are elevating the role and stature of the QA engineer; they are implementing best practice processes with rigour. However, it is surprising that with the maturity and wide availability of testing technologies, the technology dimension of many QA Business Systems is so undeveloped. The technologies available for testing are usually categorized into two categories--tools that support the manual testing process and tools that execute some automated testing.

The current "sweet spot" for QA technology--the area where technology can yield significant ROI--lies in automating as many aspects of the manual testing process as possible. According to the IT industry consultancy, Gartner, 80 percent of all applications are either not tested or are only tested manually before being delivered into production. There are a number of solutions that manage the administrative process of testing (test planning, execution tracking, defect tracking) such as [Test Director](#) from Mercury Interactive and [ClearQuest](#) from IBM's Rational division.

But, many companies are beginning to realize the powerful advantage that comes from automating the tasks of the QA Engineer, not just the QA Manager. For example, consider what happens when the QA Engineer does discover a problem--the process is particularly cumbersome. First, the problem must be replicated (to ensure that it is reproducible). Then, detailed documentation is prepared to record the exact sequence of steps executed, data entered, and so on. The process of replicating and documenting problems found in the QA process can consume 25-50 percent of the total QA effort.

This documentation is sent to the development team for analysis, which usually starts with the developer's attempt to recreate or replicate the problem situation so that traditional debugging tools can be deployed. This generally requires the developer to try to recreate the application environment in which the problem occurred and recreate the problem using the information sent by the QA team.

Unfortunately, although a lot of time is spent on documenting problems found in QA, many problems are incorrectly communicated or insufficiently described, which results in development not being able to recreate the problem to analyze for root cause. Despite a significant amount of "ping pong" communication between developers and testers, a troubling 22 percent of problems found in QA are not reproducible by development. With more complex composite applications, that number is rising towards 30 percent. The net result is a significant number of problems that were actually found in the QA process are not fixed before the product is released. As we learned from Barry Boehm's 1981 research, fixing a problem before the product reaches general availability can reduce the cost of fixing it by orders of magnitude.

One innovative organization, Cerner Corporation, has implemented AppSight, an application problem resolution system from my company (Identify Software), to automate the problem handling aspects of software bugs discovered during QA. As a world leader in healthcare software solutions, Cerner is replacing paper charts with intelligent, interactive electronic forms designed to improve patient care and business management. Until recently, Cerner took the industry-standard approach to manual QA testing, with industry-typical costs and results.

Beyond the Testing Mandate

Any organization that is serious about software quality needs to internalize the concept that no Quality Assurance business system can completely ensure that your software is free of bugs. Today's applications are complex systems, comprised of numerous loosely coupled processes owned and managed by multiple parties. There has never been an instance, in recorded history, of a complex system (organic or man-made) that doesn't sometimes fail--that's why we have doctors, car service workshops, and a hole in the ozone layer. In testing, you simply do not get the functional "stress test" of hundreds or thousands of users performing functions in unexpected combinations. And, research indicates that approximately 40 percent of application problems are not caused by a software bug, but by some condition in the system environment in which the application is deployed. But these very real problems still need to be solved--hopefully without needlessly escalating the issue to the development team, a practice that significantly affects downstream development dates. Any organization that is serious about software quality needs to extend their quality business system to quickly fix problems that occur once the application is deployed.

Conclusion

Forward-thinking companies are far beyond thinking of quality assurance as a quick shakedown period between development and delivery. They are approaching the software quality lifecycle challenge with a fulsome QA Business System that considers the dimensions of organizational issues, process, people, and technology. These organizations are delivering higher quality solutions to their constituents much faster, and supporting those applications throughout the lifecycle with less expense and higher end-user satisfaction.

Yochi Slonim is CEO of Identify Software. He can be contacted at <http://www.identify.com/>.

