A business discussion of software quality
December 2004

Rational® software

The business value of software quality

Geoffrey Bessin
Market Manager, Software Quality
IBM Software Group

Contents

Introduction

Business leaders often view software quality as a luxury—something that can be sacrificed, if necessary, for more functionality, faster development or lower costs. However, in practice, software development organizations that have a firm commitment to quality can actually speed development, reduce costs and add new features with greater ease. Organizations that develop low-quality software, whether for internal use or for sale, are always looking backward, spending time and money on fixing defects in "finished" products. In contrast, an organization that builds in product quality from the beginning can be forward-looking and innovative—it can spend its resources on pursuing new opportunities. Delivering quality is also a powerful differentiator in a marketplace where high-quality software is now the exception rather than the rule.

But where do you start? The knee-jerk reaction to a demand for better quality is to expand the testing team. But that is never enough. Instead, the entire project team must commit to improving quality throughout the software development process, including post-deployment—and that commitment must be driven by the business's uppermost leaders.

Defining quality

What exactly is quality? You might know it when you see it, but have you incorporated measures into your software development process to achieve it? Ask yourself this: when you discuss quality with your team, are you all talking about the same thing? If not, then your project is at risk. Too often, software development organizations operate with a loose, general notion of quality and tolerate defects that most engineering disciplines would not allow. In contrast, a solid definition of quality that all team members understand and accept promotes thoroughness and attention to detail. And, like a well-written requirement, a clear definition of quality leaves little room for misinterpretation. For most projects, it is important to define quality via a group of standards or metrics that the software must measure up to.

In addition to defining what you mean by quality, you also need a well-defined process to achieve it. For this process to work, everyone on your team must understand and follow it. We will discuss this in more detail later.

Creating a high-quality product

In the domain of business applications, quality must be defined in terms of the target audience: the software users. You might create a bug-free, aesthetically appealing product, but if it fails to solve the business problems for which it was designed, it is a poor-quality product. Quality is the tangible and intangible sum of functionality, usability, reliability, performance, scalability, supportability, security and other factors. James Juran aptly defines quality as "fitness for use."[1] He goes on to say that a product is not of high quality unless it adds value for both the consumer and the manufacturer.

Put slightly differently, quality encompasses added value plus attention to detail. Both a luxury car and an entry-level car will get you from point A to point B. But the luxury car offers features and capabilities that go beyond the essentials of transportation: usability, safety, comfort, reliability and so on. Product quality also reflects the process behind the product. In the software world, a high-quality process can keep development organizations from losing time reworking, re-factoring and rewriting software. These organizations can produce more innovative and creative products because they have more time to think about adding value and quality details.

Development organizations with a quality focus know that they must do more than eliminate software bugs; a quality application must do more than simply provide correct results without crashing. Does the application meet stakeholder requirements? Is it usable? Secure? Scalable? Reliable? Easily maintained? Easily extended? Simple to monitor? If development organizations can answer "yes" to these questions, then they have created a high-quality application. In short, quality is the total result of what you produce. And you will get quality results only if you constantly strive for them throughout development, integration and testing. This applies equally well to projects involving packaged or homegrown applications, upgrades or "greenfield" development work and the extending, integrating, and modernizing of legacy applications (see Figure 1).

[1] Juran, James M. and Frank Gryna. *Juran's Quality Control Handbook*. Texas:McGraw Hill. 1988

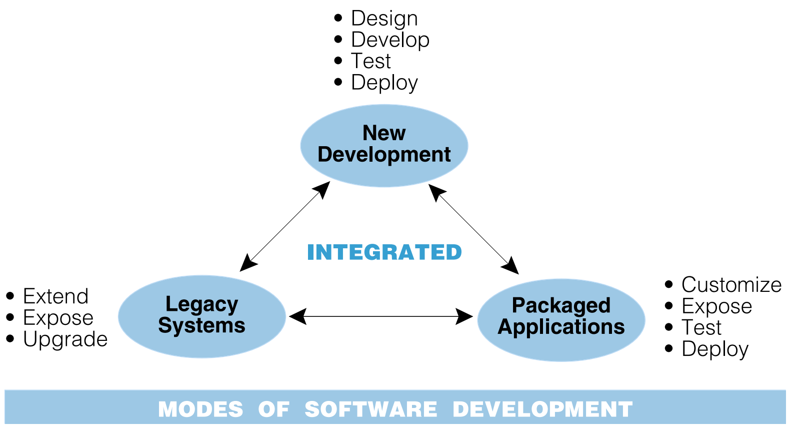

Figure 1: All modes of software development require the same attention to quality

In the long run, an organization's specific definition of quality may be less important than the fact that it has established a definition at all. But at a high level, a good definition of quality will help an organization answer some basic questions:

- *Are we building what is necessary? Is it what the customer (internal or external) wants?*
- *Does the application do its job well? Is it usable and functional?*
- *Does the application run the way it is supposed to? Is it meeting service-level expectations in deployment?*

If your organization cannot answer these questions in the affirmative based on concrete evidence, then it has an opportunity to change its ways and reap great benefits.

Business benefits of focusing on quality

As Juran's definition implies, it is not just the customer who benefits from a focus on high quality. Businesses that value quality become more responsive and innovative, increase their competitive differentiation and greatly reduce their total cost of development and ownership. Let's take a closer look at these benefits.

Quality enables responsiveness and innovation

Successful organizations recognize the importance of continuous innovation and differentiation, which require rapid responses to a constantly changing business landscape. The flexibility and malleability of an organization's software enable rapid responses, and organizations that embrace software development as a core business process quickly move ahead of the pack. However, the same flexibility and malleability are also what make software so vulnerable. One small change to one application can introduce a flaw that brings down an entire system and cripples the business. Knowing this, software development teams sometimes assume a "fix it later" mentality; they move forward with products they know are flawed—a liberty that no other engineering discipline would permit.

When 450 CEOs around the globe were interviewed about their current strategic issues, ambitions and concerns for IBM's Global CEO Study 2004[2], they ranked "responsiveness" as their number one goal, followed closely by "new, differentiated products," "operational efficiency," and "improved business models." These goals are consistent with IBM's On Demand Business model—in today's environment, businesses need to innovate, react quickly and efficiently to change and differentiate themselves from the competition. If an organization's internal software systems are flawed or if product development teams are constantly struggling to fix problems in last year's release, then the business cannot achieve these goals.

To operate successfully in an on demand world, businesses must keep a laser-like focus on quality while developing and maintaining the software systems it relies on; quality systems are what enable businesses to react, adapt, quickly deploy new solutions and maintain a competitive edge.

[2] IBM Business Consulting Services. *"The Global CEO Study 2004."* http://www-1.ibm.com/services/us/bcs/html/2004_global_ceo_study_gen.html.

Quality is a differentiator

As a discipline, software development has quality standards far below those of other engineering disciplines. Managers and developers are good at rationalizing their products' shortcomings—the functionality they offer is "better than nothing" or "better than the former system," they say, and there are manual backups in case the system crashes. But would a structural engineer reason away the collapse of a building in a similar way? Would an aeronautical engineer merely shrug his shoulders if a control panel malfunctioned in mid-flight? To support a business effectively, software engineers must also move toward a zero-tolerance policy for defects.

What difference can higher quality software make to a company in the marketplace? Consider the automobile industry in the late 1960s and 1970s. Japanese automakers had adopted a rigorous quality assurance (QA) program and soon gained a reputation for producing highly efficient, well-designed, reliable cars. This superior quality set their products apart and raised the industry's quality baseline. It also allowed Japanese automakers to develop further innovations in fuel efficiency, safety and manufacturing processes as their competitors tried to catch up.

Similarly, when any company in any industry produces higher quality software—for either internal use or external sale—it raises the bar. The quality of an organization's internal human resources, financial and customer relationship management systems affects the bottom line in ways that are harder to quantify than external sales, but are equally as important. In fact, high-quality internal systems can be a business differentiator. Focused IT organizations typically purchase applications for automating common business functions and build only software and systems that are unique to their business. If those unique systems are also of high quality, the business will have an edge on its competitors.

Quality is free (almost)

A study commissioned by the U.S. Department of Commerce's National Institute of Standards and Technology (NIST) found that software defects cost the U.S. economy almost $60 billion annually. The study also found that about 80 percent of development funds are consumed by software developers identifying and correcting defects.[3] In another study, The Standish Group reports that canceled software development projects cost organizations $55 billion dollars annually.[4] Clearly, poor software quality—and poor software development processes—are a major drain on business profitability.

The reasons for the high cost of poor quality vary according to the root cause. For example, poor problem domain analysis—and thus poor requirements—can lead to unplanned and costly reworking. Waiting until the eleventh hour to validate quality through testing inevitably extends both the schedule and budget. And operational downtime caused by reliability or performance problems can incur opportunity costs if customers are unable to access your system or the business is unable to do its work. Other root causes include: a poorly defined process with inconsistent staff buy-in; none or limited architectural and code QA; and limited post-deployment monitoring and assessment.

But what is the cost of high quality? Expert Philip Crosby insists that quality is free.[5] Ideally, a software development organization has these characteristics:

- *A well-defined process with unanimous buy-in from team members*
- *Team members who focus attention on value-added features, detail and quality across the life cycle*
- *Managers who constantly evaluate end-user feedback to ensure reinvigoration of the product line and satisfied, repeat customers*

[3] NIST News Release. *"Software Errors Cost U.S. Economy $59.5 Billion Annually: NIST Assesses Technical Needs of Industry to Improve Software-Testing."* http://www.nist.gov/public_affairs/releases/n02-10.htm.

[4] The Standish Group. *"CHAOS Chronicles, v3.0."* March, 2003.

[5] Crosby, Philip B. *Quality is Free.* New York, NY:New American Library 1992.

Although the intangibles that would make such an organization operate efficiently—good management, effective staffing, consistent process and so on—would require up-front investments in terms of management time, planning and training, they would not incur additional financial outlays.

One common misconception about quality is that you can compromise it to improve development speed, reduce costs or add functionality. Unfortunately, many people believe in Meskimen's tongue-in-cheek law of quality: "There's never time to do it right, but always time to do it over." In practice however, most organizations find that the opposite is true. In the long run, it is improved quality that enables teams to deliver more projects on time, at lower cost and with more features. A development team that continuously ensures quality does it right the first time. If you eliminate defects throughout the development process, you also eliminate the time and cost required to find and fix those defects later on.

Driving quality from the top down

When companies continuously ensure quality, they are embracing one of four imperatives required for software development organizations to become on demand businesses. This requires commitment and dedication that is driven from the top of the business leadership chain to the bottom.

Although your methodology for ensuring quality need not be as formal as organizational improvement methodologies such as Six Sigma or Total Quality Management (a precursor to Six Sigma), it will require the same kind of upper management leadership and organizational involvement that more formal methodologies demand.[6] Quality improvement is essentially a mindset issue; when managers from the top down instill a culture of quality throughout the organization, it permeates every project.[7]

[6] Although many believe that quality software requires an elaborate, formal development process, that is not always the case. Organizations that produce mission-critical or life-critical software systems require a high degree of formality, but in many organizations this is not a prerequisite for quality. RUP, a highly adaptable framework, can scale from highly structured/formal to loosely structured/informal environments.

[7] Jack Welch and John Byrne. *Jack: Straight from the Gut*. 2001. Jack Welch, former General Electric CEO and champion of the Six Sigma approach, emphasized that executive sponsorship was the key driver for the success of this initiative within GE.

Working within such a culture offers great benefits to managers. They no longer have to weigh the consequences of shipping products with known defects. And the rigorous processes, team responsibility and objective metrics that contribute to quality also create predictability. Instead of endlessly re-scoping the project and continually missing deadlines, teams can scope, estimate and schedule accurately, and they can comfortably commit to delivering on time and according to specifications.

Achieving quality with the right process and platform

Establishing a quality-oriented mindset is a top-level management responsibility, but it is up to the software development organization to adopt and execute on that mindset. IT must adopt the right process and tools, and they must seek the guidance or training necessary to produce the results that management, customers and end users are seeking.

Quality tools and procedures for RUP disciplines

The IBM Software Development Platform comprises software engineering best practices, integrated tools and expert professional services to help software development teams build higher quality software. This open and modular platform includes general-purpose IBM Rational®, IBM WebSphere® and IBM Tivoli® tools as well as specialized tools for IBM DB2® and IBM Lotus® software. At the platform's core is the IBM Rational Unified Process® (RUP®) which embodies software development best practices including an iterative approach, change management, visual modeling and continuous QA procedures.

The iterative phases of RUP emphasize one or more core process disciplines—analysis, design, development, test and deployment. Each phase and each discipline emphasizes attention to quality and seeks to help teams identify problems early in the development life cycle when they are easiest to solve. The following sections examine how RUP and tools in the IBM Software Development Platform support quality-oriented practices in each process discipline.

Analysis. META Group reports that up to 80 percent of the issues leading to customer dissatisfaction can be traced to poor understanding of requirements. For any software development project—whether new application development, packaged application integration or legacy modernization—quality starts with analyzing the business to ensure that system requirements clearly and accurately reflect business and customer needs.

If a customer writes a list of requirements, is that good enough? Probably not, because understanding such a list would likely require domain knowledge that can only be acquired through extensive experience. As the developers building the software do not typically work in that world, the requirements must be written in a manner that is understandable to them. Writing good requirements combines the efforts of customers, business users and the development team.

Tools for this activity include IBM Rational RequisitePro® software, an intuitive requirements management tool that enables teams to document requirements clearly (using Microsoft® Word) while leveraging the analytical capabilities of a database to perform requirements analysis, coverage and change impact.

IBM WebSphere Business Integration Modeler is a tool for the design, test and communication of complex business processes—the forerunners of project requirements. It simulates the operational efficiency of these processes and analyzes potential business results. IBM WebSphere Business Integration Monitor displays real-time information gathered from of the various production environments in which business processes allow decisive business performance management and optimization.

Design. In design, the primary quality focus is on architecture, the "soul" of software. Poor architecture can cause a wide array of quality problems including fragility, lack of scalability and resistance to modification. These problems become harder to solve as an application project matures—and the cost of correcting defects grows exponentially as applications progress from design to development, testing and deployment. If software developers can efficiently detect, isolate and resolve structural deficiencies during design and development, this work will pay dividends throughout the project. Developers should also ensure that the software is implemented in a way that preserves the architecture's integrity and flexibility so that developers can make changes quickly as business needs change.

IBM Rational Software Architect enables software architects and senior developers to create high-quality architectures for applications and services by leveraging model-driven development with UML. IBM Rational Software Modeler contains those features of Rational Software Architect dedicated to ensuring that specifications, architecture and designs are clearly defined and communicated to stakeholders through UML. Both products can help teams improve their understanding of the system and free up time to focus on quality issues during design and beyond.

Development. On average, developers make 100 to 150 errors for every thousand lines of code they write.[8] Of course, this number varies from developer to developer and project to project. Even if only a small fraction—say 10 percent—of these errors are serious, then a relatively small application of 20,000 lines of code will have roughly 200 serious coding errors.

Partially in response to statistics like these, development organizations in recent years have placed a stronger emphasis on developer-led testing and analysis. This is epitomized by agile practices that endorse test-first development—building tests before you code. Although unit testing and run-time analysis have become more mainstream, many managers still have the misconception that these procedures add unnecessary time to the schedule. In reality, schedules typically lengthen because of the time developers have to spend debugging code later in the life cycle, after QA or customers find problems. For teams that want to reduce risk and increase predictability, a well-formed, proactive QA approach by the development team is a good solution.

[8] Watts S. Humphrey. *Multiyear study by the Software Engineering Institute (SEI) at Carnegie Mellon University.*

For Java™ developers, IBM WebSphere Studio Application Developer for WebSphere Software provides comprehensive support for this approach. It combines Java, Web and enterprise development tools in a single, integrated development environment for building, testing, integrating and deploying applications. IBM Rational XDE™ Developer software also offers Java and managed Microsoft .NET software designers and developers a rich set of model-driven development and run-time analysis capabilities for building quality software applications based on the Eclipse platform and in Microsoft Visual Studio® .NET environments. IBM Rational PurifyPlus™ software has a complete set of automated, run-time analysis tools for improving the application reliability and performance of applications based on the Java, C/C++, managed .NET and Visual Basic® languages.

Testing. Conducting system-level functionality and performance tests is an integral part of continuously ensuring quality. The quality engineering (QE) team that performs these tests acts primarily as a customer advocate in verifying the application's ability to fulfill customer needs or business imperatives.

A development organization should neither overemphasize nor minimize the importance of system testing. Ensuring quality is not the sole responsibility of the QE team; nor is testing the exclusive domain of QE. Some tests can and should be run by developers and, in some cases, architects.

Unfortunately, many organizations treat QE team members as bug sweepers, turning them loose on an application late in the life cycle to find defects that other team members introduced. However, testing alone cannot create quality. If the application under test is riddled with architectural defects and bugs that should have been identified and eliminated during unit testing, then system and performance testing late in the life cycle will not be effective. Similarly, QE cannot be expected to test efficiently without automated tools. To be most effective, QE teams must be able to apply appropriate testing tools as they focus on system-level quality issues.

Testers and business analysts who want to improve the speed, breadth and reliability of their manual testing efforts should consider IBM Rational Manual Tester, a manual test authoring and execution tool. IBM Rational Functional Tester is a tool geared specifically toward QA professionals who perform functional and regression testing of Java, .NET, Web and terminal-based applications. And testers and IT operations professionals can use IBM Rational Performance Tester, a load generation tool designed to help ensure the scalability and reliability of Web-based applications before subjecting them to real-world user loads.

Deployment. At some point, business applications must go live. Inevitably, functionality and reliability errors slip through the cracks of even the most thorough processes. If the application is to meet or exceed service-level agreements (SLAs), then the IT operations team must assume responsibility for ensuring quality. Through constant monitoring and assessment, team members can ensure the viability of a deployed system and rapidly detect and correct performance problems.

IBM Tivoli monitoring products can help IT teams detect bottlenecks and potential problems, and also provide automatic recovery from critical situations. IBM Tivoli Monitoring for Transaction Performance provides performance management capabilities for Web and enterprise infrastructures. IBM Tivoli Provisioning Manager automates the tasks of provisioning and configuring servers, operating systems, middleware, applications, storage and network devices in both test lab and production environments.

Supporting team responsibilities for ensuring quality

Quality is not only the responsibility of each individual on the development team, but also the responsibility of the team as a whole. In an iterative process, each iteration is essentially circular—each discipline receives input from and provides output to another discipline. At a high level, this circularity ensures constant reassessment of each artifact's quality. Teams must do everything they can to integrate workflows, establish traceability and simplify communication. A breakdown in the chain that links team members leads to data loss, reworking, lack of clarity, inefficiency—and ultimately lower quality software.

The IBM Rational Team Unifying Platform™ is an integrated suite of infrastructure tools and processes that unifies development teams by providing common access to development assets, communication alerts and workflow processes. In addition to RUP and Rational RequisitePro, this platform includes integrated tools for software configuration management, change management, test management and reporting that enable requirements traceability throughout development, analysis and testing. The software configuration management solution integrates software asset management from the IBM Rational ClearCase® family of products with defect and change tracking from the IBM Rational ClearQuest® family. Effective software configuration management is an essential tool for ensuring quality; it helps organizations ensure that software builds are repeatable and reliable, and that defects and change requests are managed properly.

The IBM Rational Team Unifying Platform also includes IBM Rational TestManager, which provides control, management and reporting of all test activities from a central point. IBM Rational ProjectConsole™ enables managers to keep close tabs on quality control; it dynamically generates a project Web site and metrics dashboard with data automatically collected from the development environment. And IBM Rational SoDA® software automatically generates and maintains comprehensive project documentation and reports.

IBM also offers a wide range of services to help development teams deliver quality software. Team members can access both instructor-led and Web-based training courses that focus on tool use and skill development via the IBM developerWorks® Web portal. Professional services consultants can work with development teams to create custom quality implementation plans that span initial project assessment, installation, mentoring, training and maintenance.

Reaping the extensive benefits of better software quality

When people think about what it takes to build mission-critical and safety-critical software, they often imagine that it requires a cumbersome process with oversight, standards, formality and documentation. In truth, there are many ways to achieve quality without a heavy duty process such as Total Quality Management or Six Sigma.

Quality improvement, like software development, is an iterative process. You do not need to accomplish everything in a single step. Even small changes—including adjustments in your organization's attitude toward quality—can make a tangible difference. Although you do not need a burdensome process to achieve quality, you do need an organization mindset that focuses on quality, and that mindset must be driven by top-tier management.

The business benefits of including quality-oriented activities in all phases of your software development life cycle are both broad and deep. These measures not only facilitate innovation and lower costs by increasing predictability, reducing risk and eliminating rework, but they can also help differentiate your business from its competitors. Most important, continuously ensuring quality will always cost less than ignoring quality considerations. In fact, raising product quality costs next to nothing if you do it correctly.

For more than twenty years, IBM and IBM Rational have been helping entire project teams understand how to proactively ensure quality. Relying exclusively on a testing tool to find bugs at the end of the software development life cycle is a costly, inefficient approach that results in low-quality products. To raise the quality level, you need a predictable, quality-conscious process that involves every team member and gets to the root causes of problems early on.

In this article, we have described only a portion of the IBM Software Development Platform's rich assortment of tools, processes and services that can help teams produce higher quality software. Like its customers, IBM is constantly striving to improve its products, develop new solutions and help its clients succeed.

G507-1007-01