

## Driving Quality Throughout the Software Delivery Lifecycle

The benefits of Lifecycle Quality Management

## Contents

Executive summary . . . . .	.3
The Impact of Poor Quality . . . . .	.4
The Lifecycle Quality Approach . . . . .	.4
Overcoming Key Obstacles to Lifecycle Quality . . . . .	.5
Strategies for Better Quality Management . . . . .	.7
Plan (Define and Prioritize) . . . . .	.7
Verify and Validate . . . . .	.8
Improve (Analyze and Tune) . . . . .	.9
Manage . . . . .	.10
Borland® Lifecycle Quality Management . . . . .	.10

## Executive summary

Businesses don't wait until a product is almost available to create a strategy for how it will drive revenue. Why then do they wait until the very last moment to ensure the quality of software?

The world's reliance on software to drive businesses of all sizes has placed tremendous challenges on development organizations in almost every industry. These teams face continuous pressure to deliver software faster at less overall cost to their businesses. The growing complexity of their work environments compounds this pressure. To deliver more projects on time, organizations have continually treated software quality as an afterthought, leading to poor adoption, lost revenue or worse, project failure and abandonment.

Software quality can be described as the convergence of complete requirements, correct code and minimized defects that align to meet business goals. It applies to a micro or macro view of projects in agile or traditional development environments and also to the widely varied priorities of different organizations. There will always be a trade-off between time-to-market and high quality, but optimizing a process that aligns all critical phases of the software delivery lifecycle and prioritizes quality throughout is key to consistent project success.

To achieve higher quality software that meets business needs, a successful approach has emerged called Lifecycle Quality Management, which helps organizations infuse quality throughout the entire software development lifecycle. This approach enables teams to consistently deliver high-quality applications and services that meet business requirements, while systematically reducing costs, risk, defects, rework and time-to-market. Lifecycle Quality Management encompasses four core quality processes—plan, verify and validate, improve, and manage. The approach bolsters development team confidence by combining people, process and technology aspects to ensure that quality is built into software from the point of project definition through delivery.

This paper examines Lifecycle Quality Management, including its core processes and best practices. It also introduces Borland® Lifecycle Quality Management (LQM), the first proactive offering to infuse quality throughout the software delivery lifecycle. By integrating proven process improvement expertise and skills training services with Borland's Application Lifecycle Management (ALM) technologies, the offering breaks down the barriers between development and testing. As a result, companies mature their quality processes without increasing the burden on already strapped teams. For software organizations striving to improve quality, Borland LQM helps transform teams from being reactive testers at the end of the lifecycle to being a proactive quality organization that more consistently delivers high-quality applications that meet the performance, security and functional needs of the business.

## The Impact of Poor Quality

Quality is critical to every business. The absence of quality in any product or service leads to dissatisfied customers. That in turn, results in lower acceptance and adoption for new products and services from the same supplier, which directly impacts revenue.

Today's development organizations face continuous pressure to deliver software faster, at lower costs with fewer resources. The growing complexity of their work environments—including the use of highly iterative agile development, distributed teams and new technologies—compounds this pressure. In the rush to deliver more projects on time, software quality has continually been treated as an afterthought.

With software driving so many of today's critical business processes, the need to infuse quality throughout the software lifecycle is more urgent than ever. Yet, software failures remain high. Within corporate software organizations, only one in three software projects is considered a success, and approximately 70 percent of software projects started fail to deliver what originally was intended without going over budget, missing the deadline, or sacrificing quality<sup>1</sup>.

Despite the many known failures, ensuring software quality is still primarily the responsibility of a single department, a testing organization known as Independent Test, Quality Assurance (QA), Quality Control (QC) or simply Test. In these departments, efforts to verify quality typically begin when all code has been completed, or "frozen". Test plans are built, test cases are written or automated, manual or automated testing is run, and if time and resources allow, performance and security testing is completed. Yet all of these tests take place under enormous pressure to certify the application as quickly as possible in order for the business to meet customer and market commitments. As delivery cycles compress, this process of testing only immediately prior to deployment becomes less effective and prone to error. Additionally, though code is considered frozen, very often new features are added at the last minute or requirements are modified without the knowledge or the QA teams. This adds to the testing workload at the end of the delivery cycle, often without any room for deadline adjustments.

When organizations consider quality as an afterthought they are in effect increasing their costs and decreasing their efficiencies. Figure 1<sup>2</sup> illustrates how significantly the estimated cost per defect can increase in the latter stages of the software development lifecycle. Other experts have estimated the cost of finding and correcting a defect found in operation to be more than ten times the cost of defects found during system test.<sup>3</sup> More effective quality processes are proving to reduce business costs and increase efficiencies, enabling organizations to see the value of moving quality activities earlier in the lifecycle.

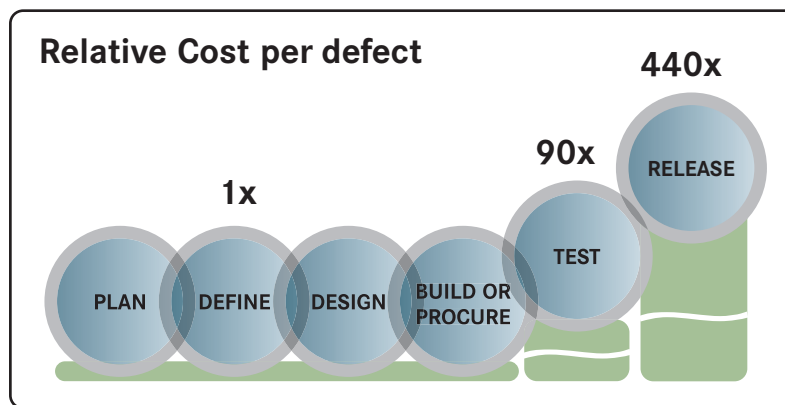


Figure 1: The cost of finding a defect significantly increases in the latter stages of the software delivery lifecycle.

## The Lifecycle Quality Approach

Industry analysts have long advocated a more comprehensive approach to software quality. Gartner recently wrote "quality is not a discrete or isolated function, it is an ongoing and cumulative effort."<sup>4</sup> Similarly in a recent presentation, Forrester reminded organizations that "software quality is everybody's business."<sup>5</sup>

Although most organizations have testing teams and processes in place at the end of a project, a more successful approach called Lifecycle Quality Management has emerged to help organizations infuse quality throughout the entire software devel-

1 The CHAOS Report, The Standish Group International, Inc., 2004.

2 National Institute of Standards and Technology, "The Economic Impacts of Inadequate Infrastructure for Software Testing." Gaithersburg, Maryland, 2002.

3 W.S. Humphrey, "A Discipline for Software Engineering." Addison-Wesley Publishing Company, Massachusetts, 1995.

opment lifecycle. A comprehensive and proactive approach to quality, Lifecycle Quality Management enables software development organizations to deliver higher quality applications and service, while systematically reducing costs, risk, defects, rework and time to market.

As Figure 2 shows, substantial involvement from all members of the software development organization ensures that software quality activities are part of every stage in the software lifecycle. Moving beyond just a focus on testing once code becomes “finished,” Lifecycle Quality Management combines skilled people, using appropriate processes and effective technology in each phase of the lifecycle, from project inception through application delivery. This shift to the complete lifecycle builds quality into the product, reducing overall development costs while addressing quality issues at their root causes.

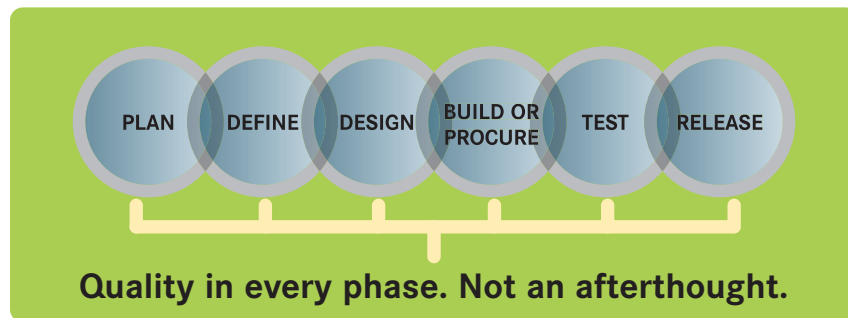


Figure 2: When quality is infused throughout the software lifecycle, organizations deliver higher quality applications and service, while systematically reducing costs, risk and time to market.

## Overcoming Key Obstacles to Lifecycle Quality

Lifecycle Quality Management requires the alignment of effective processes, skills training and the right technology to plan and manage quality activities from project definition through release into production. It also requires a commitment from the organization to be diligent about quality at every stage. Those teams that are able to overcome key obstacles and effectively institute lifecycle quality can ensure they consistently deliver high quality products, while those unable or unwilling will continue to struggle with a never-ending and costly purely-reactive approach. The following recommendations remove common obstacles to achieving lifecycle quality:

### Emphasize total quality, not just testing.

Quality itself is often hard to define because it means different things to different people, but in the context of software quality assurance, the simple and appropriate definition of software quality is the convergence of complete requirements, correct code and minimized defects that align to meet business goals.

When an organization emphasizes only testing, it typically focuses on product quality in terms of defects, narrowly defining defects as functional, performance, security and usability bugs that are to be tracked and reworked in time to meet schedule commitments. This narrow view of quality eliminates opportunities for early problem detection throughout the software development lifecycle, as well as activities that can ensure an application (and its source code) is complete, optimized, scalable, adaptable and maintainable.

4 Theresa Lanowitz, 2004. "Software Quality in a Global Environment: Delivering Business Value." The Gartner Application Development Conference: The Path to Modern AD, Phoenix, Arizona.

5 Carey Schwaber, 2006. "Software Lifecycle Quality: Adopting a Proactive Approach," A Forrester Research eSeminar Presentation.

Organizations that focus on total quality have a comprehensive definition of quality. This view enables them to address and prevent defects—all of which have real associated costs—such as the following, which cannot be detected by testing:

- Ambiguous requirements which can result in expensive rework
- Code licensing violations which can increase legal risk
- Poor design which can limit performance and scalability
- Tightly coupled interfaces which can impact ease of integration
- Use of platform-specific symbols or methods which can limit portability
- Hard-coded strings or single-byte functions which can limit localization
- Poor code readability which can increase maintenance costs
- Security vulnerabilities which can expose customers to attacks

Unlike testing-focused organizations, quality-focused organizations incorporate quality-focused tasks throughout the entire software development lifecycle. For example, they focus a great deal of time and effort early in the software lifecycle on requirements. Quality-focused organizations ensure requirements are accurately defined, explicitly documented and testable because software requirements drive software design, development, implementation and testing. Additionally because requirements change frequently, they are typically electronically captured and managed to ensure they are continually current. Concurrently, test team members link test plans and test cases with the stored requirements to ensure the final software deliverable will be tested against up-to-date customer requirements. During development, the teams designing and developing the system are also analyzing the quality of their units of work (e.g., through static analysis, simulation, unit and component test, etc.). As a result, quality-focused organizations gain greater confidence that what is delivered in the end product is accurate.

### **Be proactive, not reactive.**

Reactive approaches to software quality (testing only at the end of the cycle) provide no checks and balances as software moves through the development lifecycle.

As a result, business and IT goals can easily become misaligned, and software is delivered that does not meet customers' expectations.

Proactive quality management means that the entire organization—not just a final testing team—is thinking about quality as a requirement for all projects. Proactive organizations introduce and enforce quality standards earlier in the development cycle, without slowing down the process or hindering developer productivity. For example, they detect potential problems before they have a chance to impact other developers by reviewing the requirements against standard criteria, simulating difficult pieces of the design, conducting peer reviews of significant work products, and automatically pre-screening new code against a set of quality guidelines before it enters the build process. This ensures that each down-stream team gets useful input to maximize team productivity.

Organizations that practice proactive quality management also provide visibility into the overall health, risk and progress of projects. They provide managers with a dashboard that tracks key indicators such as project status, levels of detected and removed defects and their exact location in all types of work products, as well as review and test coverage. They also provide other local measures that predict whether a project will be delivered on time, on budget and to specification or simply if it is safe to do last minutes fixes or updates without risk.

Time and cost considerations prevent many organizations from investing in significant amounts of testing. By taking a more proactive, lifecycle approach to quality management, organizations more carefully examine quality throughout the development process. As a result, they are able to remove defects earlier and ultimately spend less time on rework and testing.

### Drive continuous process improvement.

Automation alone cannot ensure software quality; there must be skilled people using effective processes. When testers discover a software bug, they often communicate that information using a defect management system. Although that technology may be effective in tracking that bug until it is fixed, the greater gain comes from reporting the cause of the defect.

When the organization can analyze the causes and fix those through training of people with inadequate skill or by improving their processes, it not only ensures better quality for this project, but enables ongoing improvement. For example, if an issue tracking or change management system can also provide summary reports showing that 40% of the defects are coming from errors in a design process, the organization can confidently address the improvement of that process.

To support the development team as it proceeds through the software lifecycle, organizations must put in place collaborative process, workflow and infrastructure. These include a process to manage incoming requests; a process to manage project and portfolio management; a process to validate, close and signoff on change requests; and a process and infrastructure to manage and control all development artifacts.

## Strategies for Better Quality Management

A variety of practices can help organizations iteratively infuse quality into each stage of the software development lifecycle. Using these practices, they can improve teams' understanding and communications around software quality and its impact on the organization. Figure 3 illustrates, and the process areas below describe, best practices in Lifecycle Quality Management.

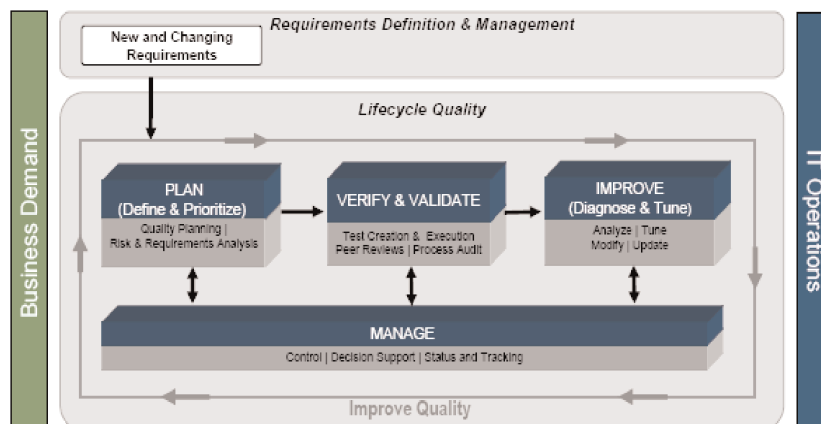


Figure 3: Organizations must address the four key process areas within Lifecycle Quality.

### Plan (Define and Prioritize)

#### Analyze and classify requirements.

High-quality requirements are the starting point for all Lifecycle Quality Management activities as they align all contributors to clearly-defined, common goals. High-quality requirements should be delivered and prioritized based on business needs. They

also should include details about testability and quality measures, for example if there is a need to support a certain response time, that time should be explicitly documented. Analyzing and classifying high-quality requirements must involve both IT and business stakeholders to balance customer need against business costs and risks.

### **Perform risk analysis and prioritize quality activities.**

Risk is inherent in any software development project. Those who analyze the risk and develop their project plans and quality plans in light of those risks have the potential to minimize their impact. An effective quality planning context includes:

- Product-related risks (coping with legacy systems, dealing with leading-edge technologies, coping with competition, ...)
- Project-related risks (level of resources, schedule constraints, financial considerations, budgets, ...)

For example, is there sufficient time to provide critical test coverage with fully manual testing, leaving adequate time for necessary rework? Or, will test automation reduce risks there, providing benefits of consistency, speed and repeatability? Is it better to conduct in-depth code reviews for the team just learning to develop in a new language instead of devoting a lot of time to unit test? Will revenue be impacted if this product is delayed by one week because of newly discovered defects? With the risks in mind, proper quality planning incorporates appropriate use of different quality activities.

To maximize the return on quality initiatives, teams must prioritize quality activities according to business requirements and risk. For example, which aspects of the software or business transactions are absolutely mission critical and thus must be tested carefully, and how do you prioritize them in order of criticality? Effective communication processes and systems are imperative to prioritizing quality activities.

### **Define and build quality plan.**

Without a map, it is hard to successfully navigate new areas. To guide their efforts, most software development organizations have detailed policies and procedures that describe their software development lifecycle (SDLC). An effective SDLC should include the development of quality plans including roles and responsibilities, resource allocation, timelines, milestones, required reports, etc., so that quality becomes a requirement in every software project from definition to delivery. Organizations should be careful to define application quality goals with agreed-upon criteria and measures that meet the application business needs.

## **Verify and Validate**

Once the quality plan is documented and approved, the project team can begin to perform the quality activities, measuring application quality status and quality progress using various methods:

### **Process audits.**

Defined processes enable teams to function efficiently and effectively, since they can see what is required of them in their project role. At a project's start, a team selects and tailors (if needed) the organization's processes to meet its project and product goals. Process audits help to ensure that software development teams follow the selected processes and procedures. Throughout the project, process adjustments can be made, as issues from process audits and defects from other quality activities are analyzed, to determine causes for the problems. By treating the process proactively, the project team is able to use data to its advantage in creating quality results.

**Peer reviews.**

Two heads are better than one when it comes to addressing most challenges. The same is true when it comes to ensuring software quality. Best practices for software quality include peer reviews of all types of work products (requirements, design, code, tests) to ensure they conform to customer requirements and organization standards. When reviewing code, for example, team members ensure that they are following coding standards and that potential defects or inefficiencies are identified before they need to be surfaced in more time-consuming testing.

**Work product analysis.**

For many software products, aspects of the architecture need to be examined and design choices made before much of the system can be coded. Proactive quality activities might include architecture modeling to examine alternate design approaches. Certain technologies or platforms, for example, may appear to be attractive until detailed modeling shows that they will not scale to meet true customer needs. This type of early analysis—using static models or simulations—can provide the assurance that a design can confidently be committed to code, knowing it will meet performance requirements.

**Create and execute test cases and suites.**

Tests measure the variance between observed and expected software behavior. Thus, testing is an effective means for identifying quality issues once code is complete, whether that issue is a functional defect, a missed service-level objective, a performance bottleneck, a usability issue or a security leak. Test execution can be manual or automated. Test automation provides the most benefit when used for repetitive testing tasks (e.g., regression, cross-platform or localization testing) or when manual testing is impractical due to repeatability and scalability (e.g., for load and performance testing). As the cost of removing a software defect grows exponentially for each downstream phase of the development lifecycle in which it remains undiscovered<sup>6</sup>, it is key to start testing early in the development phase—long before traditional end of cycle testers can get their hands on the running code.

## Improve (Analyze and Tune)

**Analyze results of process audits, reviews and tests.**

With every release, organizations can improve the quality of their software projects by analyzing the results of verification and validation activities and comparing them with their expected outcomes as defined in the quality plan and in specific requirements. Organizations seeking to improve overall quality may benefit from an automated requirements definition and management system that tracks software requirements and changes made to them throughout the delivery lifecycle.

**Diagnose and pinpoint issues.**

In the short term, analysis of defects to determine their root causes may be perceived as time-consuming and costly, but it can save significant resources in the long run. Once the root cause is known, the skill or process or technology issue can be addressed for the benefit of both the current project and for future ones. Similarly, quick and accurate problem diagnosis is important not only to accelerate problem resolution, but also to avoid any finger pointing between team members.

**Update work products and re-verify.**

The organization's understanding of the defects detected by reviews or testing is only half of the battle. The requirements, design, code or test need to be repaired, and then reviewed or tested again, to ensure that the repair correctly handled the defect. If the completed software fails a test, the gathered diagnostic information helps the responsible team members to resolve the issue by changing the application's source code or tuning its configuration. To ensure the high-quality requirements are finally

---

<sup>6</sup> Barry Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

met, a full re-test should be completed. To ensure defects have been correctly handled in a peer-reviewed document, the repaired document may need to be reviewed again.

## Manage

### **Status tracking and reporting.**

Managers must have the necessary lifecycle quality information in hand at each stage of the SDLC to make the right decisions. Organizations must be able to obtain real-time reports on quality status and project progress, including information about the results of reviews and testing, coverage, and defect find-fix rates to enable efficient resource management and further understand release readiness.

### **Control all phases of quality management from plan to improve.**

As software travels through the delivery lifecycle, organizations need support for all of their quality processes, yet software development organizations have limited time and resources. To minimize costs and time to market, all resources need to be managed efficiently. Control systems enable organizations to effectively manage activities and assets that drive quality results. Such systems may include change management for test assets (plans, scripts, and data) and the target applications; resource management for staff and the quality/test infrastructure (e.g., which machines run which tests); build/release management; traceability visualization of tests to requirements, design models and source code; and a repository of all quality records.

## Borland Lifecycle Quality Management

Borland is the leading vendor of Open Application Lifecycle Management (ALM) solutions - open to customers' processes, tools and platforms - providing the flexibility to manage, measure and improve the software delivery process. Borland offers integrated Open ALM products, services and training to help customers transform software delivery into a managed business process. One of the core development processes critical to achieving this is Lifecycle Quality Management.

Borland Lifecycle Quality Management (LQM) enables companies to effectively align business priorities and quality expectations with project requirements, development activities and software testing to support business goals such as faster time-to-market, improved customer satisfaction and increased competitive advantage. By eliminating the risks inherent in traditional quality management programs, Borland LQM helps organizations minimize costs and maximize the potential for application success by bringing discipline and structure to the overall quality assurance process.

Until now, the lack of effective processes, skills training and technology have thwarted many organizations' efforts to iteratively improve quality. However, Borland's approach to LQM is unique in that it infuses quality throughout the software lifecycle by taking into account an organization's process maturity and leveraging industry best practices to evaluate current performance and identify specific areas for improvement. Borland provides all of the components necessary to craft a complete LQM solution that encompasses an organization's business needs, development capacity and ultimately user acceptance.

Borland's effective LQM solution empowers software teams to:

### **Mitigate Risk**

- Make informed, traceable, validated Go/No Go decisions
- Identify defects earlier in the development lifecycle, as well as during the intensive testing phases
- Know that performance and scalability are up to production standards
- Validate that users are receiving business value, as expected
- Understand the impact of requirements changes on development and test activities

### **Reduce Costs and Time to Market**

- Eliminate defects in requirements – the highest source of software problems
- Capture defects on the developers desktop before testing teams ever see them
- Reduce and eliminate broken builds that often bring testing to a standstill

### **Capitalize on the efficiency, accuracy and time-savings of test automation**

- Increase Efficiency, Profitability, Competitiveness
- Support manual and automated testing with process, efficiency and reporting
- Manage heterogeneous testing environments with a single quality platform
- Gain visibility into the testing process and project readiness
- Deliver applications faster than competitors
- Deliver applications with higher quality and better performance than competitors
- Meet business needs with confidence that functionality and performance meet user expectations
- Provide traceability to defects, requirements and tests for compliance needs

Borland enables its customers to deploy an LQM solution via Borland Accelerate, a proven four-phase improvement approach for customer success. The flexible Accelerate framework, illustrated in Figure 4, allows Borland to tailor the solution to an organization's unique needs. It leverages:

- **Best-in-Class Processes** backed by assessment tools, a knowledgebase of process models, implementation toolkits, project management techniques and a repository of best practices.
- **Technology** to efficiently automate and enforce those processes, including Borland's award-winning Silk product line.
- **Skills Training** to ensure successful adoption of the processes and technology.



Figure 4: Borland Accelerate is a proven four-phase improvement approach

Using the proven Accelerate framework phases as a guide, Borland helps organizations seeking to establish an effective LQM solution for their organization:

**Phase I:** Borland experts help executives define goals and measurable criteria for implementing their LQM approach (for example, to reduced delivered defects by 20%)

**Phase II:** Borland experts help organizations architect and plan their Lifecycle Quality approach using workshops that prioritize new and changed processes and practices, as well as build skills development plans.

**Phase III:** Borland helps organizations develop and deploy an appropriate Lifecycle Quality solution that incorporates learning paths, workshops to help define processes, and technology implementation guidance.

**Phase IV:** Borland helps organizations validate their Lifecycle Quality efforts using the goals and measures set in Phase I

Borland supports organizations developing or evolving their quality initiatives, enabling them to transform software delivery into a managed business process. If improving software quality is a priority for your organization, LQM provides a proactive approach to achieving your goal. To learn more about Lifecycle Quality Management visit [www.borland.com/qualityresource](http://www.borland.com/qualityresource).

Borland is the leading vendor of Open Application Lifecycle Management (ALM) solutions - open to customers' processes, tools and platforms – providing the flexibility to manage, measure and improve the software delivery process.