Software quality management solutions
Thought leadership paper
March 2009

Rational. software

IBM

Software quality optimization: balancing business transformation and risk

Michael Lundblad, program manager, Rational software, IBM Software Group

Moshe Cohen, offering manager, Rational software, IBM Software Group

Contents

2 Introduction
4 Balancing business transformation and risk
11 Fulfilling the vision
13 Quality management solutions
16 Summary

Introduction

A recent IBM CEO study shows that 66 percent of CEOs expect their organizations to be inundated with change, largely driven by innovation and transformation. To keep pace, software development organizations need to release business-critical software in less time, but this increases risk and often results in compromised quality. *So the question becomes, How can you save time and reduce costs without sacrificing quality?*

Although aviation is more than 100 years old, most of us are still amazed by the sight of an airplane taking off and landing. What's even more incredible is how autopilot software controls and performs the whole thing with little pilot intervention. Consider the dexterity of the Joint Strike Fighter (JSF), also known as the F-35. Pointing upward, it can stop in midair, hold its position and then resume its supersonic flight. This maneuvering too is controlled with software.

What about cars? Today's vehicles contain anywhere from 100 to 150 computers. They're essentially networks of computers on wheels, continuously communicating with one another as well as with global technology that provides services such as telemetry, global positioning, in-vehicle security and system diagnostics.

Whether we realize it or not, we interact with massive software systems every day.

Every day we interact directly and indirectly with massive software systems. The inventory in our favorite supermarkets is controlled with software; our checking and savings accounts are controlled with software; our mortgages, utility bills, medical history and eligibility for prescription drugs are all controlled with software.

These advances bring convenience and progress, but they also bring risk and attendant expenses. Paul Ehrlich once said, "To err is human, but to really foul things up you need a computer." Consider these software-related issues, which were reported by devtopics.com as among the 20 worst such disasters.[1]

Highlights

On "Black Monday" (October 19, 1987), the Dow Jones Industrial Average plummeted 508 points, losing 22.6 percent of its total value. The S&P 500 dropped 20.4 percent. The greatest single-day loss Wall Street had ever suffered was caused by a software defect. As investors fled stocks in a mass exodus, computer trading programs generated a flood of sell orders, crashing systems and leaving investors effectively blind and unable to trade.

In December 2008, a leading Israeli newspaper (*Yediot Acharonot*) printed an edition with the headline, "The bug that drives hospitals insane." The article wasn't about a bacterium or virus making people sick; it was a software bug that caused blood test results to be associated with the wrong patients. Israel's health ministry office said that the defect appeared when test results were electronically transferred from labs to hospitals.

Quality software delivery has never been more critical to businesses; in fact, it's critical to business survival. In today's economy, CIOs need to delicately balance the drive toward business transformation with the management of business risk. Requirements change often, projects fall behind schedule, costs are scrutinized—but quality must improve or businesses fold. This seemingly impossible task is forcing software delivery teams to think outside the box, examine agile development approaches and challenge historically dominant methods. They're shifting their view of quality management, seeing it as an ongoing effort that demands collaborative teaming, automation techniques, and accurate, realtime reporting or dashboarding of metrics to facilitate governed discussion analysis.

In a business environment where high quality must be realized on little funding, software quality management is critical.

This paper examines the balance between transformation and risk. It looks at the reasons for software development's changing landscape, and it discusses which solutions can help your business improve quality management to overachieve in time to market, cost reductions and enhanced product quality.

Highlights

Of the 20 percent of an IT budget that's left over after operations expenses, three quarters is spent finding and fixing defects.

Balancing business transformation and risk

Software quality requires teams to find a balance between three dimensions: scope (requirements), cost and time (figure 1). Depending on which survey you read, roughly 80 percent of an organization's IT budget is spent on operations. Of the remaining 20 percent, 70 to 80 percent is spent finding and fixing defects in legacy applications. Consequently, resource funding for software quality delivery is fixed, at best.

IBM's direct experience with organizations worldwide and data gathered by researchers working with hundreds of companies show that most firms invest 25 percent or more of their development lifecycle time and cost in quality assurance. Furthermore, 30 percent of software development project costs are associated with rework, and 70 percent of that amount is related to requirements errors.[2] With present-day tooling and waterfall development processes, software quality requires more time and cost than ever. To remain competitive, businesses need to find ways to improve quality while shortening time to market for business-critical software.

Maintaining a balance in "the iron triangle" is critical for companies that want to pursue innovation without increased risk.

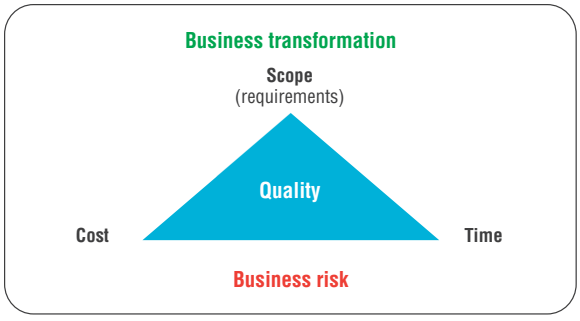

Figure 1: "The iron triangle"—managing the balance between business transformation and risk.

Highlights

When requirements are poorly defined or managed—or when they change without accurate tracking—serious defects can be introduced.

Software updates provide another fertile opportunity for defects to invade code.

Requirements are never frozen … they simply keep changing

Many of the most serious defects find their way into software through requirements: vague requirements definition, poor requirements management or incomplete dissemination of requirements to all stakeholders. And these issues don't just arise at the beginning of a project; they arise throughout the whole application lifecycle. Wishful thinking among senior-level executives would have requirements formed and then frozen so that no changes could be made to them in the future. Certainly this would help increase software quality, but today's reality is such that requirements keep changing. They reflect today's speed of business, the urgency of staying competitive and the need to comply with the regulations imposed on many of our processes.

Let's consider a case where a CIO realizes that new regulations will require that he change a number of applications. First the CIO and his team need to figure out how these changes will affect the company's software, and then they need to adjust their testing activities to address these changes. Meanwhile, the extent of the changes, the time it takes to manually test the changes, and the cost associated with making the changes convince companies that increased automation, regression testing and iterative processes like agile are the best way to manage these inevitable side effects of software updates.

Analyzing the impact of changing requirements in a geographically distributed world

Let's continue with our regulatory change example. Most companies would deal with such changes in the most natural and straightforward way: They'd have each of the teams analyze the impacts to their applications and then act accordingly. In this scenario, we'd probably end up with different teams simultaneously doing similar activities on the same products. But it's common for businesses to have multiple projects going on at once in different locations, in-house or outsourced, and all of them need to be updated to reflect the regulatory changes.

Highlights

Collaboration is essential in defending against possible defects because it allows all team members to communicate during requirements definition and software updates.

One can't help but assume that this approach creates a great deal of waste as well as increased risk. What if different teams interpret the new requirements differently? And what if they end up with test cases that reflect different behaviors when they were supposed to be identical? Dealing with multiple projects without significantly increasing the risk requires a testing process where a single set of requirements can be mapped to multiple projects. It requires the ability to define test cases from which test scripts can be generated and/or derived. It requires the ability to share and reuse test cases among projects. It requires a test plan that, like the requirements, is not a frozen document or a hierarchical Web page. It's a live, dynamic, always-up-to-date asset that allows for the definition and sharing of test processes and strategies, business-level reporting against quality objectives, and—just as important—support for collaborative team activities such as reviews and approvals.

Risks associated with execution environments

When an application has numerous configurations, it's tempting to skip testing some of them, but doing so increasing the risk of delivering contaminated software.

Today's software is so complex that in many cases it's simply impossible to guarantee error-free applications. In other cases, even if it is possible, it's not financially practical. Take, for example, a case where a customer-facing portal needs to be tested. The first step is for the testing team to develop the test cases, some of which are manual while others are automated. Depending on the complexity of the application and its intended environment, a test cycle can take anywhere from days to weeks or even months. In fact, let's assume that the portal supports only two languages and four different Web browsers or versions, and it runs on top of three databases and one application server. These few components alone still require that the test cycle be repeated 24 times (2 x 4 x 3 x 1). So what takes one hour to test, now takes 24. What takes a 40-hour week, now takes 24 weeks, almost half a year. As the number of configurations that need to be tested grows, it is tempting to eliminate testing some configurations, thus increasing the risk of delivering defective software.

Highlights

IBM has discovered that testing a small subset of configuration combinations can help businesses detect a very high percentage of defects very quickly.

IBM researched a promising way to significantly reduce the risk associated with a large number of execution environments. The results showed that there is a small subset of combinations that can detect a very high percentage of defects very quickly. Essentially, it speeds up the detection of defects. As shown in figure 2, testing under multiple execution environments follows the typical "slow" curve, where defects are detected with no special priority. The defect detection over time is mostly in the random zone, and schedules are compromised. However, applying smart optimizations brings the defect detection over time into the optimized zone, where defects are being detected much earlier in the testing cycle, saving time and significantly reducing the risk of delivering contaminated software.

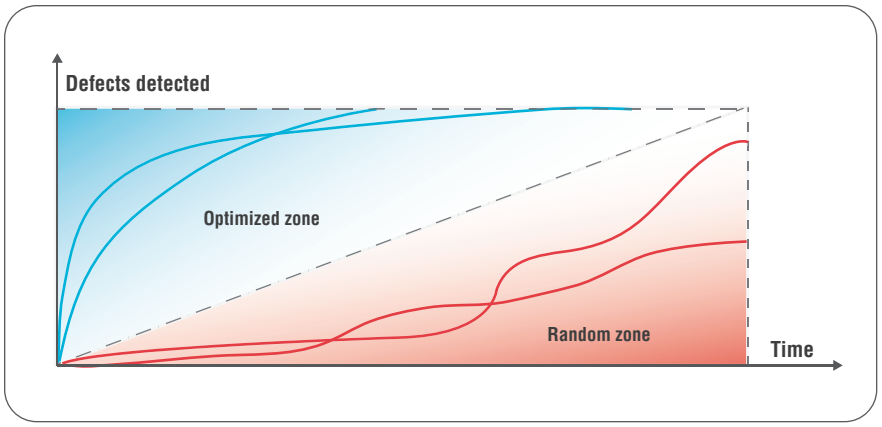

Figure 2: Speeding up the discovery of defects through smart optimizations

Highlights

Incorporating quality management earlier in the software development lifecycle can help organizations avoid the high cost of fixing bugs late in the game.

Emerging paradigm shift in quality management

To meet this quality software delivery challenge, organizations must engage in collaborative processes that are powered by automation and accompanied by actionable project governance throughout the delivery lifecycle. Traditional quality assurance (QA) testing simply validates that the software in development meets end-user expectations for functionality, availability and performance prior to deployment. Finding defects in the QA testing phase is far more expensive and time consuming than finding defects earlier in the application development lifecycle (figure 3).

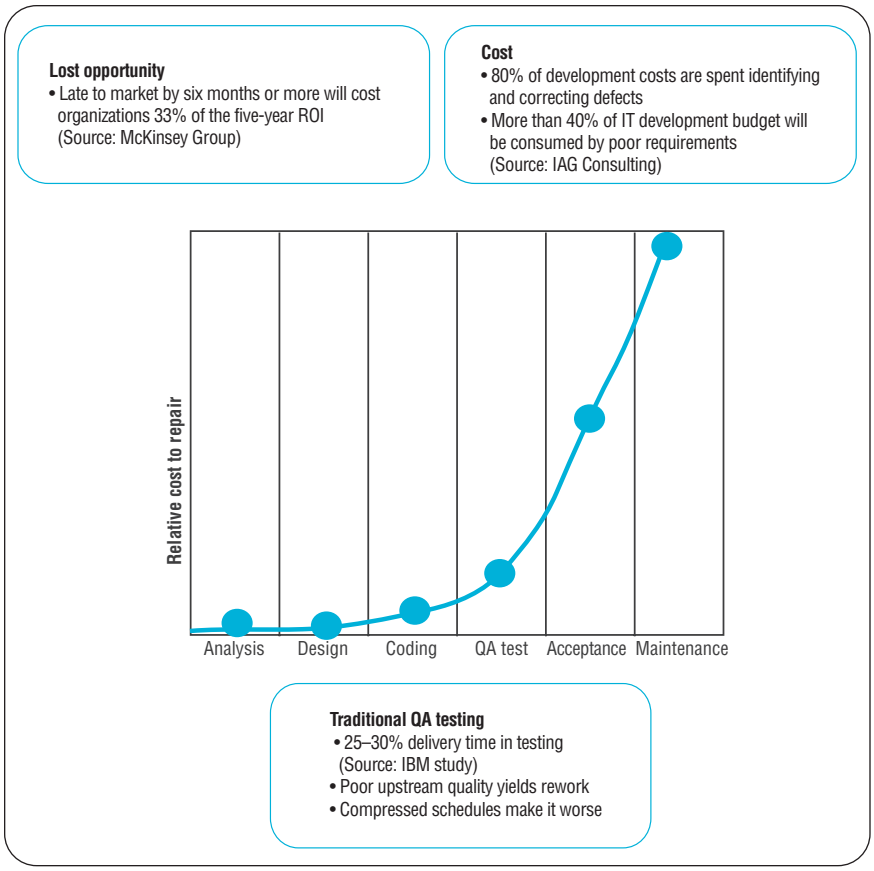

Figure 3: Opportunities to optimize quality and time to market exist throughout the application development lifecycle.

Highlights

Collaboration faces three main barriers: team dispersion across geographies, incompatible infrastructures and misalignment of staff capabilities.

Quality management has become a team sport, but collaboration faces many barriers. *Geographic barriers* block team communication across multiple time zones, and they create process gaps that result in rework. *Infrastructure barriers* create process, tooling and data integration problems that prevent teams from receiving timely information, which results in delayed schedules. Finally, *organizational barriers* introduce problems with domain expertise, weak project governance and outsourcing dilemmas like the security of intellectual property.

How are best-of-breed firms improving software quality? An IBM Global Services review[3] of 846 projects across multiple clients revealed significant differentiated value with collaborative and automated approaches. Comprehensive testing processes, integrated end-to-end lifecycle technologies, industry-based test-case and script reuse, and advanced defect analysis and quality management processes have shown the following improvement ranges:

- *Quality improvements from 30 to 70 percent*
- *Cycle-time improvements of 20 to 50 percent*
- *Cost reductions of 15 to 60 percent*

	Developing repeatable industry test solutions			Advanced defect analysis	Developing repeatable test procedures applicable to future projects		Integrating end-to-end processes		
Asset	Test cases copied	Manual scripts copied	Manual scripts reuse	Prevent and block duplicate defects	Baseline and migrate documentation	Baseline artifacts	Leveraging component reuse	Dynamic updates of test assets	Total
Quantity	343	350	1,393	905	1,365	2,023	1,029	2,227	9,635
Hours saved	167	175	696	1,755*	683	1,011	515	557	5,558
Value	$16,690	$17,514	$69,633	$175,452	$68,254	$101,125	$51,459	$55,673	
Total			$103,387	$175,452		$169,379		$107,132	$555,799

* Hours saved assumes an average of four hours to detect the duplication. In reality, it often takes much longer.

Table 1: Differentiated business value results from an automated, process-led, collaborative approach to quality management.

Table 1 summarizes some of the findings from this study, which shows how 846 of these clients' projects saved on average over half a million dollars per project.

Where did these organizations find savings? In their quality management processes.

Organizations can save time and money by paying more attention to quality management across the entire software development and delivery lifecycle.

- ***Develop repeatable industry test solutions.*** *It seems reasonable that within any given industry, test cases and manual scripts become fairly similar. That means copying and reusing those common test cases and scripts could net a substantial savings. In this study, the average savings per project was over US$100,000 (16,690 + 17,514 + 69,633).*
- ***Perform advanced defect analysis.*** *It's not easy to detect and prevent duplicate defects, but it's very important to detect them early. If you don't, you run the risk of having multiple teams working on the same defect without knowing it, often producing even more defects. Automated detection of duplicated defects not only improves quality, but also reduces risk and cost. In this study, assuming an average of four hours to detect the duplication, the average savings per project was US$175,452.*

Highlights

- ***Develop repeatable test procedures that are applicable to future projects.*** *Developing new test artifacts for every project, rather than reusing available artifacts, is time consuming and unnecessary. Standardizing on repeatable testing procedures and techniques can save a considerable amount of time. The average savings per project was about US$170,000 (68,254 + 101,125).*
- ***Integrating end-to-end processes.*** *Integrating requirements management with quality management techniques provides full bidirectional traceability from requirements to test cases and test results. It also allows for various impact analyses, especially as the requirements change throughout development. These are key to leveraging competitive advantages by increasing quality and lowering cost. The average savings per project in this study was over US$100,000 (51,459 + 55,673).*

In summary, a process-led, collaborative and automated approach to quality management not only enabled risk mitigation strategies and improved quality, but also saved, on average, over US$500,000 per project!

A strong quality management strategy can help an organization position itself for innovation and growth.

Fulfilling the vision

To fulfill the promise of innovation and growth—for example, business agility from service-oriented architecture (SOA) and component-based software development across geographically distributed teams—organizations must establish a unified and responsive quality management strategy. Software and systems delivery teams can help in this effort by examining three key areas.

Highlights

Organizations must establish a quality management strategy that involves collaboration, automation and actionable governance.

Collaboration

Teams must be kept in sync with streamlined, dynamic processes and activity-based workflows. Always-changing business requirements must ripple through the quality process to make sure that test cases are updated and that developers understand the latest requirements. QA and project managers need to ensure that their teams are working on the highest-priority tasks. Test planning should be continuous and involve a goal-oriented approach with entry/exit criteria and prioritized environmental configurations.

Automation

Software development has seen individual practitioner activities, such as functional and performance testing, become automated. And today's build engineers use timesaving build scripts and tooling. But more needs to be done to automate the process and steps between roles to improve organizational efficiencies, save money and speed time to market (e.g., test-case and test-script reuse across lines of business, test-lab automated provisioning/analysis, and automated translation of use-case models into test cases).

Actionable governance

The results of all types of testing—including unit, functional, integration and scalability testing—must be available immediately for reporting and trend analysis. The integration of technical project information with business analytics supports higher-level decision makers' influence on the allocation and use of resources for aligning IT and business. Test management and planning capabilities should be tightly integrated into the requirements analysis and definition processes. Test teams engaged in early test-case planning must

Highlights

have realtime access to the project business and functional requirements, use cases, and service-level agreements (SLAs). Integrating project status data with business analytics allows business analysts to compare project changes with business objectives and constraints. Assessing such data during each phase allows project teams to make adjustments and weigh priority risks. The overall result provides better project control and, when issues within the project arise, more accurate business risk assessment. The right information at the right time, focused and filtered for decision analysis, makes for actionable, governed software delivery.

Process guidelines, best practices and integrated team tools are essential elements in today's quality management solutions.

Quality management solutions

Today's quality management solutions provide software delivery process guidelines, best practices and integrated team tools to help organizations achieve improvements in cost efficiency, quality metrics and time to value for business-critical projects.

As demonstrated above, collaboration is key. People can collaborate using e-mail or enhanced file sharing, but these are terribly cumbersome methods that aren't even part of the core makeup of software delivery. Several organizations have developed collaborative application lifecycle management (ALM) tools that drive the transparency of team progress and govern how teams work.

Highlights

Quality management tools feature collaboration, automation and reporting capabilities to help organizations deliver higher-quality products—again and again.

The IBM Jazz™ platform includes ALM capabilities to support requirements management, development activities and quality management, while at the same time helping teams overcome the geographic, infrastructure and organizational barriers to collaboration. Additionally, Web-based hubs enable people to work together to deliver enduring software quality as a strategic business asset. New solutions allow users to:

- ***Collaborate*** *across business, development and test teams with dynamic process- and activity-based workflows for test planning and execution.*
- ***Automate*** *labor-intensive lifecycle processes and catch quality issues early, reducing time to market, cutting costs and mitigating business risk.*
- ***Report*** *prioritized metrics tailored for individuals and teams, facilitating greater visibility and enabling decision makers to act with confidence.*
- ***Deliver*** *greater predictability by mapping successful deployment patterns to operational key performance indicators (KPIs).*

Companies taking advantage of new requirements definition/management tools will be able to:

- ***Align*** *development projects with business objectives to reduce the risk of project failures that cost organizations billions of dollars annually.*
- ***Drive*** *better requirements elicitation and validation among business and technical experts using proven visual and collaborative techniques (e.g., business process sketches, user interface sketches and storyboards, and use cases).*
- ***Manage*** *change to requirements and project scope more effectively (e.g. impact analysis).*
- ***Improve*** *projects' time to market and return on investment by reducing rework due to poor and missing requirements.*

Highlights

ALM tools help build and ensure quality through architecture modeling, development, functional and system testing, automated build validation testing, manual testing, load and scalability testing, and production application support.

As part of an overall ALM quality strategy, software delivery tools such as those described in table 2 can also help ensure quality and performance of business-critical software.

Tools	Description
Architecture modeling	• Validates user-defined rules that represent architectural constraints • Automatically detects design patterns and important object-oriented structures • Detects structural antipatterns (such as tangles, hubs and butterflies) that degrade performance • Automatically refactors tangles through quick fixes
Development	• Helps developers detect memory corruption, leak detection, performance profiling and code coverage
Automated build validation testing	• Prevents faulty builds from being deployed into the test lab or system test environment • Enables developers to take advantage of off-hours cycles to test an application's stability and functionality
Manual testing	• Promotes best practices such as test modularity and reusability to transition teams from manual to automated testing
Functional and system testing	• Enables teams to build tests that manually or automatically check for regression and functional errors • Shortens automation test cycles to improve quality through broader and deeper test coverage • Encourages more accurate, reliable and reproducible tests • Extend functional testing to Oracle applications environment, including OracleForms™
Load and scalability testing	• Determines load and scalability thresholds for technologies and applications such as Java™ Platform, Enterprise Edition (Java EE); Web (particularly portals); SOA; Siebel; Oracle and SAP • Ensures that a software application can scale and perform to meet SLAs and user expectations • Allows testers to pinpoint the source of performance bottlenecks, enabling them to navigate to source code without wasting time wading through multiple levels of code • Enables a better return on hardware investments by executing pre-deployment capacity planning tests that size the server resources needed to achieve the desired performance and throughput • Extend performance testing to enable testing of Oracle NCA solutions such as Oracle Business Suite *
Production application support solutions	• Provides collaboration between operations and development teams for closed-loop application support, problem isolation and repair • Captures transaction log and trace information, as well as extended system resource data for more granular problem determination, thus reducing response time to application support diagnoses • Enables faster restoration of service levels to the business

Table 2: ALM tools contribute to quality lifecycle delivery.

Summary

In this decade we have seen a three-way collision between the quest for business transformation through innovation, the need for improved software quality to manage business risk, and the demand to reduce costs for economic survival.

The industry's shift to collaborative teamwork across geographically distributed teams, coupled with automated processes and reporting to support actionable governance, is changing the face of quality software delivery.

Quality management is no longer just about QA teams with automated testing tools supporting traditional software delivery; nor is it about quality management control processes that slow delivery or simply test and repair defects. Quality management is about inserting quality into an iterative development cycle and a proactive, closed-loop software fitness program, supported by integrated tooling, data and traceable metrics. Effective quality management makes the delivery process more manageable and less painful, and it helps build confidence with operations teams. Essentially, it breaks the iron triangle mentioned earlier in this paper because it allows managers to optimize scope, cost and time while improving quality.

Enabling quality management throughout the lifecycle and detecting defects early in the process reduce cost and improve credibility. Constant communication via a common and well-understood set of requirements—and quick responses to changes in those requirements—can infuse quality into software development right from the start. Additionally, diligent governance of the build and test execution processes to correct courses and allocate resources can help organizations become more flexible, address compliance regulations, react faster to changing marketplace conditions and, ultimately, drive business growth. In this unforgiving economy, success or failure often depends on one thing: whose product is the highest quality.

For more information

To learn more about quality management solutions from IBM, call your IBM representative, or visit:

ibm.com/software/rational/offerings/quality

Special thanks to Mr. Ron French from IBM Global Business Services for sharing key findings from his work and the work of his team.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
March 2009

1 http://www.devtopics.com/20-famous-software-disasters

2 Walker Royce, *Software Project Management: A Unified Framework*, Addison-Wesley Professional, Indianapolis, 1998.

3 IBM Global Services, SEANT (Systems Engineering Architecture & Test) ReUse Program, 2007.

RAW14125-USEN-01