from The Rational Edge: This article describes the IBM Rational proposition for improved quality, and concludes with a description of the major software development tools that help development teams continuously ensure quality throughout the development lifecycle.

For many business leaders, software quality is often viewed as a luxury -- something that can be sacrificed, if necessary, for added functionality, faster development, or lower costs. In practice though, successful software development organizations have found that an organizational commitment to quality speeds development, reduces costs, and allows new features to be added with greater ease. How can this be? It is because an organization that develops low-quality software, whether for internal use or for sale, is essentially always looking backward, spending time and money fixing defects in "completed" software. By contrast, the organization that builds in quality from the beginning can be forward-looking, able to innovate, and pursue new opportunities. As an added benefit, delivering quality is a powerful differentiator because today, high-quality software is the exception rather than the rule.

But where do you start? The knee-jerk reaction to improving quality is to add resources to testing teams. But quality is not the sole responsibility of the testing team. The entire project team must commit to improving quality throughout the software development process, including post-deployment, and that commitment must be driven from the top by business leadership. This means that an organization's pursuit of quality must reach farther than the goal of zero defects. Quality is the tangible and intangible aggregate of functionality, usability, reliability, performance, scalability, supportability, security, and any other factors important to your customers and your business. Simply put, quality software is fit for use.

Business leaders must instill a passion for quality across the organization, but they have to do more than "talk the talk." They need to provide tools, process, guidance, and training to yield the kind of results that management, customers, and users are looking for. Quality improvement,
like software development, is an iterative process. Ensuring quality does not require a formal, burdensome process, but it does require a first step and a commitment. Once you’ve made that commitment, IBM can help your team be more successful -- not just at finding bugs, but to create greater predictability, higher quality, lower costs, and more satisfied customers.

Based on more than twenty years of experience in helping software development teams build higher quality software, IBM and IBM Rational have developed a comprehensive software development and deployment solution that helps teams innovate and deliver higher quality results by simplifying, automating, and integrating the software development and deployment process. This article describes the IBM Rational proposition for improved quality, and concludes with a description of the major software development tools that help development teams continuously ensure quality throughout the development lifecycle.

Defining quality

What exactly is “quality”? You might believe that you know quality when you see it, but your ability to recognize it is not going to ensure that quality is incorporated into the software development process. Ask yourself this: when quality is discussed, are you and your team talking about the same thing? If you're not in agreement, the result is likely to be project failure.

Let’s try this definition for quality. Quality is 1) a well-defined process for 2) creating a useful product that 3) adds value for both the consumer and the manufacturer.

A well-defined process...

Within software development organizations, a loose, general notion of quality frequently allows leeway that would not be allowed in other industries and should never be allowed in safety or mission critical applications. On the other hand, a solid definition of quality helps teams focus, ensures everyone is talking about the same thing, and promotes thoroughness and attention to detail. And, like a well-written requirement, a definition expressed clearly adds value because it is less open to interpretation.

...for creating a useful product...

In the domain of business applications, quality must be defined in terms of the target audience: the software users. A bug free, aesthetically appealing product that fails to solve the business problems for which it was designed is a poor quality product. Thus, quality is the tangible and intangible sum of functionality, usability, reliability, performance, scalability, supportability, security, and other factors. Perhaps the best definition of quality in this light is the one proposed by James Juran¹: quality means "fitness for use."

...that adds value for both consumer and manufacturer

Put slightly differently, quality is a reflection of added value plus an 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 reflect the manufacturer's attention to details beyond the essentials of transportation, such as usability, safety, comfort, reliability, and so on. The process feeds itself, because a high-quality process means the business does not lose time reworking, refactoring, and...
rewriting. The business is always doing new work, innovating and creating, because it has more time to think about adding value and adding detail.

When an organization defines quality in this way, it is clear that they are focused on more than simply eliminating software bugs. A "quality" application is not simply one that provides the correct results without crashing. To address quality, we must also answer: Does the application meet stakeholder requirements? Is it useable? Secure? Scalable? Reliable? Easily maintained? Easily Extended? Simple to monitor?

Perhaps it is easy to see why quality is a differentiator: it is the total result of what the project team produces. This applies to commercial software as well as internal applications. It applies equally well to packaged or homegrown applications, to new ("green-field") development projects, or to the extension, integration, and modernization of legacy applications (see Figure 1). Throughout development, integration, and testing there needs to be a constant attention to quality.

Figure 1: Quality is a differentiator in all modes of software development

In the long run, an organization’s specific definition of quality may be less important than the fact that it has established one. 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 it run the way it is supposed to? (Is it meeting service-level expectations in deployment?)

If an organization cannot answer one or more of these questions then there is an opportunity to save money and increase quality. Any question that cannot be answered about the efficiency of the organization, the way it is operating, and the quality of what its producing represents an opportunity to improve.

Why care about quality?

As our definition in the previous section implies, it's not just the customer who benefits from a focus on high quality. Businesses that value quality become more effective at innovation, increase
their competitive differentiation, and greatly reduce their total cost of development and ownership. Let's consider the benefits that a high-quality product offers the business.

**Quality enables innovation**
Successful organizations recognize the importance of continuous innovation and differentiation, and achieving these goals requires rapid response to changing demands in a constantly changing landscape. The flexibility and malleability of software suits this immediacy, and organizations that embrace software development as a core business process will move ahead of the pack. Ironically, this great advantage -- the relative ease of making changes -- is also a great source of weakness. A small change can introduce a flaw that brings down entire systems and can cripple a business. From a broader perspective, software development teams can fall into a "we'll-fix-it-later" mentality, forgoing the discipline required in all other fields of engineering. In essence, the ability to change software and change it rapidly is good for business because it enables innovation, but it can also be bad for business if poorly managed because it can lead to poor software quality.

In "The Global CEO Study 2004"², conducted by IBM Business Consulting Services, 450 CEOs around the globe were interviewed to understand their current strategic issues, ambitions, and concerns. This study ranked a number of common business goals that emerged from the interviews. Of all key priorities, "responsiveness" was number one, closely followed by "new, differentiated products," "operational efficiency," and "improved business models." These goals are consistent with IBM's model for successful business in the on demand era: businesses need to innovate, to react as quickly and as efficiently as possible to change, and to differentiate themselves from the competition. However, if an organization's internal software systems are flawed or if your product development teams are struggling to fix problems in last year's release, how can the business respond effectively to current demands?

The answer: A business must continuously ensure a laser-like focus on quality in the development and maintenance of software systems upon which the business relies because it is this quality that makes it possible to react, adapt, and deploy quickly. If there is any breakdown, any inefficiency, any lack of quality, then the organization will fail to reach the market in a timely fashion and the organization will find itself at a competitive disadvantage.

**Quality is a differentiator**
Too often, low quality is tolerated in software development, moreso than in any other engineering discipline or business process. The reasons for this are many: for example, the functionality offered may be considered "better than nothing," or "better than the former system," even though shortcomings are obvious. Or system instability -- frequent system crashes, lost data, etc. -- may be tolerated for similar reasons, or because manual backup systems are available. But compare these typical problems regarding low software quality to low quality in other engineering disciplines. Architectural collapse would never be tolerated, for example. Nor do we expect our manufacturing or sales teams to shut down without notice, and nobody would tolerate them if they did.

What difference will higher quality software make in the marketplace? Let's take a page from the automobile industry, specifically the late 1960s and 1970s. Japanese automakers had
adopted a rigorous quality assurance program and soon gained a reputation for producing highly efficient, usable and reliable cars. This superior quality set Japanese automakers apart from their competition, and led to the establishment of a raised quality baseline for the industry. By differentiating themselves with higher quality products, Japanese auto businesses were able to develop further innovations in fuel efficiency, safety, and manufacturing processes as their competitors played catch up.

This differentiation can be realized in the software industry as well. When your competitor produces higher quality software, they raise the bar. This is true for internal systems as well. The quality of HR, financial, and customer relationship management systems affect hidden costs that are harder to quantify, but are no less important than quality issues in customer-facing software or software produced for sale. Continuously ensuring quality is necessary in any case to deliver in a timely fashion.

In fact, high-quality software that helps run a business can itself be a differentiator. Focused IT organizations typically only build the software and systems that make them unique, while they purchase applications for automating those common business functions that do not demand differentiation from competitors. So the bottom line here is simple: consistent delivery of high-quality software, whether for internal systems or for customers, will set an organization apart from its competition.

Quality is free
A study commissioned by the United States 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 costs are consumed by software developers identifying and correcting defects. The Standish Group reports that canceled projects cost $55 billion dollars. The costs associated with lack of quality are well documented and readily apparent to any organization that has to withstand extended or frequent downtime.

There are several reasons for the high cost of poor quality, depending on the root cause of the problem. For example, poor problem domain analysis -- and thus poor requirements -- leads to unplanned and costly rework. Eleventh-hour validation through testing ruins project predictability, inevitably extending the schedule and thus the budget. And there is the opportunity cost of operational downtime due to reliability or performance problems -- the inability of customers to access your system or of the business to do its work.

But what is the cost of high quality? Philip Crosby tells us that quality is free. Imagine the perfect project:

- A well-defined process with unanimous buy-in from team members
- Focused attention to value-add, detail and quality across the lifecycle
- Constant evaluation of end-user feedback to ensure reinvigoration of the product line and satisfied, repeat customers
- There is no excess cost -- the process is optimized.
Now consider the typical troubled project:

- Poorly defined process with inconsistent buy-in
- Front-loaded requirements gathering without reassessment or validation
- Absent or limited architectural and code quality assurance
- Late cycle system testing
- Limited post-deployment monitoring and assessment
- Excess cost is easy to spot at every stage.

One common misconception about quality is that it can be traded for improved development speed, reduced development budgets, or added functionality. In practice, however, most organizations find the opposite is true. In the long run, improved quality enables teams to deliver more projects on time, at lower cost, with more features. We should realize that Meskimen's law -- "There's never time to do it right, but always time to do it over" -- is a tongue-in-cheek adage for a reason. A development team that continuously ensures quality does it right the first time. By not introducing defects into the system throughout the entire development process, a team eliminates the time and cost required to find and fix those defects later on.

**Driving quality through the organization**

*Continuously ensure quality* is one of four imperatives that software development organizations must embrace to enable on demand business. More than the other three imperatives -- develop iteratively, focus on architecture, and manage change and assets -- continuous quality assurance requires that commitment and dedication be driven from the top of the business leadership chain down.

For example, Six Sigma and Total Quality Management (a precursor to Six Sigma) are structured methodologies for seeking to continuously improve quality and efficiency through a rigorous process and analysis of project metrics. Both approaches are predicated on the leadership of top management and the involvement of the entire organization. Continuously ensuring quality need not be as formal as the above quality initiatives, but it still requires management leadership. Quality improvement as a management objective is essentially a mindset issue, and that mindset must be driven from the top down to instill a culture of quality throughout the organization and throughout each project.

When poor quality software is released, someone in senior management must make the difficult decision to ship software with known defects or postpone the release and risk upsetting stakeholders or customers. It is better not to have to face this conundrum. If an organization fosters a culture of quality, then, through rigorous process, team responsibility, and objective metrics, the organization can create predictability. Teams will learn to scope, estimate, and schedule properly and more accurately. Once a team has mastered that, then it can comfortably commit to delivering on-time and on-specification.

Without these capabilities and a quality mindset held by the entire team, project teams may endlessly rescope and still miss the deadline -- the worst of both worlds. In contrast, an
organization that continuously ensures quality is able to commit to exactly what it can deliver. This organization builds software and systems that are correctly defined with few defects.

**Making a difference with a platform, a process, and a partner**

While establishing a quality-oriented mindset is the responsibility of business leadership, it is up to the software development team to adopt and execute on that mindset. Without adopting the right tools, process, guidance, or training, the development team is not likely to produce the kind of results management, customers, and users are looking for.

Based on more than twenty years of experience in helping software development teams build higher quality software, IBM Rational has developed a comprehensive software development solution that comprises software engineering best practices, integrated tools, and expert professional services. The IBM Software Development Platform is a complete, proven, open, and modular solution for developing software and software-based systems. It helps development teams to innovate and deliver higher quality results by simplifying, automating, and integrating the software development process. The IBM Software Development Platform includes general purpose tools from IBM Rational and IBM WebSphere, and specialized tools for IBM DB2®, IBM Lotus®, and IBM Tivoli®.

At the platform's core is a flexible process framework called IBM Rational Unified Process®, or RUP®, which embodies best practices of software development, including following an iterative development approach, managing change, modeling visually, and ensuring quality.

The iterative phases of RUP emphasize one or more core process disciplines -- analysis, design, coding, testing, and deployment. In each phase and each discipline, an attention to quality is emphasized to identify problems earlier in the lifecycle when they are easier to solve. Let's consider how each of these process disciplines is supported by RUP and the IBM Software Development Platform.

**Analysis**

Meta Group reports that up to 80 percent of the issues leading to customer dissatisfaction can be traced to poor understanding of requirements. Quality starts with analysis of the business to ensure that system requirements accurately reflect, with clarity, the business or customer needs. Regardless of the project -- new application development, packaged application integration, or legacy modernization -- managing requirements is the cornerstone of success. No organization can satisfy the business or its customers if its solutions do not solve the problems for which they were created.

If a customer writes a list of requirements, is that good enough? Probably not, because understanding such a list would likely require domain knowledge acquired only through extensive experience. The people building the software do not typically work in that world, therefore the requirements must be written in a manner that is understandable to software developers. Writing requirements is an activity and a skill that combines the efforts of customers, business users, and the development team.
IBM Rational RequisitePro is an effective requirements management tool, enabling teams to document requirements clearly (via Microsoft Word) while leveraging the analytical capabilities of a database to perform requirements analysis, coverage, and change impact. IBM WebSphere® Business Integration Modeler gives teams the tools to design, test, and communicate complex business processes. It simulates the operational efficiency of processes and analyzes potential business results, while IBM WebSphere® Business Integration Monitor displays real-time information from a variety of environments to allow decisive business performance management and optimization.

**Design**

In design, the primary focus is on architecture -- the "soul" of software. Poor architecture causes a wide array of quality problems including fragility, lack of scalability, and resistance to modification. These problems become more intractable and harder to solve as an application project matures, since the cost of correcting defects grows exponentially as applications progress from design, to code, system testing and deployment. Software developers need to become as architecturally aware as as possible to efficiently detect, isolate, and resolve structural deficiencies. Ensuring quality architecture during design and development pays dividends throughout the project. And after the software is developed, architects need to ensure the quality of the architectural implementation, because software needs to change quickly, as business needs change, without breaking. A well designed architecture takes this into consideration.

IBM Rational Rose XDE® Modeler enables architects and designers to manage complexity by creating technology-independent Unified Modeling Language (UML) models of software architecture, business needs, reusable assets, and management-level communication. IBM Rational Rose XDE Developer Plus provides a single, design-to-code experience via the Eclipse® framework, which is included. IBM WebSphere® Studio Application Developer and WebSphere Studio Application Developer Integration Edition IDEs improve communication and help teams focus on quality issues during design and beyond by supporting a common, industry-standard modeling language and all stages of model-driven development.

**Development**

On average, developers make 100 to 150 errors for every thousand lines of code.\(^8\) 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 figures like these, development organizations in recent years have placed a stronger emphasis on developer-lead testing and analysis, epitomized by some agile practices that endorse test-first development, where tests are built as a precursor to code. In general, unit testing and runtime analysis have become more mainstream, but they continue to lack sufficient management buy-in because of the misconception that they add unnecessary time to the schedule. In reality, schedules typically lengthen because of the time spent debugging code; much of this time is consumed performing root-cause analysis and repair after problems are detected by QA or by customers. For teams that want to reduce risk and increase predictability, a well-formed, proactive quality assurance approach by the development team offers improved reliability.
IBM Rational PurifyPlus® is a complete set of automated runtime analysis tools for improving application reliability and performance. It enables software developers to build and deploy more resilient, reliable software applications by pinpointing memory-related defects and providing them with insight into application performance bottlenecks. IBM WebSphere Studio Application Developer combines Java, Web, and enterprise development tools in a single integrated development environment for building, testing, integrating, and deploying applications. For organizations building n-tier applications, IBM Rational Rose XDE Developer Plus for Java and IBM Rational Rose XDE Developer Plus for Visual Studio offer software designers and developers a rich set of model-driven development and runtime analysis capabilities for building quality software applications based on the Eclipse or WebSphere Studio platforms, and in Microsoft Visual Studio .NET.

**Test**

System level functional and performance tests are an integral part of continuously ensuring quality. The quality engineering team that performs these tests acts primarily as a customer advocate in verifying the ability of the tested application to fulfill customer needs or business imperatives.

As a development organization, it is important to neither overemphasize nor minimize the importance of system testing. On one hand the system testing is unquestionably a key aspect of improving quality. However, quality is not the sole responsibility of the quality engineering (QE) team, nor is testing the exclusive domain of QE. Some tests can and should be run by developers and in some cases even architects.

Unfortunately, many organizations treat QE as a bug sweeper, turning them loose on an application late in the lifecycle with the goal of finding defects introduced by other members of the team. To be sure, no architecture is flawless; no code is completely bug free. However, QE cannot create quality through the testing function itself. System and performance testing cannot be performed efficiently if the application under test is riddled with architectural defects and bugs that should have been identified and eliminated during unit testing. 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 use IBM Rational Robot to create, modify, and execute automated functional, distributed functional, and regression tests. Automated regression testing verifies that changes made late in the lifecycle have not unintentionally "broken" a seemingly unrelated feature, and it does not require manually retesting the entire application. IBM Rational Functional Tester for Java and Web is designed for QA professionals who perform functional and regression testing of Java and Web applications. Using ScriptAssure™ technology, this tool produces resilient, reusable test scripts and validates the interactive data the application generates, minimizing script maintenance. And IBM Rational Performance Tester is used to ensure scalability and reliability under real-world user loads by enabling teams to execute performance tests using simulated users interacting with the application under test in a variety of ways.
Deploy and monitor

At some point, business applications must go live. Inevitably, functional and reliability errors slip through the cracks of even the most thorough processes. When the need to meet or exceed service-level agreements is also taken into consideration, it becomes obvious that a responsibility to quality must extend to members of the IT Operations team. Constant monitoring and assessment ensures the viability of a deployed system and ensures the ability to rapidly detect and respond to inadequate performance.

IBM Tivoli Monitoring provides monitoring for essential system resources to detect bottlenecks and potential problems and to automatically recover from critical situations. Tivoli Monitoring saves system administrators from manually scanning through extensive performance data before problems can be solved. IBM Tivoli Monitoring for Transaction Performance provides IT Operations with performance management for Web and enterprise infrastructure. IBM Tivoli Monitoring for Transaction Performance can help avoid critical performance problems by proactively recognizing, isolating, and repairing problems early, before they impact customers and other end users. In addition, IBM Tivoli security management solutions address two critical e-business challenges: automated identity management and security event management.

Team responsibilities

Quality is the responsibility of each individual on the development team, but it is also the responsibility of the team as a whole. Because an iterative process is essentially circular in nature, each discipline receives input from and provides output to another.

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 linking team members leads to data loss, rework, lack of clarity and 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 Rational Unified Process and Rational RequisitePro, mentioned earlier, it includes integrated tools for software configuration management, change management, test management, and reporting that enable requirements traceability throughout development, from analysis to testing. IBM Rational provides a comprehensive software configuration management solution that 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 helps organizations improve quality by ensuring that software assets are protected, software builds are repeatable and reliable, and defects and change requests are managed properly. IBM Rational TestManager provides control, management, and reporting of all test activities from a single, central point. IBM Rational ProjectConsole dynamically generates a project Web site and metrics dashboard with data automatically collected from the development environment. And IBM Rational SoDA® automates the generation and maintenance of comprehensive project documentation and reports.
IBM also offers a wide range of services to help development teams deliver quality software. IBM developerWorks, for example, offers full instructor-led and Web-based training curricula that teaches methodology and skills. And professional services consultants work with development teams to create custom implementation plans spanning initial project assessment, installation, mentoring and training, and maintenance.

**Conclusion**

When people think about what it takes to build mission-critical and safety-critical software, they usually consider the complexity of the undertaking. There is a belief that in order to have quality, an organization needs to have a cumbersome process with oversight, standards, formality, and documentation. But such assumptions are not correct. There are many ways to achieve quality. While some businesses may require Total Quality Management, or Six Sigma, or pursue Capability Maturity Model certification, most organizations need not adopt such rigorous programs as first steps. It is important to recognize that quality improvement, like software development, is an iterative process. There is no need to accomplish everything with a single step. There are many ways to increase quality, and they can be accomplished in incremental steps. Even small changes can make a tangible difference, including adjusting the organizational attitude towards quality. Quality does not require a burdensome process, but it does require a commitment from business leadership and a mindset change that must begin at the top.

The business benefits of quality are both broad and deep. Not only does quality facilitate innovation by increasing predictability, lowering risk, and reducing rework; it also serves as a differentiator, since it enables a business to set itself apart from its competitors. Most importantly, continuously ensuring quality will always cost less than ignoring quality. Quality is free when it is done right.

For more than twenty years IBM and IBM Rational have been helping the entire project team proactively ensure quality. A testing tool alone can find bugs, but it is not going to fix the root cause of problems. Genuine success means having a more predictable process, higher quality, lower costs, happier customers, and more revenue. Finding more bugs goes only so far, but eliminating them in the first place goes a long way.

This is just the beginning. Like our customers, IBM is in a constant process of improvement and innovation. We are constantly working to improve the tools, processes, and services that development teams need to deliver quality and to succeed.

**Notes**


2. Content no longer available.

3. Content no longer available.


While many believe quality software requires a formal and elaborate development process -- such as Six Sigma or Total Quality Management -- that is not always the case. Organizations that produce mission critical or life critical software systems depend on such approaches, but they are not for every organization, nor are they prerequisites of quality.
