

WHITE PAPER

Improving Software Quality to Drive Business Agility

Sponsored by: Coverity Inc.

Melinda-Carol Ballou

June 2008

IDC OPINION

Software drives competitive business success in a global economy. In that context, software quality is key, and costs of defects late in the life cycle become prohibitive. Increasingly, therefore, we see organizations pushing quality (including security) initiatives, even in a difficult economy. Yet organizations have been releasing software and dealing with defects post-deployment for years. Why do current approaches tend to be inadequate? Companies make money and support business operations with their software using traditional testing techniques and dealing with some level of bugs post-release. Why is it the case that old quality models are falling short to sustain software development and deployment?

With greater complexity from technology, software sourcing, compliance, security, and other areas, addressing software defects has never been more challenging. Keeping up with this level of complexity to retain software agility and profitability demands consistent, reenergized, and targeted approaches to quality. Education is also vital. IDC conducted a survey in 2Q08 to explore current practices in, costs of, and attitudes toward software quality. Even as organizations understand greater complexity levels and acknowledge significant labor to repair software problems, they tend to underestimate current and ongoing costs for defect repair and business impact. This level of optimism can mask the need to evaluate and rework existing quality approaches.

Users must assess the maturity of their quality strategies to provide a gap analysis and evolve toward more effective quality best practices and organizational strategies. They should also evaluate automated testing technology in line with the highest quality pain points and adopt appropriate tools in conjunction with effective process and organizational change.

METHODOLOGY

During 2Q08, IDC conducted a survey of 139 North American companies that forms the basis for data analyzed in this paper. The companies ranged in size from 250 employees to 10,000 or more employees (57% had 1,000+ employees). IT management, IT operations, and software development represented 86% of those responding (with 50% being in IT management). These survey results incorporated responses from 62% respondents who identified themselves specifically as directors and managers. The level of executive response indicates that these managers were engaged enough with quality issues to answer in-depth questions about testing, defects, and code analysis. As organizations need to shift approaches,

executive engagement is a key element. This level of engagement on the part of executives and the quality of verbatim responses are positive, therefore (within the sample size of this survey instrument). Additional sources of information for this document include in-depth user interviews and IDC market and vendor analysis.

IN THIS WHITE PAPER

This white paper presents IDC's analysis of current quality and defect amelioration strategies and market trends, and it offers guidance for best practice approaches. It also provides findings from a recent IDC survey as context for this analysis.

SITUATION OVERVIEW

The Role of Quality in Realizing Successful, Secure, Agile Enterprises

Just as the demand for business-relevant software is rising to drive agility and to respond to global competition, the complexities of scale for businesses and for software development are also rising exponentially. They include:

- ☒ The disruptive pressures of complex sourcing across geographic, temporal, and corporate boundaries with distributed insourcing, outsourcing, and offshoring, as well as open source usage
- ☒ Compliance issues, including regulatory compliance and the localization compliance demanded by competing in a global economy
- ☒ Increased technology complexity and shift in development paradigms and processes, as well as the need to address security and shifting deployment and provisioning alternatives with virtualization
- ☒ An increasingly challenged economy — even as organizations face disruption and growing complexity, they must face this complexity and address software development projects with fewer resources

As we consider **complex sourcing**, IDC's survey found that while 53% of organizations create software in-house, 47% develop using external resources (including in-house contractors, third-party offshore and onshore, and open source). Where teams are dispersed, managing quality across groups demands effective approaches and common visibility into code challenges to facilitate development and quality.

Compliance issues with regulatory, local, and other requirements also demand effective quality management and visibility. The visceral consequences and costs of poor quality with regard to security add another layer of complexity and act as yet another driver for quality adoption.

And the dramatic pace of technology change, with the evolution of multicore development, Web 2.0, SOA, demands adaptive quality approaches to leverage the benefits of emerging technology. (IDC's survey indicated that 71% of organizations are developing multithreaded applications for multicore hardware, for instance.) Additional factors are **shifting the cultural and process environment** for software creation and quality and include the rise of open source usage and development approaches, increased collaboration and reuse — which demand high quality for components being reused — and cross-system dependencies, which result in staggering consequences across multiple applications for software failure. IDC sees a shift in developer attitudes with regard to IP and increased adoption of agile approaches to both development and quality, as companies seek to be more responsive to a complex world in constant flux.

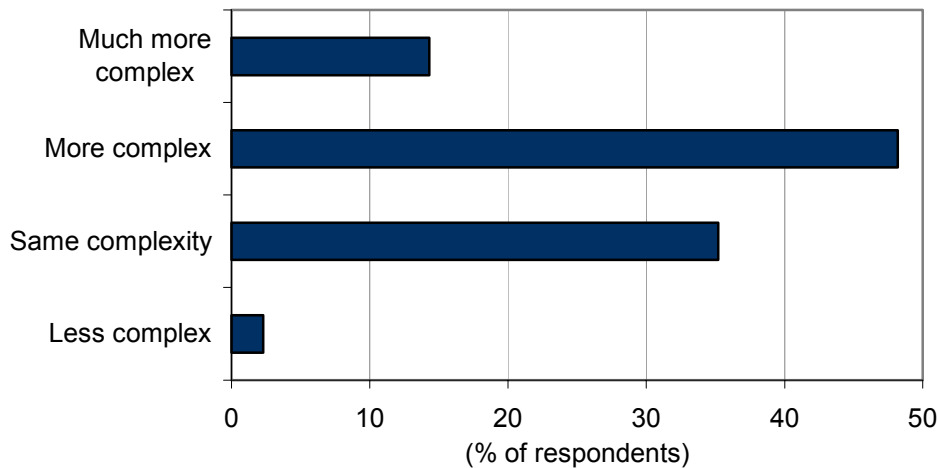
A difficult economy and fewer resources (as companies downsize) also make the costs of software failure and critical defects prohibitive. The margin for error becomes thinner as fewer resources exist to respond and as allocation of human and financial resources is stretched. This is particularly the case late in the software life cycle. IDC research indicates the cost of defects found at the deployment phase for software increases to 100+ times the costs of finding defects early on in the life cycle. Therefore, it makes good business, financial, and development sense to test early and often using effective processes, organizational, and automated tools strategies.

IDC's survey respondents recognized this growth in complexity — 63% described more or much more complexity in their environments (see Figure 1).

FIGURE 1

Code Base Complexity

Q. *Based on complexity measurements such as the number of linearly independent paths through a program's source code, do you consider your organization's code base(s) to be more complex than in the past two years?*



n = 139

Source: IDC's Software Quality Survey, 2008

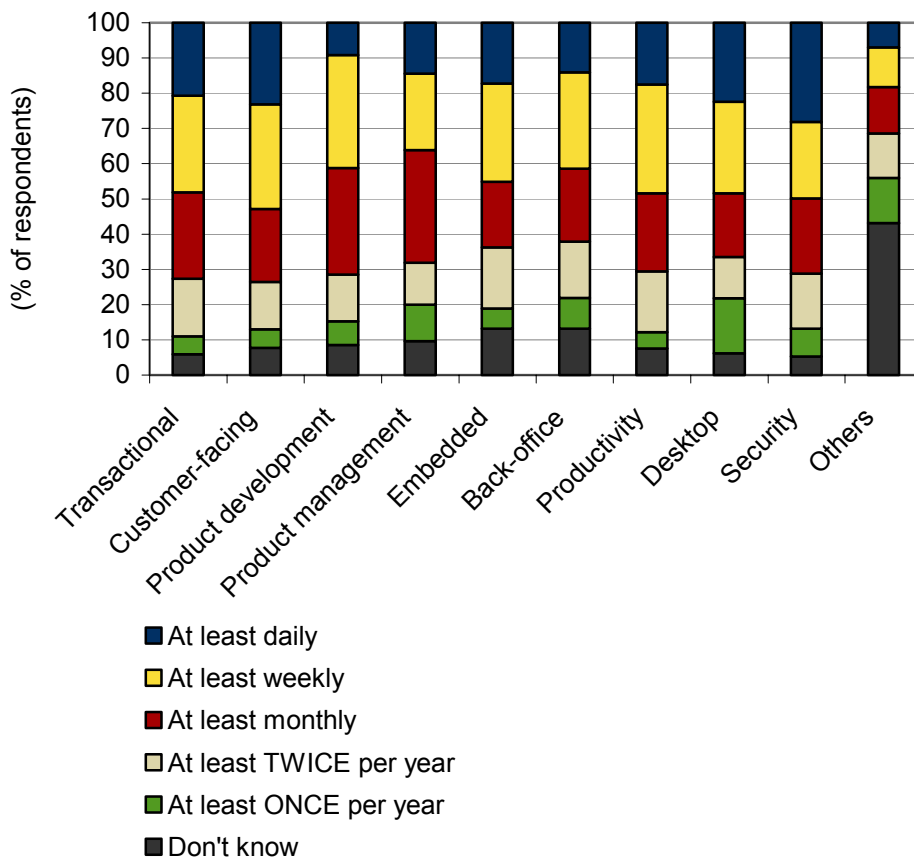
They also described a flawed debugging process — 72% described their debugging process as problematic. However, education is needed across organizations since 62% said that their defect management and testing approach either "did not require improvement" or that it wasn't possible to create change in their approaches (despite problems). And they rated the quality of their internal software developers as better than that of both commercial software developers and open source.

Also as context, the survey respondents described **security, quality, code integrity, and business relevance** as the most important factors on which to focus for software development. The rate of change and churn for key software is problematic. When asked how often applications are updated and rolled into production, respondents said that weekly and monthly are the most common time frames, but key areas of security, customer-facing, desktop, and transaction processing applications are updated at least daily by around a quarter of respondents (see Figure 2).

FIGURE 2

Frequency of Application Updates Rolled into Production

Q. How often are the following applications updated and rolled into production?



n = 139

Source: IDC's Software Quality Survey, 2008

Process change and coordination across software life-cycle areas improves quality (from requirements and modeling through software change management and build to deployment). It is helpful to consider specific quality areas that can augment existing approaches to facilitate code quality improvement and decrease defects. IDC's survey indicated that 69% of users use manual code reviews to find bugs, while 60% use static analysis and 57% use dynamic analysis tools (multiple selects were possible on this question). IDC advises users to augment effective practices with appropriate adoption of static and dynamic code analysis and other automated quality tools to facilitate early detection and repair of defects.

For instance, static analysis tools provide code analysis without requiring the developer to produce complex unit test suites. White-box testing of this kind is available via automated source code analysis. In addition, this test procedure can be performed by the developer in advance of having a complete code base. Static analysis tools let developers test incomplete code, without executables. And different types of quality defects can be identified with static analysis.

Given the code coverage and analysis provided, these tools are able to uncover defect types such as security vulnerabilities, memory leaks, boundary conditions, and crash-causing code constructs that might be missed using other testing technologies that depend on human beings to establish operational parameters. Among the historical challenges to static analysis approaches have been the number of false positives and the degree of noise produced, yet the market has been emerging to address these challenges more effectively in recent years, as we discuss later in the document.

Quality Explosion Issues and Concerns: Learning from Experience

The pressures described earlier are real and are costing customers significantly. The increased complexity of software development environments and the cost of fixing defects in the field (rather than early in the software life cycle) combine in exorbitant ways to drain income and to hamstring businesses as a result of critical software downtime. Respondents from the IDC survey indicated that the costs of debugging are significant. As one example, multiplying the mean cost of one developer per hour (\$68) from the survey, and assuming 30 hours to find and repair one defect in production, results in a cost of \$2,040. As is clear from the survey results (see Figures 3 and 4), significant numbers of respondents experienced 11–25+ bugs annually that require multiple developers and multiple hourly cycles to discover and to repair. While the scope of this survey focused on development teams, there are other, additional costs to consider — such as QA cycles, support calls, and the time it takes engineers to find the root cause of errors — that can drive up that number significantly. (These monetary averages are significant and don't take into consideration the additional business costs for downtime of critical applications [in the case of field defects]. Costs can include [but are not limited to] damage to corporate brand and reputation, product recalls, and significant revenue loss when related to transactional or other key applications as some areas of impact.) Repairing field defects is effortful. A majority of respondents (67%) said that it takes from 2 to 10 workdays to repair field defects, and 11% said that it takes from 11 to 30 days. As another example, the survey results indicated that 37% of developer time is spent

debugging for large organizations that do internal software development. Given the survey's mean cost of developer per hour (\$68), an assumed 40-hour workweek, and a distribution of the number of developers across large organizations, IDC arrives at an annual cost for debugging that ranges from \$5.2 million (based on a midpoint-median 100 developers per organization) to \$22 million (based on a midpoint-mean of 416 developers per organization). Survey respondents estimated that if 100% of defects were addressed and remediated prior to production, they would experience a 32% cost savings.

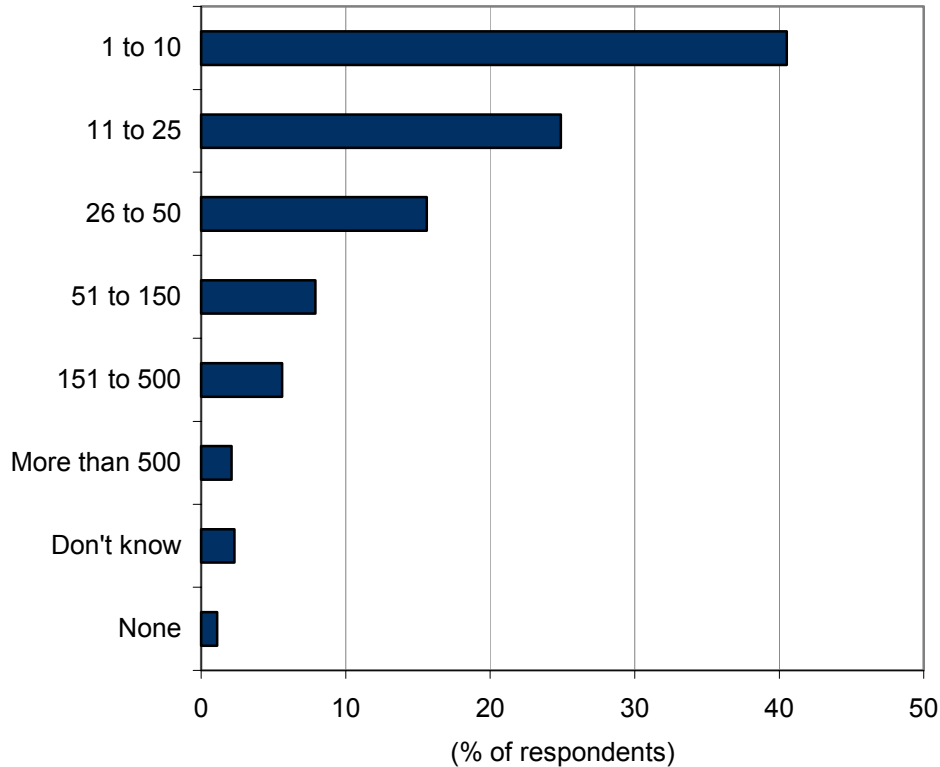
Yet organizations tend to have an overly optimistic view about defect consequences and the state of their organizational approaches. This may be related to the need to present their quality success as "better" than it is, particularly given the competitive pressure of outsourcing and offshoring. And it may also be related to the typical, all-too-human tendency to view one's own work as better than it actually is. (This underscores the need for "fresh eyes" and a separate QA/QC department organizationally. It's hard to have either the perspective or the motivation to view one's own software as flawed, particularly when the pressure and velocity of product releases are extremely high.)

More IDC survey respondents believed that the amount of time required to find and fix defects has been decreasing over the past two years (and will decrease further over the next three years) than believed that the amount of time will increase. And while 75.3% stated that they were confident that their current manual code review process identified "all potentially serious bugs," on the other hand, 31.2% had 26 or more critical defects per year *in the field* and 65.4% had 1–25 critical bugs (see Figure 3). Many of these required patches.

FIGURE 3

Number of Critical Bugs Found Within 12 Months of Release

Q. *On average, how many critical bugs requiring patches are discovered in the 12-month period following release of the software into production?*



n = 139

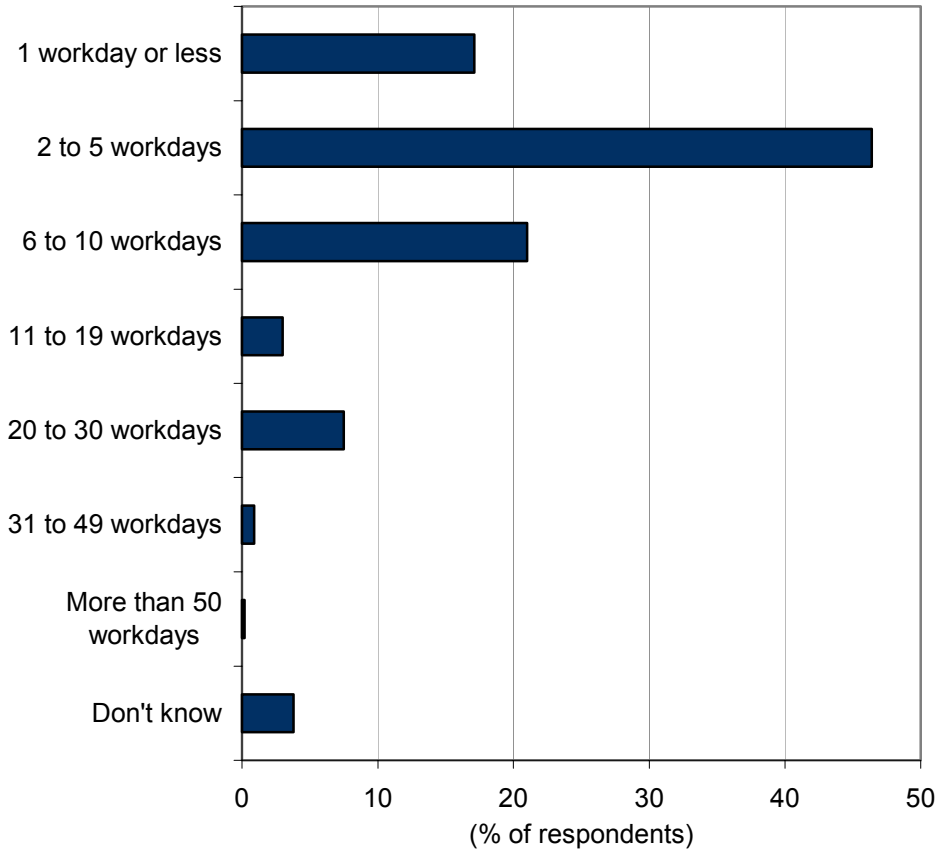
Source: IDC's *Software Quality Survey*, 2008

When asked how many organizations found serious problems post-code review, 25.5% said "very often" or "all the time," and 41% said they had issues with defects found after review. IDC's conclusion is that current, typical quality approaches are woefully inadequate to address code defect costs and issues. This means that they find problems after QA and spend significant amounts of effort and time to repair defects (see Figure 4).

FIGURE 4

Number of Days to Correct a Field Defect

Q. Typically, how many workdays does it take an average developer in your organization to correct a field defect?



n = 139

Source: IDC's Software Quality Survey, 2008

IDC also found that the following five challenges to application quality are top of mind for respondents:

- Time to implement and rapid pace of change
- Financial resources/budget
- Internal staffing/resources
- Multithreaded software
- Project prioritization

This response comes during a time frame of resource scarcity with increasing staffing and financial challenges. It sets the stage for a shift in project approaches.

When the most important drivers for adoption of software quality tools are considered, the top two areas are security concerns and business agility/speed of response. This is in line with core disruptive factors for business and for IT discussed earlier. (These factors are followed by resource constraints, compliance, and architectural complexity.)

Key Takeaways

Key takeaways from IDC's current research include:

- ☒ The challenges of increased complexity and high-end development increase code problems, increase costs, and drive debilitating consequences resulting from defects pre- and post-deployment.
- ☒ Users must become better educated about the business consequences and labor costs of poor quality and defects since inappropriate degrees of optimism mask the need for change.
- ☒ Organizations should evaluate automated tools to supplement manual review along with appropriate process and organizational approaches.

To repeat what was stated earlier, survey respondents estimated that if 100% of defects were addressed and remediated prior to production, they would experience a 32% cost savings.

In the context of these findings, we now analyze the capabilities of tool vendor Coverity, which provides an automated option for code analysis to help address quality challenges.

COVERITY SOLUTIONS

Coverity is a software vendor of automated source code analysis tools based in San Francisco. Incorporated in 2002 by founders Dawson Engler, Seth Hallem, Ben Chelf, and Andy Chou, Coverity has 120 employees, around 450 customers, and additional offices in Boston, the United Kingdom, and Japan. In early 2008, Coverity took in a \$22 million investment from Benchmark Capital and Foundation Capital. (Coverity was self-funded previously.)

Coverity products include Prevent, used for static analysis of C, C++, Java, and C# (shipping 3Q08) source code. It is a commercial application that evolved static analysis to a new generation of tools, leveraging the Stanford Checker, which used abstract interpretation to identify defects in source code. Coverity Prevent also includes defect detection for race conditions, hard-to-find errors that occur in multithreaded applications.

Coverity Thread Analyzer for Java, which was released in 2Q08, observes code as it is executed and identifies race conditions and deadlocks. The product detects problems that occur in limited, bounded testing environments, as well as problems that have the potential to occur over extended operations in field environments.

Most recently, Coverity acquired Codefast to extend the breadth of company offerings to target build management and optimization, as well as code analysis.

Initial static analysis tools evolved during the 1970s but remained challenged as an effective alternative for identifying defects. In early 2000 a second generation of tools (such as Stanford Checker) emerged that offered enough value and filtering of "false positives" to become commercially viable. By leveraging technology that expanded the capabilities of first-generation tools beyond simple pattern matching to also focus on path coverage, second-generation static analysis products can uncover more defects with real runtime implications. Coverity's product evolved static analysis, leveraging solvers for Boolean satisfiability (SAT) to complement traditional path simulation analysis techniques for results that are more comprehensive and have higher accuracy than earlier products. In this way, Coverity assists users by helping to identify defects that are difficult to find with low false positive rates and support of workflow (to facilitate effective development and business stakeholder support).

High levels of accuracy are necessary with static analysis. Developers tend to ignore results or to even refuse to use tools with a high degree of false positive results. For example, if there are 50% false positives, the extra time taken to even review — let alone remediate — issues creates a significant time sink and has a negative impact on an organization. Teams tend to be reluctant to adopt new technology even when it clearly benefits them and saves time, effort, and money. It takes little to discourage adoption of automated tools, which makes lowering the number of false positive results from static analysis tools key to their adoption.

CHALLENGES/OPPORTUNITIES

Among the key challenges for Coverity and other quality tools vendors are the process and organizational barriers to adoption. Initial static analysis tools were plagued with a high degree of both "noise" and "false positives," which discouraged early adopters. This has shifted as the market has begun to evolve. But adoption of automated tools must be accompanied by effective process content to support appropriate and timely usage of product capabilities.

In addition, companies have been interested in leveraging products that combine both static and dynamic analysis. While Coverity offered only static analysis tools, it recently released initial dynamic analysis capabilities (2Q08). As was mentioned earlier, Coverity acquired Codefast, which is both an opportunity and a challenge for the company. The opportunity resides in broadening the reach of Coverity's code analysis capabilities to build management and optimization. Products for build management are few, and some tend to focus on job scheduling primarily (such as IBM's BuildForge). Users as well as vendors (Open Make, Electric Cloud, and other alternatives including open source build tools) will benefit from an additional and stronger competitor in the space. Users facing quality issues related to build

management will benefit from a combined offering. But the acquisition also requires investment and focus, as Coverity must extend Codefast's capabilities for greater competitive value. As a smaller company, Coverity must remain targeted as it incorporates the Codefast team and pushes forward to extend its reach.

Opportunities for code analysis abound for Coverity as mainstream adoption is beginning to occur with static source code analysis tools. Large software development shops are reporting clear benefits from the use of these tools — with reductions in both security vulnerabilities and issues related to poor quality. Additionally, recent improvements in accuracy and performance have begun to increase the buy-in of developers who were skeptical about these tools initially.

Security, particularly, is key, and we see a number of vendors that have announced security-focused approaches to testing, and we expect to see other phases of the development life cycle targeted as well in the coming year (as Coverity is doing with build management).

Vendors overall are leveraging static analysis in different ways depending on the problem domain they're targeting, which of course makes sense. Some vendors focus primarily on finding code defects that impact quality. Other tools find exploitable defects that could be particularly problematic for security vulnerability. And some tools focus on a quality approach that includes uncovering both defects and security vulnerabilities.

Automation can greatly improve the task of checking for code defects, design flaws, and vulnerabilities in a software product by doing so extremely early in the application development life cycle. The benefit here is that users are able to do this prior to full integration, focusing investment on the earliest phases of creating software. Finding problems early lets users avoid the high cost of fixing those problems later in development or having them cause serious business disruption in production. So the risk of software failure and security vulnerability can be greatly reduced when using a static analysis approach up front.

In addition, managers have better visibility into each development project, which can lower the risk of cost and schedule overruns. Managers are able to get a sense of when and where they may need to reallocate resources, for example, as well as anticipating problems that haven't yet occurred. They don't have to spend extra time doing workarounds or allocating staff resources for problems late in the project that could have been avoided with better defect visibility and earlier insight. This can save time, enable effective resource allocation, and improve IT/business adaptability through quality software development.

CONCLUSION

Quality software remains a key business differentiator. Against the backdrop of a complex and financially challenged global economy along with dynamically changing technology demands, businesses and the IT organizations supporting them have no choice with regard to quality initiatives and the need to address the debilitating costs of software defects.

Users should evaluate current approaches to quality and leverage automated approaches for code analysis and testing to help enable more secure, successful, better managed software implementations. At the same time, they should make appropriate transitions to effective organizational and process strategies to support successful technology adoption.

Copyright Notice

External Publication of IDC Information and Data — Any IDC information that is to be used in advertising, press releases, or promotional materials requires prior written approval from the appropriate IDC Vice President or Country Manager. A draft of the proposed document should accompany any such request. IDC reserves the right to deny approval of external usage for any reason.

Copyright 2008 IDC. Reproduction without written permission is completely forbidden.