



Melinda-Carol Ballou
Program Director, Application Life-Cycle Management

Targeting Zero-Defect Software Development: Hype Versus Reality to Derive Business Benefit

July 2011

The business criticality of software and the need for rapid responsiveness in global markets continue to drive growth for quality assurance as the costs of software failure are visceral and prohibitive. The need to reduce spend, improve service, and be adaptive has driven and will continue to drive quality and process improvements, which are being further enabled by software-as-a-service (SaaS) and cloud testing deployment models.

The following questions were posed by Bleum to Melinda-Carol Ballou, program director for IDC's Application Life-Cycle Management research, on behalf of Bleum's customers.

Q. How do you define "zero-defect" software?

A. In striving for zero-defect software, what we're talking about is creating strategies and highly consistent approaches to application development that deliver no major or moderate defects at the point of production; in other words, software that works the way it's supposed to, as early as possible in the deployment process. Zero-defect does not mean "perfect." Perfection is also sometimes not necessary or desirable for many applications, especially given the cost and time delay associated with achieving the highest levels of software quality. Given this, zero-defect software may contain minor or cosmetic bugs that can be cleaned up after an application is in production.

Q. What are the challenges and benefits of this kind of approach to software testing and development?

A. At its best, a zero-defect software approach brings much greater rigor than is typical in development. This covers the range of software life-cycle phases from inception, through development and testing, and then deployment and operations. The approach improves and makes software quality processes systemic from requirements setting, taking into account particular business needs and environments, to an iterative approach in achieving top quality through to software production. This is accomplished by emphasizing the process of finding and fixing major bugs early in the production life cycle through strong motivation, mentoring, and team collaboration. Done right, the zero-defect approach can work well with iterative waterfall or agile development and/or mixed environments.

IDC research indicates that the cost to fix defects later in the life cycle of the software can be as much as 100 times more than the cost of finding and repairing defects early on. Code rework, exacerbated by code dependencies, becomes even more arduous and expensive given technology and global business complexity. This is especially true with business agility being driven by software delivery speed, quality, and relevance. A zero-defect approach can

bring significant benefit to the extent that it enables organizations to improve the quality of their software early on and then consistently maintain such a strategy. That's a good thing.

On the other hand, where the zero-defect approach is more slogan than reality, teams may not actually engage in significant commitment to the disciplined approaches involved. Lip service can substitute for actual engagement with quality approaches. This can inadvertently lead companies to shut down their longer-term strategies for software quality and think they've solved their problems. Then they may no longer have a consistent approach to quality across the board, resulting in neglect of essential rigorous, consistent iterative types of approaches to software quality. Another challenge occurs where there are fewer problems to find, making it harder to locate and repair those thorny, remaining defects. One can actually come to believe that there are zero defects when, in reality, the defects have gone undetected. This leads to a false sense of security and to additional code problems down the road.

An additional challenge can be building software quality and rigor and at the same time stifling creative, dynamic approaches to software creation. Companies can mistakenly straitjacket developers and their partnering with business teams. Iterative, collaborative approaches to improved quality then become key in these situations. Although not necessarily a consequence of a zero defect-oriented approach, it is important to enable your most innovative teams to communicate well to unite creative code development with rigorous testing.

It is also crucial to identify and choose the most appropriate areas for meticulous oversight. In a time where both human and financial resources are at a premium, placing too great an emphasis on software quality where applications or capabilities are of lesser consequence is wasteful and inefficient. As organizations move toward incorporating a zero-defect approach, it becomes increasingly important to ensure a pragmatic and realistic strategy. Companies need to build a continuous improvement strategy into their software quality programs throughout these types of initiatives to help prioritize and deliver new software efficiently.

Q. What are the best practices for developing software with zero defects, and what roles can agile and/or waterfall development play?

A. A zero-defect strategy is more than an approach. It is also a way of thinking. It underscores the idea that software architects and developers — the people creating, deploying, and maintaining code longer term (through the software life cycle) — should do things right the first time from a quality perspective. The associated philosophy is that one can increase profits by seeking to eliminate the cost of failure and grow revenue by increasing customer satisfaction early on and consistently as software is created and then extended. A key aspect of quality, of course, is making certain that what you create aligns with customers' imperatives — their core business needs and individual requirements.

The most important principle is understanding the extremely high cost of quality problems, both initially and late in the life cycle especially. A zero-defect concept should be adaptable to relevant situations and changing business dynamics that drive the development, creation, and delivery of software. The sooner you find a problem and repair it, the more effective your software will be. In addition, this will increase your business' efficiency in delivering software effectively and meeting competitive market demand.

It's critically important to be both rigorous and continuously iterative, with an appropriate feedback loop between the business, development, and deployment. (This approach is typically associated with agile and can be leveraged in waterfall environments as well.) You need to examine, from an end-to-end perspective, the places where flaws and problems may be introduced. This should happen not only late in the life cycle but early and often (from

inception through to and including operations). There's a kind of relentlessness to the zero-defect approach when it comes to iterative testing. If migrating to agile, which uses continuous feedback loops as well as regular and recurring approaches to testing, utilizing a zero-defect strategy can prove helpful in making the transition. (An iterative testing approach can also and should be used in waterfall environments.)

Additional challenges, however, are faced when incorporating an appropriate testing approach into agile development. The constant iterations of quickly creating smaller pieces of code with agile development require a commensurate shift in thinking about quality and change management. Rigor and frequency with regard to agile quality tie in well with a zero-defect approach but also demand organizational transition and strong management.

Zero-defect approaches can represent a significant change in existing development processes, so there has to be effective cultural and organizational change management. Companies need to set the stage for a more rigorous approach to quality. This requires proactive engagement by management, as well as implementing a quality improvement team to help drive this cultural adoption. Initial baselines and metrics to evaluate the evolution of your organization to this type of strategy are also helpful. In this way, you can perform before-and-after analysis of benefits, as well as more easily observe where flaws typically occur.

As mentioned earlier, there is concern that this type of rigor can stifle creativity and innovation in software development. Companies will need to be clear about the interplay between a rigorous approach, which reduces risk and security challenges, and engaging software developers and analysts creatively. All involved should remember that although the quality process is fixed and stringent, some degree of flexibility to address dynamic business needs is key to the creative craft of adaptive, relevant software development.

Q. For which types of applications and industries is the zero-defect approach most appropriate?

- A. The investment in a zero-defect approach is most appropriate for instances where software cannot fail. This includes situations where failure could create either a human or business life-or-death situation. Examples run the gamut from air traffic control systems to manufacturing environments where people depend on software products to keep them safe (jets, cars, medical devices, etc.). We've seen recalls in a number of products recently, most notably automobiles where software was the issue around accelerators or brakes. These are obvious examples where a zero-defect approach would be best suited.

Business-critical systems are also relevant in this context, including transactional, customer-facing applications and key business analytic systems that drive executive decision making and business competitiveness.

Q. How should a business balance the interest in zero defects versus the necessary investment and time to establish such a strategy?

- A. This is the other side of the previous question. Not all software defects are created equal. It's important to remember that there are environments where lesser software performance, defects, irrelevance, and architectural issues don't matter as much. In some environments, the impact of a software problem is so much less that it's not cost-effective to exert the effort in development hours and financial resources to address it as consistently or as thoroughly. Companies need to look at how business critical or life critical their software is and decide on the level of urgency or requirement for a zero-defect approach.

Companies must evaluate the cost/benefit ratio of implementing this approach for certain types of defects rather than expending energy and resources on addressing problems that aren't going to seriously impact users or customers. A risk-based and value-based analysis and understanding should be a part of this assessment so that one understands where best to expend resources in establishing a zero-defect strategy.

ABOUT THIS ANALYST

Melinda-Carol Ballou serves as program director for IDC's Application Life-Cycle Management research. In this role, Ms. Ballou provides thought leadership, expert opinion, research, and analysis through comprehensive research on application life-cycle management (ALM), with specific focus on software life-cycle process configuration and management, software quality, and IT governance software. Ms. Ballou also offers competitive intelligence and consulting on key aspects of the ALM market to service providers, investment firms, and G2000 end-user companies.

ABOUT THIS PUBLICATION

This publication was produced by IDC Go-to-Market Services. The opinion, analysis, and research results presented herein are drawn from more detailed research and analysis independently conducted and published by IDC, unless specific vendor sponsorship is noted. IDC Go-to-Market Services makes IDC content available in a wide range of formats for distribution by various companies. A license to distribute IDC content does not imply endorsement of or opinion about the licensee.

COPYRIGHT AND RESTRICTIONS

Any IDC information or reference to IDC that is to be used in advertising, press releases, or promotional materials requires prior written approval from IDC. For permission requests, contact the GMS information line at 508-988-7610 or gms@idc.com. Translation and/or localization of this document requires an additional license from IDC.

For more information on IDC, visit www.idc.com. For more information on IDC GMS, visit www.idc.com/gms.

Global Headquarters: 5 Speen Street Framingham, MA 01701 USA P.508.872.8200 F.508.935.4015 www.idc.com