A Forrester Consulting Thought Leadership Paper Commissioned By Coverity

Development Testing: A New Era In Software Quality

Demands For Speed And Innovation Are Driving Quality Earlier Into The Software Development Life Cycle

November 2011

FORRESTER®

Headquarters | Forrester Research, Inc.
60 Acorn Park Drive, Cambridge, MA 02140 USA
Tel: +1 617.613.6000 | www.forrester.com

Table Of Contents

About Forrester Consulting
Forrester Consulting provides independent and objective research-based consulting to help leaders succeed in their organizations. Ranging in scope from a short strategy session to custom projects, Forrester's Consulting services connect you directly with research analysts who apply expert insight to your specific business challenges. For more information, visit www.forrester.com/consulting.

Executive Summary

Software is increasingly becoming a key part of any business' ability to compete. Speed and innovation have risen to the top of the to-do list for software delivery teams. In parallel, software delivery teams are looking to deliver their applications on a growing list of platforms and channels. The result is the need to build software faster, demonstrating more innovation in an environment of increased complexity. Those objectives translate into an ever-changing software development team that is applying more Agile approaches, using different technologies, and executing different practices. But what of quality? In August 2011, Coverity commisioned Forrester Consulting to survey firms in North America and Europe to answer the question, "What are the implications of expanding testing in development, and what does it mean to development quality?". The survey targeted IT leaders from more than 200 companies.

Key Findings

Forrester's study yielded six key findings:

- **Development testing has dramatically increased in importance.** Eighty-seven percent of people surveyed believe that development testing is more important than it was two years ago, and 97% have plans to increase their investment in development testing. Respondents cited improved product quality, improved product security, and cost reduction as the top three items driving their development testing initiatives.

- **The longer you leave defects, the more they cost.** The survey reported that 73% of respondents believe that increased cost is the most serious consequence of finding defects late in the software development life cycle. Seventy percent of respondents believe that development testing is more important today because they have an increased awareness of the time savings of finding and fixing defects during development. Forty-one percent of respondents cited the ability to resolve defects more quickly as the primary benefit of development testing, followed by the ability to better meet time-to-market schedule.

- **Developers are under pressure to deliver innovation quickly to market but must maintain quality.** Sixty-four percent of managers responded that developers are under more pressure today to deliver innovation. Fifty-eight percent cited increased pressure to deliver faster to the market. Yet developers cannot sacrifice quality for the sake of speed.

- **The most important quality measures are subjective in nature.** Seventy-seven percent of respondents believe the number of defects introduced compared with peers is a key measure in the success of a development project. This reinforces the relative nature of quality that without firm comparisons, development teams continue to look to their peers to provide context to their results.

- **Testing is a critical component of the developer's role and is expanding beyond its traditional definition.** Seventy-nine percent of managers believe testing is a key part of the developer's role. And the nature of testing methods deployed during development is expanding beyond unit testing as the sole safety net. The survey indicates that security (69%), performance (67%), and functional (50%) testing are being undertaken more often than unit testing (48%).

- **Collaboration between development and other groups is a problem.** Sixty-three percent of managers reported that a lack of collaboration between QA and development has increased the risk in a project, and 46% of

respondents say it has resulted in increased project costs. A lack of visibility and conflicting priorities were cited as the primary barriers to collaboration.

- **Delivery capability is not improved by architecture.** It comes in a poor fourth after process, skills, and tools. This highlights the weakness of the discipline and the disconnect by development teams between good architectures and improved delivery capability. Technical debt, a key measure of software maintainability, was also not measured by 31% of respondents as a part of project success.

Speed And Innovation Are Driving Quality Earlier In The Life Cycle

The pace of business innovation continues to increase. Forrester has observed that businesses that traditionally took years to bring new products and services to market now strive for those changes to be introduced in months and perhaps days. Innovation is also increasingly the responsibility of the software delivery group, with the business looking to technology to create new products, connect to the customer in different ways, and exploit new information and sales models (see Figure 1). For many software delivery teams, this has resulted in a fundamental shift in their working practices.

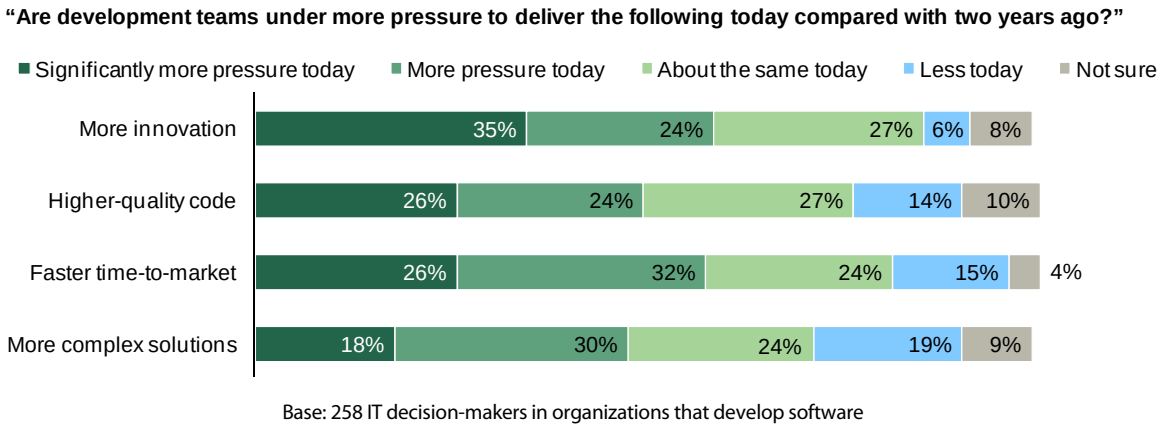
In a recent survey, Forrester found that more than 38% of developers are using an Agile method, with increased velocity being cited as the primary reason for such a change.[1] Agile encourages cross-functional teams to deliver software while focused on a common goal. In the same way that Agile methods drive development and the business to have a closer working relationship, they also encourage the practice of testing to be both started earlier and more integrated. But starting testing earlier is not just the province of Agile development teams; even delivery organizations that are not using Agile, when challenged with increased velocity, have a strong focus on quality in development (see Figure 2). The importance of development quality continues to grow. Eighty-seven percent of organizations surveyed describe an increase of importance for development testing, with 31% describing that increase as significant. Motivations include the following:

- **Finding bugs early saves you time and money later.** It may come as no surprise that the majority of organizations surveyed describe finding bugs earlier as the reason for development testing, with 70% describing their motivation as time and cost (see Figure 3). The idea that finding bugs earlier has a direct impact on cost and schedule is nothing new, with Capers Jones describing the impact on software development productivity in his studies during the ’90s, but without the motivations of increased innovation and speed organizations might not act on it.

- **Finding performance and security bugs late can be showstoppers.** Security and performance bugs are perhaps the hardest bugs to fix, and they also have the most impact on the overall architecture of the software. By focusing on discovering these bugs early, development teams reduce their impact.

- **Quality, quality, and more quality.** Modern users of software have considerably less willingness to put up with bugs and problems. This is even more important when end customers are using the software. Customer experience is increasingly including not only physical interactions and the product but also electronic channels. It is therefore crucial that quality is baked into the application from day 1.

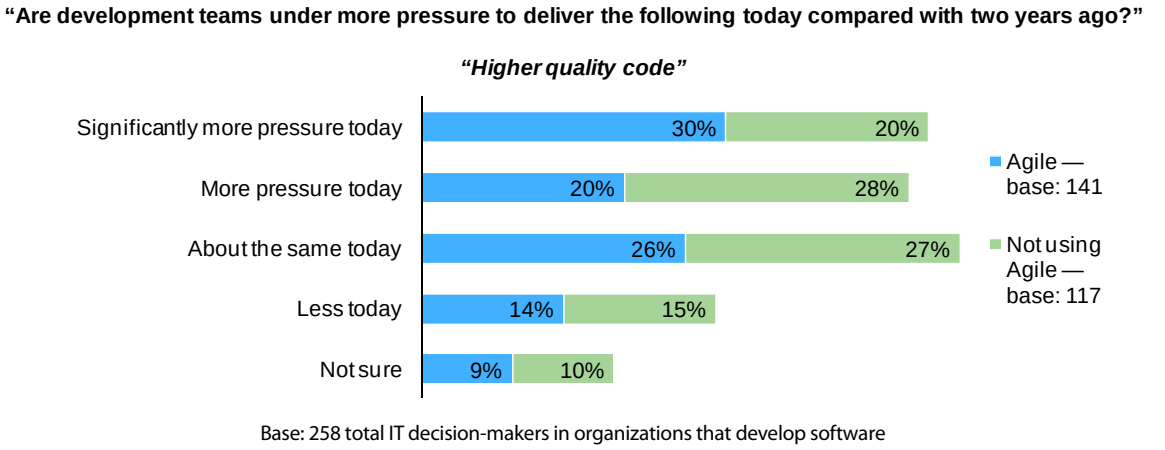
Figure 1
Innovation And Speed Are The Biggest Pressures For Development



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

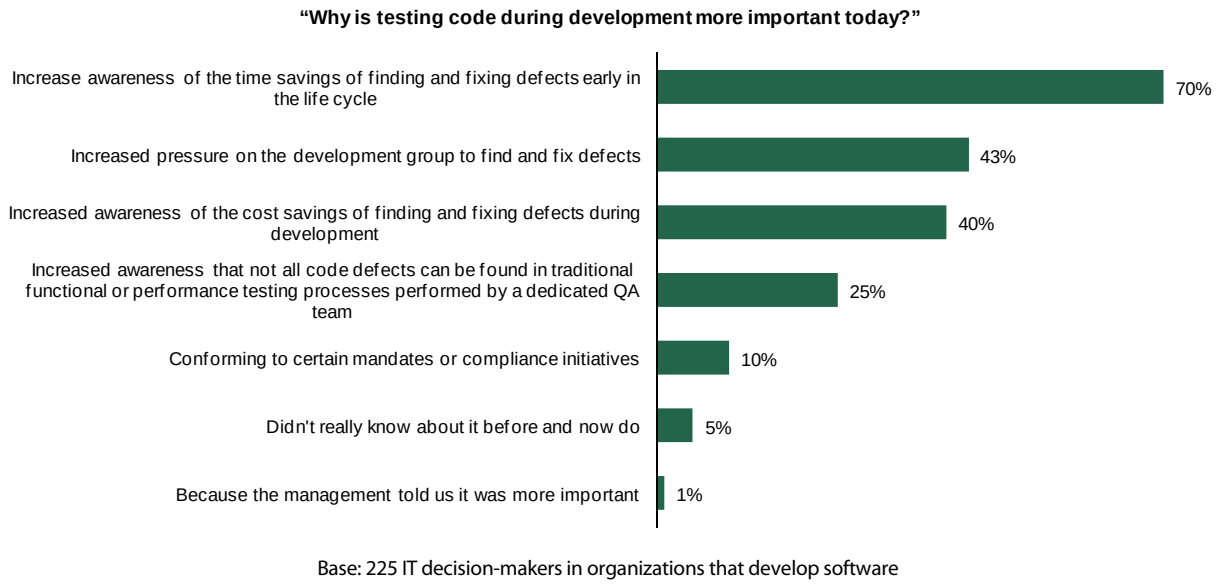
Figure 2
Agile And Traditional Development Approaches Value Development Quality



Base: 258 total IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Figure 3
Increased Savings Of Time And Money Drive Development Quality



Base: 225 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Finding Defects Later Costs Money And Affects Speed

It comes as no surprise to anyone involved in software that the longer a defect is left undiscovered and resolved, the more money, effort, and time it takes to fix. Defects, unlike fine wine, do not age well. But with increased pressures on the development team to deliver more innovative software faster, it often seems much more prudent to leave discovery of defects until a fixed point, allocating focused time on their discovery and resolution. This process may even be given a cool name such as "hardening sprint" or "quality smackdown." Barry Boehm and Philip Papaccio found that an error created early in the life cycle and not fixed costs you 50 to 200 times more to fix later in the life cycle than if fixed in the stage it was created.[2] What is more telling is not the cost but the architectural impact of late discovery. As time pressures become more pressing later in the project, defect fixes tend to be *scrappier*. Architectural rules, so easily followed in the earlier stages of the life cycle, suddenly look like an overhead when everyone is watching the clock.

Innovation Is Not Just About Adding More Stuff

When many organizations think about innovation, they focus on adding new capabilities and functions. But innovation is also about building software that is easy to change and flexible to use (see Figure 4). Increasing software flexibility is a key characteristic of the value of addressing technical debt, a metaphor for describing the value of improving not only what the software is but also how it does it. Technical debt is the cost incurred to change the software, which, if left, will only increase. Increasing architectural flexibility is a key characteristic of technical debt, with debt being the metaphor to describe the increased cost, time, and effort to make changes to the software. Increased technical debt:

- **Slows down time-to-market.** The more complex the solution, the harder it is to change but also to test. For example, many organizations' legacy systems have evolved to a point where the only way to know what your change has impacted is to test the software and see what breaks. This increased test burden makes the process of delivery harder and harder.
- **Makes it harder to swap out resources.** The more complex the solution, the harder it is to describe. That leads to organizations having to rely on certain key workers who have a strong knowledge of a particular system and makes changing staff or managing transitions increasingly hard.
- **Reduces the overall value of the software.** Software is like any other asset within an organization and, unfortunately, may depreciate if not maintained. As the cost of making a change increases, the return on that change reduces. At a certain point, the cost will outweigh the value and make it hard to justify change. Innovative ideas will stall because the cost of change is so hard, thus making the initial investment a significant barrier.

Figure 4
Making Factors Affect Improving Delivery

"What are the factors currently preventing your development teams from delivering the following? (Select all that apply)"

To deliver:	Top factor preventing delivery	%
Faster time-to-market	**Lack of connection with the business**	44%
	Development teams are siloed.	40%
	Not enough time to test	38%
More innovation	**Existing technical debt**	52%
More complex solutions	**Lack of visibility and control over projects**	40%
Higher-quality code/more innovation	**Lack of collaboration between development and QA**	39%
Higher-quality code	**Legacy software makes developer testing really hard.**	39%

Base: 258 IT decision-makers in organizations that develop software

(Only top factors shown)

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Security, Complexity, And Functional Quality Affect Long-Term Viability

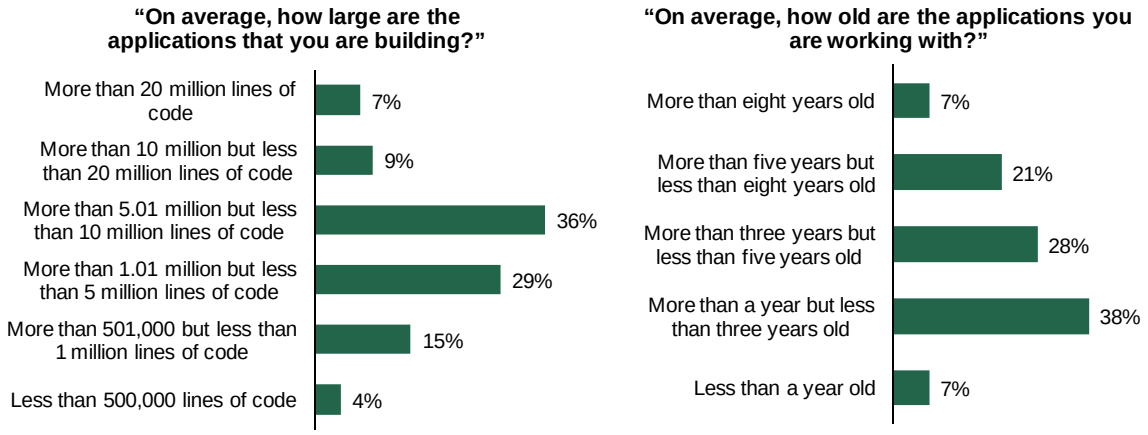
The project culture that exists in many software delivery organizations encourages many teams to forget that software has a long shelf life. Software older than three years is the norm, with more than 28% working on software that is older than five years (see Figure 5). Like many Hollywood actresses, software only ages well when treated with great care. But considering how long software lives, for many organizations, many of the things that are important to manage age are ignored or underinvested in, such as:

- **Documentation.** If the software has to be maintained then it is crucial that documentation describing the form and function of that software is maintained. For many development teams, documentation is considered as an

afterthought, being left until the last minute or considered an ancillary task undertaken by more junior members of the team.

- **Consistent staff.** Without strong documentation, it is important to keep teams together, allowing tacit knowledge to support the maintenance of the software. However, the majority of software delivery organizations follow a project-based approach, moving resources as necessary to accommodate the project. This results in a transient culture for software teams with lack of ownership for the software assets they work on.
- **The architecture.** Like any good city, it is easier to navigate the software when it follows a good plan. However, the majority of organizations do not measure the software's adherence to the plan in a formal way, relying on code reviews and the role of the architect to ensure compliance. These tests are subjective and allow human factors to creep into the evaluation.

Figure 5
Old And Large Is The Norm For Software



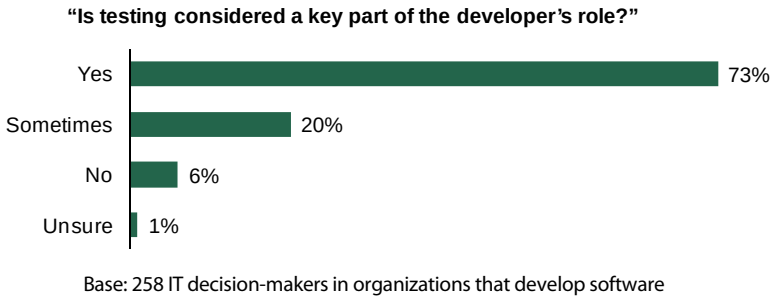
Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

The Relationship Between Development And Testing Is Changing

With increased speed and innovation, the relationship between testing and development is changing from one of a service to a partnership. The lines between testing and development have blurred. Developers are doing more testing, and testers are being involved much earlier in the life cycle. Testing and quality are now considered a key part of the developer's role (see Figure 6). The extension of the role allows developers to extend the reach of testing, including additional tests focused on performance, security, and service. It also enables quality professionals to utilize the skills of developers in building more comprehensive automation suites.

Figure 6
Test Is A Crucial Role For Development



Base: 258 IT decision-makers in organizations that develop software

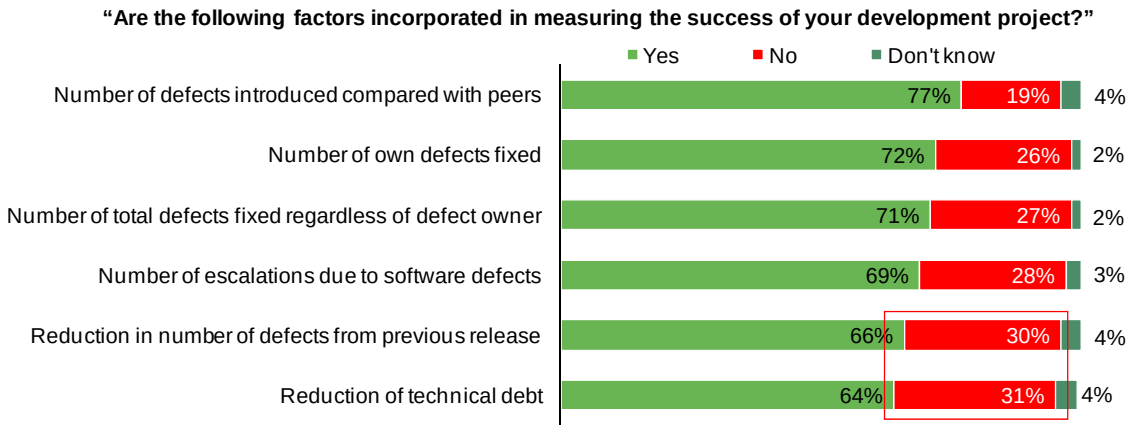
Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Developers Are Measured For Today, Not Tomorrow

Though software quality is a key responsibility of the developer, there is a tendency to focus on the immediate rather than the long-term quality. Business-driven quality measures are often ignored in favor of comparative measures with other development teams and a strong focus on their own related defects (see Figure 7). Functional, performance, and security tests are the emphasis of development testing (see Figure 8). Longer-term views of the software in the areas of usability and flexibility are ignored. Software is rarely short-term, with many software systems having longer tenure than the people who work on them. Installing a long-term view to developer quality ensures that:

- **Complexity does not get out of hand.** Without clear guidelines on complexity, it is very easy for development teams to increase the complexity of the system without any regard to the impact of that complexity. Measures such as cyclomatic complexity and approaches that focus on high cohesion and low coupling help development teams understand the impact of the changes they are making.
- **Defect numbers trend down.** Absolute measures on defects provide little reference for development teams. It is therefore crucial to see relative improvement in quality. Trending information is a great mechanism to compare a team's overall quality today with where it was, with a view to continually improve quality.
- **Customer insight is not ignored.** Without clear insight into how effectively customers use the product, it is difficult for developers to have a clear idea of the overall quality of the product. In the absence of real customer data, measures provided by operations and support can provide a proxy for real usage information.

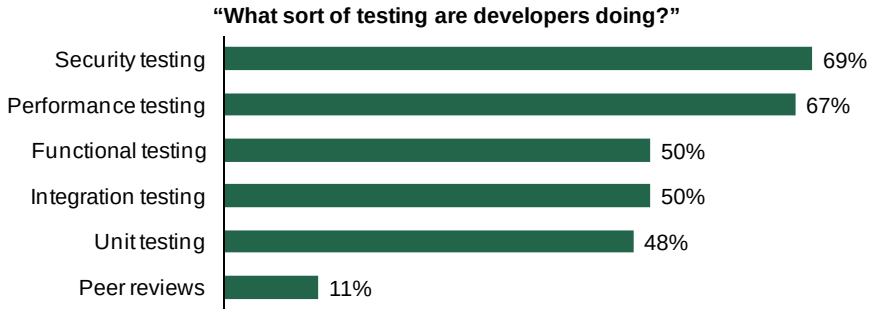
Figure 7
Development Quality Focuses On Comparison Of Peers



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Figure 8
Security And Performance Testing



Base: 240 IT decision-makers in organizations that develop software

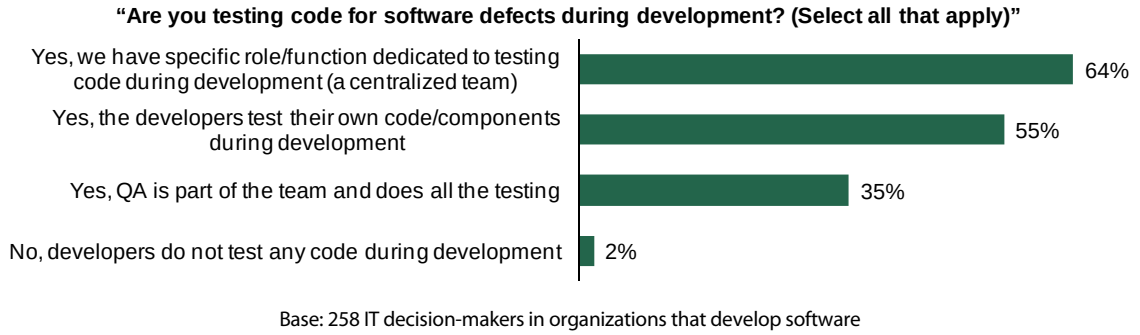
Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Developers Are Not The Only People Doing Testing During Development

The term "development testing" often leads us to believe that the focus is developers doing testing when in fact, the word "development" is meant to signify the stage of the life cycle or, in the case of Agile, the work that happens within a sprint. Development testing, like many activities during a project, is a team sport, with many team members working on it. Developers are a key resource, but their skills are augmented with professional QA practitioners, business analysts, and software architects (see Figure 9). The whole team shares the objective of increasing the quality of the software, with different practitioners doing tasks that make the most sense to them. For example, developers might spend time creating performance and security scripts, while QA professionals build out the functional testing environment or create test data. This partnership model allows for great flexibility and ensures that the right resources

are involved. But it also encourages a real team-based approach with shared goals and collaborative working environments.

Figure 9
Testing Happens During Development In Many Different Forms



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Future State Requires Collaborative, Automated, Integrated Test Practices

It is clear that leaving testing to the end of the life cycle and disconnecting this from the practice of software development only has a negative impact on the project. But to optimize the software manufacturing process requires a change to technology, roles, and organizational models. It also requires delivery teams to focus on measures and metrics that encourage collaboration and aligned software delivery efforts to business value. It is far too easy for development organizations to invest in development testing without that investment being aligned to value. Code coverage is an example of a development testing practice that can generate large amounts of work with diminishing returns. By setting test coverage levels that are nearly complete, development teams will spend a great deal of time building tests to cover each bit of code — code that might be simple, infrequently used, or tested in some other way. It is therefore crucial that development testing is considered in a holistic way, focusing on practices that provide a balance between value and effort.

Automate The Simple Tasks To Ensure Consistency And Visibility

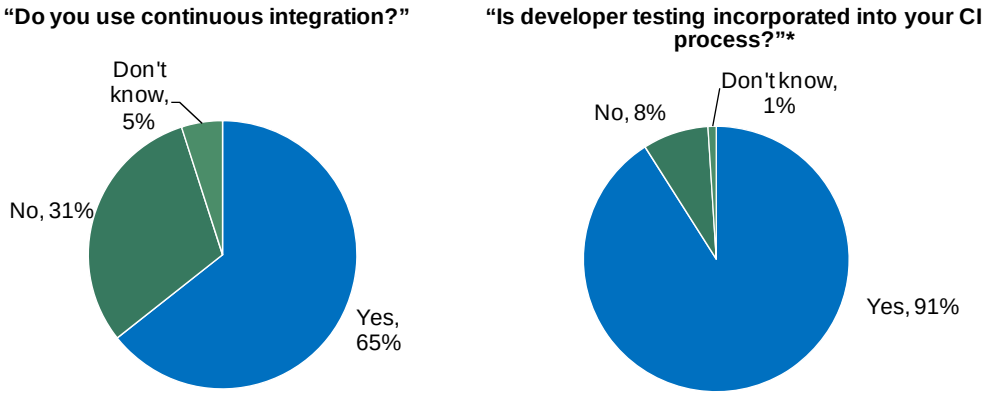
The delivery team undertakes many algorithmic tasks — in particular, in the areas of build, configure, and deploy. By focusing on the routine tasks, automating and providing dashboards, delivery teams can increase the consistency of these tasks while ensuring that the team has visibility of what is happening. This principle holds even truer when the team is distributed. Distribution increases the lack of transparency in the project and reduces the overall amount of trust. By automating tasks, distributed teams will get consistent outcomes across locations. In the area of development quality, automation should focus on:

- **Continuous integration (CI).** CI is a growing trend within the broad development community, and 65% of the people surveyed are using CI on their development work. Though 65% sounds like a great number, it means that

more than 30% are not (see Figure 10). By increasing the frequency of integration that CI provides, delivery teams will improve their visibility of the overall quality of the software. Integration issues build problems, and code conflicts will be front and center, allowing these problems to be resolved.

- **Integrating testing into CI.** Once the delivery team has built the foundation of frequent build and integration, the next logical step is to integrate development testing into that process. Automatically executing unit tests as part of the regular process allows teams to test not only their code but also their code in the context of other people's code. Code that worked so well in the developer's sandbox may suddenly stop working or behave in strange ways when connected to a much more complex set of changes.
- **Guarding entry to CI with instrumentation.** Code coverage and static code analysis provide great ways of ensuring that the code that enters the CI process is of an appropriate quality. This ensures that the team does not spend wasted time dealing with immature code or incomplete tests. By using instrumentation to gate the process of CI, delivery teams increase the value of the CI process, removing many of the time-consuming simple issues that often plague its use.

Figure 10
Continuous Integration Is A Key Development Quality Practice



Base: 258 IT decision-makers in organizations that develop software
*Base: 167 IT decision-makers in organizations that use CI

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

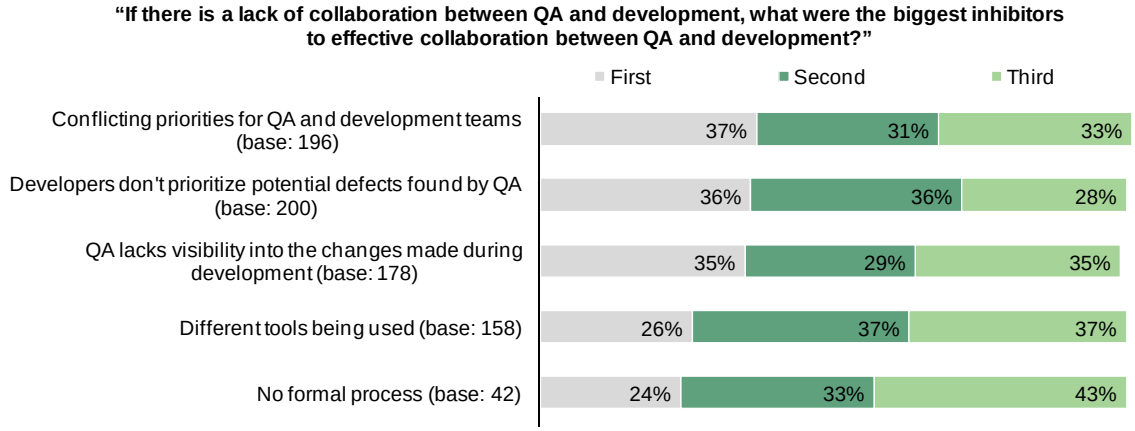
Collaboration Between QA, Operations, And Development Is Key

Software quality, by its very nature, is an abstract concept determined not by machines but by its stakeholders. Developers, managers, business analysts, product owners, and quality professionals all have strong ideas of what quality means, and they express those quality definitions in their test plans and acceptance criteria. In development, those test plans form the basis of development testing activities such as unit testing and integration/architecture testing. QA builds separate sets of test plans, which describe quality from its perspective (see Figure 11). Operations, which may have its own preproduction tests, creates its own definition. Often, each of these groups operates as silos. By increasing the collaboration between the groups and connecting QA and operations to development quality, delivery teams will

fundamentally reduce the risk to the project (see Figure 12). Improving collaboration between QA, operations, and development requires the following:

- **Use consistent deployment models within CI.** Forrester has observed within many delivery organizations that deploying to integration test uses one deployment protocol, deploying to preproduction another, with the final production deployment using a completely different approach. This inconsistency leads to a large number of errors, with some organizations describing about 30% of production tickets being related to infrastructure inconsistencies. By using consistent processes and tools, it is possible to increase quality and reduce time wasted trying to resolve issues that are only issues because of inconsistent deployment models.
- **Put in place dashboards that everyone can view.** Nobody likes surprises, and often, the relationship between these groups is one of the surprises. Development surprises QA with incomplete software. QA surprises operations with what is in the release and how long it has to deploy it. Operations surprises development with production tickets and scheduling conflicts. By having clear dashboards that are shared by both parties, it is possible to reduce those surprises and encourage collaboration between those organizations.
- **Access to development quality results after software is live.** Understanding how the software runs in production and then sharing that with QA and development teams really help provide context for any change decisions going forward. It also allows QA to prioritize tests based on actual usage patterns.

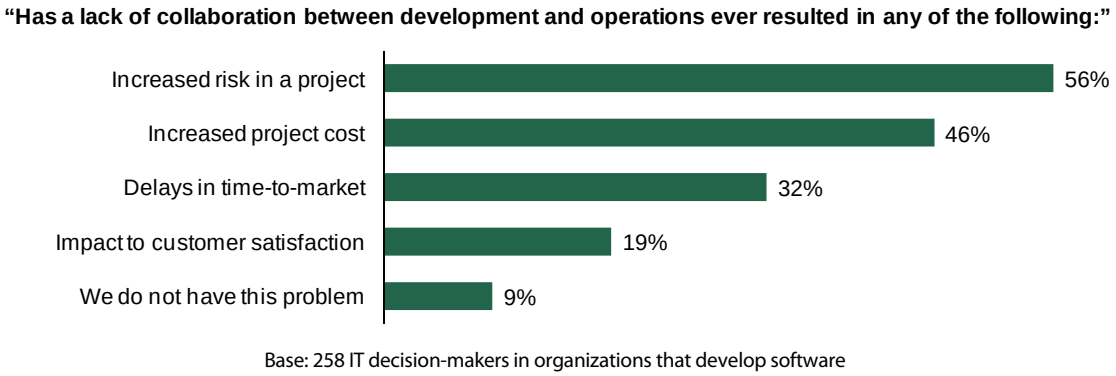
Figure 11
Conflicting Priorities Are Top Inhibitors Of Collaboration



Base: IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Figure 12
Increasing Collaboration With Operations Will Reduce Development Risk



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Technical Debt Is Measured

Quality is a long-term rather than a short-term attribute of the software. It is far too easy to focus delivery teams on short-term quality measures such as functionality or performance. Long-term value also needs to be measured, reflected on, and if appropriate, acted upon. "Technical debt" is a term used to describe this long-term software quality. By putting in place practices that measure that debt with a process for dealing with its results, software delivery teams will actively manage the long-term quality of the software over time. To implement a technical debt practice within development quality, software professionals should:

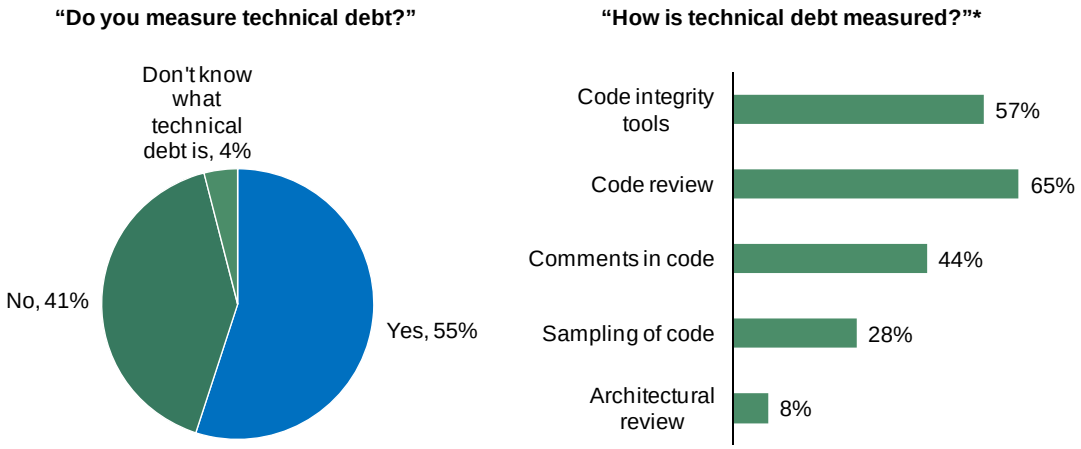
- **Educate themselves on what technical debt is and why it is important.** Without a clear understanding of the impact of technical debt, it is very hard for practitioners to invest time and effort into its resolution. As time pressures mount, doing work that does not directly improve delivery for today is hard to justify. By having a good understanding of the long-term impact of that debt, it equips software delivery pros with tools to balance the short-term needs with the long-term objectives. For the majority of people, no one wants to do a bad job, but the trick is knowing what "bad" is, and technical debt adds to that understanding.

- **Measure debt and report on it.** For many organizations, technical debt is an abstract concept associated with style, comments, and architectural models. These concepts, though valid, are often hard to make tangible. Not only does this make it hard to execute on reducing technical debt, but making the tradeoffs between change today and change in the future is also really hard to make. By having a tool that clearly provides a set of metrics associated with debt, software delivery pros can get a clear understanding of the state of their debt and review how it is changing over time (see Figure 13).

- **Balance working on debt items versus new functionality.** The key to success is not to build perfect code but instead to make intelligent decisions about the level of perfection versus getting the software out the door. On a project, these decisions are made at many levels, ranging from at an application or product level to individual lines of code within one component or service. Peer review helps provide a mechanism to review the majority of the decisions, allowing peers to discuss the quality of the code and its impact. By using objective metrics within

that peer review, software delivery pros can base those discussions on some level of fact rather than just stylistic considerations.

Figure 13
Measuring Technical Debt



Base: 258 IT decision-makers in organizations that develop software
*Base: 143 IT decision-makers in organizations that measure technical debt

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

KEY RECOMMENDATIONS

It is clear that software delivery is becoming more important and more complex. Software that traditionally was hidden from customers is now front and center to their lives. Software is not just running businesses, but for many, it is running the relationship between the customer and that business. It is the place for innovation and value. But that requires software delivery organizations to fundamentally change their way of thinking about the processes, skills, and tools being employed to deliver that software. By moving quality earlier in the life cycle, software delivery pros can build software of a higher quality but more importantly, deliver innovative software faster. To introduce a more robust developer quality practice:

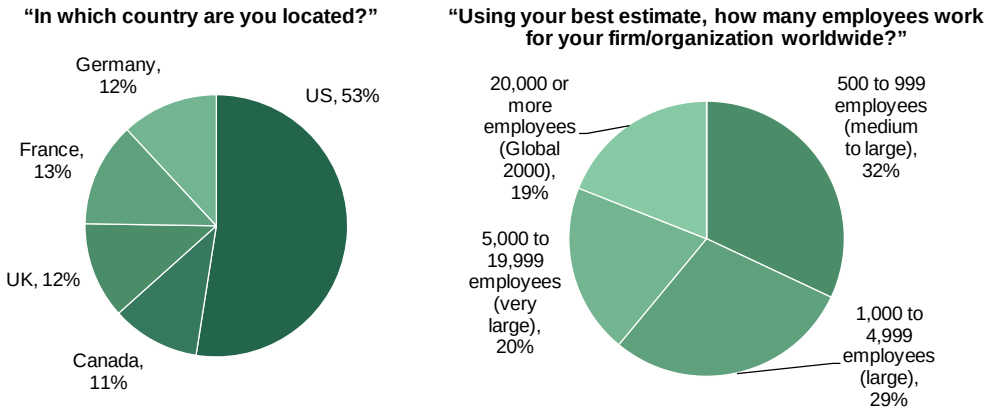
- **Educate developers on testing and quality.** Testing and quality are a key responsibility of developers, but for many, their testing skills were picked up while doing other things. In fact, for many, testing is considered to be a sign of weakness or a discipline practiced by developers who cannot develop. These ideas need to dispelled, with developers not only being educated on great practices for development quality but also understanding why it will help their career and professional development.
- **Automate with tools, and connect them into your build and release process.** As delivery velocity and complexity increases, it is very difficult to measure development quality without using tools. Those tools provide objective measures that can help drive behavior within the delivery team. By using this information, it is possible to better enable the CI process, allowing CI to only select code that has achieved a certain level of quality. This reduces the amount of churn in integration testing, reducing the number of false defects and wasted time.
- **Integrate the life cycle for information sharing.** Quality is not an abstract concept but should be something very tangible to both the developer and the delivery team. By treating development quality as a first-class citizen within the ALM tools and sharing it in the same way that code and requirements are being shared, software delivery teams get a better understanding of the applications and the state they are in.

Appendix A: Methodology

In this study, Forrester conducted an online survey of 258 IT decision-makers in the US, Canada, the UK, Germany, and France to evaluate trends in development testing. Survey participants included decision-makers in organizations that develop code in-house or commercially. Questions provided to the participants asked about testing practices, development methods being used, technical debt, and areas that developers are measured against. Respondents were offered a small incentive as a "thank you" for time spent on the survey. The study began in August 2011 and was completed in the same month.

Appendix B: Demographics

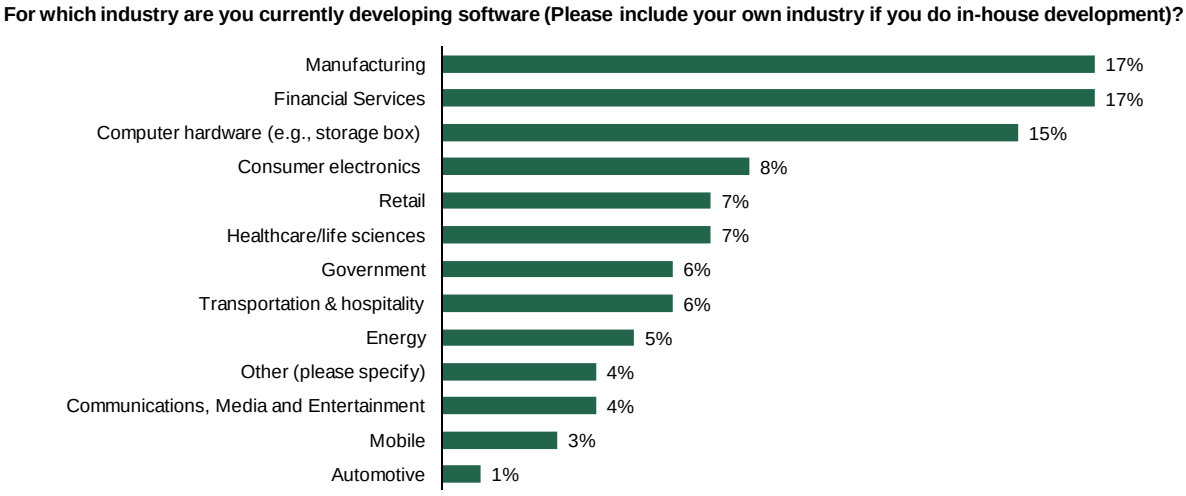
Figure A
Country And Company Size Profile



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

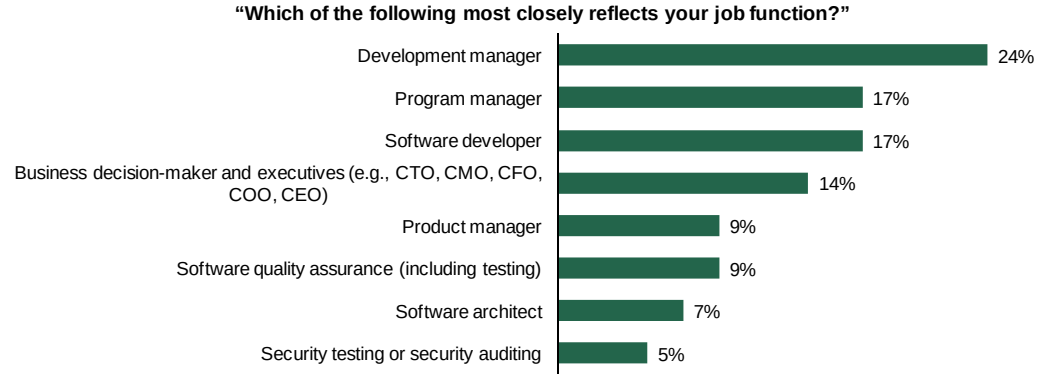
Figure B
Industry Profile



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Figure C
Role Profile



Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Figure D
Types Of Software Being Produced

Base: 258 IT decision-makers in organizations that develop software

Source: A commissioned study conducted by Forrester Consulting on behalf of Coverity, September 2011

Appendix C: Endnotes

[1] Read about adoption and the implication to Agile in "It's Time To Take Agile To The Next Level," Forrester Research, Inc., March 25, 2011.

[2] A great book on the whole economics of software engineering that describes the cost model is *Software Engineering Economics* by Barry W. Boehm.