IDC TECHNOLOGY SPOTLIGHT

Optimizing Quality Analysis to Deliver Business Value Amid Complexity: Code Visibility Cuts Software Risk

Adapted from *Worldwide Automated Software Quality 2014-2018 Forecast and 2013 Vendor Shares: Some Growth in ASQ with Ongoing Adoption Projected for Mobile, Cloud and Embedded, IDC #251643* and *Establishing Software Quality Analysis Strategies to Help Address Third Platform Complexity, IDC #253257*

Sponsored by: SonarSource

Melinda-Carol Ballou Jennifer Thomson
January 2015

INTRODUCTION: UNDERSTANDING THE IMPACT OF QUALITY ANALYSIS GAPS

The shift to a digital world, the impact of digital transformation, and demand for continuous deployment across technology platforms are putting huge pressure on IT organizations as they address dynamically evolving business needs. Time to market of high-quality applications becomes critical, but delivering software releases and developing new customer-facing applications quickly is a growing challenge. This is particularly the case for large, multinational organizations that must contend with a complex web of changing, multimodal technologies combined with legacy systems and resources across thousands of users that are geographically distributed.

For the CIO, the goal is not merely to improve IT agility — it is about using IT to successfully enhance business agility, innovation, and customer experience across the "3rd Platform," which ranges from mobile, to social systems of engagement, to the cloud, while incorporating Big Data analytics. At an operational level this approach puts increased pressure on companies to restructure, modernize, and transform software development and testing practices. This can allow for faster delivery of enterprise consumer applications with appropriate quality, risk, speed, and cost levels.

Yet despite the obvious consequences of poor quality software on customer access, revenue, and business reputation in these influential mobile and other 3rd Platform environments, many organizations have slipped into pitiful software hygiene habits. Deficient levels of visibility into the causes for defects and their impact have created a culture for many where developers don't take responsibility for the defects they create. "Throwing code over the wall" for deployment remains a frequent, visceral issue for companies caused in part by the pressing demand for immediate software delivery in these critical business environments. Yet poor approaches to development are occurring even as the need to improve code quality and remove defects early in the cycle has increased dramatically.

At IDC we have been tracking the success rates of key business change projects within large organizations over the past few years, and our 2014 QA survey across more than 200 large enterprises finds problematic project success rates. We define "success" as timely, relevant, and on budget delivery of individual projects. Our research shows that approximately 40% of business change-oriented software development projects failed to meet these requirements. And the 60% that are delivered on time and on budget typically required post-hoc adjustments to ensure company deadlines were met (for example, more employees needed to be thrown at the project, or

project scope was reduced). Beyond these budgetary and timeliness challenges, lie code quality issues.

IDC research shows the costs of defects found late in production are exponentially more expensive to the organization to repair than those found earlier in the cycle (10–100 times as much or more, depending on other dependencies and how late in the life cycle the issues are discovered). The ability to find code problems, to have common access to issues, and to take responsibility for fixing these issues and addressing the backlog becomes paramount in an environment unforgiving of poor user experience and failed functionality. The immediacy of customer feedback in terms of user engagement, user fall-off rates, and app store ratings brings a new level of visibility and importance to drive adoption of common access to code problems for context, for quick remediation, and for visibility into security challenges.

These factors combine with increased software project complexity. Enterprise adoption of complex, geographically distributed development using internal and external resources such as systems integrators (SIs) has increased significantly, along with use of a variety of development tools (including open source), a range of languages, and agile processes. This means that the need for code analysis with common access to information about software problems and quantitative and qualitative metrics across disparate teams has never been greater.

And in this era of digital consumerization and mobility, projects become both more challenging and critical to manage. Why? Organizations must do more in these more complex environments in a global competitive environment with few and/or declining resources. They must bridge the ever-increasing gap between IT and the business with new forms of social engagement, and contend with a greater volume of requests for innovative applications that meet both business and consumer needs.

Enterprise projects in the context of fierce global pressures with unprecedented levels and rates of technology change demand that departments work together, encompassing business stakeholders and executives across IT, development, quality, and operations in order to survive competitively.

And as we see organizations turning to complex sourcing to address these business and technology needs, we see the need for software analysis, automation, and common process adoption. For IDC, complex sourcing means leveraging the resources of external service providers (such as onshore and offshore SIs), internal resources from business, IT, infrastructure, contractors, and use of open source software (OSS) solutions and components.

The need for partnering and the use of a range of resources demands management, automation, and coordination for quality. Visibility into code quality becomes critical as a metric to help improve behavior to produce better software and to map teams to the appropriate kinds of projects given experience levels and execution (even as the opportunity becomes available to educate teams to execute more effectively). Service providers and internal staff alike need to "up their game" with regards to software creation, appropriate quality, and defect backlogs. "Technical debt" can only be identified and addressed if it is visible, which can be enabled via appropriate automation tools and process strategies for adoption and analysis to understand the impact.

This IDC Technology Spotlight will discuss the need to improve approaches to software analysis, defect management, security, and metrics to gain business and IT benefits via proactive visibility. Those who don't know history are doomed to repeat it; that adage applies also to ineffective approaches to software creation, quality, and defect mitigation and resolution.

THE CHANGING QUALITY PERSPECTIVE

As businesses become more dependent on technology and on software to deliver core offerings, the job of the CIO is shifting. Budgets are not increasing and yet the business expects IT to be able to deliver new business value more quickly than before in highly complex environments with increased expectations of quality. In addition, there is an expectation that IT become more aware of emerging business objectives and align IT priorities to match. A key challenge is how to enable faster delivery of better quality applications that address business and customer needs. IT and CIOs must move beyond improving IT efficiencies to enable business innovation and provide superior customer experience with continuous deployment and improved software practices. And a basic building block for doing that is code analysis and management.

Lack of visibility into code quality and resulting code that is rife with problems is both more obvious and more debilitating to businesses now because of the visibility and exposure demanded by mobile and other customer-facing applications. This has become a core issue for executives and those at the code creation level of organizations.

Developing and delivering high-quality software releases and new customer-facing applications on time is a growing challenge, particularly for large enterprises that must contend with a complex web of modern technology combined with legacy systems and resources. Organizations must address the needs of the internal business — the extension and availability of enterprise apps to mobile/social/cloud platforms — as well as satisfying client/consumer demand for the availability of innovative applications. Yet how is that possible without providing insight into problems as they are inadvertently created?

What can be done to improve overall software quality to reduce issues and risk while at the same time accelerating release cycles to bring new services and products to market faster? This demands effective and earlier management of software vulnerabilities and defects to enable proactive quality strategies and to cut costs. Yet how can poor existing behaviors be mitigated to best accomplish those goals?

MANAGING AND ENHANCING LARGE-SCALE SOFTWARE ENVIRONMENTS

Challenges in Software Development

As executives seek to evolve business-critical applications, strong coordination of quality management at the earliest software life-cycle phases through to operations can help lead to successful, continuous deployments, and corporate and IT productivity. Organizations must make this transition by observing current challenges and leveraging strategies based on those issues to create the impetus for change.

IDC identifies the following as the top challenges facing software developers and organizations that depend on speedy, innovative software creation for competitive success:

- The business has ever-increasing needs both for upgrades and fixes as well as new customer-facing apps. IT and development teams must shift approaches to code and quality analysis to produce a higher level of confidence in the software delivered (with fewer defects).

- Enable faster time to market – at right cost, quality, and risk with appropriate resource allocation – and shift the view that software development, code analysis, and testing are merely a "cost center" to being a core "business value enabler."
- Limit business risk and technical debt – help ensure the delivery of high-quality "secure" applications that work well "first time" through proactive, iterative code examinations, and common visibility for software practitioners and management.
- Shift from defect detection (often too late) to proactive defect prevention and upfront defect identification with common analysis and a "source of truth" across teams and executives.
- Flexible, contextualized dashboards can enable teams to view data that is relevant to them and can empower individual responsibility and collective collaboration across groups.
- Transparency is key to understanding quality, code completeness, and testing activities that are underway – coordinate with iterative build management and set appropriate thresholds.
- Ever-growing needs for governance and regulatory compliance are factors that affect the market and put pressure on companies to automate code assessment capabilities; this can help to drive workflow improvement, traceability, reporting, and metrics.
- Where there are limited or no unit tests and where there is little idea about whether the current design allows for the quick addition of new features or the ability to make improvements, concern about refactoring can straightjacket organizations and act as a spur to change.
- When bugs are discovered late in the development life cycle causing "red-alert" situations and when releases become "dragons" that QA and customer support have to fight before software can be deployed to customers, that negative impact can also motivate behavioral shifts.
- Leverage code analytics information to provide the opportunity to improve behavior through transparency and iterative change (do not use this information as a "wall of shame" to punish teams or individuals).
- IT, quality, operations, and business teams should unite and use these challenges as a jumping off point and as an impetus to shift poor existing software development habits and behavior. Establish effective software analysis and inspections, create coding rules and boundaries for builds and releases that require certain quality levels, and leverage metrics to improve quality strategy as well as the business outcomes that are reliant on high-performing, innovative software.

Enhancing Application Value and Reducing Software Development Risk

Successful organizations are moving quality analysis up front in the process and making it iterative as part of the overall software development life cycle. Development and test worlds and business approaches as well are evolving toward agile models – continuous value generation/continuous improvement/continuous inspection are essential strategies to improve quality. These are some of the key strategies that IDC is observing for both end-user customers and service providers:

- Traditional approaches to code quality typically fail when testing becomes a checklist item at the end of the development process with the QA team running a tool and going back to development teams with a list of actions they need to take before the software goes to production. This can result in delays and budget over-runs. So what can happen all too frequently is that management signs off to move the project along with no quality gates for development.
- Collecting appropriate metrics with code analysis from projects starting at project inception is critical. Without the right metrics (or any metrics) the code base may be deteriorating

without anyone noticing, or they only notice when the technical debt has reached a certain level where it's too expensive to address, given the time and budget constraints. Collecting code metrics continuously can provide visibility and give the team the advantage of keeping the technical debt of the code base under control.

- Understanding which code to fix is also essential. Frequently, teams start refactoring because they believe the code base is bad in terms of performance, brittleness, instability, difficulty to maintain, and/or to extend. But without the right contextual analysis it is impossible to detect which part of the code base is responsible for the issues encountered. So changes may be applied to the wrong code, or the right code gets refactored in the wrong way, or only part of the problem gets fixed. This is an area where metrics and tools can help by identifying the parts of the code that are causing the problems.

Ensuring Continuous Software Quality Can be the Key to Success

Both end-user companies and services organizations must run software development as a business. Part of doing that is managing the long-range maintainability of software – i.e., what you develop today you need to maintain tomorrow. Being proactive about the architecture and also the quality of what is being created is key to managing long-term expenditures for software maintenance; it is more expensive to keep faulty, poorly structured software alive.

In this context, application maintenance teams also need visibility into software to be able to better maintain code to reduce costs, to enable better quality, and to improve customer responsiveness and ROI. Before any code gets sent to the customers, establishing consistent processes for code analysis can help ensure long-term improvements for companies that must run software development effectively.

SONARSOURCE AND THE SONARQUBE PLATFORM

SonarSource: An Introduction

SonarSource is a Swiss company founded in 2008. The company was born out of a desire to tackle and resolve the growing issues related to software quality, and to bring to market a solution that could track code quality in the software development process. In a market with competitive offerings from a handful of providers, SonarSource's portfolio is differentiated by its roots in open source, its accessibility, and a range of engaging options from a packaging and pricing perspective.

The SonarQube platform was created and taken to market (initially as "Sonar"), with SonarSource releasing its first commercial plug-in for the platform in October 2009. By March 2010, SonarSource began to see both community and enterprise acceptance of the SonarQube platform, which by then was being downloaded more than 2,000 times a month. In May 2010, SonarSource released COBOL and Visual Basic plug-ins, followed a few months later by a SQALE plug-in, the C# plug-in (June 2011), and the PL/SQL plug-in (September 2011).

The company's key intent is to bring affordable and intuitive quality solutions and analytics to developers that also provide capabilities for broad, distributed use. Today, SonarSource has approximately 350 customers worldwide, including Deutsche Bank, Bank of America, Michelin, Telefónica, BNP Paribas, Thales, and EADs. The SonarQube platform is used by around 300 customers, with between 30,000 and 40,000 installations. The company has seen explosive growth over the past couple of years, and it now employs more than 30 staff, up from 20 people just 12 months ago.

SonarQube Platform

SonarQube is a continuous quality analysis platform running as a web server that tracks metrics by analyzing code and code structure. SonarQube is an open source platform and was developed with a main objective in mind: "to make code quality management accessible to everyone with minimal effort." The SonarQube ecosystem is made up of the SonarQube platform and a suite of plug-ins hosted on public infrastructure.

SonarQube essentially provides code analyzers, dashboards, reporting tools, issues tracking, and TimeMachine as core functionality, but it also has a plug-in mechanism enabling the community to extend the functionality (currently there are more than 60 plug-ins available).

SonarQube can become a coordinating hub for source code quality as it does not need to be restricted to developers or the technically savvy but can provide helpful information broadly to project managers, technical leads, IT, and even business leadership within an organization with customizable dashboards. SonarQube's architecture and plug-ins (such as SQALE) and the opportunity to manage and track technical debt can provide key information to managers and the business to proactively address defects iteratively throughout the software life cycle. They provide a high-level overview of the project as it relates to quality and cost, and help address risk.

SonarQube provides more than mere high-level indicators about software health. Since developers are provided with information at a granular code level, SonarQube enables those building software to find and drill down to where code problems exist. SonarSource's product portfolio enables feedback and impact analysis on areas of software change, and provides feedback on how to improve development approaches. The products also provide coordination with build management (with Jenkins support) to help enable continuous integration for deployment.

SonarQube integrates with tools such as FindBugs, Checkstyle, PMD, FXCop, and CppCheck out-of-the-box, or with provided plug-ins. It then can act as a central hub for code analysis tools, thereby providing historical insight and trend analysis for multiple projects. The all-in-one-place analytics and reports are a plus even if organizations choose not to act on the reports that are sent. Having access to historical data about code complexity and the number of issues tools spotted in the code can provide visibility into whether software is being built effectively or if the approach needs to be changed.

In terms of languages, SonarQube supports analysis of Java in the core, but also of more than 20 languages such as COBOL, C++, PL/SQL, and C# through plug-ins (open source or commercial) as the reporting engine is language agnostic.

SonarQube enables organizations to cover quality on seven axes and to report on:

- Duplicated code
- Coding standards
- Coverage by unit tests
- Complex code
- Potential bugs
- Comments
- Design and architecture

The most current version also has improved the evaluation of software quality attributes and does a better job of scoping technical debt and isolating the problems that create technical debt, according to references. SonarQube is configurable and can give a "grade" to the code (from A to E) and can identify what it would typically cost in terms of effort and the type of effort needed to improve the software. For instance, the code might have a low rating due to lack of unit testing, or due to high amounts of duplicate code or security violations. SonarQube will show what the violations are and will estimate costs to address them. Users can change the effort estimations and they'll be calculated in, which is helpful. This release can also make comparative team assessments across projects and parts of the organization as needed.

While SonarQube can be used tactically for one-off audits, it can be leveraged more strategically as a shared, common source of information for quality analysis as was just described, to help support a continuous improvement strategy for code quality.

A wide variety of organizations use SonarQube given the range of offerings of the portfolio (from the company's free "open source" option to the enterprise, site-wide "Ultimate" license version). However, a target at the high end is large and very large corporations with enterprise, distributed development teams, and partner coordination. One-person teams can use the open source version obviously and this can act as an on-ramp to adoption. But once an organization has crossed the line in terms of the number of development projects and users there is a need to move to the commercial enterprise solution. IDC spoke with three enterprise customer references for SonarQube that mainly began with open source adoption that gained a foothold and then evolved through to deployments of 1,000+ users.

Typical Size of Implementations and Benefits

Large international organizations can be running analysis of more than 10,000 projects and analyzing 650–700 million lines of codes in 14 languages with 8,000 visits a day on the website.

One of the customers IDC spoke to is tracking 1,200 projects with 160 million lines of code being scanned through SonarQube, plus a further 300 projects with another 160 million lines of code being scanned.

Another customer ramped up from two dozen projects to over 2,230 projects now for registered users with many more browsing dashboards anonymously.

Why SonarQube?

The SonarQube references IDC spoke to needed a way to measure and enforce software and code quality metrics. A key goal was to have quantitative measurements of code quality and to analyze those metrics to come up with a set of benchmark measurements — essentially to utilize the platform to encourage good practices (and to discourage bad ones).

When evaluating competitive products they looked for the following: the features for quality analysis provided (such as dead code analysis, impact analysis, cross-platform analysis); languages supported (SonarSource supports 20+); the flexibility of code review; and dashboard offerings and reporting analytics. Service organizations also evaluated the tools based on commercial constraints and engagement constraints.

The advantages of SonarQube typically include its overall ease of use, requiring less time to learn and to adopt. Packaging options with SonarQube were also a benefit for both end users and service providers — enterprise licenses with "no strings attached" are a help to end users with

dynamic distribution needs and for service providers, providing the freedom to be able to leverage SonarQube flexibly as part of engagements.

SonarQube is still evolving its support for impact analysis, though in the meantime some customer references have created workarounds to address the issue.

The Benefits of Using SonarQube

The functionality that SonarQube customers described as being most significant in solving their core problems included the following:

- Code and quality visibility to be able to see where the hot spots are in an application to proactively include app quality "up front" as an initial and iterative part of the development process dashboards where users can pick and choose the metrics to contextualize and customize reporting.
- The ability to consolidate metrics at varying levels with different views — at the customer level, at the developer level, and/or the business unit level — and to roll them up into "one source of truth"; a single portal/single point where everyone can go and see what they need to know.
- Managers/directors can customize and use SonarQube to measure the performance of individual groups — service providers can customize the dashboards for each customer organization where they are working to address different kinds of needs and standards. They can also augment existing rules and integrate the results because SonarQube gives that level of flexibility. At the same time, organizations must be careful not to use SonarQube as a "bludgeon" to force "good behavior" — successful companies have leveraged the information to encourage better practices rather than to establish a "wall of shame" to punish individuals for poor coding behavior. This means using SonarQube as "diagnostic metrics" rather than "outcome based" metrics which can better drive success.
- Overall, these capabilities have enabled customers to manage and alleviate technical debt through a cost-effective solution that can scale to an enterprise level and be broadly distributed. SonarQube helps organizations to benchmark code quality and to understand how well their organizations are doing and how they can and have improved over time via information that is both qualitative and quantitative.

Broader Implications and Opportunities With the Use of SonarQube

Beyond scanning, an interesting outcome seen at some organizations is that individuals have started to act on the information made available by SonarQube to change and improve their behavior with regards to quality code creation. A number of teams have become engaged — and even excited — about going and checking factors such as test code coverage and have been able to improve it dramatically. Directors have used SonarQube information about rules violations to educate teams about shifting poor habits to improve code creation. So a key outcome of SonarQube use at those organizations has been the opportunity to encourage and even to drive the right behavior.

Some organizations have helped this along by mandating the integration of metrics with the build and release process. This means that certain standards have to be met to allow the build and release process to move forward. While you can't force people to look at poor outcomes generally, when project deadlines and software release dates slip, there is an opportunity to work with engineering managers with specific data points and benchmarking to understand the impact of prior work. (The organizations doing this are judicious in the standards they put in place for stopping the build and release process.)

Overall, a tangible benefit has been the doubling of test coverage seen by some of the testing component teams using SonarQube. Some organizations have observed test team coverage increases of 4–5 times for unit tests, as well an improvement of the thoroughness and rigor of those completed unit tests with the use of SonarQube. This, in turn, helps drive accurate benchmarks for team progress along with the delivery of higher quality code.

In short, SonarQube can provide a single point for basic code analysis where users from developers to managers can go and see what they need to know to help improve code quality, and potentially to integrate with code review tools to include code review metrics in the dashboard.

The Challenges for SonarQube

One of the key challenges for SonarSource is convincing organizations about the ROI benefits of implementing and using a code analysis solution. Part of the issue in this context is that it's challenging for organizations to comprehend long-term benefits at a time when many companies are seeking quick, iterative deployments. So it is important to understand the long-term benefits across the lifetime of the code, rather than merely the short-range advantages (which are significant in their own way but are not strategic). The evolution of the product to address functional capabilities such as impact analysis and cross-platform analysis are also areas where SonarSource needs to focus, according to customers. Integration with code review tools was also mentioned as an area of future focus.

SonarSource's small size can be a barrier to adoption for enterprise deployment decisions. But the fact that the base product is open source has typically jump started adoption for initial usage, and provides some reassurance for enterprise use of the commercial product as well. The enterprise customers that have demonstrated adoption of SonarQube with bigger deployments also serve to reassure those with concerns about the ability of the product portfolio to be adopted broadly.

CONCLUSION: EVOLVING AND IMPROVING CODE QUALITY ACROSS THE LIFE CYCLE

Now, more than ever, software drives competitive advantage and corporate success. As companies increasingly need to improve quality and as the consequences of poor approaches to software development are visible and can be deeply damaging to revenue generation and to customer and prospect engagement, we see an urgent need for improvements in behavior with regards to code quality. Just as the "unexamined life is not worth living," so is unexamined code not worth deploying. Neither companies nor the customers and prospects they are seeking to engage can afford to continue with ineffective quality approaches.

Continuous code inspection with the ability to customize rules can provide a good vehicle to empower engineering managers with diagnostic metrics (and with judicious, careful usage, outcome metrics). Visibility into what's occurring is dispassionate and helpful to enabling behavioral change to help drive quality improvements for development teams, placing a higher onus on better hygiene.

A single portal such as that provided by SonarSource with SonarQube – with the ability to automate data gathering – is not just about quality per se but is about enabling more thorough testing. Visibility into code quality provides the basis for effective decision making. Products like this can help organizations examine and understand software development through a single hub to start managing software development as a business.

Many organizations have become too accepting of poor quality software development and readily – or reluctantly – agree to the delivery of software products that are late, over budget, and rife with defects. When used properly, automated code analysis tools can enable a starting point to treat software development as a real business. Benefits can include product releases that are more stable and enhance trust and confidence in development teams and partner service organizations (for those with outsourced development).

As we conclude, it is important to underscore that while these types of products can help to measure the quality of software projects they should not be used as "blaming tools." Pointing the finger will not lead to effective team collaboration and better execution. The focus for quality metrics must be about improving the code quality to drive positive action to improve overall team collaboration and software development with a goal to improve corporate execution as a whole. It is the whole team's responsibility to detect and correct code quality problems, just as it is to the team's benefit and to the benefit of the organization when software succeeds and drives business innovation and strong execution.

In short, code analysis products can help organizations analyze the situation with regards to code development, take action, and quantify improvements. IDC recommends evaluation and adoption of these kinds of automated approaches along with the organizational and process change necessary to enable effective adoption and to improve software quality strategies.

About IDC

International Data Corporation (IDC) is the premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications and consumer technology markets. IDC helps IT professionals, business executives, and the investment community make fact-based decisions on technology purchases and business strategy. More than 1,100 IDC analysts provide global, regional, and local expertise on technology and industry opportunities and trends in over 110 countries worldwide. For 50 years, IDC has provided strategic insights to help our clients achieve their key business objectives. IDC is a subsidiary of IDG, the world's leading technology media, research, and events company.

Global Headquarters

5 Speen Street
Framingham, MA 01701
USA
508.872.8200
Twitter: @IDC
idc-insights-community.com
www.idc.com
