CONTINUOUS INSPECTION

A Paradigm Shift in Software Quality Management

By Olivier Gaudin,
CEO & Co-Founder of SonarSource SA

CONTENTS

INTRODUCTION

Software Quality is a matter of increasing concern for every commercial enterprise because of the escalating role software plays in running business-critical systems. Software quality consists of both external and internal quality. External, or functional, quality describes how well the software matches its defined functional requirements – does it perform as intended? Internal quality describes key internal attributes of the code, such as robustness, standards-compliance, and maintainability. Industry statistics show that on average, 80% of the lifetime cost of a software product is spent on maintenance, and that maintenance costs have a high variability depending on internal quality. This means that the level of maintainability a software product has today will determine the level of its cost liability tomorrow.

Traditional approaches to code quality control involve so-called punctual audits or quality gates, which are periodic audits of the source code. These audits are usually performed by external auditors during the "last mile" of the development process – during or after the functional tests. By their nature, punctual audits can lead to disruptions in the development cycle because they result in changes to "completed" software. In the best case, this quality control approach leads to delays and rework. In the worst case, it leads to the release of poor quality software. In either case, the traditional approach fosters the perception that building quality software is overly complex and expensive.

There is an urgent need for a newer model, one that emphasizes quality throughout the development cycle, and has shorter feedback loops to ensure rapid resolution of internal quality issues; in short, a model that builds in quality from the start, rather than considering it after the fact.

Continuous Inspection is a holistic, fully-realized process designed to make internal code quality an integral part of the software development life cycle. By raising its visibility for all stakeholders throughout the life cycle, Continuous Inspection enables enterprises to embrace code quality whole-heartedly. Supported by SonarSource, the Continuous Inspection paradigm is highly effective, and has been proven to work in the real world at organizations ranging from small companies to Fortune 100 businesses, across all industries. This paper details the key challenges in code quality management. It then introduces the Continuous Inspection paradigm and illustrates how it addresses those challenges, supporting thousands of enterprises in improving their software quality.

KEY CHALLENGES IN CODE QUALITY MANAGEMENT

Punctual audits occur, by design, at specified intervals and not continuously. This approach to code quality management has four major types of shortcomings, which will be detailed in this section.

Too Little, Too Late

Punctual audits identify two kinds of improvements: cosmetic and structural changes. Whereas cosmetic changes require minor modifications, structural changes may include major software re-engineering. While such changes may be needed, action plans resulting from punctual audits are defined too late in the process to do anything but disrupt the development cycle; either the software release date needs to be extended to include the software re-engineering, or worse the software will be pushed to production with sub-par quality, and therefore have decreased maintainability and adaptability when new business requirements arise.

Pushback from Development Teams

Developers tend to push back on action plans generated from punctual audits, because they:

- √ Are generated outside the team, and are seen as a new constraint in daily work
- √ Are subjective; findings rely on the judgement of the auditors rather than on objective measures
- √ Miss contextual and historical information, and are therefore seen as irrelevant
- √ Are invalidated by on-going changes and quickly become out-dated
- √ Do not involve developers and other stakeholders in the review & audit process
- √ Intervene too late in the process; by the time a feature is audited, developers need to “relearn” the code to address a finding

Lack of Process Ownership

There is a clear lack of ownership of the quality process within the organization. Auditors cannot own the process, because they neither own the code nor have control over issue resolution. Similarly, the command-and-control nature of the model prevents the development team from owning the process because it is not involved in the reviews. Thus you have two disconnected groups which are both responsible for quality while neither of them is accountable.

Heterogeneous Requirements

Traditional approaches which measure software on absolute values, such as total number of issues found during a quality gate, force evaluators to measure each application against different requirements depending on its origin. For instance, a legacy project may not be held to the same high quality standard expected of a greenfield project, and in-house development might be judged differently than outsourced code. This is due to the fact that you still need to allow software to ship to production, and requiring each project to reach the same absolute values for quality thresholds before release is often impractical. Using such absolute values, it is almost impossible to work out common requirements for all applications, and therefore difficult to adopt good practices across the board.

The challenges in code quality management outlined above combine to create an often self-fulfilling perception that “creating quality software is expensive.” Under the traditional approach, quality management comes too late in the process to be effective, and disrupts the development cycle, causing unexpected delays, unplanned rework or missing features. Compounding the problem, development teams push back on the quality assessments, which they see as reducing the team’s productivity and collaboration. Furthermore, enterprises do not gain long-term improvements in overall quality, because the traditional approach does not take into account the need to educate developers. As a result, the same or similar quality issues arise repeatedly during the project lifecycle, and perhaps even from project to project as developers are re-assigned within a company.

ABOUT CONTINUOUS INSPECTION

We, the founders of SonarSource, are quite familiar with the shortcomings of the traditional model, having worked for many years within its confines. But with the rise of Continuous Integration, we envisaged that a different model was possible. Just as continuously integrating the changes from multiple developers prevents integration headaches, we realized that continuously applying the quality gate standards prevents the problems of the punctual audit model.

Continuous Inspection is a new paradigm in code quality management designed to make internal software quality an integral part of the software development lifecycle. It is a holistic, fully-realized process which increases both the internal software quality of a project, and the visibility of software quality for all stakeholders. Continuous Inspection provides continuous code quality management, and drastically raises the ROI of a development project. The key concept in Continuous Inspection is finding problems early–when fixing them is still cheap and easy. Under this model, automated code audits are performed on a daily basis and made available within an organization. These objective, automated audits analyze a project's code along multiple maintainability axes, test it for bugs, and compare it to team coding standards. Audits are completed by tools that detect those issues directly in the developer's environment, much like the spell checker in Microsoft Word. Team members are alerted as soon as new issues are found so they can be addressed as quickly as possible–while the code is still fresh in the developer's mind. The timeliness of these alerts has the added benefit of training coders out of bad habits and leading them to good ones.

Continuous Inspection enjoys a grass-roots adoption among development teams, because its collaborative nature leads to a truly collective code ownership, and helps teams deliver better software. With its small, rapid cycles of issue identification and treatment, it has been proven to increase development team efficiency, and raise the longevity of applications by fostering the development of higher-quality code. The most important aspects of Continuous Inspection can be summed up in ten principles.

The 10 principles of Continuous Inspection:

1. All stakeholders in the development process – not just developers or managers - must have ready access to meaningful data about software quality.
2. Managing software quality must be everyone's concern from the beginning of development, but is the development team's ultimate responsibility.
3. Software Quality must be part of the development process, meaning that meeting quality standards is one of the hard requirements to be able to declare development complete.
4. Software Quality requirements must be objective and not require a subjective pass / fail decision.
5. As much as possible, software quality requirements must be common to all software products, regardless of their specifics.
6. Software Quality data must be up to date, i.e. measured on the very latest version of the code.
7. Software products must be continuously inspected, so that errors are found quickly, when they are easy to correct. Developers must be able to spot new quality flaws as soon as they are introduced, i.e. within the IDE as they write code, similar to how spell checkers highlight misspellings.
8. Whether through push or pull, stakeholders must be alerted when new quality flaws are injected, whether that's by sending email, breaking the build or by other methods. Injection of new issues must be tracked, enabling teams to make quick, informed decisions about quality.
9. Software quality data must be available both as absolute (on all code) and differential (new code only) values so that the development team can be in full control of the incoming flow of issues.
10. All new issues and existing critical issues must be assigned a clear path and timeline for resolution.

The Continuous Inspection paradigm is highly effective, and has been proven to work in the real world for companies ranging from offshore software factories to Fortune 100 businesses. These companies have successfully used the Continuous Inspection model to manage internal software quality on projects of all sizes. One Fortune 100 company with more than 20,000 developers uses it to manage more than 600 million lines of code, in an environment where more than 5,000 applications are analyzed each day. In all cases, Continuous Inspection has helped these companies significantly improve software quality and stability, typically saving millions of dollars that would otherwise be spent on root cause analysis and crisis management.

Continuous Inspection, the new software quality paradigm, addresses and resolves the key challenges in code quality management:

Key Challenges	Solution Offered by Continuous Inspection
Too little, too late	√ The team receives continuous feedback on quality, including against a set of quality requirements √ A clear, updated picture of quality evolution over time is available, including comparisons among versions √ Teams can track issues from introduction, and provide feedback √ Stakeholders are notified of quality flaw injection as soon as it happens √ The Quality Gate is executed daily √ The final Quality Gate iteration becomes a non-event √ Continuous education of developers leads to a virtuous circle of improvement
Pushback from Development Teams	√ Quality Action Plans are generated directly within the team and integrated in the development process √ Software Quality is part of the development process √ Reviews include contextual and historical information including different versions and various changes made to the software √ Stakeholders can access meaningful information about their software quality in real time √ Development teams receive information about quality flaws as soon as they are added (via email, visible in the IDE, …) enabling issues to be fixed immediately √ Teams gain the ability to develop better software
Lack of Process Ownership	√ Ownership of code quality belongs to development team √ Software quality is embedded in the development process and becomes everyone's responsibility √ Access to the software quality tool is provided throughout the organization to every stakeholder √ Quality requirements can be shared, updated and reviewed among team members and across the organization √ Quality judgements are made in an automated fashion based on objective criteria that were published to the organization beforehand. √ Reporting clearly shows software maintainability, and is immediately understandable without the need for external consultants √ Ongoing education of developers leads to significant software quality improvements in the long run
Heterogeneous requirements	√ Teams have the ability to measure Software Quality on new and changed code as well as on the entire code base √ Teams can track injection of new issues

CONCLUSION

Designed and realized by SonarSource, Continuous Inspection of internal quality is a holistic, fully-realized process designed to make code quality an integral part of the software development life cycle and raise its visibility for all stakeholders throughout the life cycle. The Continuous Inspection paradigm is highly effective, and has been proven to work in the real world, across all industries at organizations ranging from one-man shops to Fortune 100 businesses.

Continuous Inspection is a new model for software quality, one that incorporates shorter feedback loops to ensure rapid resolution of quality issues. In short, it is a model that builds in quality from the start, rather than considering it after the fact. With Continuous Inspection, quality flaws are found – and corrected – very early in the development process, while the impacts are small and manageable. Some problems will be caught in the developer's IDE, before ever being checked in. The rest will be reported within a day, while the code is still fresh in the developer's mind and the fix is still cheap and easy. This quick feedback cycle has the double benefit of improving quality and educating developers.

Continuous Inspection fits well into both agile and waterfall development environments, and addresses the shortcomings of the traditional approach. Continuous Inspection offers improved quality with minimum disruption to the development process and time line.

Continuous Inspection fosters stronger team collaboration and productivity, and engenders a strong sense of team ownership of code quality because the quality process, like the code itself, is owned by the team. Where punctual audits are derided for being quickly out of date and ignoring the incremental nature of software development, Continuous Inspection provides an immediacy and a clear picture of software quality over time.

With Continuous Inspection, the perceived cost of quality is zero because quality is blended so seamlessly into the development process itself. With Continuous Inspection, enterprises can finally embrace code quality whole-heartedly and maximize their software ROI.

SonarSource SA
Carrefour de Rive 2
1207 Geneva, Switzerland

Tel: +41 (0)22 510 2424
Email: contact@sonarsource.com
Web: www.sonarsource.com