A Forrester Consulting Thought Leadership Paper Commissioned By Klocwork

The Value And Importance Of Code Reviews

It's Time To Exploit Modern Technology For Improved Code Reviews

March 29, 2010

FORRESTER

Headquarters | Forrester Research, Inc.
400 Technology Square, Cambridge, MA 02139 USA
Tel: +1 617.613.6000 | Fax: +1 617.613.5000 | www.forrester.com

Table Of Contents

Executive Summary

Code reviews have been a fundamental part of the process of development since the birth of software engineering. However, the practice of software development has changed greatly since its creation in the 20th century. Has the process of code reviews kept up with the changes to software development? In January 2010, Klocwork commissioned Forrester Consulting to answer that question. The resulting study finds that the use of code reviews is widespread and considered to be of high value, but the practices and processes exploited to undertake code reviews have not significantly changed since their inception.

Key Findings

Forrester's study yielded six key findings:

- **Code reviews are valuable, but approaches tend not to be formal.** Even though software is a key asset, the process to review that code is often informal and ad hoc in nature. The result is poor visibility of the review process for management and reduced effectiveness of those reviews themselves.
- **Modern development is increasingly complex, and code reviews are not keeping pace.** Teams are spread out, and many organizations contribute to the development of a single software product. Add to this the increased amount of software complexity, and the result is a compounded organizational and technological challenge. Code reviews, a key part in sharing best practices and knowledge, have not kept up with these changes and continue to be mostly executed in one location with infrequent use of tools and supporting infrastructure.
- **Code reviews don't always include the right people.** The advent of more complex architectures and increased cross-functional dependencies increase the importance of reviewing code in a much larger context. Architecture and QA are often the functions that understand the bigger picture but are often not included in the code reviews, with teams instead relying on local software engineers to provide those insights.
- **Tools don't play a significant role in code reviews.** Code coverage and static code analysis are just two of the tools that provide valuable insights into code structure and behavior, but these tools currently play a small part in code reviews overall.
- **Process and time limits inhibit code reviews.** Like other parts of the development process, code reviews suffer from a lack of time to devote to them. This is made particularly acute because both the planning of development completion and the need for developers with certain skills make preparation for this activity difficult.
- **Social media plays an increasingly important role in technology research.** IT professionals surveyed are heavy users of social media for technical support and social interactions, but they don't apply this technology to the "social" activities of code reviews. The result is that code reviews tend to rely on traditional approaches both to document findings and to resolve questions and issues.

Code Review Definition

Code review is a systematic examination of computer source code intended to find and fix mistakes overlooked by the original developer.

When computer machine time was at a premium, code reviews were used to ensure that code was of a high quality before being processed. As the cost of processor time came down, the timing and importance of code reviews changed, and the process became a more flexible set of activities executed at certain points during the process. Code is no longer reviewed prior to being loaded onto the computer. However, not much else has changed; it's still the case that:

- **Code is collected and reviewed by another developer.** The process of code reviews focuses on one developer taking another developer's work and reviewing or "desk-checking" that code for omissions and defects.
- **Code is reviewed with a view to requirements.** The context of the review is the requirements, and thus the reviewer will follow the logic of the requirement within the code, ensuring that the code conforms to the needs expressed in the specification. To highlight the code and its relationship to the requirement, many developers include comment statements that map the requirement to the actual code.
- **Code standards and design are considered.** Other than functioning as specified, code reviews also consider the standards employed by the organization. This is often the only step within the development process that manages the code for consistency and maintainability.

Everything Changes, But Code Reviews Stay The Same

In direct correlation with Moore's Law, the volume of computer code has increased exponentially over the past 40 years.[1] That volume includes not only an increase in the number of lines of code but also in the power provided by each statement within those programs. Increased focus on object-oriented and service-oriented architectures also adds to the level of complexity, with program paths taking increasingly complicated routes, through both developer- and vendor-created source code. The result is that proofing code has become more and more difficult.

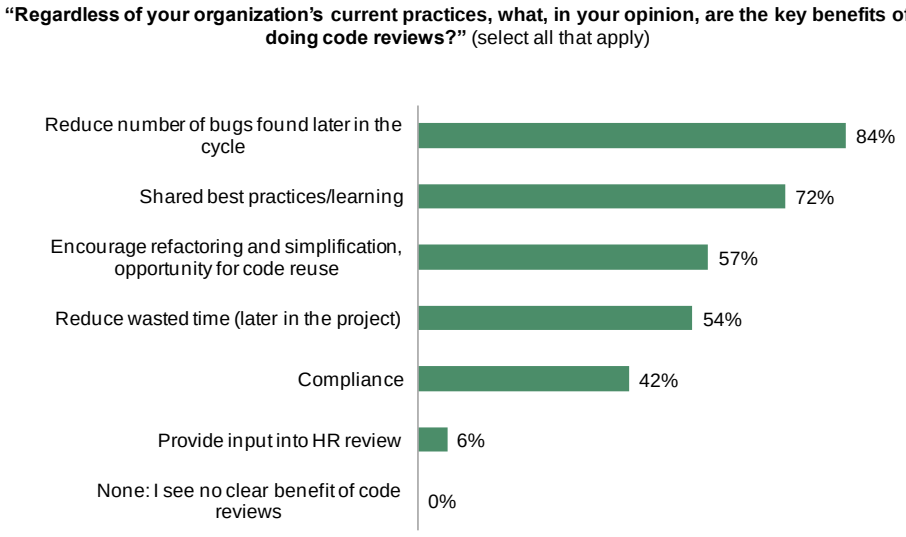
In direct response to increases in complexity, software development tools have improved. Integrated development environments (IDEs) provide developers with a dazzling array of functions for editing, reviewing, and debugging source code. Code overage and static analysis tools provide insight into execution paths and function usage. And testing tools provide functions to help developers automate the rigors required to test complex functionality. To better understand the state of code review practices and the extent to which they have kept pace with the changes in other parts of development, Klocwork commissioned Forrester Consulting to conduct a study of 159 IT professionals directly involved in code reviews. Two-thirds of these IT professionals were involved in commercial software for sale or included in a physical product. The remaining third were involved in Web development or enterprise IT. More information on the methodology of the study can be found in Appendix A.

Code Reviews Add Value

All IT professionals believed that code reviews added value to the final deliverable (see Figure 1). They cited finding bugs and shared learning as the most valuable benefits attributed to the practice. A lack of a clear relationship between code reviews and HR reviews was pronounced, with only 6% of people surveyed seeing input into HR review as a benefit of code reviews. This is interesting to note, considering that one of the key skills related to developers' professional growth is writing code. IT professionals believed that code reviews improve projects in the following ways:

- **Bugs are uncovered earlier in the life cycle.** By providing a clear inspection of the source code, IT professionals believe that bugs will be uncovered earlier in the development process. This reduces rework for both development and test resources.
- **Best practices are shared.** Although the result of code reviews did not feature in formal HR processes, IT professionals believe that code reviews provide some value in sharing best practices and highlighting new techniques within the code.
- **Refactoring and code simplification are encouraged.** The Agile practice of refactoring has become popular in both Agile and non-Agile processes, with the desire to continually improve the design through a process of incremental refinement.[2] By providing additional sets of eyes on the code, it's possible to identify areas that would benefit from refactoring.

Figure 1
Perceived Code Review Benefits



Base: 159 IT professionals who participate in their organization's software development and code reviews

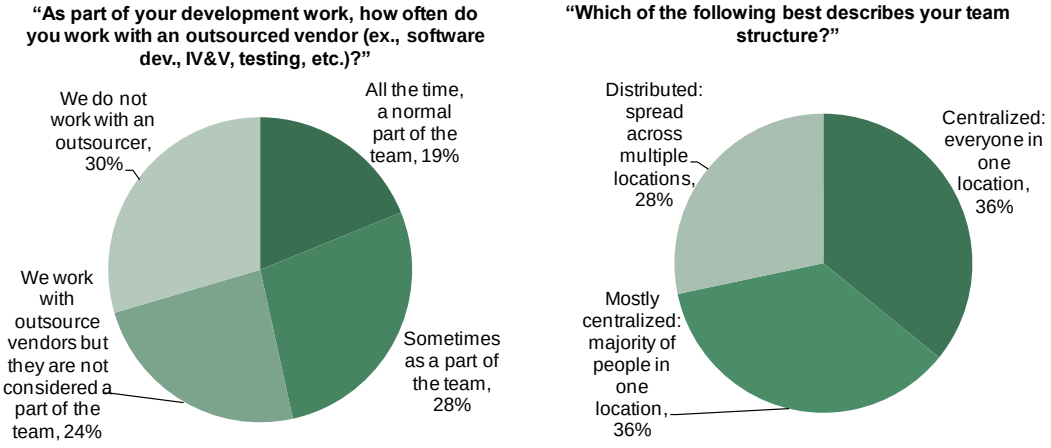
Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Code Review Practices Don't Match Up To IT Organizational Realities

The study highlighted the realities of modern software developments, including the use of outsourced vendors and some geographical distribution of the reviewing team (see Figure 2). Many firms look to flexible working structures to provide business agility, encouraging the use of the right skills at the right time without limitations of geographical location or organizational alliance. However, as respondents described current code review practices, they revealed an approach that does not consider these same flexible working practices. In fact the following still holds true for code reviews:

- **The majority of code reviews are done in person.** Sixty percent of IT professionals undertake code reviews in person (see Figure 3). This implies that code reviews are primarily drawing on local resources rather than prioritizing use of resources with the right skills or experience. This will become increasingly problematic as organizations adopt more and more flexible organizational structures.
- **Technology is not pervasive in the review process.** Although only 18% of IT professionals surveyed don't use any tools in their review process, the use of tools overall is inconsistent, with only one-third using static analysis tools and even fewer using other tools. Code analysis tools focus on the simple problems for code associated with structure and behavior, thus allowing code reviews to focus on more complex architectural and style issues.
- **QA and architects are not always involved in the process.** Less than half of IT professionals surveyed reported regular participation of architects in their code reviews, and roughly 70% of those surveyed regularly perform code reviews without QA represented (see Figure 4). Modern software increasingly exhibits the characteristics of an ecology rather than an engineering system, with small changes in one program having huge ramifications in other modules. QA and architecture often have a view of the whole system and an appreciation of the dependencies between systems and the impact of change.

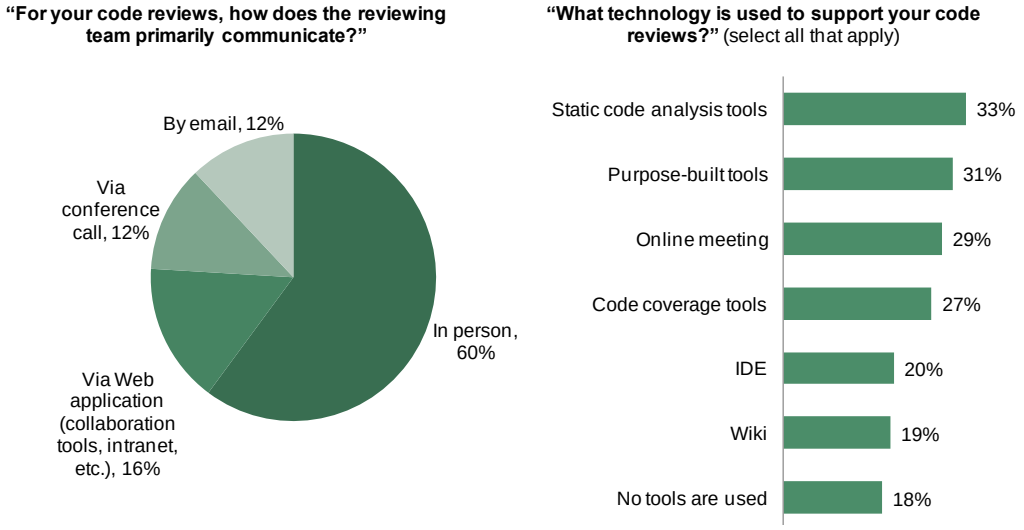
Figure 2
Flexible Working Structures Prevail



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

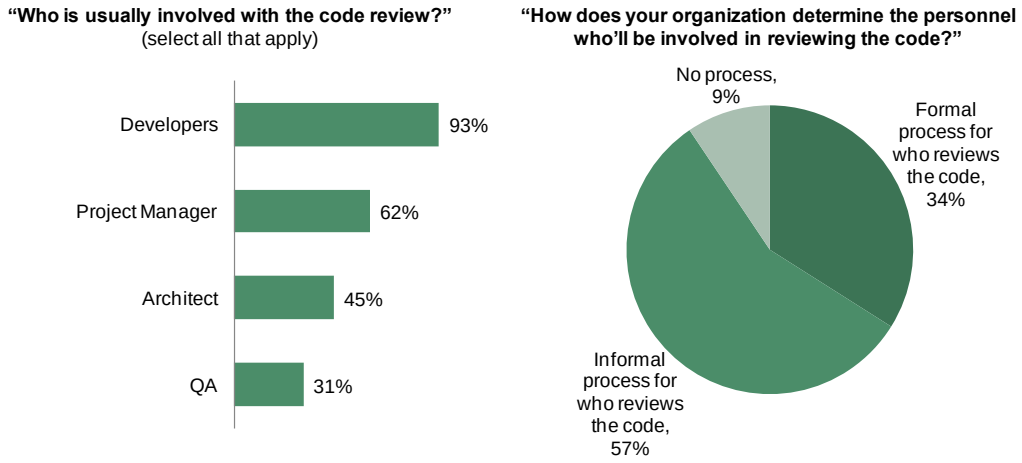
Figure 3
Communication Strategy And Tools Used



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Figure 4
Process And Allocation of Resources For Code Reviews



Base: 159 IT professionals who participate in their organization's software development and code reviews

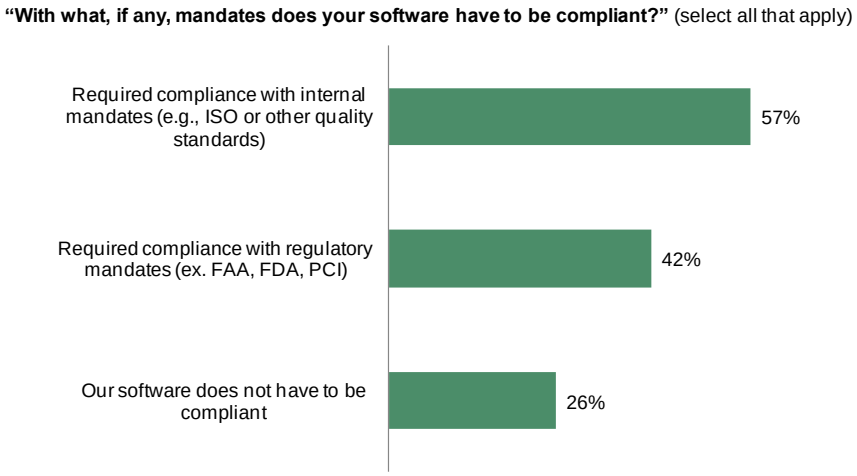
Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Despite Proliferation Of Mandates, Software Development Remains Decidedly Informal

As the importance of software increases, the impact of problems becomes more visible to the business and the industry. Compliance mandates are introduced to offset this risk (see Figure 5). The nature of these rules varies, ranging from introducing clear separation between functions to ensuring that tests are executed prior to programs going live. The one thing they all have in common is a clear policy and a requirement of reviewing the policy in the context of the program. Code reviews often become the place where compliance mandates are reviewed, providing a way for another software engineer to sign off that the code complies. Although often driven by both internal and external mandates, code reviews continue to be:

- **Reviewed on an ad hoc basis.** Thirty percent of the IT professionals don't have a formal process for determining when a code review happens, allowing developers to select the timing based on their needs (see Figure 6). However, more than half state that code can't go live without a code review happening. Without a clear timing and standard process, planning challenges are to be expected, and often.
- **Staffed in an informal manner.** Fifty-seven percent of IT professionals stated that there is no formal process for selecting staff for code reviews, allowing developers to select the available resources as appropriate. The risk of selection by developers is that they may likely pick their friends or people who are physically close to them rather than selecting the software engineer with the right skills. From an audit perspective, allowing the developer to select the reviewer can also inadvertently allow inappropriate code to slip into the software.

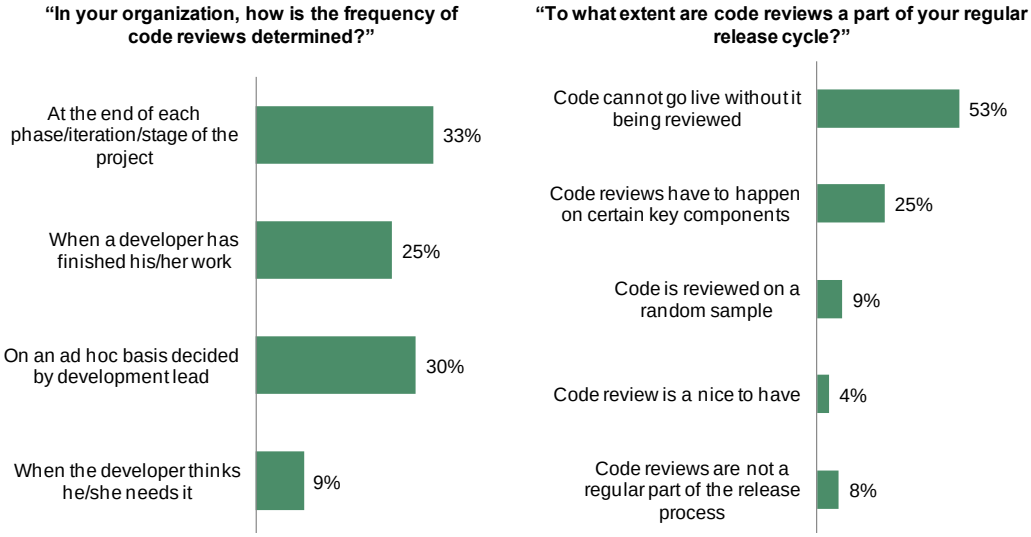
Figure 5
Compliance Mandates



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Figure 6
Code Review Process



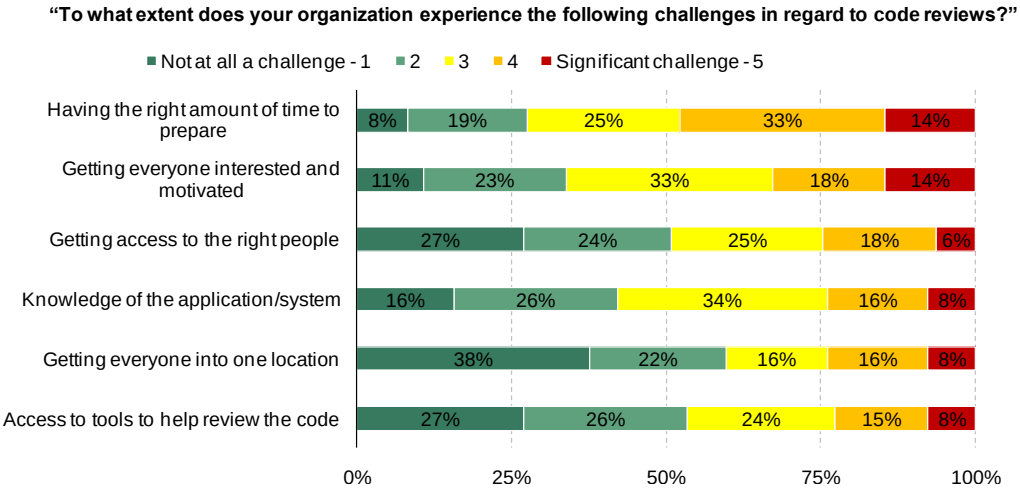
Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Bringing Code Reviews Of Age: The Right People, The Right Time, And Beyond

IT professionals describe the top challenges to code reviews as: No. 1, having the right amount of time; No. 2 and No. 3, getting the right people involved and motivated; and No. 4, knowledge of the system being reviewed (see Figure 7). At face value, improving code requires improved people practices, ensuring that the right people with the right skills are involved and want to be. However, by taking a step back to look at the broader approach to code reviews, it's possible to solve these issues and improve code reviews with a combination of development tools, process, and social tools.

Figure 7
Key Challenges: Time And Motivation



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Make Reviews Easier By Providing More Data

Knowledge of the application system is considered a significant challenge by many IT professionals surveyed. There are two ways of increasing knowledge of the application: 1) gaining years of experience in the code base, and 2) employing practices that quickly provide insights. There is no quick fix for gaining experience, but by increasing informational output about the code, IT professionals can increase their knowledge of the application. Without clear information, it's hard to make informed decisions about the quality of the code. Building a good understanding of the code requires careful, often painstaking, analysis of the code and its execution. With the application of certain key technologies, it's possible to jump-start this process, providing the developer with insights into the code and its execution. IT professionals should:

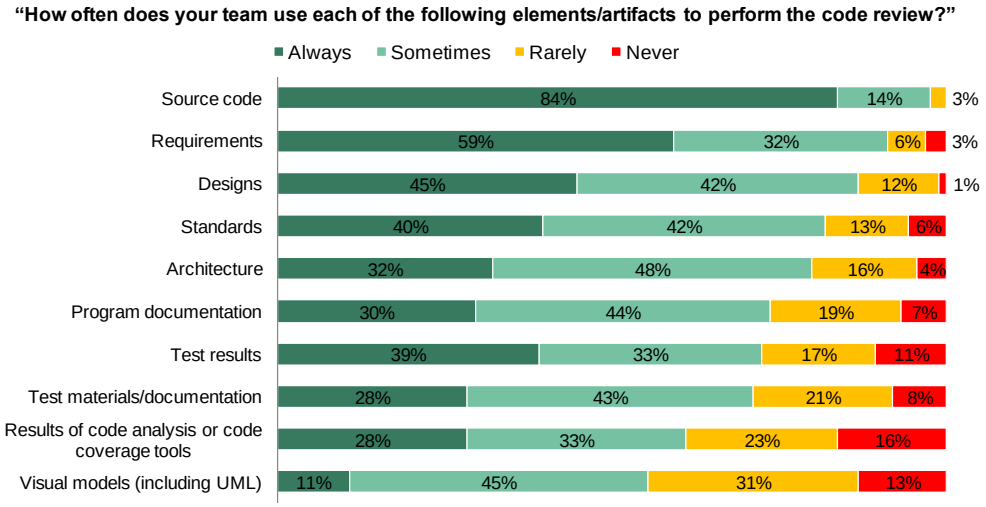
- **Exploit code coverage and code analysis tools.** Developers surveyed reported the results of code coverage and analysis tools among the least often used elements in their code reviews, with 39% rarely or never using these (see Figure 8). Rather than looking at static code and building a working execution in their heads, developers should exploit tools that provide analysis of the code structure and execution. By reviewing these metrics, it's possible to understand quickly how the code works and, often more importantly, which bits of the code are significant. This allows the reviewer to prioritize the code, evaluating the elements that are used most. Static analysis tools add additional detail to structure and pattern usage.

- **Avoid "dumb" mistakes with automation.** Exploiting code analysis tools makes it possible to focus scarce development resources on important discussions associated with style, architecture, and the reuse of design patterns, leaving simpler code corrections to these automated tools.

- **Include testing materials.** Proof of the pudding is in the eating: By including the results of tests, code reviewers can quickly see the effectiveness of the code, yet test results have not seen a high frequency of use with respondents surveyed. Test coverage tools add another dimension to the test results, providing detailed analysis of code execution and paths through the code base. This information provides valuable insights into how the code actually works and helps the reviewer gain an understanding of the application.

- **Use designs and architectural materials as appropriate.** Lack of knowledge around the existing systems can be a major stumbling block for effective code reviews. By abstracting key functionality into a series of design patterns and software architectures, reviewers can quickly come to terms with the code in the context of a broader system.

Figure 8
Elements Used In Code Reviews



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Make More Time For The Review

IT professionals described the No. 1 challenge of code reviews as finding enough time to prepare. This challenge is not unique to code reviews and often hampers all development activities. The additional constraint imposed on code reviews is that because this is a downstream activity it's very hard to predict upfront when the code will be ready for a review. To make more time for code reviews, application development professionals should:

- **Make code reviews a formal part of the process.** The activities associated with code reviews apply to all forms of process, including Agile, iterative and waterfall. If organizations are following an Agile approach, code review should be included as part of the "done" criteria for a story. If an iterative process is followed, code reviews should be included before an iteration is complete. For waterfall models, the stage build cannot be finished until code is reviewed. By actively planning for code reviews, you increase the likelihood that there will be ample time to undertake them. By making code reviews a milestone, it's possible to report on the status of the milestone. This helps ensure that code review, although a very technical activity, has the same level of visibility of other activities in the process.

- **When automating processes, include code reviews.** Once code reviews are a part of the process, one way of ensuring they happen is to include them in the application life-cycle management (ALM) or change management tool that automates the process. Often they can be associated with the promotion model for source code, ensuring that evidence of the code review is provided prior to code being promoted into an integration or system test stream. By coupling code reviews to the source code management (SCM) controls, code will be reviewed more frequently and not postponed until the end when the volume of code is large.

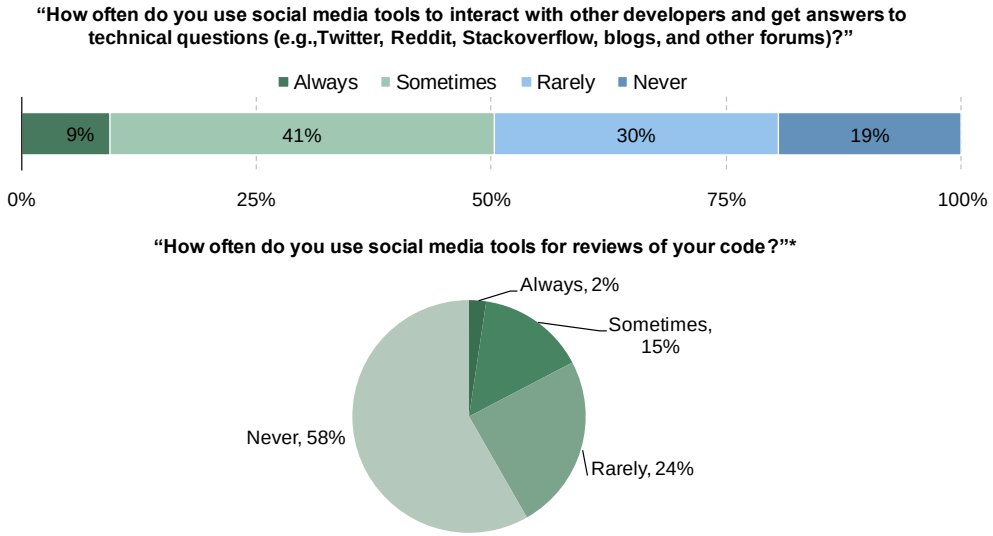
- **Capture code issues in the backlog.** Why should code review results be treated differently from other development areas? Add findings from code reviews to the team's work list or backlog, the same as you would other work items like defects, etc. These items can then be selected and worked on by the development team in the same way as bugs. This ensures that management has visibility of the work happening on the project, and that learning from a design issue or programmatic style problem is shared.

Take Advantage Of Social Tools

Social media is making a frequent appearance in the lives of IT professionals surveyed, with 50% saying that they use social media tools to interact with other developers to answer difficult questions (see Figure 9). The stereotype of a software engineer as a lone geek may not be true, but often the activities of software development require focus and are by their very nature antisocial. Code reviews, like answering technical questions, require some level of social interaction. Code reviews in particular require a high degree of organized interaction, with a group of developers reviewing code and discussing program elements and structures. As results of this study show, it's increasingly unlikely that all the right participants will reside in the same building. But taking a cue from how they use social media to answer technical questions, IT professionals can also exploit new interactive tools in the following ways:

- **Make code reviews a pull rather than push activity.** Primary work tasks never flawlessly divide up into a work day, often leaving small chunks of time that can be filled with other activities. Also, following the maxim of "a change is as good as a rest," looking at a different problem can help focus on the problem at hand. By providing a queue of code review work broken into small chunks, IT professionals can quickly pick up a section of code, review it, and make comments concurrently with their regular work tasks. By organizing queues into particular technologies or systems, it's possible for engineers to sign up for a particular selection of code to review, allowing the right skills to be applied.
- **Exploit wikis to store experience.** The back and forth of many code reviews can be captured in the form of a wiki thread, allowing engineers to add references, comments, and documents. Threads can be connected to a code set, allowing future developers to gain valuable insights into the code's evolution and review. Interested parties can sign up for a thread, allowing them to be made aware of when new discussions have occurred.
- **Utilize social media to increase your review group.** Technology such as microblogging enables discussions to include a much larger population, allowing developers to contribute experience and knowledge in the most appropriate way, considering their availability and the subject. Searching these resources after the fact will also provide invaluable guidance to engineers who will have to maintain the code or create new similar applications, classes, or services.

Figure 9
Social Media Use By Developers



Base: 159 IT professionals who participate in their organization's software development and code reviews
*128 IT professionals who participate in their organization's software development and code reviews and who use social media tools to interact with developers and answer technical questions

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

BUILDING A 21ST-CENTURY APPROACH TO CODE REVIEWS

The study shows that code reviews are pretty much the same now as they were when invented more than 40 years ago. They are a valuable and important activity providing significant insights into the code base, allowing engineers to share best practices and reuse ideas. But, code reviews also continue to be run in person, with local staff who don't have enough time to do as good a job as they would like. As software projects become more complex in terms of both technology and organization, IT professionals should question their approach and look to modernize the practice of code reviews in the following ways:

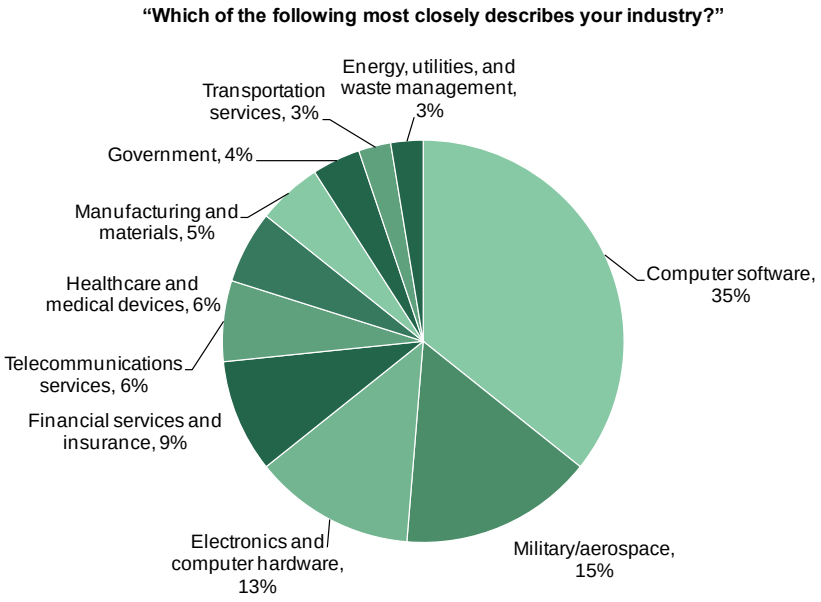
- **Make code review contribution part of technical development.** To increase motivation and interest in the practice, make it front and center in the development of engineers. Include personnel development objectives for engineers based on their contribution to code reviews. Introduce skills profiles for engineers that describe the subjects they are competent to review. Using the output of reviews as an input to HR reviews allows the creation of technical objectives that can be tied directly to engineering practices. For example, an HR objective may be to learn to use a particular design pattern or refactor a layer of the architecture. This can be brought out as part of the code review and used to demonstrate technical development.
- **Use tools to automate understanding and increase information.** Instead of making code reviews a solely manual process, add automation to ensure the activity happened and confirm the results of that activity. Add code analysis and coverage tools and test results to the inputs for code reviews, ensuring that the review team has detailed information to supplement the static code. Once additional metadata is being captured, it's important to carry that information around in the same way as other key project artifacts, maintaining version control and connecting it to the change management practices of the team. When issues are found during the code review, the way in which they are treated should be twofold: 1) Interaction and discussion should be captured in a wiki, allowing notes and decisions to be shared with a broader community; and 2) more fundamental, architectural, or design issues should be treated in the same way as other defects and managed accordingly. By equipping developers with the same code analysis tools used in the code review, code can be of a much higher quality prior to being reviewed, with automated tools helping developers find and eliminate simple mistakes on structure and behavior.
- **Increase participation with social media.** In direct response to dispersed development teams and the need to get access to the right development resources, consider including the use of social media tools such as wikis and microblogs and use discussion forums to augment and document the review process. Store this information with the code and system documentation, providing a valuable context for any engineers working in this code space in the future. Long-term approaches could include opening up code reviews to a much larger development community and sharing code and insights with developers outside of the organization. By exploiting modern social approaches, IT professionals can extend the value of code reviews, reduce their cost, and make them a valuable part of a software engineer's development.

Appendix A: Methodology

In this study, Forrester conducted an online survey of 159 US IT professionals to evaluate the current state of code review practices and ideas for the future and improving the code review process. Survey participants included IT professionals who participate in their organization's software development and code reviews. Questions provided to the participants asked about current code review team practices and tools used, opinions on the value of code reviews, current challenges, and ideas for increasing code review effectiveness. Respondents were also asked specifically about their use of social media tools outside and within the development and code review environment. The study was conducted in February 2010.

Appendix B: Demographics/Data

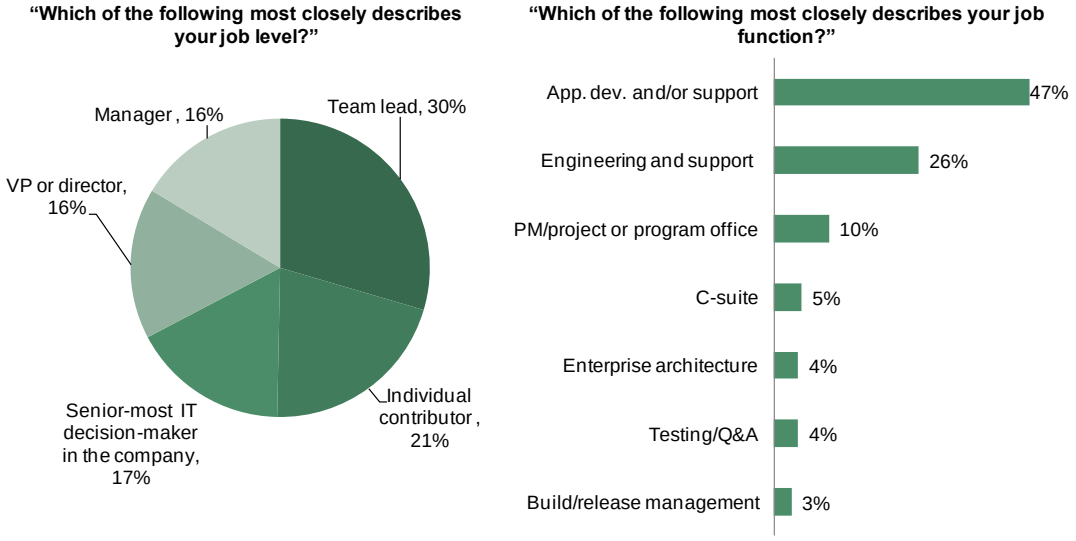
Figure B1
Industries Surveyed



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Figure B2
Respondent Level And Role



Base: 159 IT professionals who participate in their organization's software development and code reviews

Source: A commissioned study conducted by Forrester Consulting on behalf of Klocwork, February 2010

Appendix C: Endnotes

[1] Moore's Law describes a trend in computing hardware in which the number of transistors that could be placed on an integrated circuit board doubled every two years. Rather than just talking about the growth in transistor power, the law is used to describe the incredible growth of computing power, capability, and context within the past 40 years.

[2] Code refactoring is the practice of incrementally changing the external functional behavior of code in order to improve the nonfunctional capability of that software. Examples include performance, security, and often in the context of code reviews, standards and readability. The best book on the subject is *Refactoring: Improving Design of Existing Code,* by Martin Fowler, published by Addison-Wesley (1999).