</>

5 practices to make code quality a shared focus of the entire organization

www.sig.eu

Software Improvement Group

Think **big**, start small

If you've read our previous article, you likely have a pretty clear understanding that an organization can create maximum value with its tools and the coders who use them only when it commits to making code quality an integral part of its ethos and operating practices. In this eBook we elaborate on five practical measures you can undertake to make the health of your software a companywide priority.

Obviously, it's neither realistic nor necessary to conquer the world in one day. If software-related incidents hamper your progress on legacy renewal and hinder your digital transformation efforts, it's time to take action and embed code quality into your daily routines by making small adjustments in the way you work. This will eventually enable you to benefit from the great advantages of high-quality code such as increased time to market, lower costs and more secure and future-proof software.

The key is to pick the right solution for your situation. Don't opt for an approach just because it's what you're 'supposed to do' (Microservices! Low code! SaaS! Cloud!). Rather, determine what way of working is most effective both for your team and the system you're building.

Using the right metrics and tools helps you focus on improving what matters instead of running around and firefighting symptoms of the true underlying problem. In the next section, we zoom in on some striking data and tool features that will make your life easier.

5 practices to make code quality a **shared focus** of the entire organization

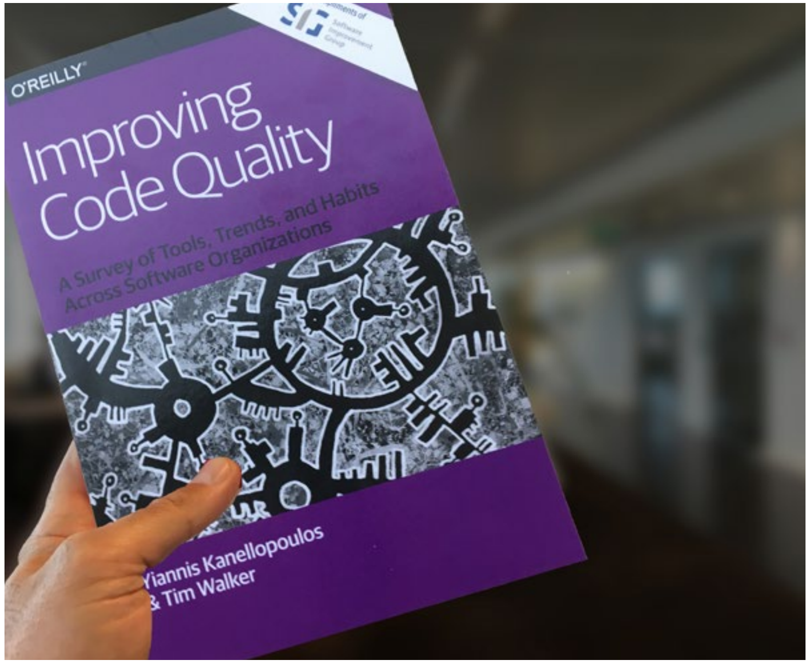
As research suggests, the good news is that code quality is often seen as a shared responsibility of an entire team. Delivery of high-quality code and products then highly depends on the team's ability to develop common standards, metrics and techniques that create a shared definition of done across an entire organization. Here are five measures you can take to actively start steering on code quality, based on research, as well as our industry benchmark and 18 years of experience.

1 Put your **money** where your mouth is

Make **resources** available

In a survey conducted by SIG and O'Reilly with 1400+ developers, 80% of all respondents said they address code quality "during coding". Yet, more than half of them do not use any code quality tools at all. This is largely due to the fact that they lack the budget to acquire them. This indicates that lots of organizations - regardless of what they say about the value and importance of code quality - are not putting their money where their mouth is in terms of making resources available to actively steer on quality.

Investing in the usage of certain tools will help address some of the reasons as to why you're losing sleep at night: it will improve performance, stability, security and maintainability of enterprise software. It might be time to reconsider your budget priorities and make sure there are dedicated resources available for development teams so they can continuously deliver high-quality code using a consistent, empirical methodology.


O'REILLY
Improving Code Quality
A Survey of Tools, Trends, and Habits Across Software Organizations
Yiannis Kanellopoulos & Tim Walker

Improving Code Quality, O'Reilly Media and Software Improvement Group, 2017

2 Use the right standards and metrics to measure code quality objectively

Another striking result of the aforementioned research is that more than 70% of poll respondents said they apply some form of quality standard, but only half of them check for source code metrics. This means that around only one third of the developers perform metrics-based evaluations of their source code, even though more than two-thirds of them think a measurable quality standard would help them achieve better software. We can conclude there is definitely room for improvement there.

So how can you measure code quality objectively?

First and foremost, in order for metrics to be effective, they must measure properties that have the most positive impact. As it turns out, maintainability is the most important aspect of code to measure. That may sound like you need a silver bullet, but the truth is that maintainability requires following 10 simple guidelines. You don't need to strive for perfect maintainability (whatever that may be), but you have to know when the quality of your code guarantees sufficient maintainability. On the next page you can see what the 10 guidelines we're referring to include.

10 guidelines for **better code**

Write short units of code

Write code once

Seperate Concerns in Modules

Keep Architecture Components Balanced

Automated tests

Write simple units of code

Keep Unit Interfaces small

Keep your code base small

Keep your code base small

Write clean code

These 10 guidelines are derived from the SIG benchmark, which consists of more than 8 billion lines of code in more than 180 different technologies. Analyzing around 15 million lines of code every week, we can confidently say that we know a thing or two about maximizing maintainability and making IT future-proof. The benchmark provides the software industry with a definition of a maintainable system. When a systems scores above market average, it's easier than average to maintain. Since the industry is improving continuously, this unique benchmark is recalibrated every year, making sure it reflects the state of the art in software engineering.

3 Select relevant tools. With relevant features

Using the relevant (static analysis) tools will make your lives and development process easier. And your developers happier. But how do you go about selecting the right tool for your teams? Based on the results of the study, it seems that developers have specific requirements when it comes to qualifying and selecting static analysis tools; for example, they have to support a wide variety of technologies, the price should be fair and it should have specific features, including:

- Indication of software design guidelines
- Indication of security vulnerabilities
- Metrics on test coverage
- Integration with automatic build pipeline

The biggest pitfalls of these tools boil down to three main issues:

- They generate too many false positives
- They flag too many warnings
- They come at too high a price

These insights can help you select the right tool for your situation.

4 Use the right tool to improve what matters

Lots of tools generate an exhausting list of violations, suggesting that each and every violation is bad and should be resolved. However, resolving everything is neither necessary nor beneficial. Flooding developers with a ton of violations will only demotivate them, and they'll want to ignore everything altogether.

We believe it's more valuable to take a risk-based approach and only focus on improving what matters.

A clean codebase gives development teams a running start, makes it easy for developers to begin contributing and helps maintain a high velocity. But when is a codebase clean enough? When does refactoring and fixing bugs makes sense, and when is it not worth the effort? You need context. You need to know what "done" means, when you're there, and what you need to change in the codebase to be above market average.

Integrating tools like Better Code Hub into your CI/CD pipeline will lead to continuous benchmarked software quality assurance for your codebase, happier developers and a strong competitive advantage.

Better Code Hub
by Software Improvement Group

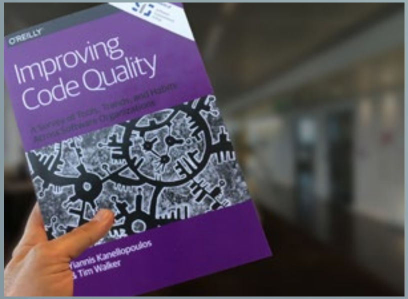
Better Code Hub was developed by SIG to help software development teams make code quality a shared focus of the entire organization. Better Code Hub indicates exactly the right amount of quality improvements to achieve that Definition of Done, so you won't be spending your time fixing bugs but instead shipping new features. The tool checks your GitHub codebase for compliance against 10 software engineering guidelines – and gives you immediate feedback on where to focus for quality improvements.

5 Focus on developer happiness

It's quite simple: happy developers solve problems better and deliver higher quality software faster. Software development is a highly intellectual activity and emotions and mood deeply influence the cognitive processing abilities and performance of software developers, including creativity and analytical problem solving. Research tells us that the top 3 causes for developer unhappiness are:

1. Being stuck in problem solving
2. Time pressure
3. Bad code quality and coding practice.

As the practices mentioned above suggest - you can actively improve the third cause of developer unhappiness. Using the right metrics and standards, making dedicated resources available, and taking advantage of the relevant tools will put the right focus on code quality and make it a shared priority of your entire organization.



Improving Code Quality, O'Reilly Media and Software Improvement Group, 2017

Software Improvement Group
Fred. Roeskestraat 115
1076 EE Amsterdam
The Netherlands
www.sig.eu

Software Improvement Group