How BMW Improved Software Quality and Agility of a Mission Critical Software System

About BMW:

With the brands, BMW, MINI and Rolls-Royce Motor Cars, the BMW Group has its sights set firmly on the premium sector of the international automobile market. To achieve its aims, the company deploys its strengths with an efficiency that is unmatched in the automotive industry. From research and development, to sales and marketing, BMW Group is committed to the very highest in quality for all its products and services. The company's success to date is proof that this strategy is the correct one.

About hello2morrow

Hello2morrow is a leading provider of comprehensive static analysis tools for the validation and enforcement of rules related to architecture, structure and technical quality of software systems. Our products support Java, C#, C/C++ and other languages, and are used by more than 300 clients all over the world.

The Challenges:

In 2012 BMW decided to redevelop their "Integrated Aftersales Platform" (IAP). Integrating required new features and changes into the old version of the system was not considered to be feasible. IAP-1 already had a significant defect backlog and it was very difficult to change the system without introducing new defects. The system was suffering from severe structural erosion and high levels of technical debt.

The new system has a staff of about 75 developers, 30 of them working with Java. A key goal for the new system was to avoid running into the same problems that crippled its predecessor. In particular, the following challenges had to be met:

- New releases every 4 months
- Strict rules for architecture and code quality standards that would be enforced in an automated fashion
- Contracts with 3rd party development service providers now had to include architecture rules and quality standards. Rule violations had to be fixed in the scope of the provider warranty.
- Avoid the buildup of a defect backlog. All known defects had to be fixed with the upcoming release, at the latest.

The Solution:

First, acceptable code quality standards had to be defined. It became obvious that it would only make sense to enforce standards that can be checked efficiently and, ideally, in a fully automated way. It was also considered important that developers not be overwhelmed with too many rules and guidelines. The goal was to improve the quality of project outcomes without slowing down the development process in a significant way.

The standards had to cover the following aspects:

- Coding and formatting rules
- Testing and test coverage
- Software metrics
- Architecture rules and dependency structure

Each of these aspects needed tool support to ensure the automated enforcement of rules. The first two aspects could be covered easily with the help of Open Source tools like PMD, CheckStyle, FindBugs and Cobertura. These tools are conveniently combined by "SonarQube", another open source tool. SonarQube acts as an aggregator which allows various kinds of code analysis tools to provide data to a unified dashboard.

There was no viable open source option for the last two aspects, so BMW needed a commercial tool to cover the gap. Sonargraph had already been used successfully in the context of other smaller projects and could also be integrated with SonarQube. Therefore, the decision was made to use Sonargraph in all stages of the development of IAP-2.

While SonarQube collects data daily during the nightly build, Jenkins is used to check coding and architecture rules with every commit. Rule violations will break the build. Developers will receive an email notification from the build server so that the problem can be fixed while it is still easy to fix.

Here are the code quality standards used by BMW to enforce a high level of code quality:

Rules to limit architectural debt:

- No architecture violations (Sonargraph)
- No dependency cycles (Sonargraph)
- Limit for metric "Unassigned Types": 0 (Sonargraph)

Rules to verify solid test coverage:

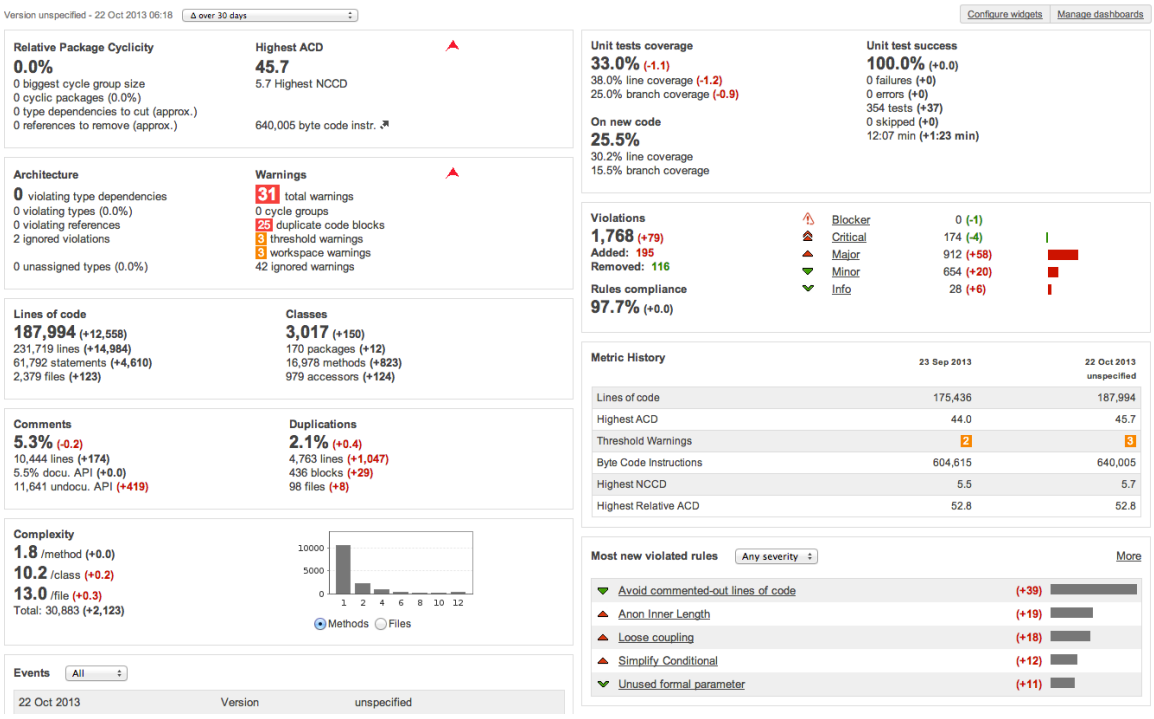
- Branch code coverage: at least 50% (SonarQube)

Rules regarding code complexity and comprehensibility:

- Number of SonarQube findings: 0 (SonarQube)
- Limit for metric "Cyclomatic Complexity": 20 (Sonargraph)
- Limit for metric "Average Component Dependency": 20 (Sonargraph)
- Limit for metric "Lines of Code" per compilation unit: 700 (Sonargraph)

Once the technical infrastructure was put in place to check and enforce rules, projects were reviewed frequently for compliance. This usually took place once every two weeks, and was done by the Quality Assurance Manager, a role that was specifically created to ensure the enforcement of the above standards. The review process was driven by the quality data derived from the SonarQube dashboard and Jenkins build results. If standards were not met the issue is first monitored by an issue tracking system. If an issue is not resolved within a limited time frame, it gets escalated to the attention of the project manager. Additionally, any application that fails to meet the quality standards does not go into production.

An important aspect of successfully implementing software development based on quality standards is to make sure that all stakeholders are on board. This begins with senior management being fully in support of the new approach, and then moves on to the



Typical SonarQube Dashboard featuring findings from Sonargraph and other tools

appropriate training of developers and architects. It is important to explain the established rules so that the benefits of a development approach centered on quality is clearly understood. Also important, is selecting development partners based upon their willingness to comply with the prescribed standards.

As a rule, "quality before time" was implemented by senior management, meaning that it is more important to enforce quality rules than it is to rush to meet deadlines. This makes it mandatory for a software system not only to pass all functional tests, but to also comply with all quality rules before it can go into production.

Of course, sometimes situations will occur that make sacrificing of quality standards a necessity for reasons outside of the technical scope of the project. In such cases it is important to plan for a repair phase after the initial goal has been met. A clean foundation and stable structure also makes repairs easier.

The Results:

All stakeholders consider the introduction of a quality centered development process into IAP-2 to be great success.

In particular, with comparison to IAP-1, the following improvements have been achieved:

- Significantly shorter release cycles. IAP-2 is delivering a full release every 4 months. IAP-1 needed between 12 and 15 months between releases with a tendency toward longer periods between releases. This is *especially* significant when taking into consideration the "quality before time" rule.
- The number of functional defects per function point has been reduced by a factor of 10.

- All defects are fixed much faster, usually within the next sprint. In IAP-1 there was a growing pile of defects moved from release to release.
- Since code quality and architectural rules are now part of the contract, BMW does not have to pay for fixes of rule violations anymore. They have to be fixed in the context of the development service provider warranty. The software will only be approved by BMW if all contractual quality standards are fully met. This is automatically verified by scanning the code base with Sonargraph and other related tools.
- Greater development efficiency and lower maintenance effort make space for new applications and improved functionality. Improved agility!
- No more panic mode. In IAP-1 the most critical issues had to be fixed by specialized task forces with direct reporting to the BMW board of directors. Avoiding these kinds of situations is not only good for company morale, it is also good for bottom line of the business.

Since a clean software architecture is a precondition for many other aspects of code quality, Sonargraph is considered to be a mission critical part of the tool chain used to enforce quality standards for IAP-2.

Related Research:

Capers Jones wrote a highly recommended book about "The Economics of Software Quality". The gist of it is nicely condensed in the following paragraph from an interview with the author:

„And not only for software is quality free, but it actually has a positive return on investment. So by emphasizing quality in software, you get more than just a free ride. You actually gain market share, you lower your warranty costs, you shorten your development schedules, and you improve team morale. You get a relatively powerful return on investment for a relatively small outlay of cash and energy. So I think software can actually go beyond the concept of "quality is free," to the concept that "quality pays a handsome profit."

The interview can be found here:

http://www.informit.com/articles/article.aspx?p=1824791
http://www.informit.com/articles/article.aspx?p=1824792

Contacts:

BMW AG

Erwin Wagner

Email: erwin.Wagner@bmw.de
Web: hhtp://www.bmw.de

hello2morrow

Alexander von Zitzewitz

EMail: a.zitzewitz@hello2morrow.com
Web: http://www.hello2morrow.com