CODE REVIEW AUTOMATION

CODE REVIEWS | BY Balachandra Tarodi

CODE REVIEW AUTOMATION

Code Reviews

Code Review Automation

Automation of code review plays vital rule in the quality and robustness of any application and is one of the important corner stones in the DevOps. Over the past decade, with tremendous advancement of intelligent IDE's (mostly open source tools like Eclipse) and billions of lines of reference code available freely and accessible on the internet (GIT), majority of average developers spend little time learning the craft of programming. In other words, Developers spend less and less time learning new API's or optimize their code during development. The problem also gets compounded by rapid pace of industry deploying latest versions of Software Product almost every year and developers have hard time to keep up with the pace. In short, the code quality both in efficiency and robustness, suffer over time. This has inevitably lead to enforcing automated code reviews across Industry in many forms, specifically for making DevOps highly effective.

Automated Code Review Tool

Reviewing manually thousands of lines of code for pitfalls and incorrect use of API can be extremely tiresome exercise for any Peer Reviewer. Code review comments provided by team members can give rise to conflicts in some situations. Several of the automated code review tools have successfully deployed in Enterprises to review continuously code base as new code are checked in and provide feedback and insights to team. Automated code review tools should be used more like a friendly assistant (similar to a smart IDE). In this white paper we will look in details of leveraging SonarQube and its ability to craft custom rules for applications.

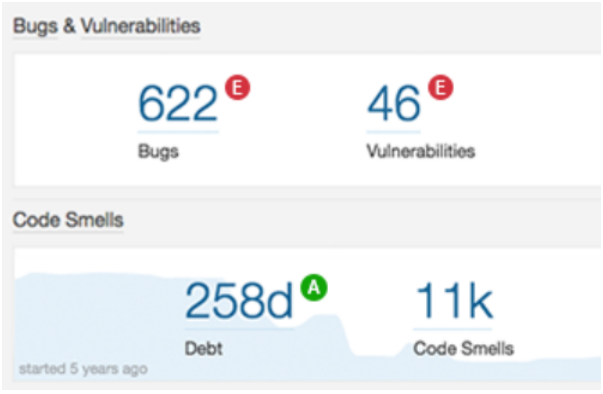
The other aspect is uniformity of enforcement of standards in both coding and design not only within team, but across multiple teams and vendors. The quality of code can suffer due to attrition of experienced programmers and it hard to detect and quantitatively measure in large scale applications in the enterprise. Robustness, extensibility, modularity and maintainability of code base is only be possible by enforcement of good coding principles, right from the start of development phase by the project team. Automated code review tool along with project specific tailored rules will help the team in this journey of DevOps.

SonarQube

Many open source tools as well as Commercial Off the Shelf (COTS) solutions are available currently to help with automated code review challenges. One of the most widely used tool in our industry is SonarQube.Below are some of the strengths and flexibility of this tool which makes it one of the preferred tools in DevOps.

Visualize Historical Trends

Code quality can be only as good as the members who code it. Code quality can be adversely affected by high rate of attritions of developers. It is imperative in large scale projects to continuously check the entire code base, so that even after several years there will not be significant diminishing Return on Investment (ROI) for maintaining the large code base. SonarQube maintains code review statistics of every code base scanned for a project, thus can show the historical trends of quality of code base over the course of project executions



Support for DevOps integration with tools like JENKINS, MAVEN, ANT etc

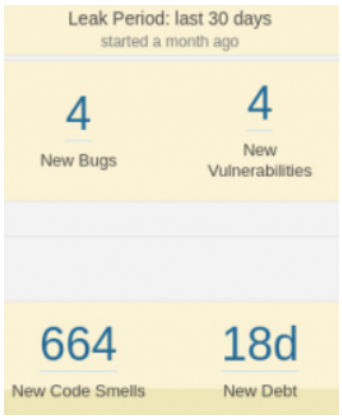
Quality Profile

Adhering to high code quality standards will generally help to keep production defects to minimum. Sonarqube methodology is through collections of code review rules which is called Quality Profile. SonarQube also allows exporting and importing the Quality Profile from one instance to another, so that it is easy to test the existing code base over additional new rules and its impact.

Ability to configure Custom Rules specific to s/w application or project

Although SonarQube offers hundreds of OOTB rules that most of applications can use, the enterprise applications are far too complex and use many custom vendor software. These will have custom API which the developers may not use effectively leading to not just performance issues but also severe deadlocks. The ability to enforce rules over custom in-house or vendor API is the greatest strength of automated code review tool

Analyze and show statistics only on changed code base (Leak Period)

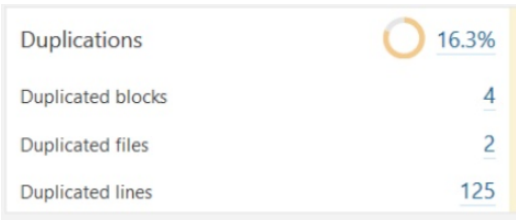


Ability to view the issues directly in the context of the source code.

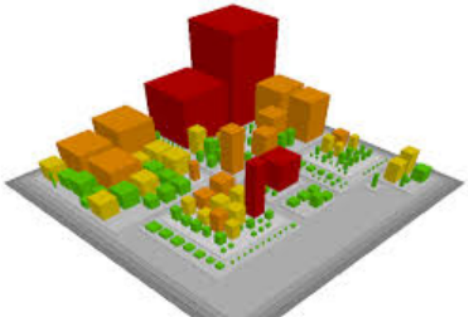
Supports multiple programming languages (JAVA, C#, VB.NET, JavaScript, Python, etc)

Detect duplications across entire code base (not just within a single source file)

Code duplications occur due to many reasons. Copying entire methods or even large portions of a class file and making few changes is not uncommon practice by junior developers who have been pushed to meet challenging deadline. Proactive refactoring of code base is generally a nice to do thing in ever changing landscape and team dynamics.



Ability to use various open source plugins including 3D-Visualization of code base for quick drill down.



Our Value Proposition using SonarQube

We have been demonstrating to our clients the ROI of automated code review as part of DevOps in the projects. Sogeti team has actively engaged as partner to implement custom SonarQube rules written in JAVA, that are tailored for software applications (eg ENOVIA 2012x till 2018x).

Sogeti team has also done several demos showing how to leverage SonarQube for large customers and implement custom rules to maximize the value for the clients. As part of our SonarQube engagement, we follow 3 step process to implement SonarQube for clients

Step-1: Identify application specific value add Rules.

This is the first step that helps document the gaps in their current code review process and identify the checklist of rules and other manual methods on improving code standards. Then all the possible violations are identified along with severity. Sogeti team has several years of experience in identifying common coding pitfalls (checklists) and convert them into SonarQube rules.

Rule Name	Description
CheckMethodName	Method name should be minimum 5 characters and not exceed 20
CheckClassName	Class name should be minimum 10 characters and begin with “ABC_”

Step-2: Categorize & Prioritize the identified rules

This is the next step to bucket the rules into implementation complexity categories like low, medium and high. This is accompanied by detailed scenario or examples of each violation. The violations and discussed and prioritized for implementation for the customer.

Rule Name	Description	Priority	Complexity
CheckMethodName	Method name should be minimum 5 characters and not exceed 20	2	Low
CheckClassName	Class name should be minimum 10 characters and begin with “ABC_”	3	Low

Step-3: Implement, Package and Deploy

This is the final step of implementation of SonarQube rules, updating the Quality Gates as per project needs, followed by packaging and deploying in client environment.

Also as value add, team can demo the code base quality and violations in 3D to end-customers with ability to drill down the hot spots in various complex modules without going through various reports.