WIE GELANGT MAN ZU EINEM EFFIZIENTEN PRODUKTIVBETRIEB

SonarQube for Enterprise

Mit SonarQube die Softwarequalität im Entwicklungsprozess überwachen.

Die unterschiedlichen Qualitätsmerkmale von Software werden in der ISO 9126 beschrieben. Sie können grob in funktionale und nicht funktionale Anforderungen unterteilt werden. Die Umsetzung von funktionalen Anforderungen lässt sich durch passende Teststrategien sicherstellen. Darüber hinaus muss sich ein Entwickler mit den folgenden Fragen beschäftigen: Wie einfach lässt sich die Anwendung warten? Wie ist die Testabdeckung des Codes? Werden die Richtlinien eingehalten? Bei der Beantwortung dieser Fragen kann das serverbasierte Tool SonarQube helfen.

SonarQube liefert nicht nur Warnungen, die Qualitätsprobleme aufzeigen, sondern darüber hinaus weitere Kennzahlen, wie zum Beispiel die technische Schuld. Sie beschreibt, wie viel Zeit investiert werden muss, um die bestehenden Qualitätsmängel zu beheben. Eine weitere Kennzahl steht für die Komplexität der Anwendung. Mit ihr kann darauf geschlossen werden, ob die Anwendung leicht zu warten ist. Die Werte lassen sich zeitlich verfolgen und sind somit wichtige Indikatoren, wie sich die Qualität des Quellcodes über einen Zeitraum entwickelt. Zusätzlich kann durch Grenzwerte sichergestellt werden, dass die Qualität nicht weiter absinkt. Dies ist insbesondere bei Legacy-Systemen wichtig.

Auf dem Weg hin zu einem effizienten Produktivbetrieb gibt es allerdings einige Stolpersteine, die überwunden werden müssen. Beginnend bei der Konfiguration über die Integration von externen Analysewerkzeugen, wie ReSharper, StyleCop oder unterschiedlichen Roslyn-Analyzern, bis hin zur Vereinheitlichung von Richtlinien der SonarQube-Analyse und der lokalen Entwicklung mittels SonarLint. Die Integration von SonarQube in den Continuous-Integration-Prozess (CI-Prozess) spielt dabei eine ebenso wichtige Rolle wie die Rechtevergabe unter den Nutzern und die Organisation von unterschiedlichen Regelsätzen.

Was ist SonarQube?

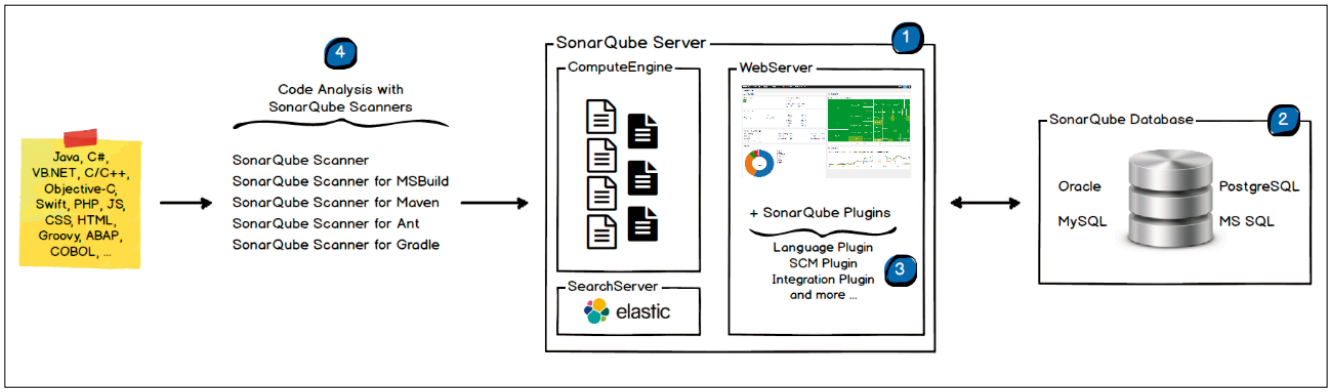
SonarQube ist eine serverbasierte Plattform, um die Codequalität zu überwachen. Es bietet individuell konfigurierbare Dashboards, um Daten entsprechend den Bedürfnissen unterschiedlicher Stakeholder anzuzeigen. Durch die Historisierung der Daten ist es zudem einfach möglich, ständig ein Auge auf die Entwicklung der Softwarequalität zu haben. So können auch Softwarearchitekten, Projektmanager und CIOs von den Analysen profitieren. Darauf aufbauend können sie Maßnahmen einleiten, um die Softwarequalität zu erhöhen und um Risiken frühzeitig zu erkennen und zu vermeiden. SonarQube ermöglicht dadurch ein zentralisiertes Management der Softwarequalität.

SonarQube deckt die sieben Achsen der Codequalität ab. Durch die Analyse werden Kennzahlen zu Architektur und Design, Duplikaten, Unit-Tests, Komplexität, potenziellen Bugs, Entwicklungsrichtlinien und Kommentaren zur Dokumentation erfasst und gespeichert. Anhand dieser Kennzahlen wird eine achte Kennzahl ermittelt, die bereits erwähnte technische Schuld. Sie beschreibt den Aufwand, der betrieben werden muss, um den Quellcode von jeglichen technischen Mängeln zu bereinigen, und ist damit der Hauptindikator für die Softwarequalität [1].

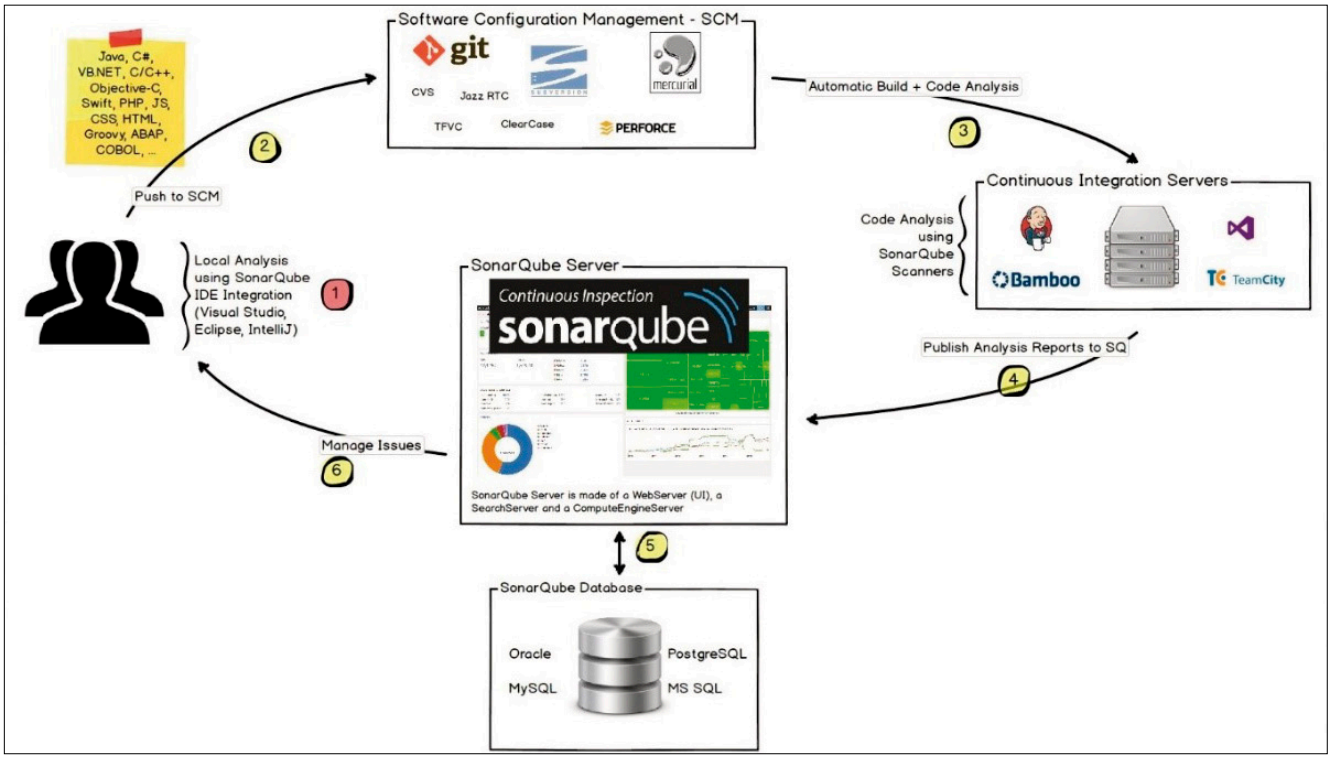
SonarQube ist Open Source und kann frei verwendet werden. Wer zu den freien noch erweiterte Funktionen und zusätzlich Support und eine erste Einweisung wünscht, kann für bis zu 50000 Euro pro Jahr unterschiedliche Lizenzen erwerben. Es besteht aber auch die Möglichkeit, individuelle Anpassungen und Erweiterungen entwickeln zu lassen.

Architektur

Die Architektur von SonarQube (Bild 1) basiert auf vier Komponenten. Die erste ist der SonarQube-Server, der seinerseits



Die Architektur von SonarQube (Bild 1)



SonarQube: Continuous Inspection (Bild 2)

aus einem Webserver, einem Search Server und einem Compute Engine Server besteht. Der Webserver stellt eine Oberfläche bereit, mit deren Hilfe man die SonarQube-Instanz verwalten und konfigurieren kann. Außerdem werden die bereits durchgeführten Code-Analysen übersichtlich dargestellt und lassen sich durch Entwickler und Manager auswerten. Der Search Server liefert die passende Engine, um die Datenbank nach Analysen zu durchsuchen und diese auf der Oberfläche anzuzeigen. Der Compute Engine Server verarbeitet schließlich die durchgeführte Analyse und speichert diese in der angebundenen Datenbank.

Die Datenbank stellt zugleich die zweite Komponente dar. In ihr werden sowohl die Einstellungen als auch die Analysen gespeichert. SonarQube liefert eine Datenbank von Haus aus mit, unterstützt allerdings auch die Verwendung von Microsoft SQL Server, MySQL, Oracle und PostgreSQL. Die mitgelieferte Datenbank sollte nur für Testzwecke verwendet werden. Im Produktivbetrieb wird daher empfohlen, auf eine der unterstützten Datenbanken der Drittanbieter zurückzugreifen [2].

Die dritte Komponente stellen die unterschiedlichen Plugins dar. Mit diesen kann man SonarQube um unterschiedliche Fähigkeiten erweitern. So können Analysen für weitere Programmiersprachen integriert oder auch die Authentifizierung und Autorisierung an das Active Directory delegiert werden. Eine Integration in unterschiedliche Quellcodeverwaltungssysteme wie Git, TFVC oder auch SVN kann ebenso umgesetzt werden.

Der SonarQube-Scanner ist die vierte Komponente. Dieser führt letztendlich die Code-Analyse durch. Der Scanner wird auf der Build-Maschine ausgeführt. Außerdem stehen unterschiedliche Versionen bereit, die verschiedene Build- beziehungsweise ALM-Tools unterstützen. Damit und durch die Unterstützung von Apaches Maven und Ant, MSBuild, Gradle und Jenkins erschließen sich vielfältige und einfache Möglichkeiten, die Analyse in den CI-Prozess zu integrieren, siehe Bild 2. Die Analyse kann aber auch auf der lokalen Entwicklungsmaschine ausgeführt werden.

Continuous Inspection

Wie genau sieht die Integration der Analyse in den CI-Prozess aus?

- Entwickler schreiben Quellcode in ihrer IDE und können mithilfe von SonarLint die Codequalität lokal mit den in SonarQube konfigurierten Regeln überprüfen, ohne die Ergebnisse auf den Server zu laden.
- Hat die Software einen gewissen Stand erreicht, wird der Code in das verwendete Quellcodeverwaltungssystem eingespielt.
- Durch das Einspielen des Quellcodes wird auf dem Continuous Integration Server automatisch ein neuer Build-Prozess gestartet. Dieser muss so konfiguriert werden, dass während des Build-Prozesses der SonarQube-Scanner auf dem Build-Agent gestartet und ausgeführt wird.
- Nach Abschluss der Analyse im Build-Prozess werden die Ergebnisse zum SonarQube-Server hochgeladen.
- Die Ergebnisse der Analyse werden in der Datenbank gespeichert und können über den Webserver abgerufen und angezeigt werden.
- Anschließend können die Ergebnisse überprüft und kommentiert sowie die qualitativen Mängel behoben werden, die während der Analyse entdeckt wurden. ▶

Installation & Konfiguration

Bevor jedoch die Anwender von den umfangreichen Analysen profitieren können, die SonarQube bereitstellt, ist einiges an Konfigurationsaufwand zu bewältigen.

Zunächst muss die Installation von SonarQube erfolgen. Das geschieht über das Entpacken des Installationspakets und Kopieren des Inhalts in das gewünschte Verzeichnis. Vor dem ersten Start müssen in der Konfigurationsdatei nur noch der gewünschte URL und der Port eingetragen werden, unter welchen die Weboberfläche des Servers erreicht werden soll. Für den Produktionsbetrieb ist zusätzlich noch der Connection String für den Zugriff auf die gewünschte Datenbank einzutragen. Anschließend kann SonarQube entweder als Konsolenapplikation oder mit wenigen Eingaben als Windows Service gestartet werden.

Quality Profile

Nun können die Regeln konfiguriert werden, die während der SonarQube-Analyse angewendet werden sollen. Dies geschieht über sogenannte Quality Profiles. Über den Reiter *Quality Profiles* und den Button *Create* kann ein neues Profil erstellt werden. Über den Reiter *Rules* lassen sich anschließend die Regeln konfigurieren. Dazu werden die ausgewählten Regeln über den Button *Bulk Changes* in einem Quality Profile aktiviert oder deaktiviert (Bild 3).

Quality Profiles können auch von bestehenden Profilen abgeleitet werden, sodass man beispielsweise einen Mindestregelsatz bereitstellt und je nach Anwendungsfall zusätzliche Regeln konfiguriert [3].

Quality Gate

Im Quality Gate werden Grenzwerte für beliebige Kennzahlen eingerichtet, welche die Mindestanforderungen an die Qualität bestimmen. Dabei kann zwischen Bedingungen unterschieden werden, die immer erfüllt werden müssen, und solchen, bei denen dies über einen gewissen Zeitraum der Fall sein muss. So kann man zum Beispiel festlegen, dass die Code-Abdeckung bei Unit-Tests jederzeit mindestens 80 Prozent betragen muss oder dass die technische Schuld im Monat um nicht mehr als zwei Stunden zunehmen darf. Dabei lassen sich zwei Stufen festlegen: ein kritischer Bereich und ein Bereich, in dem die Softwarequalität als ungenügend angesehen wird. Dementsprechend kann das Quality Gate nach der Analyse die Werte *passed*, *warning* oder *error* aufweisen. Über ein Benachrichtigungssystem kann man die betroffenen Nutzer über den Zustand der Analyse informieren. Für eine sehr restriktive Verwendung ist es sogar möglich, dass bei einer ungenügenden Analyse im CI-Prozess ein Build-Fehler erzeugt wird und der neue Stand somit nicht ausgeliefert werden kann [4].

Verwendung im Produktivbetrieb

Nach der erfolgten Konfiguration sind allerdings noch weitere Punkte für einen erfolgreichen Produktivbetrieb zu beachten. Besonders wichtig ist dabei, wie SonarQube in den bestehenden Continuous-Integration-Prozess integriert werden kann oder wie das Zusammenspiel mit Analysetools von Drittanbietern funktioniert.

TFS–Build–Integration

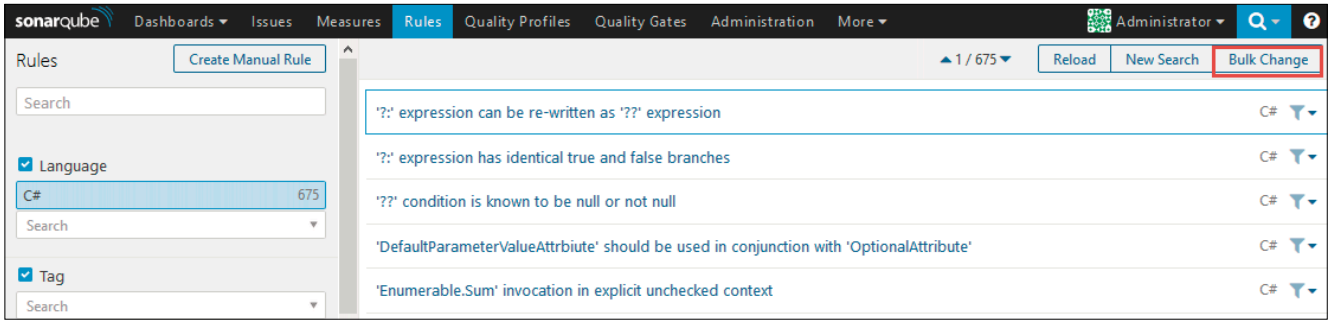
Die Integration der SonarQube-Analyse in den CI-Prozess des Team Foundation Server (TFS) erfolgt über zwei Build Tasks (Bild 4). Der Task *Begin Analysis* wird vor dem Build Task *MSBuild* beziehungsweise *Visual Studio* eingefügt und der Build Task *End Analysis* nach der Durchführung der Tests. Über die Schaltfläche *Manage* muss zunächst ein neuer generischer *Service Endpoint* erstellt werden. Dazu wird die Adresse benötigt, unter welcher der SonarQube-Server erreichbar ist, sowie Benutzername und Passwort, unter welchen die Analyse durchgeführt und hochgeladen werden soll.

Über den Projekt-Key wird ein SonarQube-Projekt eindeutig identifiziert. Alle durchgeführten Analysen werden unter dem eingegebenen Schlüssel gespeichert. Sollte noch kein Projekt mit dem angegebenen Key bestehen, wird automatisch ein SonarQube-Projekt angelegt, sofern der Benutzer, unter dem die Analyse durchgeführt wird, die dafür erforderlichen Berechtigungen besitzt. Durch die Vergabe einer Projektversion können die Änderungen für diese eine Version verfolgt werden. Es wird also zum Beispiel sichtbar, dass für eine bestimmte Version elf Issues hinzugekommen sind [5].

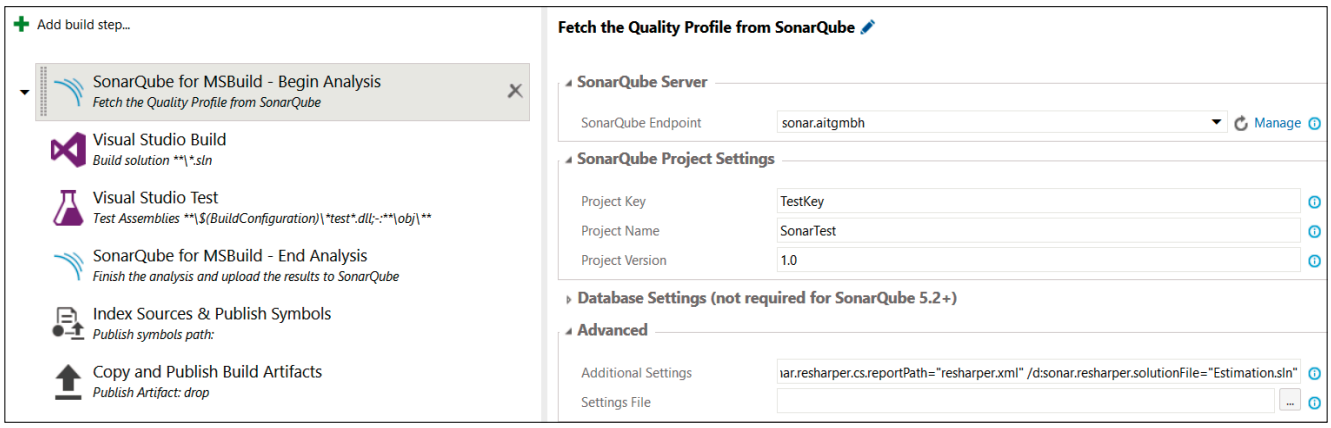
So konfiguriert wird in jedem ausgeführten Build automatisch die SonarQube-Analyse durchgeführt und die Ergebnisse hochgeladen. Diese können anschließend über die Weboberfläche betrachtet werden.

Rechteverwaltung

Das Erstellen und Verwalten von Anwendern und das Zuweisen zu Nutzergruppen bringt SonarQube schon mit. Benutzer und Benutzergruppen werden folglich über die SonarQube-



Regel eines Quality Profiles bearbeiten (Bild 3)



SonarQube-CI (Bild 4)

Oberfläche verwaltet [6]. Dies bedeutet allerdings für so gut wie alle Unternehmen einen doppelten Pflegeaufwand, da in der Regel ein organisationsübergreifendes System zur Nutzer- und Rechteverwaltung vorhanden ist.

Zur Vermeidung des doppelten Aufwands ist die einfache und schnelle Erweiterbarkeit von großer Bedeutung. SonarQube stellt dafür ein Active-Directory-Plug-in bereit, mit dessen Hilfe man SonarQube direkt an ein Active Directory (AD) anbinden und sich somit direkt mit den vorhandenen Nutzern anmelden kann. Aufgrund dieser Verknüpfung wird nach dem Login direkt ein Nutzer auf dem SonarQube-Server erzeugt. Der neue Nutzer hat zunächst keine Rechte. Diese können allerdings auch über die AD-Gruppen vergeben werden. Dazu müssen die AD-Gruppen, die verwendet werden sollen, zunächst eins zu eins über die Administrationsoberfläche von SonarQube angelegt werden. Über diese Gruppen lassen sich dann die Berechtigungen vergeben. Nach einem erneuten Login wird der Nutzer in SonarQube automatisch denjenigen Nutzergruppen zugewiesen, denen er auch im AD angehört. Die manuellen Zuordnungen von Benutzern zu Gruppen werden bei jedem Log-in automatisch überschrieben, da diese Konfiguration ausschließlich beim Active Directory liegen soll [7].

Wer auf den Clouddienst Azure Active Directory setzt, um Nutzer, Nutzergruppen und Berechtigungen zu verwalten, bekommt ebenso ein Plug-in geliefert, wie Verwender von Apache DS, Open LDAP oder OpenDS, die auf das LDAP-Plug-in zurückgreifen können [8].

Integration von Third-Party Analyzern

Die Integration von Third-Party Analyzern ist ein ebenso wichtiges Thema. Wer lokal Werkzeuge wie ReSharper oder StyleCop nutzt, will die Ergebnisse dieser Analyse natürlich auch auf dem Server haben, um die Softwarequalität zentral verwaltbar zu machen. Auch hier bietet die einfache Erweiterbarkeit durch Plug-ins wieder Lösungen an. Die Regeln für ReSharper können nach der Installation, wie gehabt, für bestehende Quality Profiles aktiviert und deaktiviert werden. Zum erfolgreichen Durchführen der Analyse ist allerdings die Installation der ReSharper-Command-Line-Tools auf dem Build-Server erforderlich sowie eine zusätzliche Konfiguration im Build-Prozess, worauf an dieser Stelle allerdings nicht weiter eingegangen wird [9].

StyleCop basiert in seiner neuesten Version auf den seit Visual Studio 2015 verwendeten Roslyn-Analyzern. Durch die jüngst erfolgte Fokussierung auf die Integration eben dieser Analyzer ist es auch möglich, StyleCop in der neuesten Version in SonarQube zu integrieren. Aber nicht nur das. Mittels des SonarQube Roslyn SDK können aus allen Roslyn-Analyzern SonarQube-Plug-ins erstellt werden, welche auf dem SonarQube-Server installiert werden können und die Regeln aus dem referenzierten Roslyn-Analyzer enthalten. Aktuell gibt es hierbei allerdings noch Einschränkungen. So muss der Analyzer als NuGet-Paket und in der Version 1.0 oder 1.1 vorliegen. Außerdem werden nur C#-Regeln unterstützt [10].

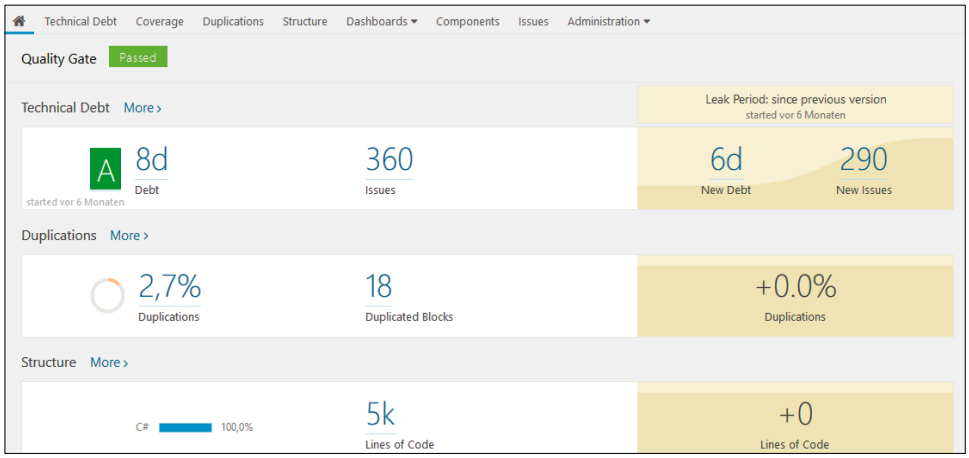
Die Liste von Drittanbieter-Analyzern lässt sich noch um NDepend erweitern. Wer in seinem Entwicklungsprozess andere Tools verwendet, muss auf ein Update hoffen.

Vereinheitlichen der Regeln

Ein Problem von SonarQube besteht in der Vereinheitlichung von serverseitig konfigurierten und lokalen Regeln. Will man durch die SonarQube-Analyse entdeckte Issues beheben, steht man vor dem Problem, dass diese in der lokalen IDE nicht als Warnung angezeigt werden. Im Entwicklungsprozess ist es allerdings sehr hinderlich, für jeden einzelnen Fehler in die Oberfläche des Web Service zu wechseln, ihn zu analysieren und in der IDE zu beheben. Viel komfortabler wäre es, wenn die Issues direkt als Warnung in der IDE angezeigt werden würden.

Dies wird durch die Visual-Studio-Extension SonarLint for Visual Studio ermöglicht. Mittels SonarLint lässt sich eine Verbindung zu SonarQube aus Visual Studio heraus herstellen und das lokale Projekt direkt mit einem Projekt auf SonarQube verknüpfen, um die Server-Regeln lokal anzuwenden. Auch hier gibt es allerdings eine Einschränkung: Die Synchronisation der Regeln funktioniert ausschließlich mit Roslyn-Analyzern, wobei die nativen SonarQube-Regeln auch als Roslyn-Analyzer zur Verfügung stehen. Ein Abgleich von ReSharper, NDepend oder auch der nativen CA-Regeln ist nicht möglich. Durch die Verknüpfung eines Projektes mit dem SonarQube-Projekt werden automatisch die benötig- ▶

ten Analyzer als NuGet-Pakete installiert und der im Quality Profile konfigurierte Regelsatz übernommen [11]. Trotz der gleichen Regeln ist es im Praxisbetrieb in seltenen Fällen zu leichten Abweichungen bei der Erkennung von Warnungen gekommen, was sich durch die unterschiedlichen Analysemethoden erklären lässt.



Ergebnis einer SonarQube-Analyse (Bild 5)

Auswertungen

Aber was kann man nun mit den Daten tun, die man mittels der unterschiedlichen Analysen gesammelt hat? Allein durch das Ausführen der Analysen entsteht noch kein Mehrwert.

Für einen ersten und schnellen Überblick über die Gesamtsituation aller laufenden Projekte dienen die globalen Dashboards. Diese liefern insbesondere für Projektleiter, Entwicklungsleiter oder auch CIOs einen erhöhten Nutzen. Die Dashboards lassen sich individuell gestalten, sodass jeder die Informationen erhält, die er gerade benötigt. So lassen sich beispielsweise für einen Projektleiter alle für ihn relevanten SonarQube-Projekte mit den für ihn relevanten Kennzahlen darstellen. Durch die Historisierung hat er ebenfalls jederzeit einen Überblick darüber, wie sich das Projekt entwickelt, und kann rechtzeitig qualitätssichernde Maßnahmen einleiten, um die technische Schuld unter Kontrolle zu behalten.

Für Entwickler dagegen sind die detaillierteren Projekt-Dashboards interessanter. Hier erhalten sie genaue Informationen zu allen Kennzahlen im Projekt und können durch die Drilldown-Funktionalität bis zum Quellcode navigieren, um der Ursache der technischen Schuld oder von Tests nicht abgedeckten Code-Pfaden auf den Grund zu gehen (Bild 5). Aufgrund dieser Möglichkeit der Überprüfung kann sich ein Prozess der Selbstkontrolle etablieren, wodurch sich die Fähigkeiten des Entwicklers durch ständiges Feedback verbessern und durch die Anwendung des Feedbacks die Projektqualität in gleichem Maße erhöht wird.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass SonarQube ein geeignetes Werkzeug ist, um die Softwarequalität im Entwicklungsprozess zu überwachen. Mithilfe der umfangreichen Erweiterungen und Unterstützungen von Third-Party-Tools lässt sich die Anwendung optimal in den bestehenden CI-Prozess integrieren, wodurch sich Continuous Inspection umsetzen lässt. Dank der individualisierbaren globalen und detaillierten Projekt-Dashboards können alle Beteiligten des Projektes von den Analysen profitieren und zu einer besseren Qualität beitragen.

Über die wenigen Probleme in der Praxis, die bei der Synchronisierung der lokalen Entwicklungsrichtlinien mit den serverseitigen Richtlinien auftraten, lässt sich vorerst hinwegsehen. Die jüngsten Annäherungen an die Roslyn-Analyzer durch SonarQube weisen in die richtige Richtung, wodurch noch einiges an Verbesserungen zu erwarten ist.

Bei Legacy-Systemen gilt es zu beachten, dass es bei einer Anzahl von mehr als 300 000 Issues zu Timeouts und OutOfMemory-Exceptions kommen kann. Hier sollte man sich schrittweise herantasten.

Für den Einsatz von SonarQube sprechen noch weitere Faktoren, wie das verbesserte Wissen über Qualitätsprobleme der Entwickler und die frühzeitige Erkennung von Qualitätsmängeln. Beides vermeidet Wartungsarbeiten, die stets mit erhöhten Kosten in Verbindung stehen. Wann starten Sie Ihre erste Code-Analyse? ■

[1] SonarQube, www.sonarqube.org
[2] SonarQube – Architektur, www.dotnetpro.de/SL1701SonarQube1
[3] SonarQube Quality Profiles, www.dotnetpro.de/SL1701SonarQube2
[4] Quality Gate, www.dotnetpro.de/SL1701SonarQube3
[5] CI-Integration, www.dotnetpro.de/SL1701SonarQube4
[6] Authorisation, www.dotnetpro.de/SL1701SonarQube5
[7] Active-Directory-Plug-in, www.dotnetpro.de/SL1701SonarQube6
[8] LDAP-Plug-in, www.dotnetpro.de/SL1701SonarQube7
[9] ReSharper-Integration, www.dotnetpro.de/SL1701SonarQube8
[10] SonarQube Roslyn SDK, www.dotnetpro.de/SL1701SonarQube9
[11] SonarLint, www.sonarlint.org/visualstudio

Lukas Schwendemann
ist Consultant bei der AIT GmbH & Co. KG. Seine Schwerpunkte liegen im ALM- und Prozess-Consulting und in der Entwicklung von Anwendungen mit Microsoft .NET. Sie erreichen ihn unter **Lukas.Schwendemann@aitgmbh.de.**

dnpCode A1701SonarQube