Sonar

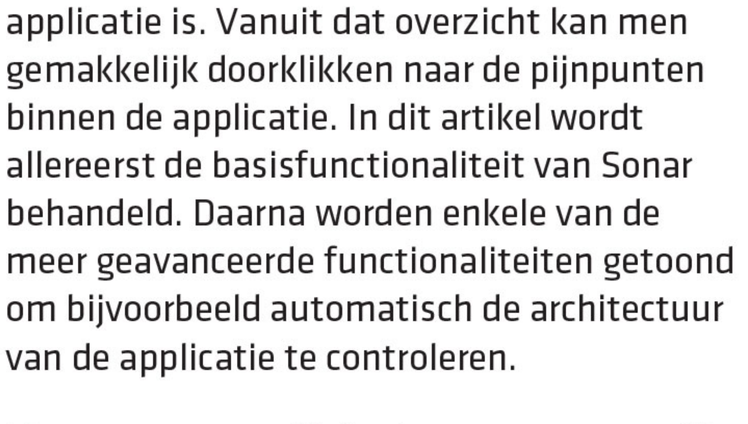
Kwaliteitsinjectie voor applicaties

Code kwaliteit is erg belangrijk om succesvolle software op te leveren. Het is noodzakelijk dat ontwikkelaars zo snel mogelijk feedback krijgen op de code die ze schrijven. Sonar, een open platform voor het automatisch analyseren van code-kwaliteit, biedt niet alleen inzicht in de huidige kwaliteit, maar toont ook het verloop hiervan.

Met behulp van Sonar kunnen ontwikkelaars goed zien wat de algehele kwaliteit van de applicatie is. Vanuit dat overzicht kan men gemakkelijk doorklikken naar de pijnpunten binnen de applicatie. In dit artikel wordt allereerst de basisfunctionaliteit van Sonar behandeld. Daarna worden enkele van de meer geavanceerde functionaliteiten getoond om bijvoorbeeld automatisch de architectuur van de applicatie te controleren.

Betere ontwikkelaars en applicaties met behulp van Sonar

Om de focus op kwaliteit zo hoog mogelijk te krijgen, is het van belang dat iedereen kwaliteit als iets essentieels ziet voor een succesvolle oplevering. Vaak probeert men kwaliteit af te dwingen door bepaalde eisen aan de code te stellen. Het aantal 'major Checkstyle' violations moet bijvoorbeeld nul zijn en de coverage minimaal 80 procent. De ontwikkelaars zien dit vervolgens als een verplichting en proberen aan de eisen te voldoen 'omdat het moet'. Het zou echter beter zijn als ontwikkelaars uit zichzelf proberen om kwalitatief betere code te schrijven. Sonar biedt de ontwikkelaars hierbij hulp door aan te geven waar er bijvoorbeeld violations in de code zijn, waar men afwijkt van object georienteerde best practices, waar de architectuur niet gevolgd wordt etcetera.



Figuur 1: De zeven assen van code kwaliteit die Sonar ondersteund.

Sonar

Wat is Sonar nu eigenlijk? Zoals op de website te lezen valt is Sonar een open platform voor het managen van code kwaliteit. Sonar analyseert de code en verifieert die op basis van de zeven assen van code kwaliteit die in figuur 1 te zien zijn. Sonar en de meeste plugins voor Sonar zijn als opensource software beschikbaar. SonarSource, het bedrijf achter Sonar, verdient zijn geld met support en enkele betaalde plugins. De Sonar community is erg actief, van Sonar zelf zijn er ieder jaar ongeveer zes releases en er zijn inmiddels een heel aantal plugins beschikbaar. Op de mailinglijsten worden veel onderwerpen behandeld en vragen worden meestal snel beantwoord.

Johan Janssen
is Java ontwikkelaar bij Info Support

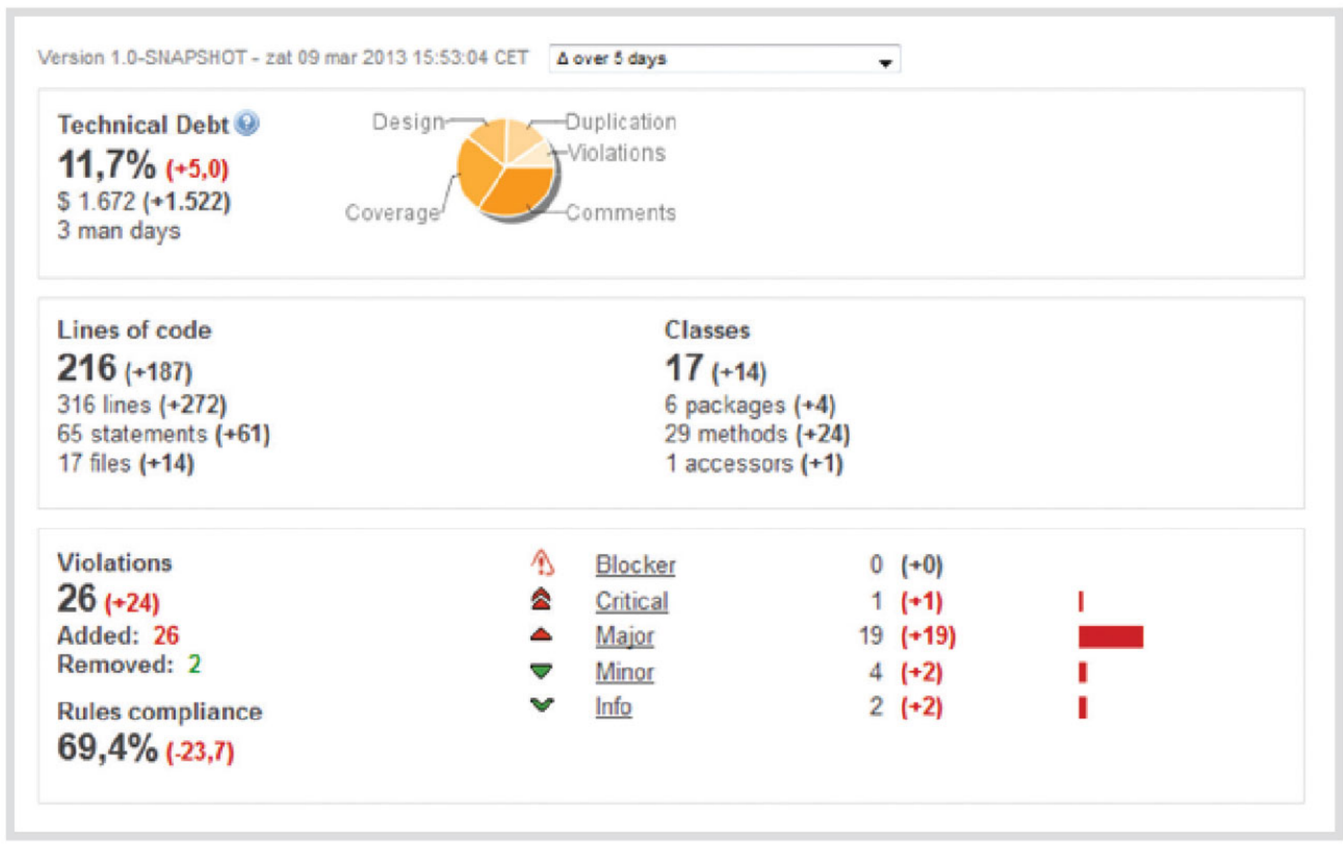
Sonar integreert met veel populaire componenten zoals Maven, Ant, Jenkins, Eclipse, Jira etc. Verder biedt Sonar ondersteuning voor verschillende talen zoals Java, C#, PHP en Cobol.

Het Sonar dashboard

Het Sonar dashboard is het belangrijkste onderdeel van Sonar, hierop kan men zien hoe het met de onderhoudbaarheid van het project gesteld is. In figuur 2 is een stuk van het Sonar dashboard te zien waar een aantal metingen getoond worden. Het dashboard is te configureren zodat alleen de informatie getoond wordt die belangrijk is voor het project. Sonar werkt met Quality profiles. Daarin wordt ingesteld welke regels van bijvoorbeeld Checkstyle, PMD (een source analyzer) en Findbugs gecontroleerd moeten worden. Regels kunnen geselecteerd en bewerkt worden binnen een Quality profile. Als er een fout in de code gevonden wordt, dan levert dat een violation op. Op een violation kunnen verschillende acties uitgevoerd worden. Zo kan er commentaar op een violation gegeven worden en is het mogelijk om de violation aan iemand toe te kennen of af te vinken als false positive, waarmee de ontwikkelaar expliciet toont dat erover nagedacht is. Binnen het Quality profile kunnen ook alerts ingesteld worden. Op die manier kan er bijvoorbeeld een alert gemaakt worden, zodat er een duidelijke melding verschijnt als de unit test coverage minder dan 80 procent is.

Authenticatie en autorisatie

Om verschillende Sonar features zoals codereviews goed te kunnen benutten, is het noodzakelijk om eerst de security goed in te richten. Standaard is er binnen Sonar een admin account waarmee de beheertaken gedaan kunnen worden. Voor alle overige zaken, zoals het bekijken van projecten, is het niet nodig om in te loggen. Het is echter wel aan te raden om de beveiliging beter in te richten. Niet alleen kan op die manier sourcecode afgeschermd worden, maar gebruikers kunnen zich dan ook uniek identificeren in Sonar. Als iedere gebruiker met eigen gegevens inlogt, is het mogelijk om Sonar items zoals violations aan specifieke gebruikers toe te wijzen. Als iedereen inlogt, kan men ook zien wie een codereview heeft toegevoegd of wie een violation als false positive heeft aangevinkt.
Door in Sonar gebruikers en groepen te definieren, is het mogelijk om een betere beveiliging in te richten, maar het is aan te raden om gebruik te maken van de LDAP plugin. Met de LDAP plugin kan Sonar aan LDAP gekoppeld worden en hoeft er binnen Sonar geen gebruikersbeheer meer plaats te vinden.



Figuur 2: Sonar dashboard met differential view, hiermee is te zien hoe de code de afgelopen 5 dagen veranderd is.

Codereviews

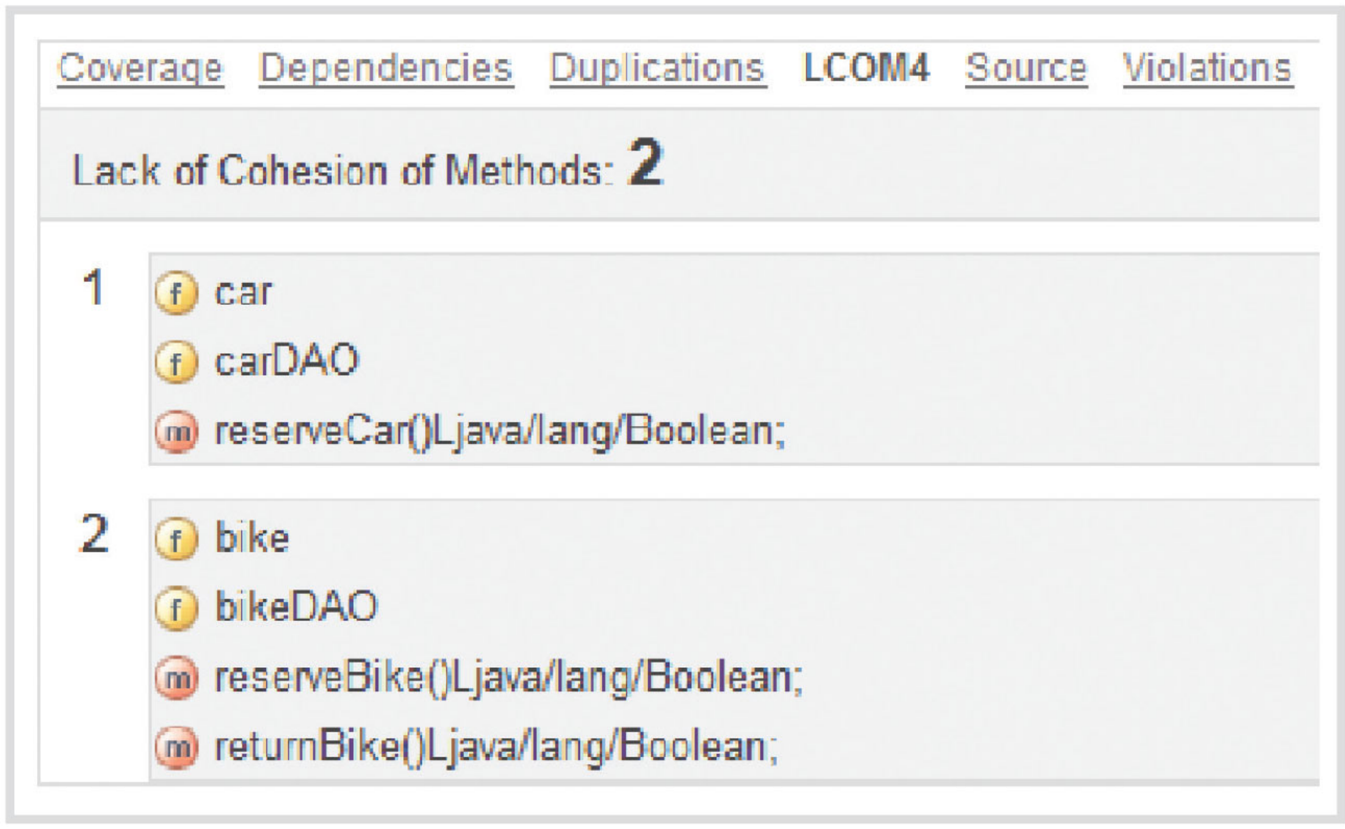
Er zijn allerlei tools ter ondersteuning van codereviews en het bijhouden van de gevonden fouten. Tools die hier regelmatig voor gebruikt worden zijn bijvoorbeeld Gerrit, Crucible en JIRA. Mocht er binnen het project geen behoefte zijn aan een aparte codereview tool, dan is het ook mogelijk om Sonar hiervoor te gebruiken. De eerste keer dient een admin een nieuwe violation aan te maken en die violation bijvoorbeeld de naam 'Codereview' te geven. Vervolgens kunnen alle ingelogde gebruikers codereviews aanmaken op een specifieke regel code. Deze codereviews hebben dezelfde functionaliteit als de violations. Zo is het mogelijk om een code review aan een persoon toe te kennen of om de code review als false positive te bestempelen.
Het is een best practice om gebruik te maken van de Sonar codereview functionaliteit, met name om kleine problemen aan te geven. Daardoor kunnen fouten niet per ongeluk vergeten worden.

Advanced Sonar: Ondersteuning voor het ontwerp en de architectuur

In de komende paragrafen worden een aantal meer geavanceerde features van Sonar behandeld. Sonar features zoals LCOM4, package cycles, architectural rules en libraries kunnen gebruikt worden om automatisch het ontwerp en de architectuur van de applicatie te controleren.

LCOM4

LCOM4 (Lack of Cohesion of Methods) is een metriek die kijkt of de methoden in een bepaalde klasse ook daadwerkelijk bij elkaar horen. Als de methoden bij elkaar horen,elkaar aanroepen en allemaal een relatie met elkaar hebben dan is de LCOM4 waarde 1. Wellicht zou je dan denken dat de LCOM4 waarde bij een klasse met alleen getters en setters erg hoog is, maar dit is echter niet het geval, omdat deze methoden eruit worden gefilterd.
In Figuur 3 is een klasse te zien met daarin 2 groepen methoden die niets met elkaar te maken hebben. Dit is een voorbeeld van een klasse met een LCOM4 waarde van 2. Deze klasse heeft 2 verschillende verantwoordelijkheden, het was waarschijnlijk beter geweest om daarvoor 2 aparte klassen te maken.
Het is vooral belangrijk om naar de LCOM4 waarden te kijken om ervoor te zorgen dat iedere klasse zijn eigen verantwoordelijkheden heeft. Is dat niet het geval dan dient de code opgesplitst te worden.



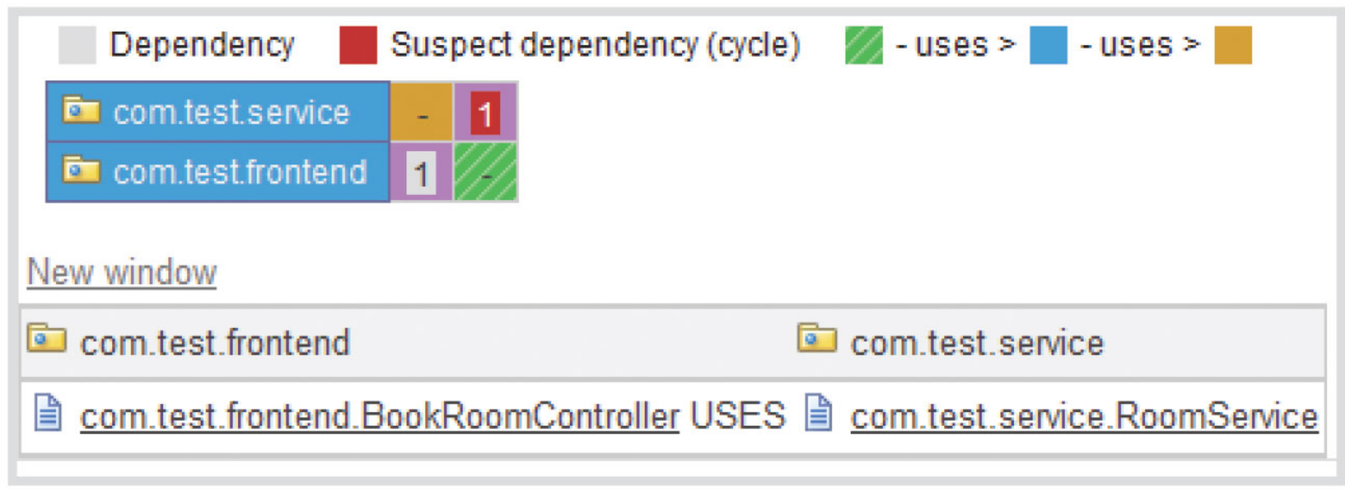
Figuur 3: Omdat de methoden van deze klasse in 2 groepen in te delen zijn is de LCOM4 waarde 2.

Package cycles

Package cycles komen voor als klassen uit twee packages elkaar aanroepen. Dit kan gebeuren door ontwerpfouten of door een verkeerde package indeling. Deze cycles zorgen ervoor dat de code slechter onderhoudbaar is.
Figuur 4 en Figuur 5 laten een voorbeeld van deze cycles zien in een zogenaamde Dependency Structure Matrix. Vanuit het frontend package wordt een klasse van het service package aangeroepen. Echter wordt vanuit het service package ook het frontend package aangeroepen.
Waarschijnlijk is het niet de bedoeling dat een service een controller aanroept, dus dit dient aangepast te worden. Let er wel op dat Sonar automatisch een van de twee aanroepen als 'suspect dependency' aanmerkt, maar het kan ook zo zijn dat de andere dependency incorrect is. In het voorbeeld geeft Sonar aan dat het gebruik van RoomService vanuit de BookRoomController waarschijnlijk niet correct is, terwijl deze aanroep juist wel correct is (zie figuur 4 en 5).
Het aantal package cycles zou nul moeten zijn. Op die manier is de code namelijk beter gestructureerd en makkelijker te lezen.

Architecture Rule Enginge

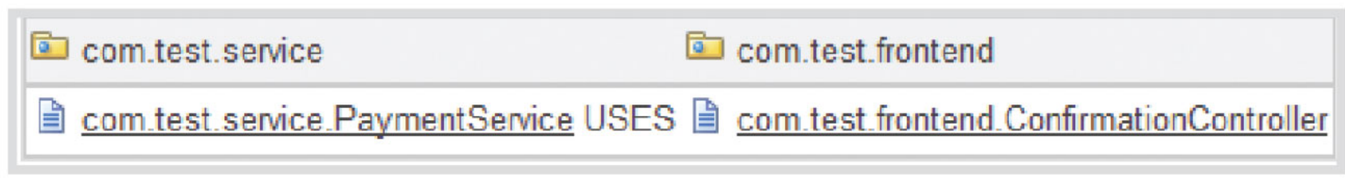
Architecture Rule Engine kan gebruikt worden om een bepaald ontwerp af te dwingen. Hiermee is het mogelijk om het voorbeeld van figuur 5, waarbij de PaymentService de ConfirmationController gebruikt, te verbieden.
Dat kan door een kopie te maken van de 'Architectural constraint' violation in het Quality profile. In deze regel kan aangegeven worden welke klassen niet gebruikt mogen worden vanuit bepaalde andere klassen. In Figuur 6 is een Architectural constraint te zien welke verbiedt dat klassen uit het service package de klassen uit het frontend package gebruiken. Door middel van deze architectuur regels controleren we automatisch of onze projecten voldoen aan de gekozen architectuur. Als een ontwikkelaar per ongeluk of expres afwijkt van deze architectuur dan kunnen we dat direct zien.



Figuur 4: Een klik op de 1 rechtsboven toont dat de BookRoomController de RoomService gebruikt

Het overzicht behouden

Hiervoor zijn allerlei features van Sonar getoond, maar hoe kun je het overzicht behouden? Wat doe je bij bestaande code met vele violations, en hoe monitor je de code kwaliteit over een langere periode met meerdere releases? Hoe verhoudt de kwaliteit van het project zich tot de kwaliteit van



Figuur 5: Een klik op de 1 linksonder toont dat de PaymentService de ConfirmationController gebruikt.

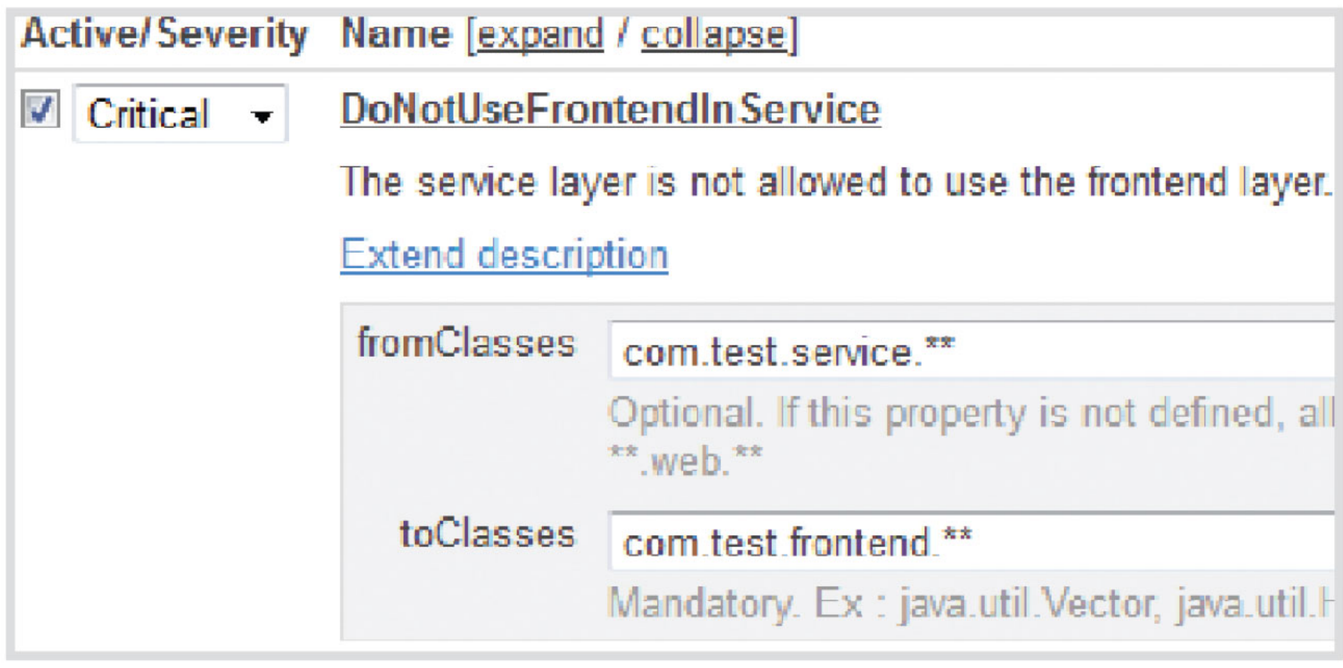
andere projecten? In de komende paragrafen wordt getoond hoe Sonar hiervoor het beste in te zetten is.

Totale kwaliteit van het project in één oogopslag

Sonar voert vele controles uit en het kan lastig zijn om in één oogopslag te zien wat de kwaliteit van een project is. Om een beter totaalbeeld te krijgen, kan er gebruik gemaakt worden van de Technical Debt plugin die in figuur 2 te zien is. De Technical Debt plugin berekent de kosten in dollars en mandagen om bepaalde kwaliteitsproblemen op te lossen. Voor deze berekening wordt onder meer gekeken naar het aantal violations. Op nieuwe projecten kan men dit makkelijk in de hand houden door violations en andere kwaliteits controles direct op te lossen. Daardoor blijft de technical debt laag en de onderhoudbaarheid van de code hoog. Als je echter een bestaand project hebt met vele violations en andere kwaliteitsproblemen dan kan de technical debt enorm zijn. Hoe je daar efficiënt mee om kan gaan binnen een bestaand project is te lezen in de volgende paragraaf.
Voorbeelden van andere plugins die gebruikt kunnen worden om een beeld te krijgen van de totale kwaliteit van het project zijn de SIG Maintainability Model plugin en de commerciële SQALE Quality model plugin.

Gebruik van Sonar voor bestaande code

Op een nieuw project is het relatief eenvoudig om Sonar in te zetten. Als een violation gevonden wordt, dan dient die zo snel mogelijk verholpen te worden. Maar hoe doe je dat als je een bestaand project krijgt? Wat als daar duizenden violations inzitten, waar de test coverage richting de nul gaat en waar nog veel meer mee mis is?
Ten eerste is er de mogelijkheid om aparte profielen te configureren met daarin minder controles. Op die manier kan de code stapsgewijs verbeterd worden. Ten tweede is het mogelijk om violations en code reviews in te delen in zogenaamde Action Plans. Daarmee kan een Action Plan voor de volgende release gemaakt worden, met daarin de items die voor die release opgelost moeten zijn. Ten derde heeft Sonar de Hotspots functionaliteit, waarmee gekeken kan worden welke klassen de meeste aandacht nodig hebben.
Tot slot is het mogelijk om de Cutoff plugin te gebruiken. Met de Cutoff plugin is het mogelijk om alleen files te analyseren die na een bepaalde datum zijn gewijzigd. Op die manier kan de kwaliteit van de nieuwe code beter gecontroleerd worden.



Figuur 6: Architectural constraint welke verbiedt dat klassen uit het service package klassen uit het frontend package gebruiken.

Measures

Met de 'measure' functionaliteit kan uitgebreid gezocht worden binnen de Sonar projecten. Zo is het bijvoorbeeld mogelijk om te zoeken op bepaalde metingen binnen projecten, maar er kan ook gezocht worden op metingen binnen files. Zo kan er gezocht wordt binnen projecten naar alle projecten die een coverage hebben van minder dan 80 procent en die meer dan 10000 violations hebben.

Conclusie

Sonar bevat erg veel features die ontwikkelaars kunnen gebruiken om betere code te schrijven en daardoor betere applicaties op te leveren. Er komen steeds weer nieuwe interessante onderdelen bij, welke gebruikt kunnen worden om beter te sturen op de onderhoudbaarheid van de applicaties. Zelf gebruiken we de tool op alle projecten. In het begin was men nogal sceptisch, maar inmiddels kan men niet meer zonder en wordt Sonar als een best practice gezien.
Sonar is een compleet platform voor het managen van code-kwaliteit en dankzij de vele interessante plugins een goede aanvulling op elk software-ontwikkelproces. ■

HET IS EEN BEST PRACTICE OM GEBRUIK TE MAKEN VAN DE SONAR CODEREVIEW FUNCTIONALITEIT, MET NAME OM KLEINE PROBLEMEN AAN TE GEVEN.

INFORMATIE

Officiële opensource site: **sonarsource.org**

Beschikbare Sonar plugins:
docs.codehaus.org/display/SONAR/Plugin+Library/

Sonar Confluence wiki:
docs.codehaus.org/display/SONAR/Documentation