Softwarequalität sicherstellen mit Sonar

Aktuelle Technologien zur Entwicklung verteilter Java-Anwendungen

Hochschule München

Michaela Lutz
München, den 07. Juni 2013

Gliederung

- Was ist Sonar?
- Komponenten von Sonar
- Installationsprozess
- Analyse
- Dashboards
- Metriken
- Hotspots
- Reviews
- Time Machine
- Fazit

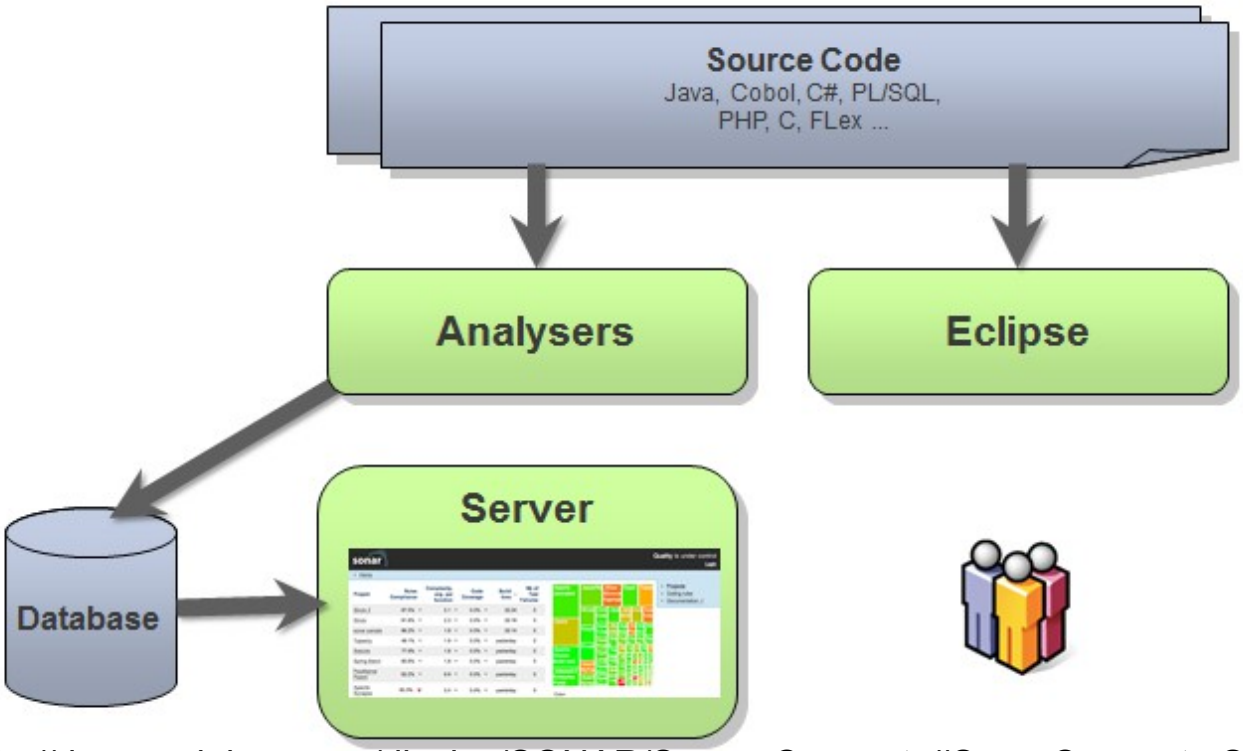
Was ist Sonar?

- Tool zum Verwalten und Visualisieren von Qualitäts-Metriken in Software-Projekten
- Verwaltung und Verbesserung der Software- und Codequalität steht im Vordergrund

- Sonar deckt die **7 Säulen der Qualitätssicherung** ab:

  - Architektur und Design
  - Komplexität
  - Kommentare
  - Potentielle Bugs
  - Duplikate
  - Code-Regeln
  - Unit tests

Aus welchen Komponenten besteht Sonar?



Quelle: http://docs.codehaus.org/display/SONAR/Sonar+Concepts#SonarConcepts-SonarArchitecture

Installationsprozess:

- Auf der Seite http://www.sonarsource.org/downloads/ finden sich der **Sonar-Server** und der **Sonar Runner** (Default Client) als zip-Datei zum Download.

- Wenn man seine Projekte mit **Maven** macht, braucht man keinen Client installieren!

- Einzelheiten zum Installationsprozess finden sich auf der Seite http://docs.codehaus.org/display/SONAR/Installing+Sonar

Wie wird eine Analyse gestartet?

- Auf der Konsole:

```
r158144:Verteilte_Java-Anwendungen michaelalutz$ cd shareit_sonar
r158144:shareit_sonar michaelalutz$ ls
Eclipse-Plugins.p2f     etc                     target
License.txt             pom.xml
README.md               src
r158144:shareit_sonar michaelalutz$ mvn sonar:sonar
```

Startbefehl für die Analyse!!

Startbefehl für Maven : mvn clean install -DskipTests=**true**
mvn sonar:sonar

```
[INFO] [10:06:50.786] Analysing /Users/michaelalutz/Documents/NewWorkspace/TheGarage/target/jacoco.
exec
[INFO] [10:06:50.930] No information about coverage per test.
[INFO] [10:06:50.931] Sensor JaCoCoSensor done: 150 ms
[INFO] [10:06:51.267] Execute decorators...
[INFO] [10:06:52.852] Persist graphs of components
[INFO] [10:06:52.940] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000
[INFO] [10:06:52.941] Executing post-job class org.sonar.plugins.core.batch.IndexProjectPostJob
[INFO] [10:06:53.013] Executing post-job class org.sonar.plugins.dbcleaner.ProjectPurgePostJob
[INFO] [10:06:53.025] -> Keep one snapshot per day between 2013-04-19 and 2013-05-16
[INFO] [10:06:53.026] <- Delete snapshot: 2013-05-15T12:14:35+0200 [38]
[INFO] [10:06:53.150] -> Keep one snapshot per week between 2012-05-18 and 2013-04-19
[INFO] [10:06:53.151] -> Keep one snapshot per month between 2008-05-23 and 2012-05-18
[INFO] [10:06:53.152] -> Delete data prior to: 2008-05-23
[INFO] [10:06:53.154] -> Clean Shareit Beispielimplementierung [id=4]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 33.459s
[INFO] Finished at: Fri May 17 10:06:53 CEST 2013
[INFO] Final Memory: 21M/313M
[INFO] ------------------------------------------------------------------------
```

Projekt wurde erfolgreich angelegt!

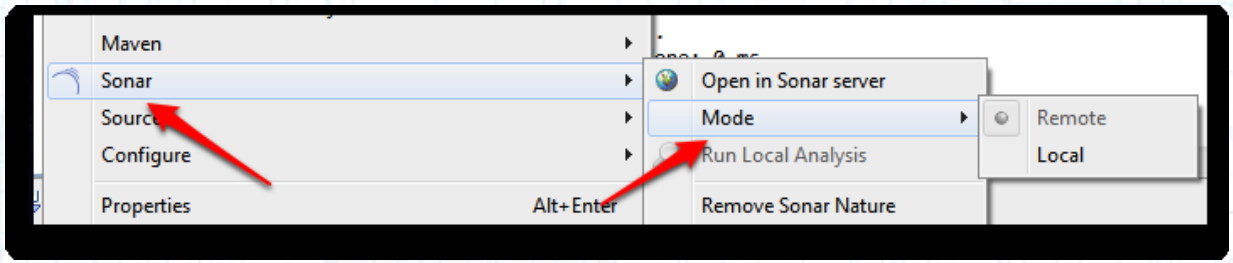
Analyse in starten im Terminal:

Demonstration am Rechner

Wie wird eine Analyse gestartet?

- In Eclipse:

1. Modus auswählen:



Quelle: http://docs.codehaus.org/display/SONAR/Browsing+Sonar+in+Eclipse

2. Analyse starten:

Quelle: http://docs.codehaus.org/display/SONAR/Browsing+Sonar+in+Eclipse

Analyse in starten im Eclipse:

Demonstration am Rechner

Wie wird der Sonar-Server gestartet?

- Der Server wird auf der Konsole gestartet!

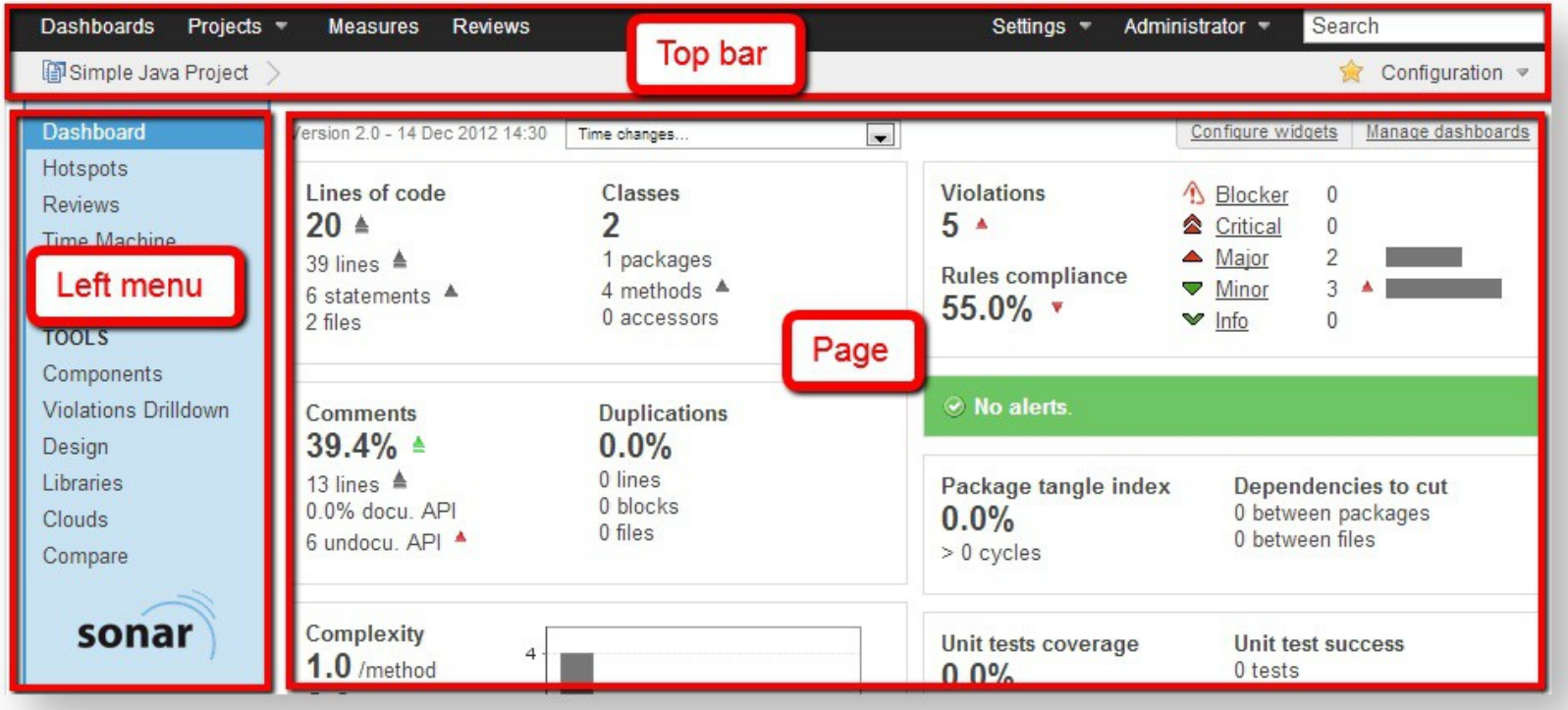
Startbefehl:

Linux/Mac OS: bin/<YOUR OS>/**sonar.sh start**
Windows: bin/windows-x86-32/**StartSonar.bat**

- Die Ergebnisse der Analyse werden auf der Web-Oberfläche des Sonar-Server angezeigt
- **http://localhost:9000**

  ↳ In Dashboards

Was ist ein Dashboard?



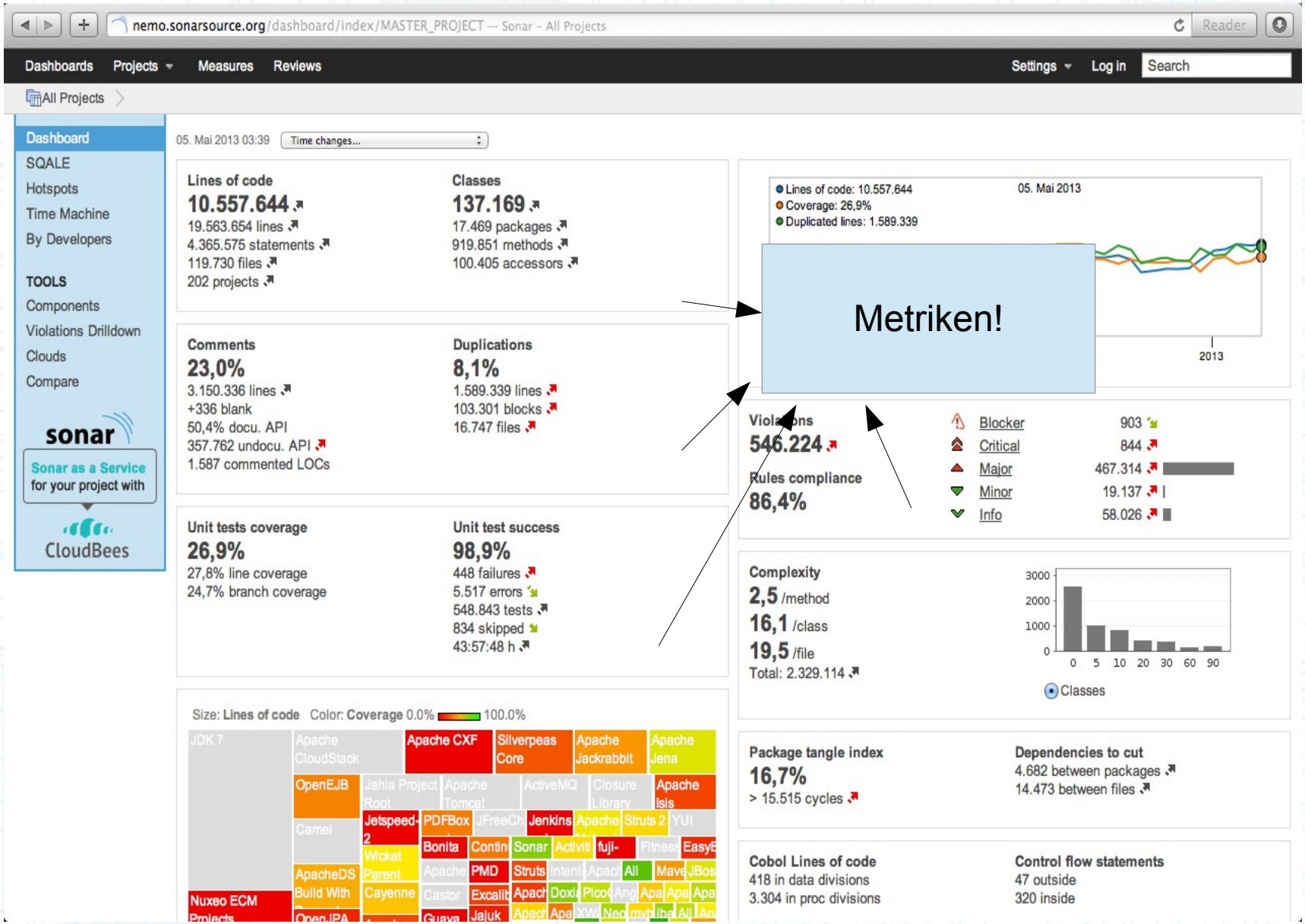
- **„Dashboard“** = Web-Seite auf der die Daten aus der Datenbank angezeigt werden
- Enthalten die Auswertungen der Analyse
- Einstiegspunkt in die Auswertungen eines Projekts

Was sind Metriken?

Metrik = (griech.) Messung

- In Sonar heißen die jeweiligen Messgrößen Metriken!
- Ergebnis der Analyse wird an den gesetzten Metriken und an den Verletzungen von Code-Regelungen festgemacht!
- Sonar unterscheidet 8 verschiedene Bereiche von Metriken:

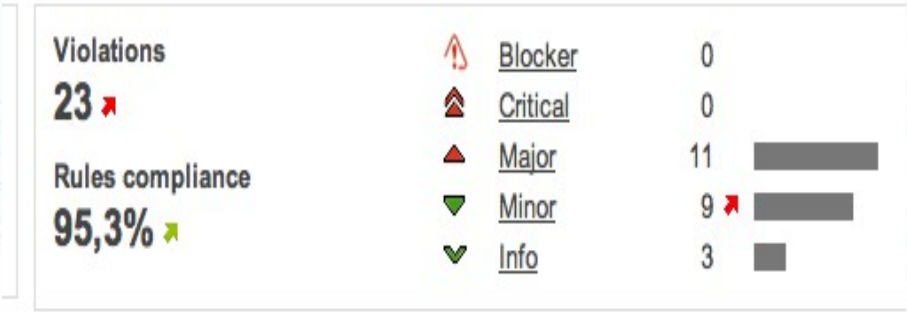
  Complexity, Design, Documentation, Duplications, Reviews, Rules, Size, Tests



Quelle: nemo.sonarsource.org/dashboard/index/MASTER_PROJECT

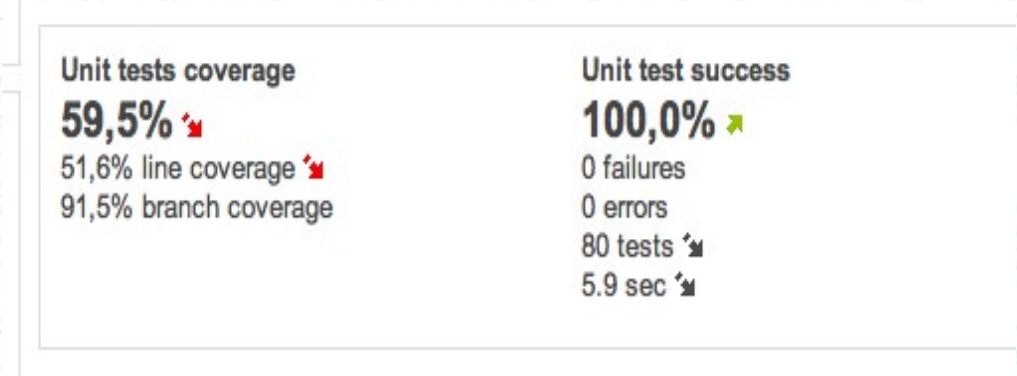
Welche Metriken sind besonders wichtig?

- Den Bereich den man sich als erstes im Projekt immer anschauen sollte ist **Rules**:



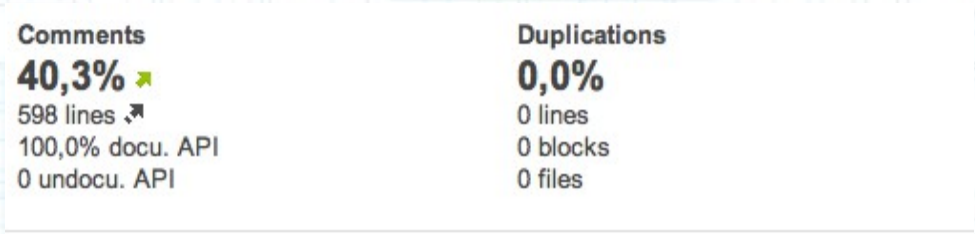
- **Violations** = Anzahl der Regelverletzungen
- **Rules compliance** = Prozentzahl der vorhandenen Regelverletzungen (0% = rot (schlecht) , 100% = grün (gut))
- **Severity** = wie schwerwiegend sind die Regelverletzungen (Skala : von Blocker bis Info)

- Als nächstes sollte man sich die **Unit tests** anschauen:



- **Unit tests coverage** = wie viel Quellcode wird durch die Unit tests abgedeckt
- **Line Coverage** = wurde jede Zeile während der Ausführung der Unit tests aufgerufen?
- **Branch Coverage** = prüft ob bei jedem boolschen Ausdruck beide Seiten im Test abgedeckt wurden
- **Unit test success**: wie viele Tests (%) sind erfolgreich durchgelaufen?
- Anzahl der Failures, Errors, Tests und Dauer des Testdurchlaufs

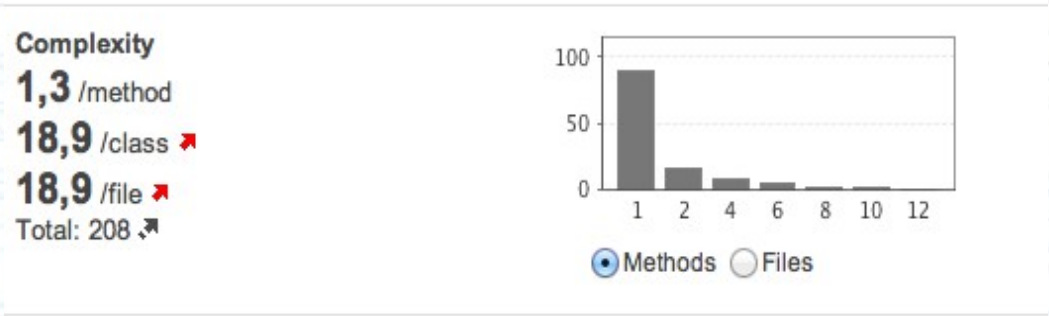
- Die Bereiche **Kommentare** und **Duplikate**:



- **Comments** = Anzahl der Kommentare in %
- **Lines** = Anzahl der Codezeilen incl. Leerzeilen
- **Docu. API** = % des dokumentierten Codes
- **Undoc. API** = % des nicht-dokumentierten Codes

- **Duplications** = Anzahl des duplizierten Codes in %
- Einteilt nach Anzahl der Duplikationen in **Codezeilen**, **Codeabsätzen** und **Dateien**

- Der Bereich **Complexity** :



- **Complexity** = Berechnet Komplexität für **Methoden**, **Klassen** und **Dateien**. Immer wenn der Kontrollfluss auseinander geht, erhöht sich die Komplexität.
- Beispiel:

```
public void process(Car myCar){              <- +1
    if(myCar.isNotMine()){                   <- +1
        return;                              <- +1
    }
    car.paint("red");
    car.changeWheel();
    while(car.hasGazol() && car.getDriver().isNotStressed()){    <- +2
        car.drive();
    }
    return;
}
```

Ergebnisse der Analyse in Sonar betrachten:

Demonstration am Rechner

Was sind Hotspots?

- Stellen im Quellcode an denen sich Violations häufen:
- **Hotspots** werden in Bereiche aufgeteilt :

  - **Most violated rules** = Regelverletzungen die gehäuft auftreten
  - **Most violated ressorces** = Klassen in denen gehäuft Violations auftreten
  - **Hotspots by Unit tests duration** = Zeit die benötigt wurde um alle unit tests durchzulaufen
  - **Hotspots by Uncovered lines** = Anzahl der Zeilen, die nicht durch unit tests abgedeckt wurden
  - **Hotspots by Complexity** und **Complexity/methods**
  - **Hotspots by Duplicated lines**
  - **Hotspots by Public undocumented API**

Wie wird eine Violation behandelt?

Demonstration am Rechner

Was ist eine Review?

- Listet eine Historie aller Regelverletzungen im Projekt auf
- Managet die Abarbeitung der ganzen Violations

- Abarbeitung wir nach **3 Kategorien** geordnet:

1. Bedrohungen die **sofort** behoben werden müssen
2. Bedrohungen die **im nächsten Sprint** behoben werden
3. Bedrohungen die im Auge behalten, aber noch **nicht behandelt** werden

Wie wird eine Review erstellt – Sonar Server?

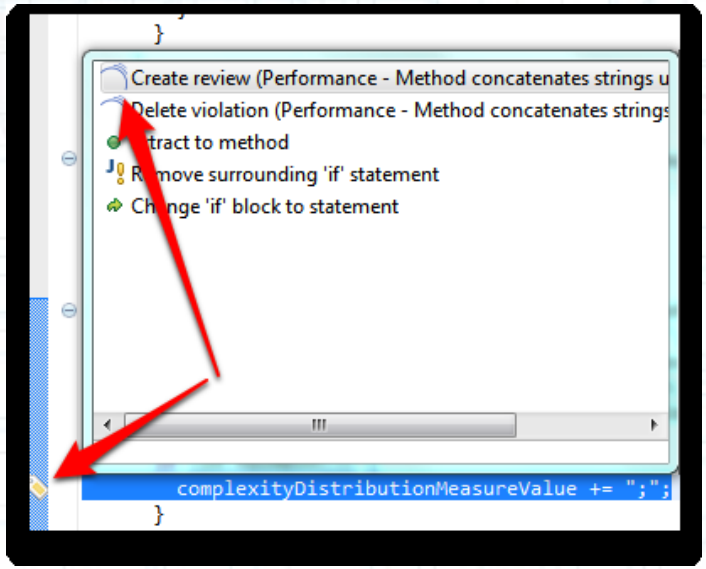
Demonstration am Rechner

Wie wird eine Review in Eclipse erstellt?

- 2 Möglichkeiten:

  1. Ein Klick auf das Sonar-Zeichen links außen neben der Codezeile

  => „Create Review“ auswählen

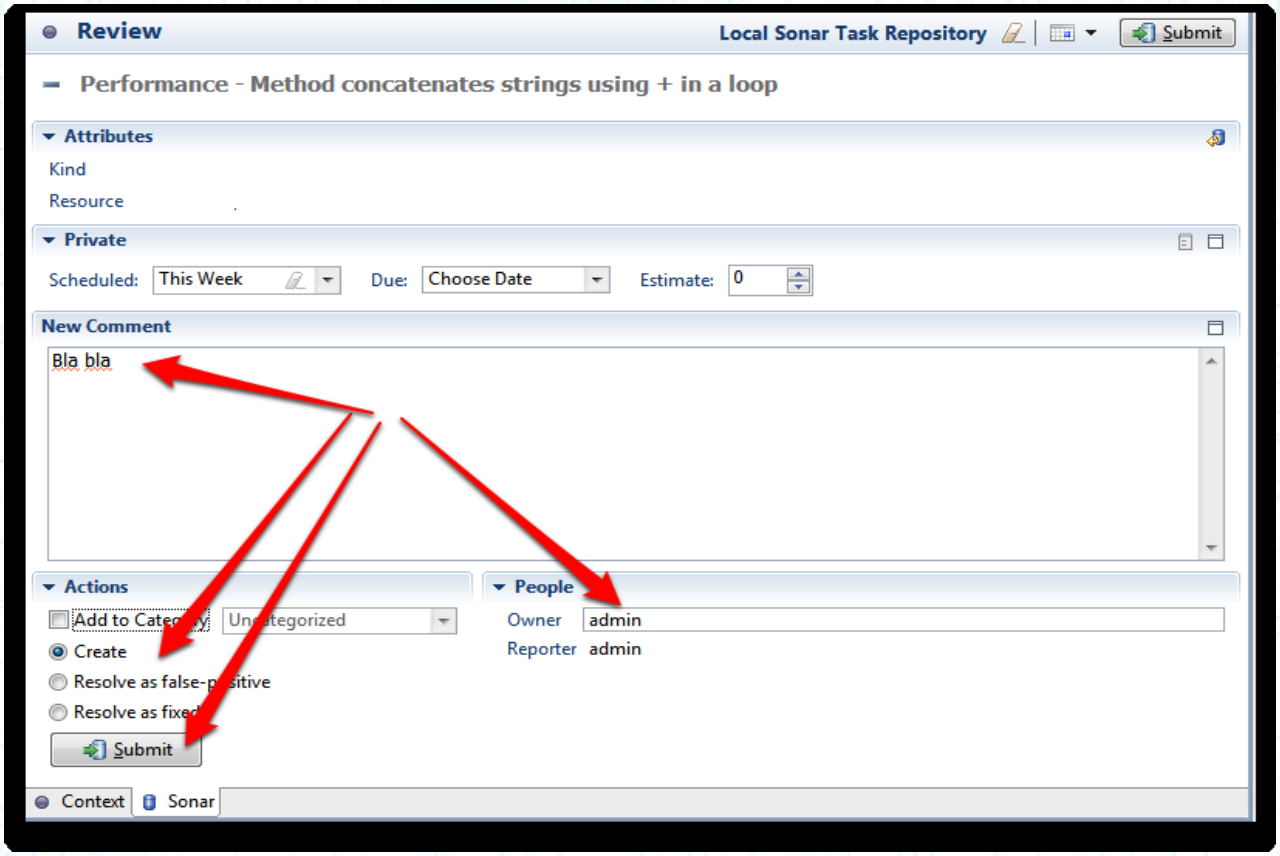


Quelle: http://docs.codehaus.org/display/SONAR/Working+with+Sonar+in+Eclipse#WorkingwithSonarinEclipse-WorkingthroughReviews

2. Direktes anlegen von der Violation – Ansicht aus:

   => Rechtsklick => „New Task from Marker“

- In beiden Fällen erscheint ein neuer Task angelegt:



Quelle: http://docs.codehaus.org/display/SONAR/Working+with+Sonar+in+Eclipse#WorkingwithSonarinEclipse-WorkingthroughReviews

Wie wird eine Review erstellt - Eclipse?

Demonstration am Rechner

Was ist die Time-Machine?

- Verwaltet die getätigten Analysen
- Zeigt standardmäßig immer die erste, die vorletzte und die letzte Analyse im Vergleich an
- Im **Timeline-Wigdet** können die Daten graphisch betrachtet werden
- In dem **History-Table-Wigdet** können von bis zu 10 Metriken die historischen Daten angezeigt werden

Was zeigt die Time-Machine an Informationen?

Demonstration am Rechner

Fazit:

- **Allgemein:**

  + praktisch zum Verwalten mehrerer Projektmitglieder

  - dauert bis man die vielen Metriken kennt

  - Anfangs muss viel Arbeit investiert werden, bis man mit allem vertraut ist und sinnvoll damit arbeiten kann

- **Persönlich:**

  + Time Machine und Hotspots sind sehr praktisch

  - In Eclipse etwas aufwändiger bis alles läuft

  - Hat man schon andere Softwarequalitätstools in Eclipse laufen ist Sonar bis auf wenige Features fast überflüssig

Danke für eure Aufmerksamkeit!

- Noch Fragen??