

Verteilte Verarbeitung

Einschub

Continuous Integration

Kontinuierliche Integration nach Fowler

An important part of any software development process is getting **reliable builds** of the software.

Despite it's importance, we are often surprised when this isn't done.

We stress a **fully automated** and **reproducible build, including testing**, that **runs many times a day**.

This allows each developer to integrate daily thus reducing integration problems.

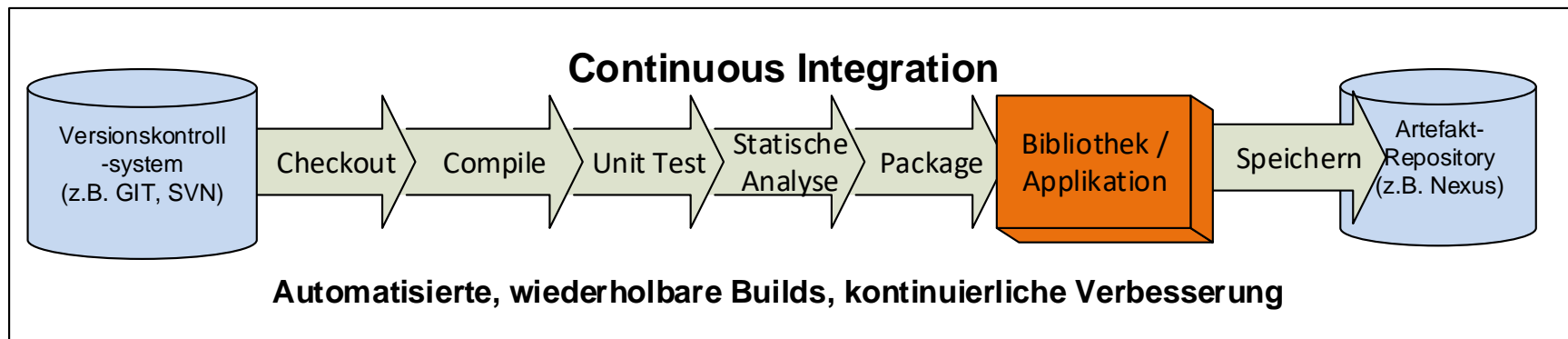


[Martin Fowler]

Bild aus Wikipedia,
https://en.wikipedia.org/wiki/Build_light_indicator

Continuous Integration: Automatisierung

Computer übernehmen Routinetätigkeiten, schnelles Feedback zur Qualität



- Checkout aus Versionskontrolle dann **automatisierter Build mit einem Knopfdruck** Inklusive: Datenbankschema (-Migration), Testdaten, ggf. Infrastruktur wie Docker-Container
- **Automatisiertes Testen**, soweit möglich und sinnvoll (TDD)
- Automatisierte Qualitätsmetrik / Statische Analyse
- Werkzeuge: z.B. Gitlab CI, Jenkins, Teamcity, ..., Teamscale, Sonar

Bei uns Gitlab CI Marktführer ist Jenkins



GitLab Pipelines - vorlesungen / apt2019

https://inf-git.fh-rosenheim.de/vorlesungen/apt2019/pipeline_schedules

GitLab Projects Groups Activity Milestones Snippets Search or jump to...

apt2019

Project Repository Issues Merge Requests CI / CD Pipelines Jobs Schedules Charts Schedules Security & Compliance Operations Packages

vorlesungen > apt2019 > Pipelines

All 26 Pending 0 Running 0 Finished 26 Branches Tags

Run Pipeline Clear Runner Caches CI Lint

Status	Pipeline	Triggerer	Commit	Stages	
passed	#22541 latest		master -> 06148039 success Build	✓ ✓	00:01:40 14 hours ago
failed	#22540		master -> d7f256b5 tach	✓ ✗	00:01:40 14 hours ago
failed	#22539		master -> 8b0c10af tach	✓ ✗	00:01:40 14 hours ago
failed	#22538		master -> 31ab885a tach	✓ ✗	00:01:39 14 hours ago
failed	#22537		master -> a0ff29f2 tach	✓ ✗	00:01:41 14 hours ago
failed	#22536 yml invalid error		master -> addc649a Update .gitlab-ci.yml		15 hours ago

https://inf-git.fh-rosenheim.de/vorlesungen/apt2019/pipeline_schedules

Pipeline in Gitlab gut sichtbar

`.gitlab-ci.yml`

```
stages:
- compile

build:
  stage: compile
  image: gradle:5.3.1-jdk8-alpine
  script:
    - echo "compiling"
    - cd katas
    - gradle clean
    - gradle assemble
  artifacts:
    paths: build/libs/*.jar
    expire_in: 1 week
```

...

success Build

🕒 2 jobs for `master` in 1 minute and 40 seconds (queued for 3 seconds)

📄 latest

🔗 06148039 ... 📄

Pipeline Jobs 2

Compile

Test

✅ build

✅ testing

QS-Werkzeuge in build.gradle

1. Haben wir genügend Testtreiber geschrieben?

```
jacoco { ...  
}
```

2. Finden sich im Code typische Fehler oder Verstösse gegen “guten” Code?

```
pmd { ...  
}
```

3. Finden sich im Code Kopien (Copy&Paste)

```
cpd { ...  
}
```

4. Einhalten der Coding Conventions

```
checkstyle { ...  
}
```

Weitere Werkzeuge in Java

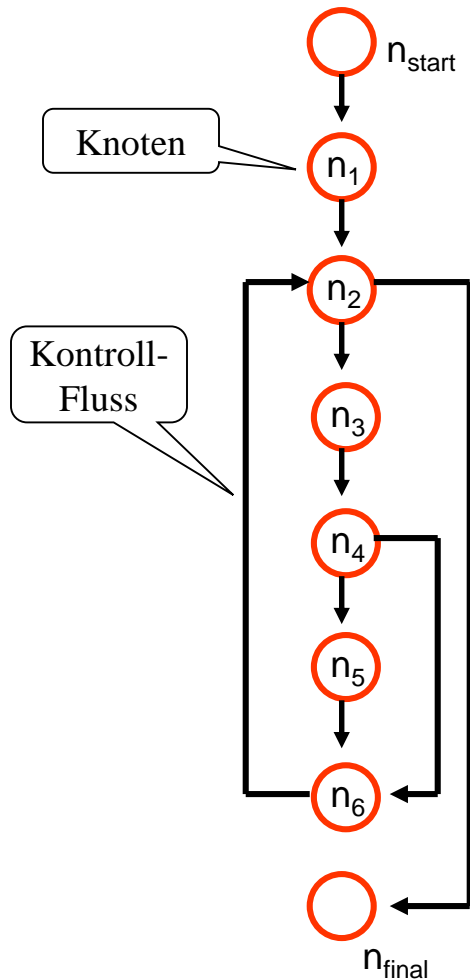
- Checkstyle: Einhalten der Coding Conventions
- Findbugs / SpotBugs: Gibt es im Code Anzeichen für typische Programmierfehler?
- Pitest: Mutationstest – Sind unsere Testfälle ausreichend oder entgehen uns eventuell Fehler im Code?
- Sync: Finden sich Sicherheitslücken im Code?
- OwaspDependencyCheck: Eingesetzte Bibliotheken sicher?
- ...

Wann sind wir mit dem Testen fertig?

Testüberdeckung

- Jede Anweisung mindestens einmal durchlaufen?
- Jeder Zweig mindestens einmal durchlaufen?
- Jeder Boolesche Ausdruck ganz und in Teilen true/false?

Anweisungs- und Zweigüberdeckung



Programmstück in C++

```
cin >> Zchn; // Zeichen einlesen
```

```
while ((Zchn >= 'A') && (Zchn <= 'Z')  
      && (Gesamtzahl < INT_MAX)) {
```

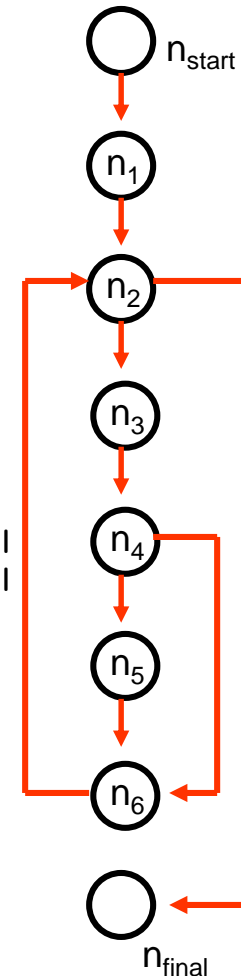
```
    Gesamtzahl = Gesamtzahl + 1;
```

```
    if ((Zchn == 'A') || (Zchn == 'E') ||  
        (Zchn == 'I') || (Zchn == 'O') ||  
        (Zchn == 'U'))
```

```
        VokalAnzahl = VokalAnzahl + 1;
```

```
    cin >> Zchn; // Zeichen einlesen
```

```
}
```



Bedingungsüberdeckung

■ 3 Varianten

- **Einfache Bedingungsüberdeckung:** Alle atomaren Bedingungen einmal true einmal false
- **Minimale Mehrfachüberdeckung:** Jede Bedingung muss einmal true oder false gewesen sein
- **Mehrfach-Bedingungsüberdeckung:** Alle Variationen der atomaren Bedingungen werden getestet

■ Beispiel

```
if ((Zchn == 'A') || (Zchn == 'E') ||  
    (Zchn == 'I') || (Zchn == 'O') ||  
    (Zchn == 'U'))
```

- Einfache Überdeckung:
Zchn in {A, E, I, O, U} -> 5 Testfälle
- Minimale- Mehrfachüberdeckung:
Zchn in {A, E, I, O, U, anderes Zeichen} -> 6 Testfälle
- Mehrfachüberdeckung nicht erreichbar

Code Coverage mit Jacoco

Anweisungs- und Zweigüberdeckung

- Ist der Code ausreichend getestet?
- Idee: Test Coverage messen
 - Rot: Nicht ausgeführt
 - Gelb: Halb ausgeführt (ein Zweig)
 - Grün: Ganz ausgeführt

```
35.  /*
36.   public static String of(final int value) {
37.   ◆ if (value < 1) {
38.       throw new NumberFormatException(
39.           "Value " + value + "must be greater t
40.       }
41.   ◆ if (value > 100) {
42.       throw new NumberFormatException(
43.           "Value " + value + "must be smaller t
44.       }
45.   final StringBuffer result = new StringBuffer();
46.
47.   ◆ if (value % 3 == 0) {
48.       result.append(FIZZ);
49.   }
50.   ◆ if (value % 5 == 0) {
51.       result.append(BUZZ);
52.   }
53.   ◆ if (value % 7 == 0) {
54.       result.append(WHIZZ);
55.   }
56.
57.   ◆ if (result.length() == 0) {
58.       result.append(Integer.toString(value));
59.   }
60.
61.   return result.toString();
62.   }
63. }
```

Ist der Code gut änderbar / verständlich?

Code Qualität

- Werden die Coding – Conventions eingehalten? (Java: Sun oder Google, eigene?)
- Werden typische Bug-Patterns vermieden?
- Wird ein guter Programmierstil gepflegt?

PMD



PMD is a *static source code analyzer*. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth.

PMD Konfiguration build.gradle

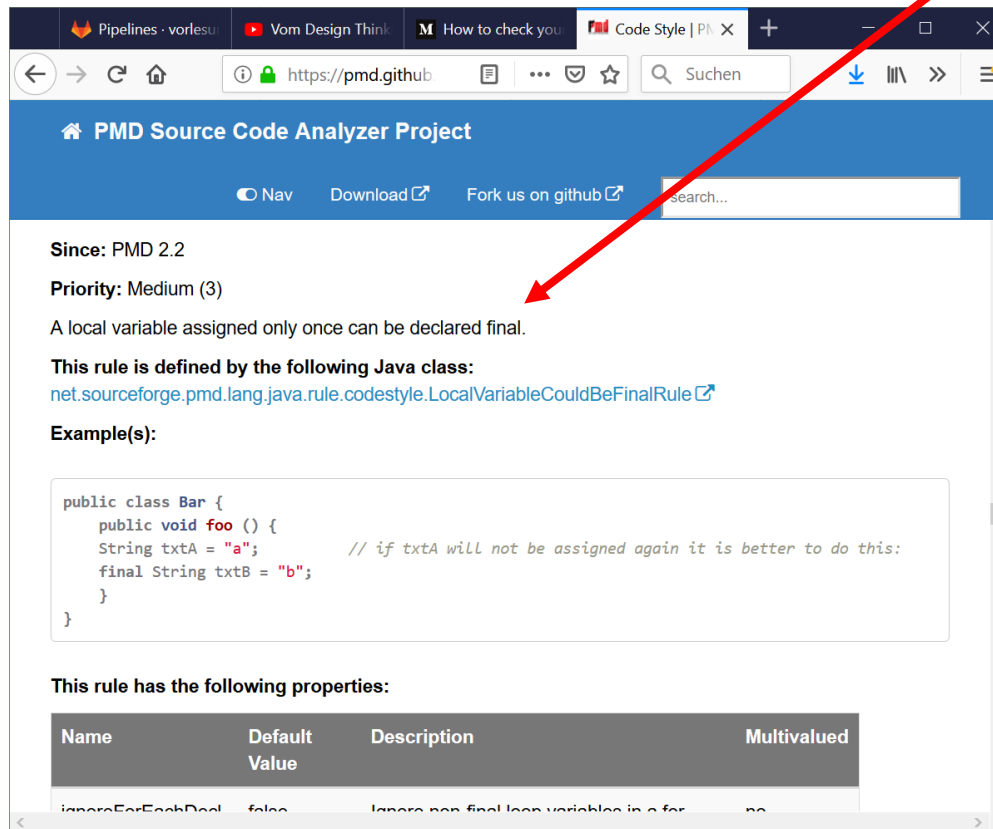
```
plugins {  
    id 'pmd'  
}  
pmd {  
    toolVersion = '6.18.0'  
    ignoreFailures = true  
    ruleSets = [  
        'category/java/errorprone.xml',  
        'category/java/codestyle.xml',  
        'category/java/multithreading.xml',  
        'category/java/bestpractices.xml',  
        'category/java/performance.xml',  
        'category/java/documentation.xml',  
        'category/java/design.xml'  
    ]  
}
```

PMD report

Problems found

#	File	Line	Problem
1	D:\Sandbox\apt2019\apt2019\katas\src\main\java\de\thron\inf\kata\ fizzbuzz\FizzBuzz.java	10	The utility class name 'FizzBuzz' doesn't match '[A-Z][a-zA-Z0-9]+(Utils? Helper)'
2	D:\Sandbox\apt2019\apt2019\katas\src\main\java\de\thron\inf\kata\ fizzbuzz\FizzBuzz.java	27	Comment is too large: Too many lines
3	D:\Sandbox\apt2019\apt2019\katas\src\main\java\de\thron\inf\kata\ fizzbuzz\FizzBuzz.java	40	Avoid using short method names
4	D:\Sandbox\apt2019\apt2019\katas\src\main\java\de\thron\inf\kata\ fizzbuzz\FizzBuzz.java	41	Avoid using Literals in Conditional Statements
5	D:\Sandbox\apt2019\apt2019\katas\src\main\java\de\thron\inf\kata\ fizzbuzz\FizzBuzz.java	45	Avoid using Literals in Conditional Statements
6	D:\Sandbox\apt2019\apt2019\katas\src\main\java\de\thron\inf\kata\ fizzbuzz\FizzBuzz.java	49	Local variable 'result' could be declared final

Configuration errors



Since: PMD 2.2

Priority: Medium (3)

A local variable assigned only once can be declared final.

This rule is defined by the following Java class:
[net.sourceforge.pmd.lang.java.rule.codestyle.LocalVariableCouldBeFinalRule](#)

Example(s):

```
public class Bar {
    public void foo () {
        String txtA = "a";        // if txtA will not be assigned again it is better to do this:
        final String txtB = "b";
    }
}
```

This rule has the following properties:

Name	Default Value	Description	Multivalued
ignoreForEachDecl	false	ignore non-final loop variables in a for	no

PMD Regeln im Überblick

https://pmd.github.io/latest/pmd_rules_java.html

Best Practices

Rules which enforce generally accepted best practices.

- **AbstractClassWithoutAbstractMethod**: The abstract class does not contain any abstract methods. An abstract class suggests an incomplete...
- **AccessorClassGeneration**: Instantiation by way of private constructors from outside of the constructor's class often causes...
- **AccessorMethodGeneration**: When accessing a private field / method from another class, the Java compiler will generate a acc...
- **ArrayIsStoredDirectly**: Constructors and methods receiving arrays should clone objects and store the copy. This prevents f...
- **AvoidMessageDigestField**: Declaring a MessageDigest instance as a field make this instance directly available to multiple t...
- **AvoidPrintStackTrace**: Avoid printStackTrace(); use a logger call instead.
- **AvoidReassigningLoopVariables**: Reassigning loop variables can lead to hard-to-find bugs. Prevent or limit how these variables ca...
- **AvoidReassigningParameters**: Reassigning values to incoming parameters is not recommended. Use temporary local variables inst...

Copy & Paste

Korrektheit des Codes – Bad Smells

Copy & Paste Code (Code Clone)

- Häufig (insbesondere in der Einarbeitung)
Copy & Paste Entwicklung
 - Funktionierender Code / ein Beispiel wird kopiert
 - Kopie wird ggf. angepasst
 - Variablen umbenennen
 - Algorithmus leicht ändern
 - Beispiele: GUI-Code, Datenbankzugriff, Netzwerkzugriff, XML-Parsing, ...
- Problem: Änderbarkeit
 - Kopien müssen möglicherweise gefunden und geändert werden
 - Im Einzelfall entscheiden, was angepasst werden muss
 - „Grep“ oder Clone-Suchprogramme helfen
- Mögliche Lösungen:
 - Basisklasse einführen
 - Delegation an ausgelagerte Klasse (Strategy Pattern, Utility-Klassen)

Beispiel für Bedeutung von Code Clones

```
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
    boolean found = false;
    StringBuffer res = new StringBuffer();
    if (elements != null) {
        Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
        for (int i = 0; i < elements.length; i++) {
            ModelElement el = elements[i];
            res.append(showElementLink(el)).append(HTML.LINE_BREAK);
            found = true;
        }
        Index.getInstance().resetCurrentRenderer();
    }
    if (!found && nomsg != null && nomsg.length() > 0) {
        res.append(HTML.italics(nomsg));
    }
    return res.toString();
}

// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
    boolean found = false;
    StringBuffer res = new StringBuffer();
    if (elements != null) {
        Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
        for (int i = 0; i < elements.length; i++) {
            ModelElement el = elements[i];
            res.append(showElementLink(el)).append(HTML.LINE_BREAK);
            found = true;
        }
        Index.getInstance().resetCurrentRenderer();
    }
    if (!found && nomsg.length() > 0) {
        res.append(HTML.italics(nomsg));
    }
    return res.toString();
}
```

Figure 1. Missing null check on right side can cause exception (Sysiphus).

Resultat der umfangreichen Studie von E.Jürgens (TUM) et al.

Wenn sich Code-Clone in einer oder mehr Zeilen unterscheiden, handelt es sich mit ca. 50%iger Wahrscheinlichkeit um einen Fehler!!

E. Juergens, et al: Do Code Clones Matter? ICSE 2009

Code Clone z.B. mit CPD (aus PMD)

- CPD =
Copy & Paste
Detector
- Diverse
Wrapper zum
Einbau in Gradle

CPD Report

Analysis run on 2020-04-15 16:41:03.509

Number of duplications: 4

Total number of duplicated lines: 237

Duplications

Duplication of 76 lines / 268 tokens

File	Line
D:\Sandbox\training\benekengerd\training.kata\src\main java\training\lecture\serialize\csv\Customer.java	13-88
D:\Sandbox\training\benekengerd\training.kata\src\main java\training\lecture\serialize\jser\Customer.java	15-90

Duplication of 62 lines / 243 tokens

File	Line
D:\Sandbox\training\benekengerd\training.kata\src\main java\training\lecture\serialize\jser\Address.java	6-67
D:\Sandbox\training\benekengerd\training.kata\src\main java\training\lecture\serialize\json\Address.java	5-66

Duplication of 60 lines / 241 tokens

File	Line
D:\Sandbox\training\benekengerd\training.kata\src\main java\training\lecture\serialize\csv\Address.java	9-68
D:\Sandbox\training\benekengerd\training.kata\src\main java\training\lecture\serialize\json\Address.java	7-66

Namenskonventionen

Namenskonventionen einhalten und prüfen ...

Beispiel: Java aus SUN Coding Conventions

- Namen von Methoden, Variablen, Attributen und Packages starten immer mit Kleinbuchstaben
- Namen von Klassen und Interfaces starten immer mit Großbuchstaben
- Konstanten nur Großbuchstaben (z.B. `PI`, `JAVA_HOME`, ...)
- Groß/Kleinbuchstaben-Mix, um Namen **lesbarer** zu machen (z.B. `isReadyForUse()`, oder `getFirstElement()` für Methoden oder `UniqueNameSignature` für eine Klasse) = ***Camel Case***
- Ganze Namen verwenden, welche die Variable, Klasse oder Methode sinnvoll beschreiben. Beispiel `firstName` anstelle von `fName` oder `xPosition` anstelle von `x1`.

Checkstyle

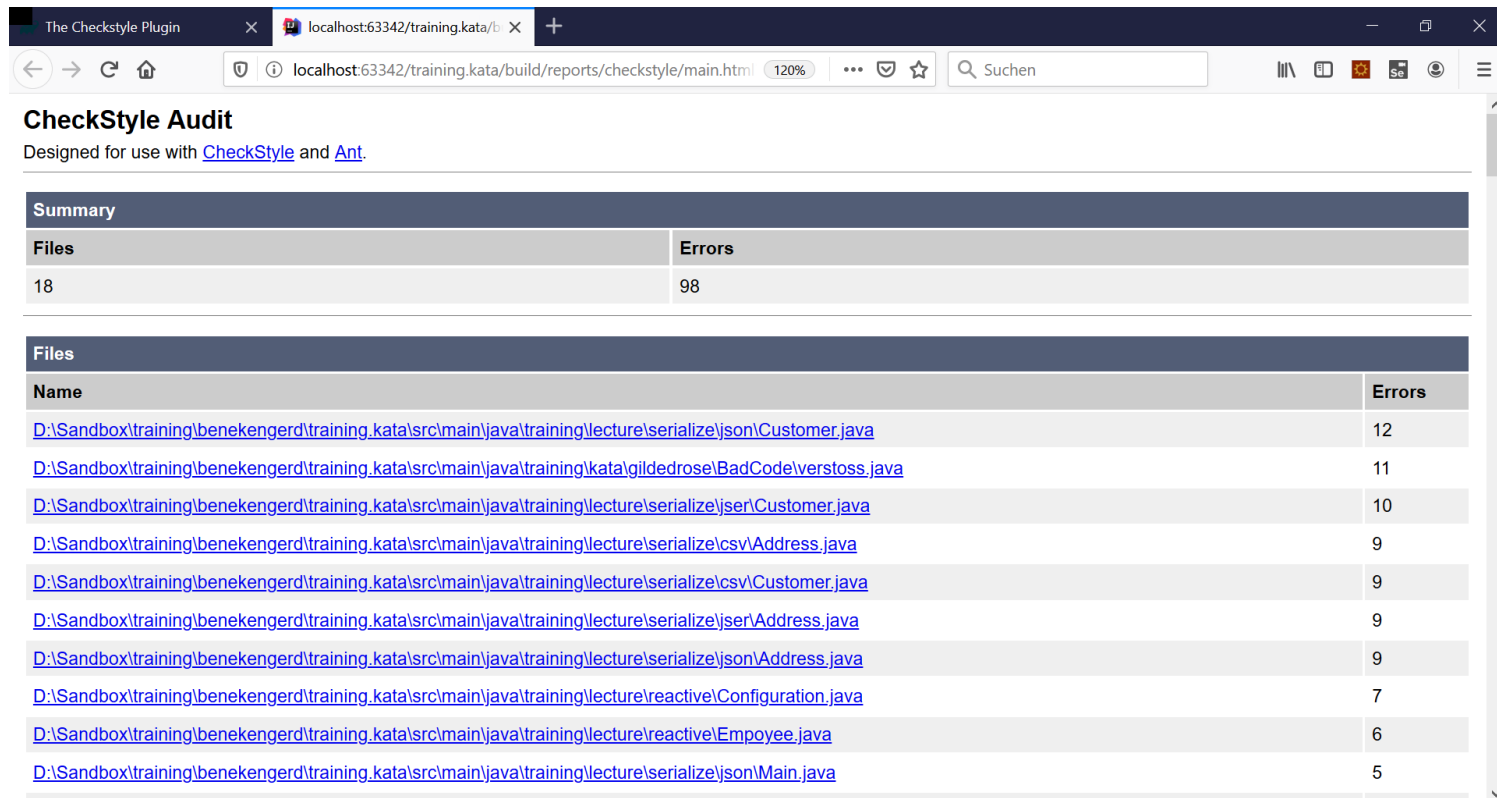
- Werkzeug zur Prüfung von Namenskonventionen

```
plugins {  
    id 'checkstyle'  
}  
  
checkstyle {  
    toolVersion '8.31'  
    config = rootProject.resources.text.fromFile(`  
        config/checkstyle/rules.xml`)  
}  
  
checkstyleMain {  
    source = 'src/main/java'  
}  
  
checkstyleTest {  
    source = 'src/test/java'  
}
```

Beispiel aus rules.xml (Google-Styleguide)

```
<module name="MemberName">
  <property name="format"
    value="^[a-z][a-z0-9][a-zA-Z0-9]*$"/>
  <message key="name.invalidPattern"
    value="Member name '{0}' must match pattern '{1}'."/>
</module>
<module name="ParameterName">
  <property name="format"
    value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
  <message key="name.invalidPattern"
    value="Parameter name '{0}' must match pattern '{1}'."/>
</module>
```


Checkstyle Report im Buildprozess



CheckStyle Audit
Designed for use with [CheckStyle](#) and [Ant](#).

Summary	
Files	Errors
18	98

Files	
Name	Errors
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\json\Customer.java	12
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\kata\gildedrose\BadCode\verstoss.java	11
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\json\Customer.java	10
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\csv\Address.java	9
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\csv\Customer.java	9
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\json\Address.java	9
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\json\Address.java	9
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\reactive\Configuration.java	7
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\reactive\Employee.java	6
D:\Sandbox\training\benekengerd\training.kata\src\main\java\training\lecture\serialize\json>Main.java	5

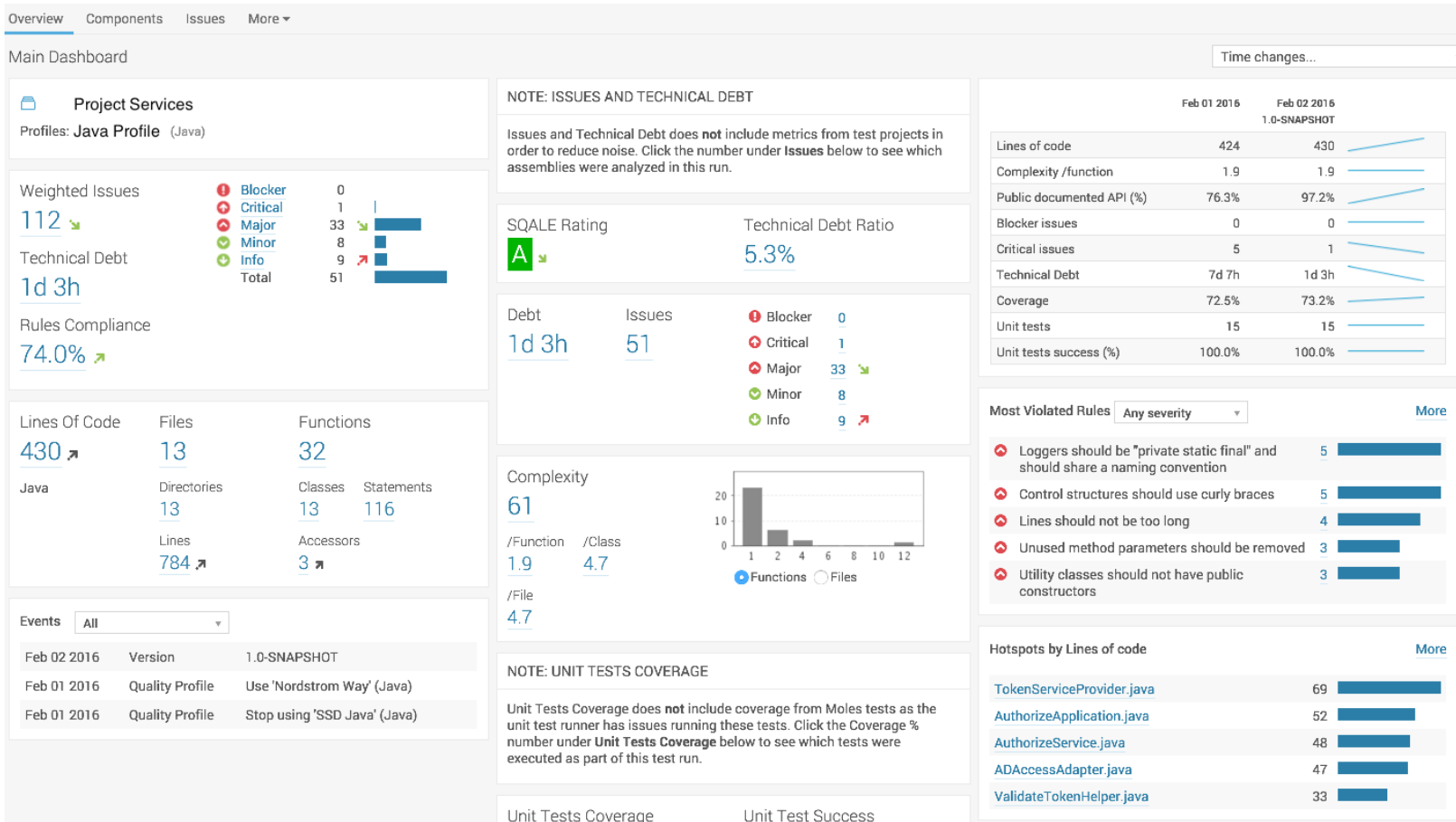
Qualitäts-Dashboard

SonarCube, ehem. Sonar



- Qualitätsdashboard
- Gut erweiterbar über (selbstgeschriebene) Plugins
- Integriert mit Maven, Gradle
- Speichert historische Qualitätsdaten, macht damit Änderungen z.B. in Testüberdeckung sichtbar
- Populär: SQALE Rating

Ziel: Qualitätsdashboard

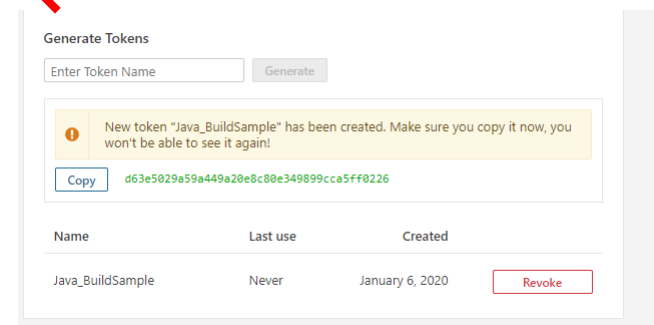


Beispiel mit Docker-Container von Thomas Mildner

<https://github.com/Thomas-Mildner/JavaBuildSample>

```
plugins { id "org.sonarqube" version "2.8" }

sonarqube {
    properties {
        property "sonar.projectKey", "Java_BuildSample"
        property "sonar.host.url", "http://localhost:9000"
        property "sonar.login",
            "d63e5029a59a449a20e8c80e349899cca5ff0226" }
    }
```



Buchempfehlungen

/2

