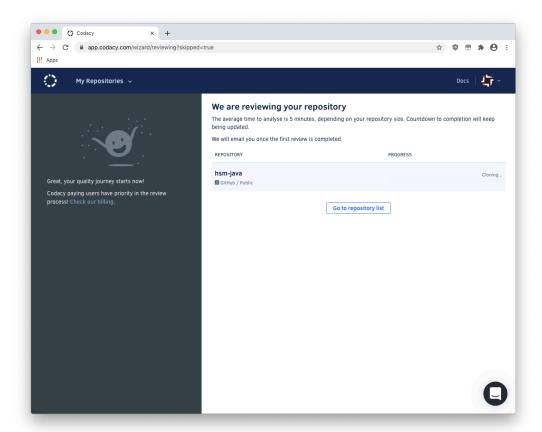
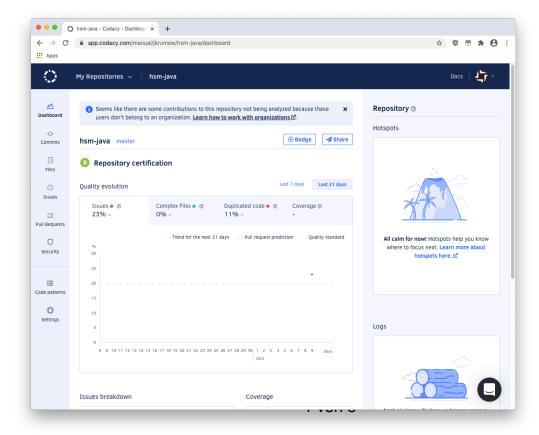
Analyse mit Codacy

Ich habe ein Github-Repository, an dem ich mitgewirkt habe, analysiert: https://github.com/jkrumow/hsm-java

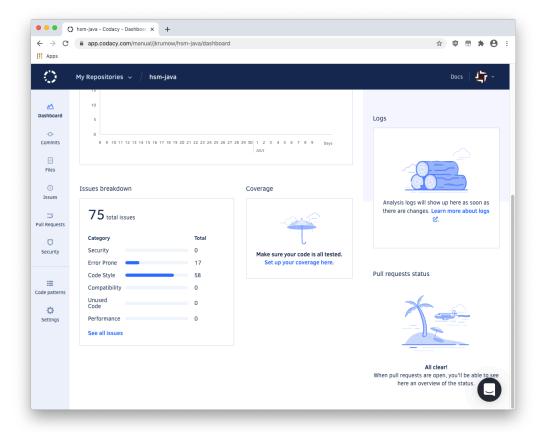
Login via Codacy und klonen des Repositories:



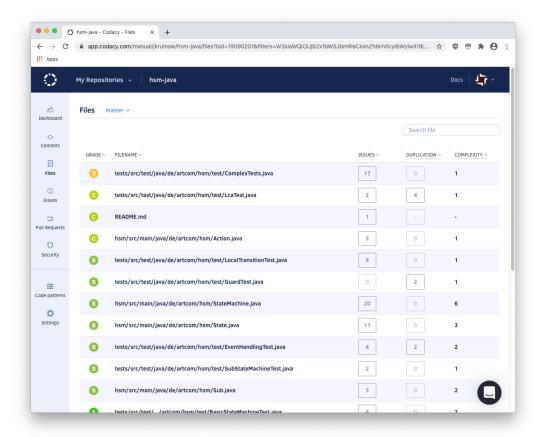
Übersicht über das Projekt auf dem Dashboard. Das Projekt hat die Schulnote B bekommen:



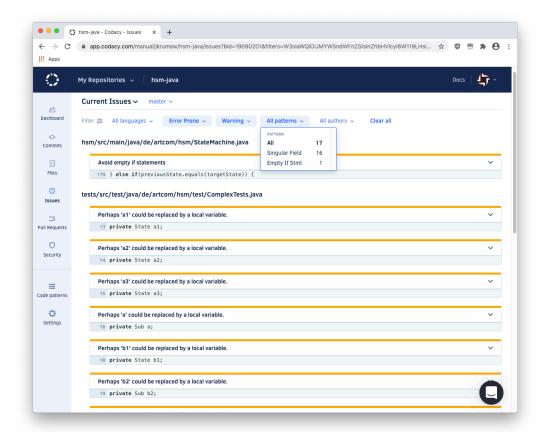
Zum Großteil wurden nur Probleme mit dem Code Style gefunden (58). Potenzielle Fehlerquellen wurden wenige ausgemacht (17):



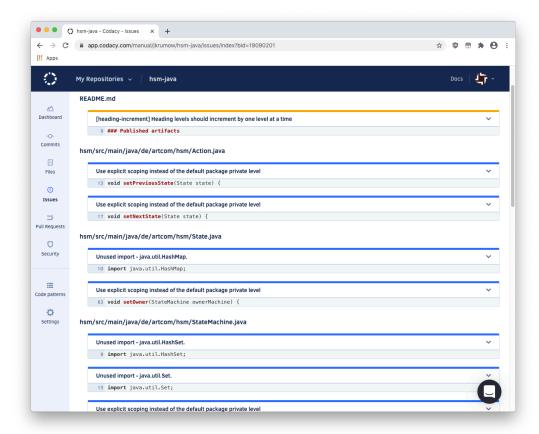
In der Dateiliste sieht man, dass sich die Benotung der Dateien vorwiegend zwischen A und B bewegt:



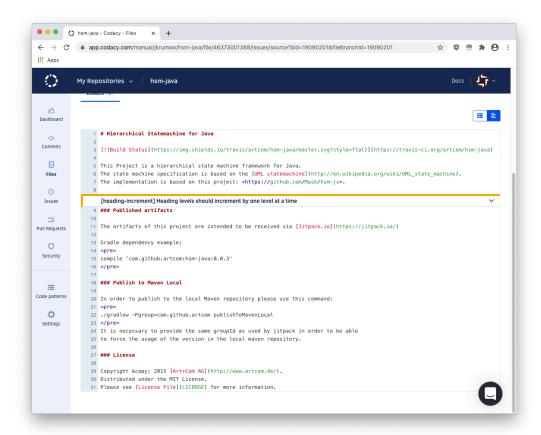
Lässt man sich die schwersten Mängel anzeigen (Error Prone, Warning) sieht man unvollständige Verzweigungen und unnötige Attribut-Deklarationen:



Unter den Coding Style Issues sind Sichtbarkeit der Attribute und unnötige Importe:



Auch die REAME-Datei wird geprüft. Hier wird bemängelt, dass die Ebenen der Überschriften nicht schrittweise durchlaufen werden, sondern sprunghaft:



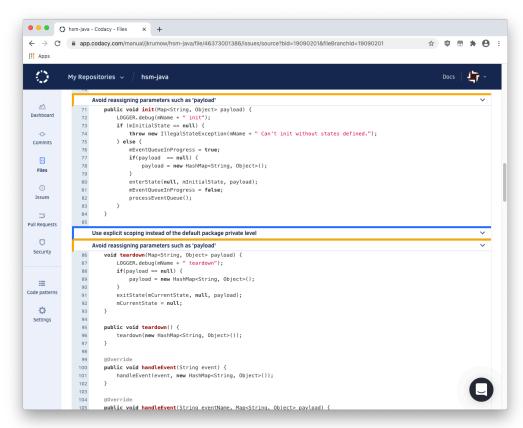
Hier die Details zu einem leeren if-Zweig:

```
hsm-java - Codacy - Files
 ← → C app.codacy.com/manual/jkrumow/hsm-java/file/46373001386/issues/source?bid=19090201&fileBranchid=19090201
                                                                                                                                                                                                                 ☆ ♥ 🖶 🖈 \varTheta
    0
                     My Repositories 🗸
                                                           hsm-java
                                                                                                                                                                                                                         Docs | 👆 ~
                                  Switch statements should have a default label
                                              switch (handler.getKind()) {
   case External:
     doExternalTransition(mCurrentState, targetState, action, event);
     break;
  -O-
Commits
                                                    case Local:
                                                            doLocalTransition(mCurrentState, targetState, action, event);
     o
     Files
                                                     oreak;
case Internal:
    executeAction(action, mCurrentState, targetState, event.getPayload());
    break;
     1
                           154
155
      =
                           156
157
158
159
160
161
162
163
164
165
166
167
170
171
172
173
174
175
176
                                      private void executeAction(Action action, State previousState, State targetState, Map<String, Object> payload) {
   if (action != null) {
      action.setPreviousState(previousState);
      action.setMextState(targetState);
      action.setPayload(payload);
}
 Pull Requests
     \Box
   Security
                                                     action.run();
     Code patterns
                                       private void doExternalTransition(State previousState, State targetState, Action action, Event event) {
                                               StateMachine lca = findLowestCommonAncestor(targetState);
     Φ
                                              lca.switchState(previousState, targetState, action, event.getPayload());
                                       private void doLocalTransition(State previousState, State targetState, Action action, Event event) {
   if(previousState.getDescendantStates().contains(targetState)) {
      StateMachine stateMachine = findMextStateMachineOnPathTo(targetState);
      stateMachine.switchState(previousState, targetState, action, event.getPayload());
   } else if(targetState.getDescendantStates().contains(previousState)) {
                                                    int targetLevel = targetState.getOwner().getPath().size();

StateMachine stateMachine = mPath.get(targetLevel);

stateMachine.switchState(previousState, targetState, action, event.getPayload());
                                  Avoid empty if statements
```

Hier wird ein Überschreiben eines Parameters bemängelt:



Unit-Tests werden ebenfalls geprüft. Hier fehlt die Assertion:

```
hsm-java - Codacy - Files
                                                        × +
 ← → C 🕯 app.codacy.com/manual/jkrumow/hsm-java/file/46373001376/issues/source?bid=19090201&fileBranchid=19090201 🖈 🕲 🖪 🕻 😉
Apps
    0
                                                                                                                                                                                                                     Docs | 👆 ~
                     My Repositories 🗸 / hsm-java
                           31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
                                      private StateMachine stateMachine;
                                            a1 = new State("a1");
a2 = new State("a2");
a3 = new State("a3");
  Commits
     o
                                             a = new Sub("a", a1, a2 , a3);
                                            b1 = new State("b1");

b21 = new State("b21");

b22 = new State("b22");

b2 = new Sub("b2", b21, b22);

b = new Sub("b", b1, b2);
     (1)
                                            c11 = new State("c11");

c12 = new State("c12");

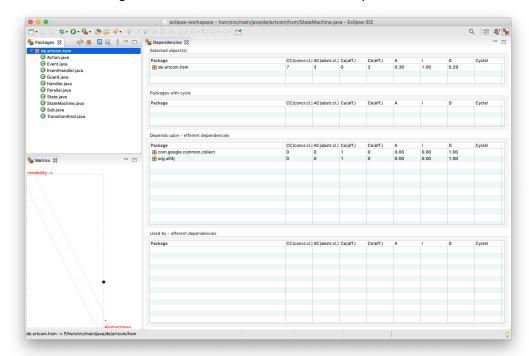
c21 = new State("c21");

c22 = new State("c22");
Pull Requests
     U
   Security
                                             CZZ = new Statet CzZ /,
c1Machine = new StateMachine(c11, c12);
c2Machine = new StateMachine(c21, c22);
c = new Parallel("c", c1Machine, c2Machine);
     \equiv
Code patterns
     Ö
                                 JUnit tests should include assert() or fail()
                        59
60
61
62
                                             //when:
stateMachine.init();
                                             stateMachine.teardown();
                                             //then: no exceptions
```

Analyse mit JDepend

Das selbe Projekt wurde jetzt mit dem JDepend-Plugin für Eclipse analysiert.

Zuerst das Package de.artcom.hsm in dem die HSM implementiert ist:



CC: Das Package enthält 7 Klassen

AC: Das Package enthält 1 abstrakte Klasse und 2 Interfaces

Ca: kein abhängendes Package

Ce: 2 Packages werden benötigt: com.google.common.collect und org.slf4j. Diese haben wiederum ein *Ca* von 1 da sie vom analysierten Package anhängen

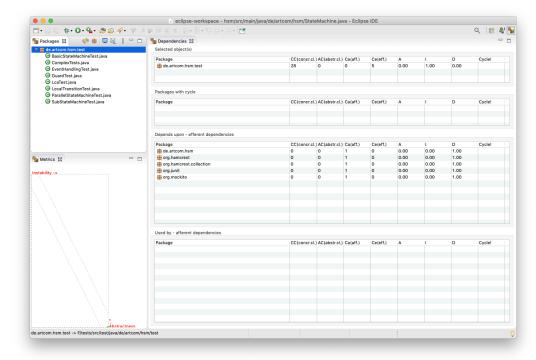
A: Ein drittel sind abstrakte Klassen und Interfaces: 0.3

I: hohe Instabilität, da abhängig von anderen Packages, jedoch keine gleichzeitigen eingehenden Abhängigkeiten: 1

D: Mit 0.29 relativ nahe. Hohe Tendenz zu Instabilität, mit niedriger Abstraktion

Cycle: zirkuläre Abhängigkeiten kommen nicht vor

Auch die Unit-Tests wurden analysiert:



CC: Das Package enthält 28 Klassen

AC: Das Package enthält keine abstrakte Klasse Interfaces

Ca: kein abhängendes Package

Ce: 5 Packages werden benötigt. Darunter die zu testende HSM und unsere Testumgebung: de.artcom.hsm, org.hamcrest, org.hamcrest.collection, org.junit, org.mockito. Diese Packages haben wiederum ein *Ca* von 1 da sie vom analysierten Package anhängen.

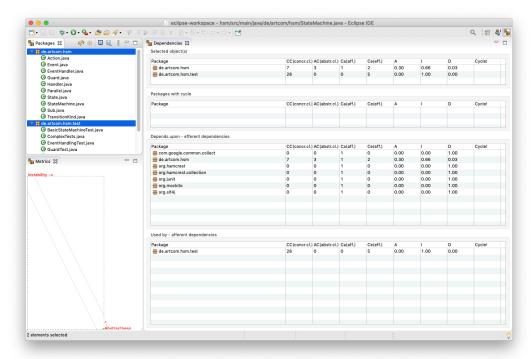
A: es gibt nur konkrete Klassen

I: hohe Instabilität, da abhängig von anderen Packages, jedoch keine gleichzeitigen eingehenden Abhängigkeiten: 1

D: Mit 0 sehr eindeutig. Hohe Instabilität ohne Abstraktion

Cycle: zirkuläre Abhängigkeiten kommen nicht vor

Analysiert man beide Packages gemeinsam gibt es folgende Werte:



Ca für die HSM steigt an, da das Test-Package diese nutzt. Dies senkt die Instabilität unter 0.7, was schlecht ist, da das HSM-Package nun eingehende und ausgehende Abhängigkeiten aufweist, was Veränderungen am Code erschwert. Es muss sowohl auf Änderungen in den importierten Packages geachtet werden und gleichzeitig die eigene API stabil gehalten werden. Dies ist insofern wichtig, da die HSM als Bibliothek für andere Projekte bereitgestellt werden soll.

Für diesem konkreten Fall ist es natürlich unerheblich, da im Test-Driven-Development beide Packages ständig aufeinander abgestimmt werden müssen. Die Werte für unser Test-Package bleiben unverändert.