Seminar Software Qualität im SS 2020

Vorbesprechung am 5. Februar

Dr. Maximilian Junker,

Markus Schnappinger,

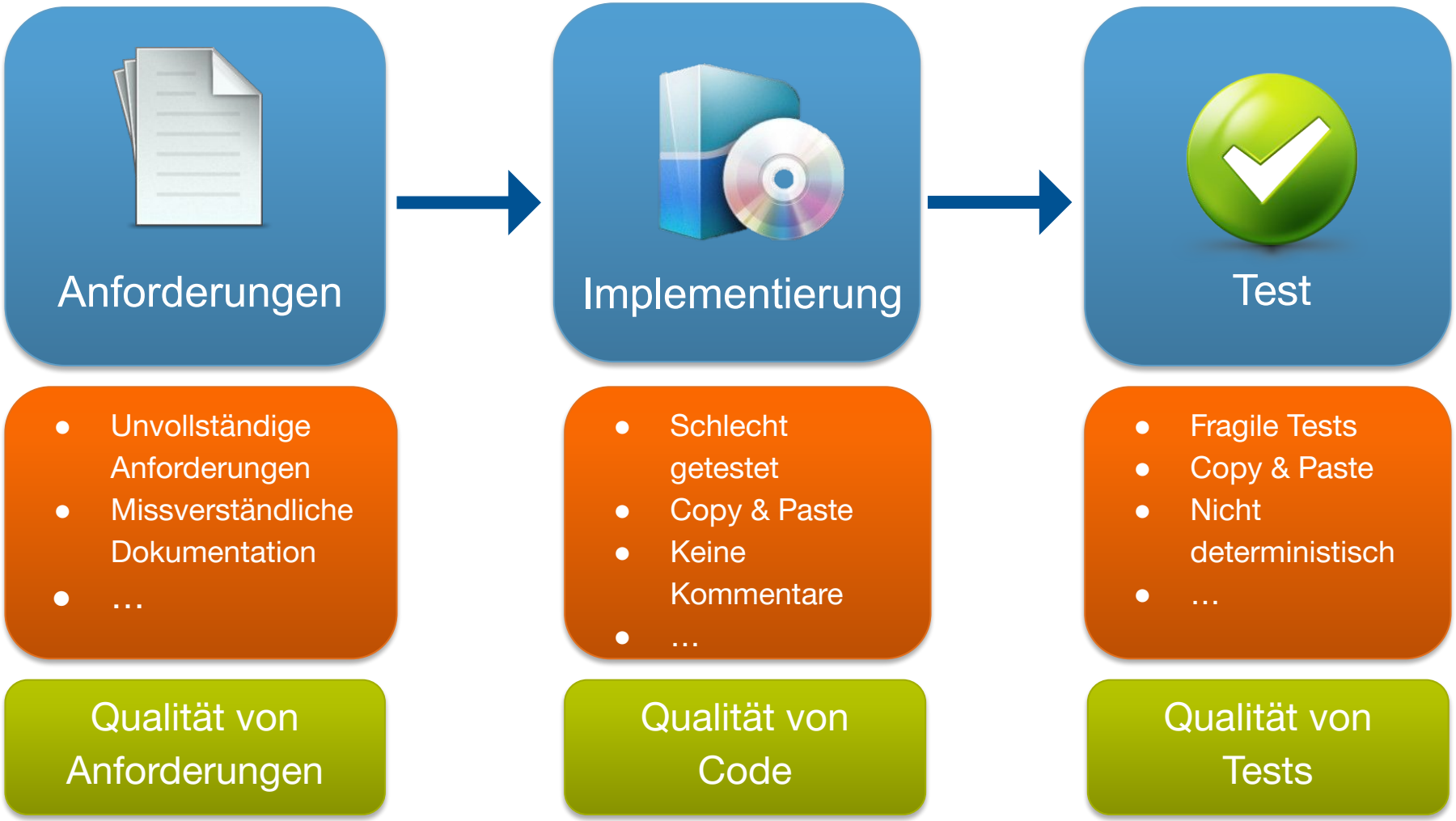
Dr. Benedikt Hauptmann,

Daniel Veihelmann,

Jakob Rott,

Daniel Elsner

Seminar Software Qualität



Teilnahme am Seminar

Mail an **junkerm@in.tum.de**

bis **Mittwoch, 17.02.**

Inhalt

- Motivation für Seminarteilnahme u. Themen (<1 Seite)
- Stand im Studium
- Erfahrungen (Arbeit, Praktikum, Open-Source-Projekt, …)
- **Themenwünsche** (sortiert nach Priorität)

- Eintrag im Matching System
- Rückmeldung Ende Februar über das Matching System
- Zuweisung Betreuer per Email durch uns

Ablauf

1. Ende Februar (nach Matching): Rückmeldung bei Betreuer und Festlegung des genauen Themas und Vorgehen
2. Semesteranfang (vstl. Mitte April): Auftaktveranstaltung
   - Überblick Software Qualität
   - Literaturrecherche
   - Effektives Präsentieren
3. Vstl. Juli 2020*: Seminar / Vorträge

   - Blockveranstaltung: 2-3 Tage Vorträge mit Diskussion

*Terminfindungen via Doodle

Seminararbeit und Vortrag

- **Ausarbeitung:** max. 15 Seiten (Text)
- **Vortrag (Blockveranstaltung):** 25min + 20min Diskussion

Inhalt

- **Theorie**
- **Anwendung des Seminarthemas in der Praxis**
  (Ergebnisse, Erfahrungen, Probleme & Grenzen)

Abgaben

- Seminararbeit (vollständige Fassung): 2 Wochen vor Vortrag
- Probevortrag (verpflichtend): 2 Wochen vor Vortrag
- Seminararbeit (Endfassung): 1 Woche nach Vortrag

Sonstiges

- Bewertung / Zusammensetzung der Note: 50/50 (Ausarbeitung/Vortrag)
- Vorlagen werden zu Verfügung gestellt
- Anwesenheitspflicht bei Blockveranstaltungen

Themen (I)

Qualitätssicherung von Anforderungen

- **Automatische Qualitätssicherung von Anforderungen**
  (Maximilian)

- **Requirements Detection: Wo im Dokument befinden sich Requirements?**
  (Maximilian)

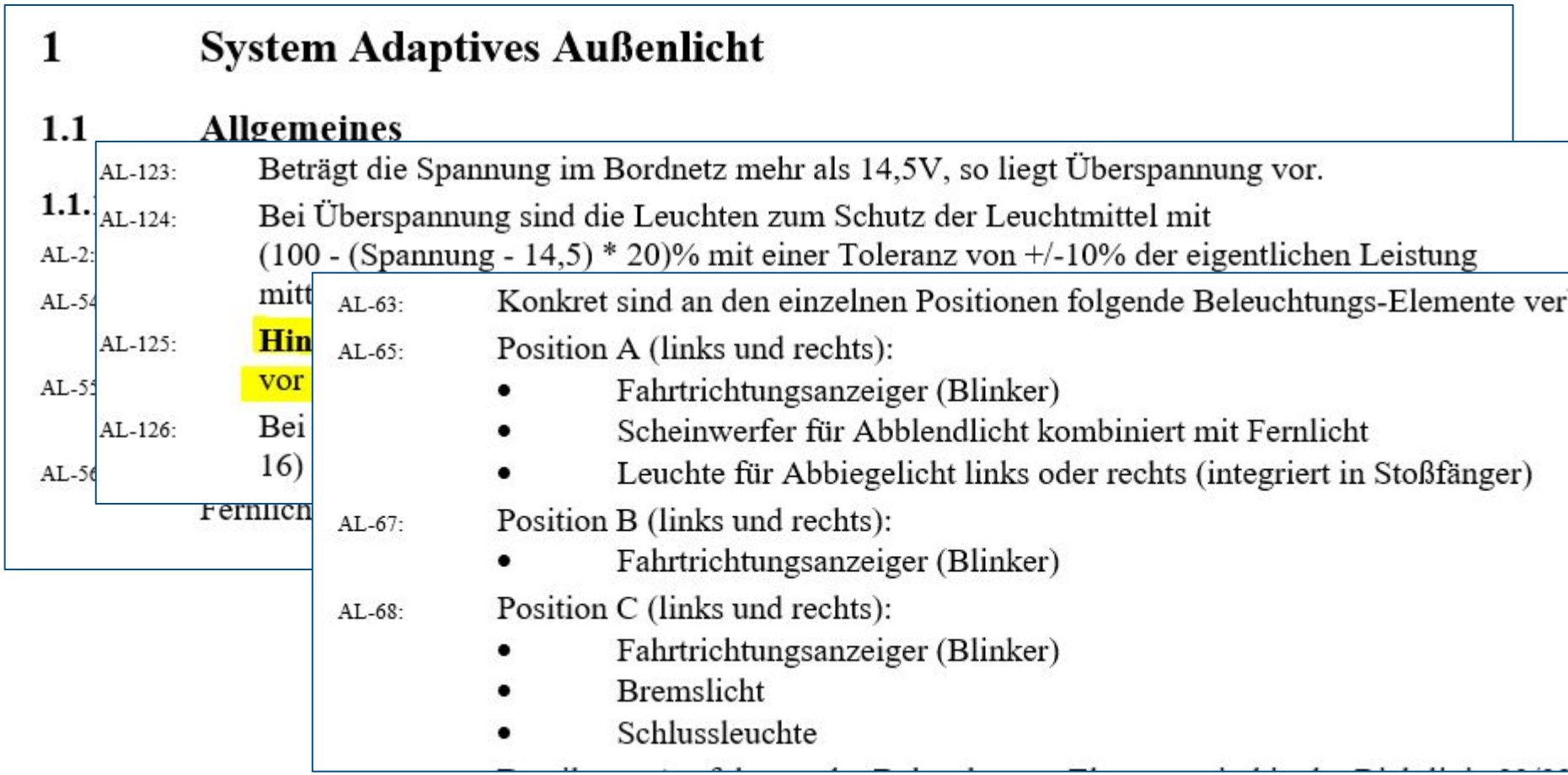
Automatische Qualitätssicherung von Anforderungen

```
Dies bedeutet, das E-Mails, die nicht im
Ordner ”gesendet” (oder irgendeinem
Unterordner) gespeichert wurden oder bei
denen der SPAM Filter anschlägt (Bayes),
in die entsprechende Ordnerstruktur im
privaten E-Mailverzeichnis verschoben
werden.
```

Frage: Kann man Lesbarkeit automatisch bewerten oder Verletzungen erkennen?

Betreuer: Maximilian Junker

Requirements Detection: Wo im Dokument befinden sich Anforderungen?



1 System Adaptives Außenlicht

1.1 Allgemeines

AL-123: Beträgt die Spannung im Bordnetz mehr als 14,5V, so liegt Überspannung vor.

AL-124: Bei Überspannung sind die Leuchten zum Schutz der Leuchtmittel mit (100 - (Spannung - 14,5) * 20)% mit einer Toleranz von +/-10% der eigentlichen Leistung

AL-63: Konkret sind an den einzelnen Positionen folgende Beleuchtungs-Elemente ver

AL-65: Position A (links und rechts):

- Fahrtrichtungsanzeiger (Blinker)
- Scheinwerfer für Abblendlicht kombiniert mit Fernlicht
- Leuchte für Abbiegelicht links oder rechts (integriert in Stoßfänger)

AL-67: Position B (links und rechts):

- Fahrtrichtungsanzeiger (Blinker)

AL-68: Position C (links und rechts):

- Fahrtrichtungsanzeiger (Blinker)
- Bremslicht
- Schlussleuchte

Themen (II)

Qualität von Code

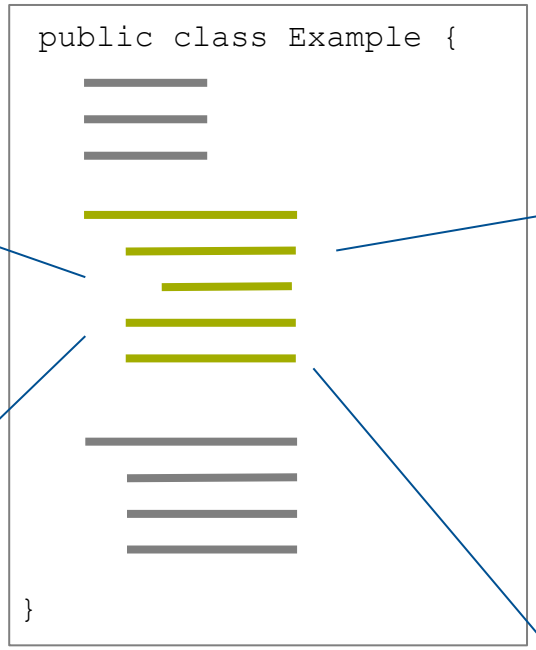
- **Qualitative Bewertung von Methoden:** Wie können wir die Qualität einer Methode bestimmen - und was ist überhaupt Qualität? (Markus)
- **Software Architektur:** Automatisch Entwurfsmuster erkennen (Markus)
- **Clone Detection:** Wo ist kopierter Quelltext im System? (Jakob)

Analyse von Java Methoden

*Wie messen und **interpretieren**?*

Strukturmetriken

- Nesting Depth
- Loop Count
- ?



Kommentare

- Dokumentation
- Commented-out code

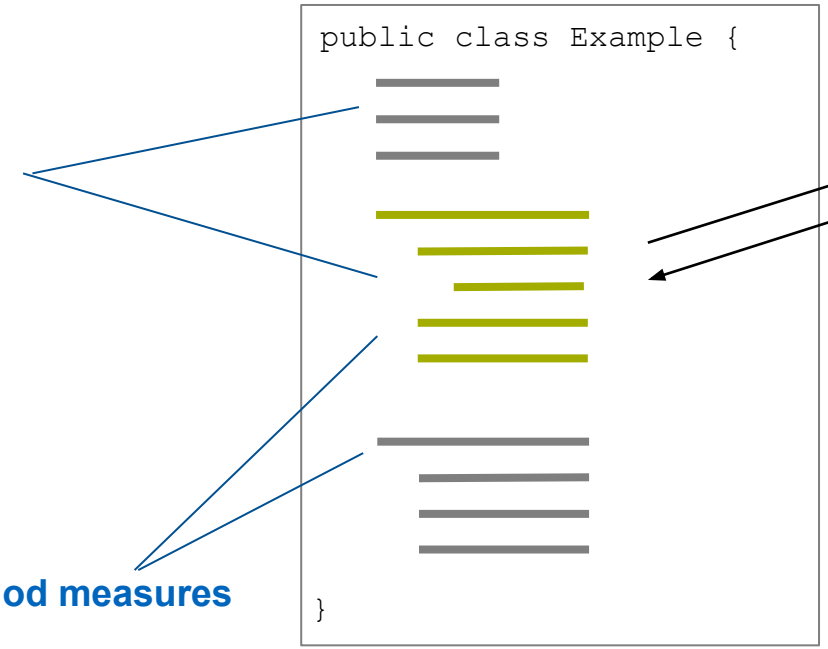
Aggregierte Metriken

- Smells
- ?

Kohäsion

Analyse von Java Methoden

Wie messen und ***interpretieren****?*



Class-method interactions

- ?

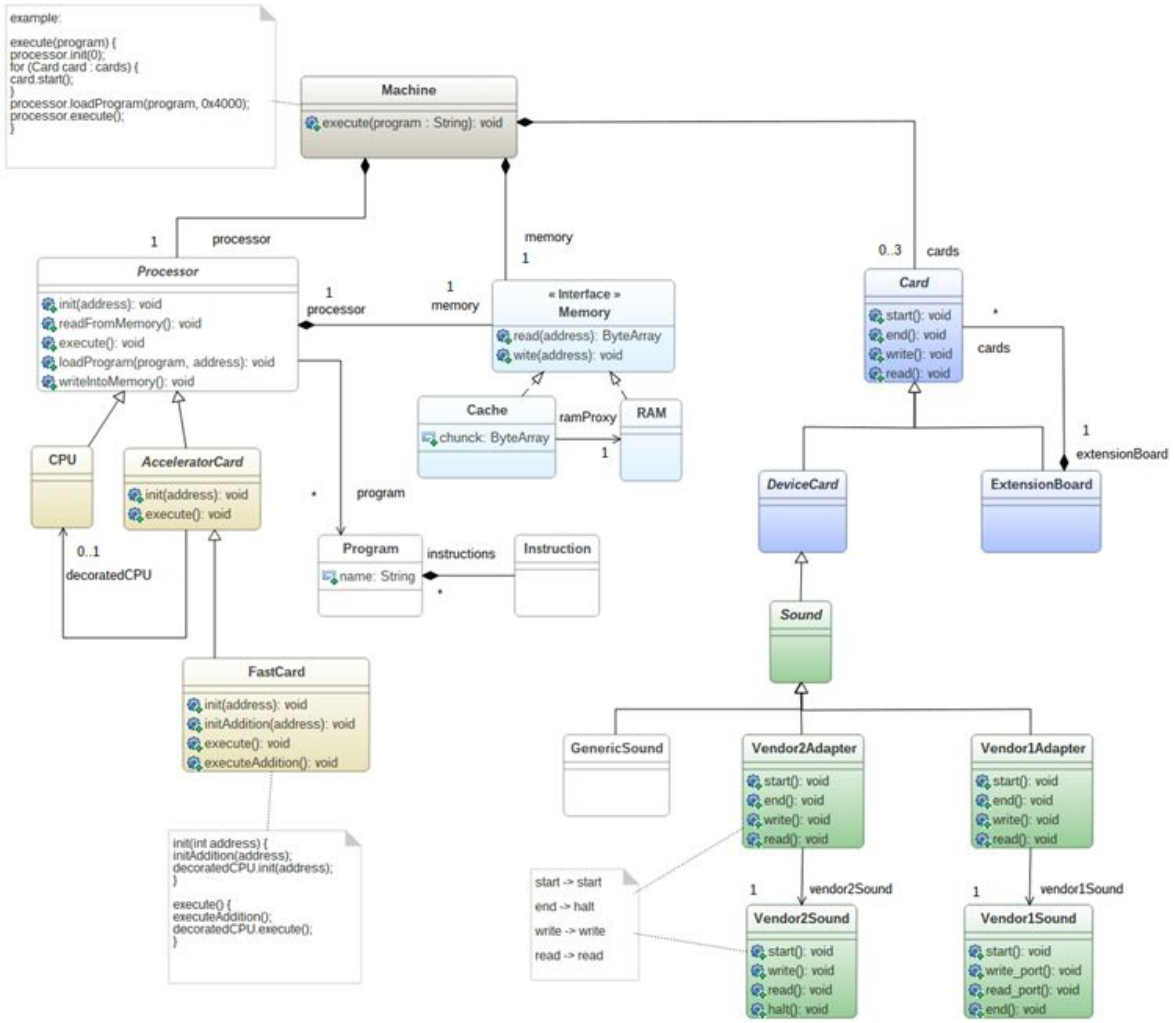
Inter-method measures

- Cloning
- ?

Inter-class relationships

- Cloning
- Coupling
- ?

Qualität einer SW-Architektur

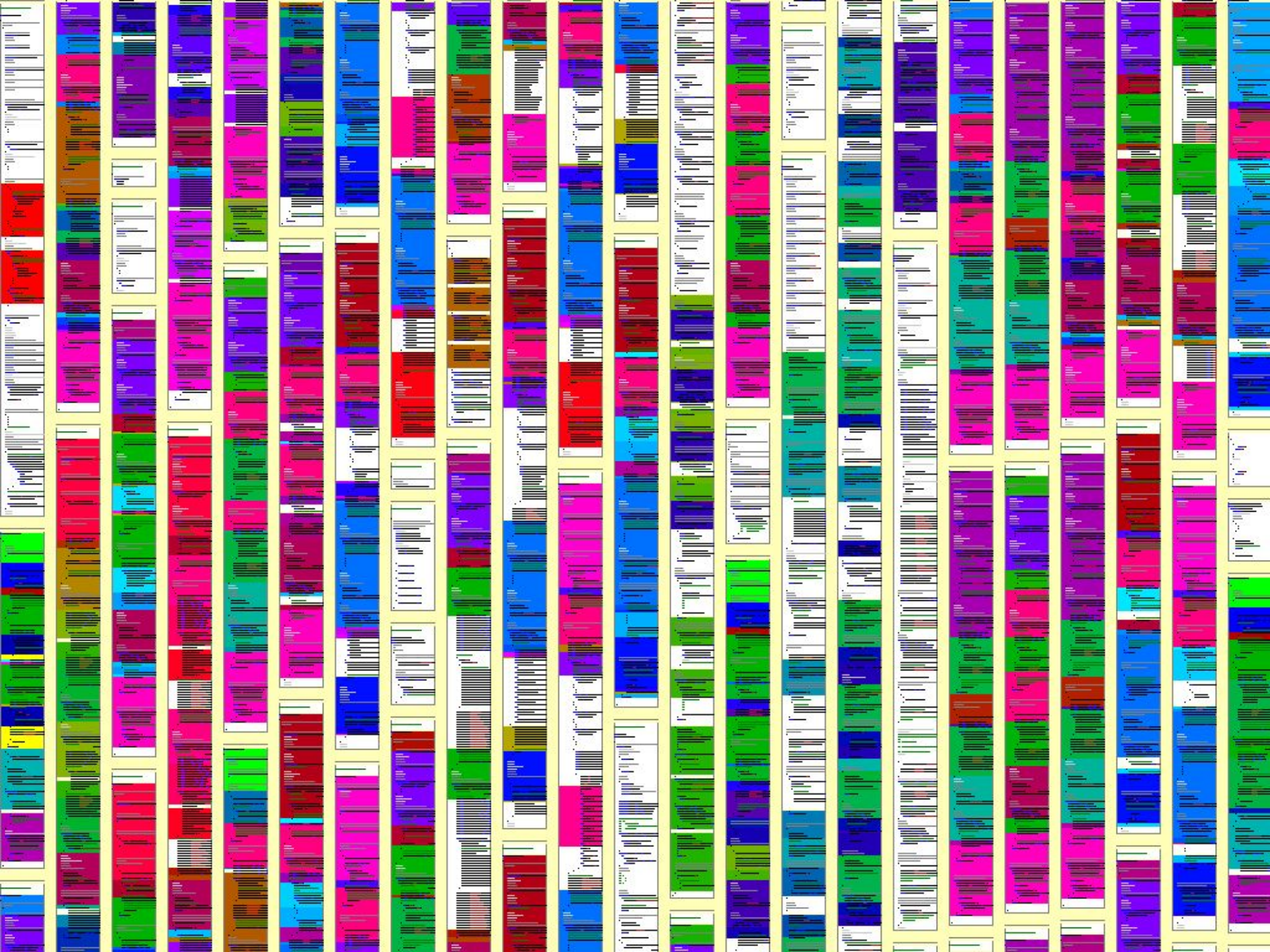


- Was macht eine gute Architektur aus?
- Welche Rollen spielen Pattern und Antipattern?
- Wie kann man automatisch Qualität bewerten?

```
// 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();
}
```

```
// 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();
}
```



Themen (III)

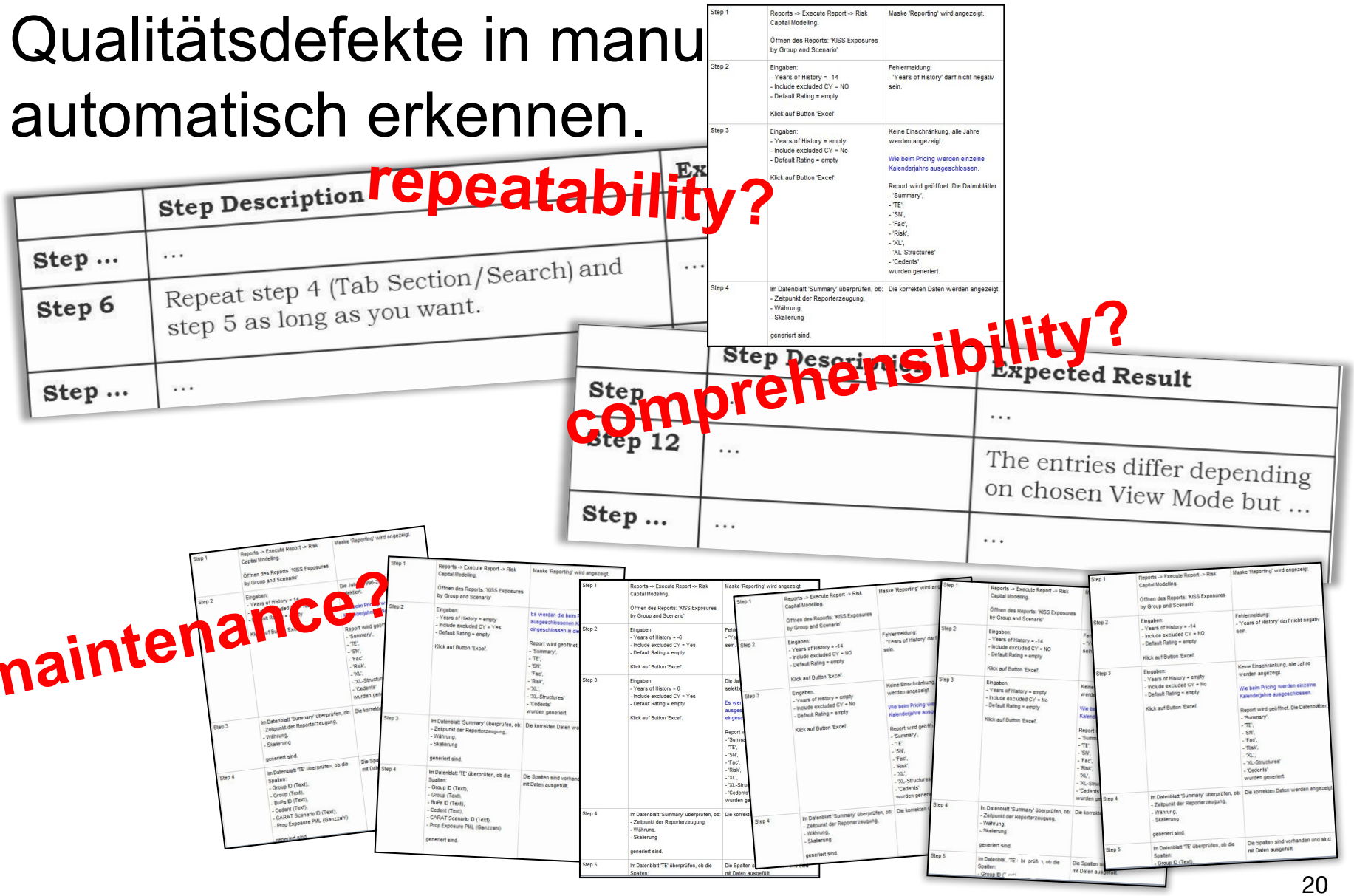
Thema Tests

- **Natural Language Test Smells**: Qualitätsdefekte in manuellen Tests automatisch erkennen.
  (Benedikt)

- **Warum zerbrechen meine automatisierten GUI-Tests?** Statische Analyse von automatisierten Tests.
  (Benedikt)

- **Was macht ein gutes Code-Review aus?** Untersuchung von quantitativen Metriken (Daniel Elsner)

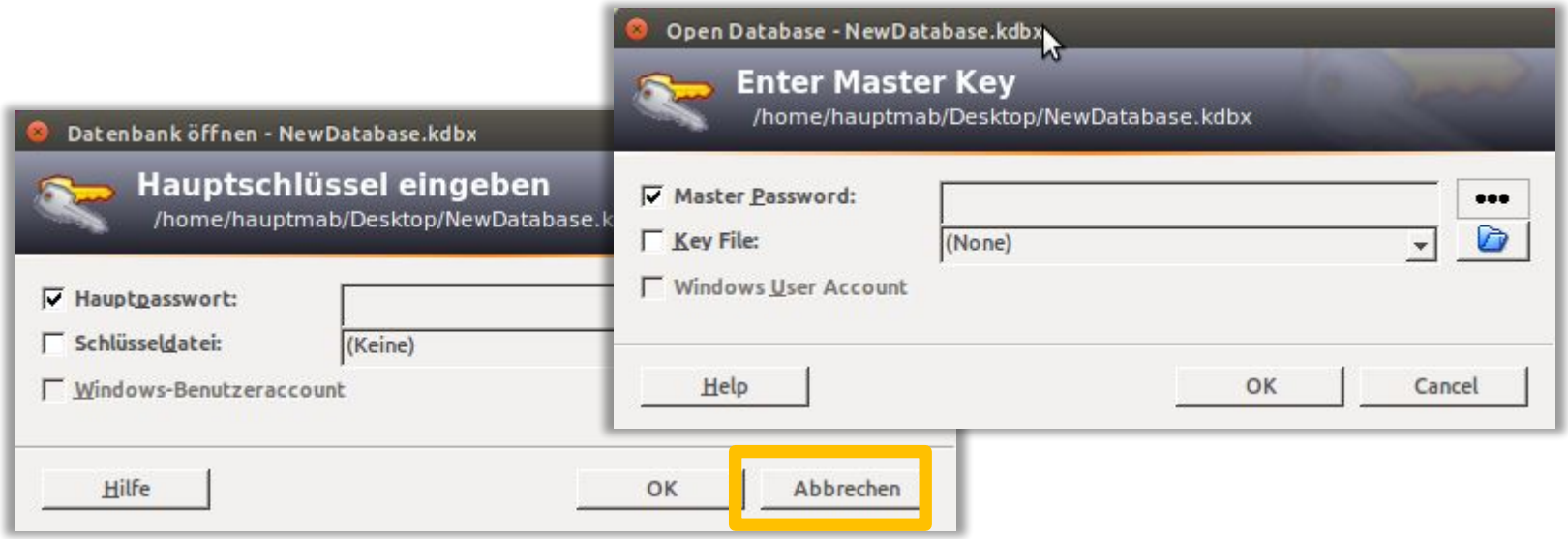
Natural Language Test Smells:
Qualitätsdefekte in manu
automatisch erkennen.



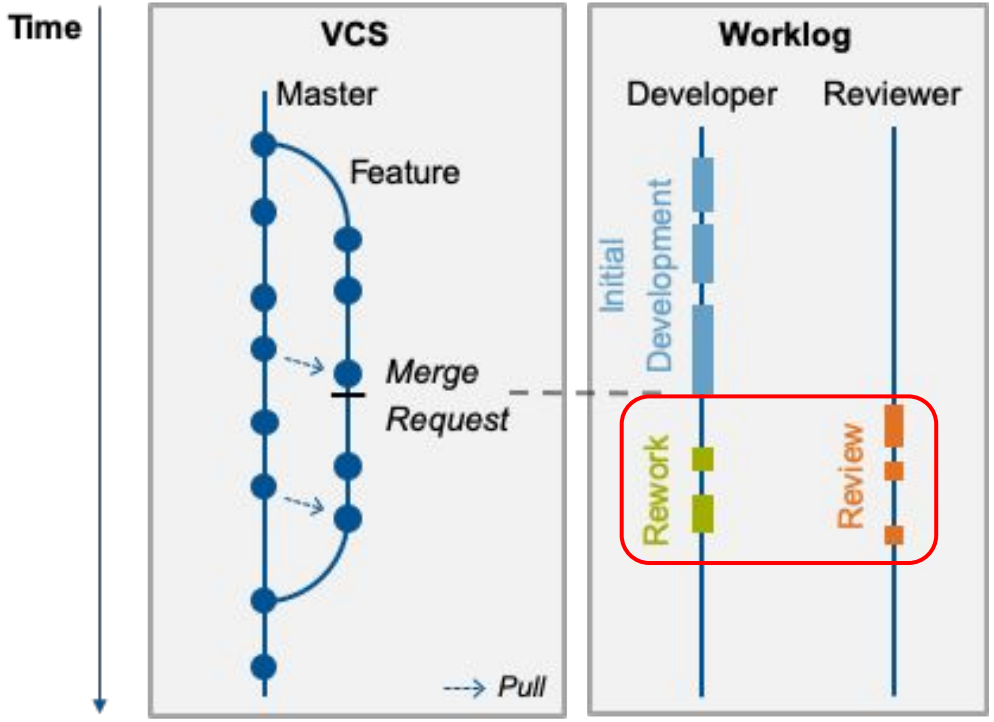
Warum zerbrechen meine automatisierten GUI-Tests?

Statische Analyse von automatisierten Tests.

Automated Test 01 – Login Dialog			
1.	Open application	C:\programs\keepass\2.0\keepass.exe	
2.	Validate	Windows is visible	“Open Database – New Datbase.kdbx”
3.	Enter value	“Master Password”	“123mysecret$”
4.	Click	Button	“Cancel”
	…	…	…



Was macht ein gutes Code-Review aus? Untersuchung von quantitativen Metriken



Open Source vs. Commercial?

SLOC/h?
#Comments?
#Rework-commits?
TotalDuration/SLOC?
Commit-Content?
...

Unerwartet lange Code-Reviews führen zu Verzögerungen im Entwicklungsprozess.
Was macht ein gutes Code-Review aus?

Themen (IV)

Qualität von Tests

- **Qualitätssicherung von Tests durch Mutation Testing**
  (Daniel Veihelmann)

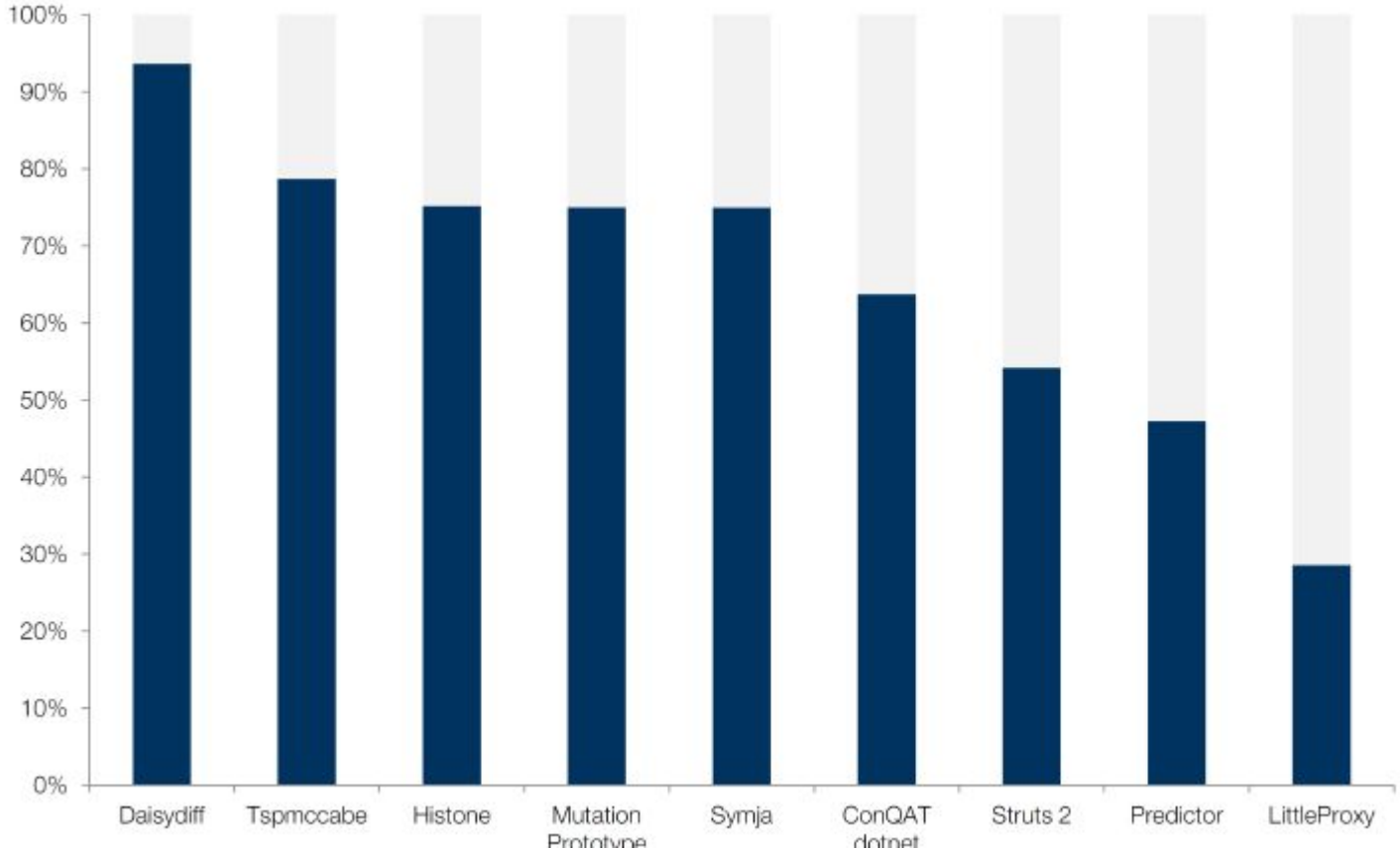
- **Test Gap Analyse:** Werden Systemänderungen auch wirklich von Tests erfasst?
  (Daniel Veihelmann)

- **Test Impact Analyse** (Jakob)

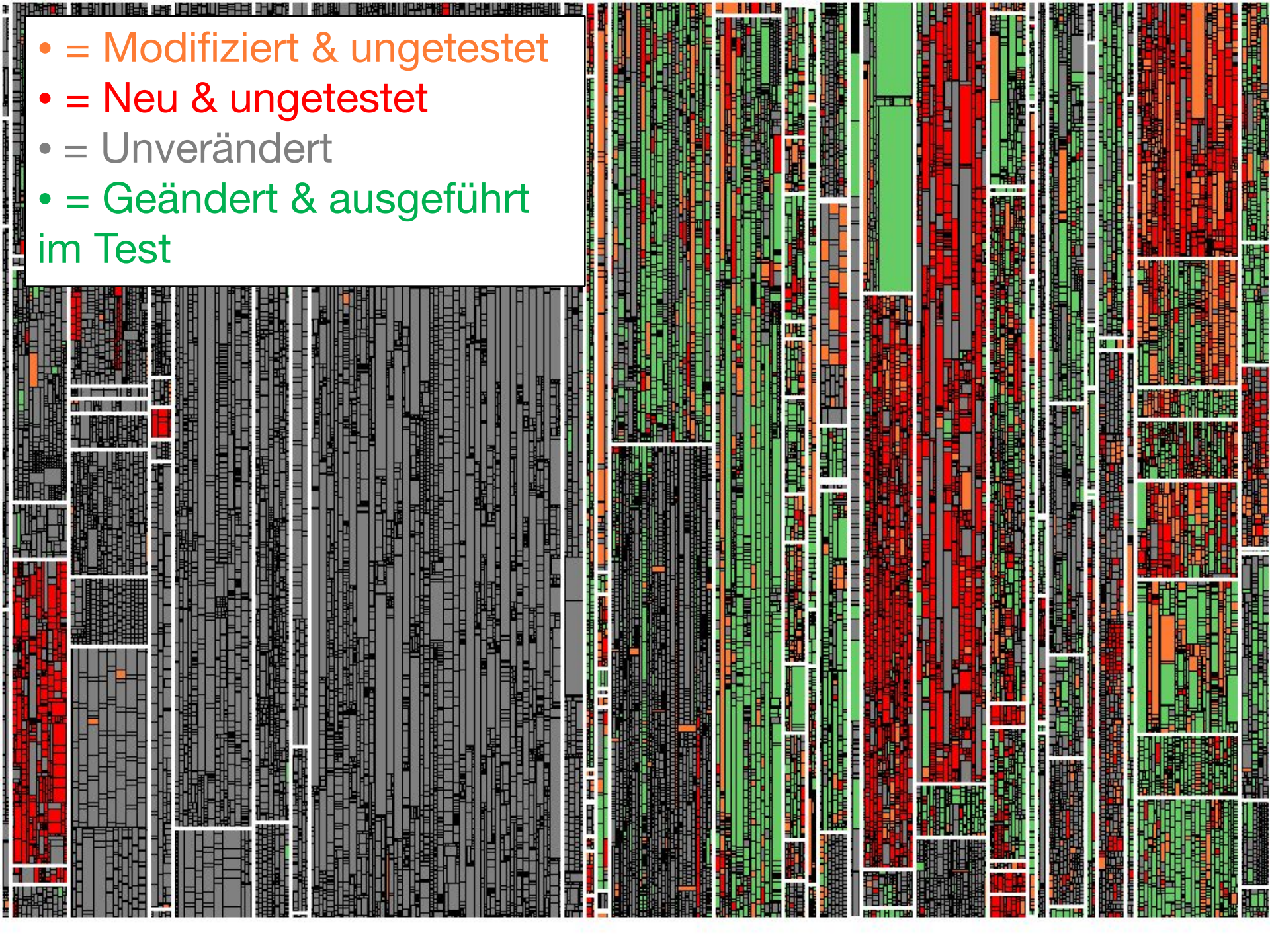
- **Smarter Testing:** Auswahl der relevanten Testfälle durch statische Analyse (Daniel Elsner)

Mutation Testing

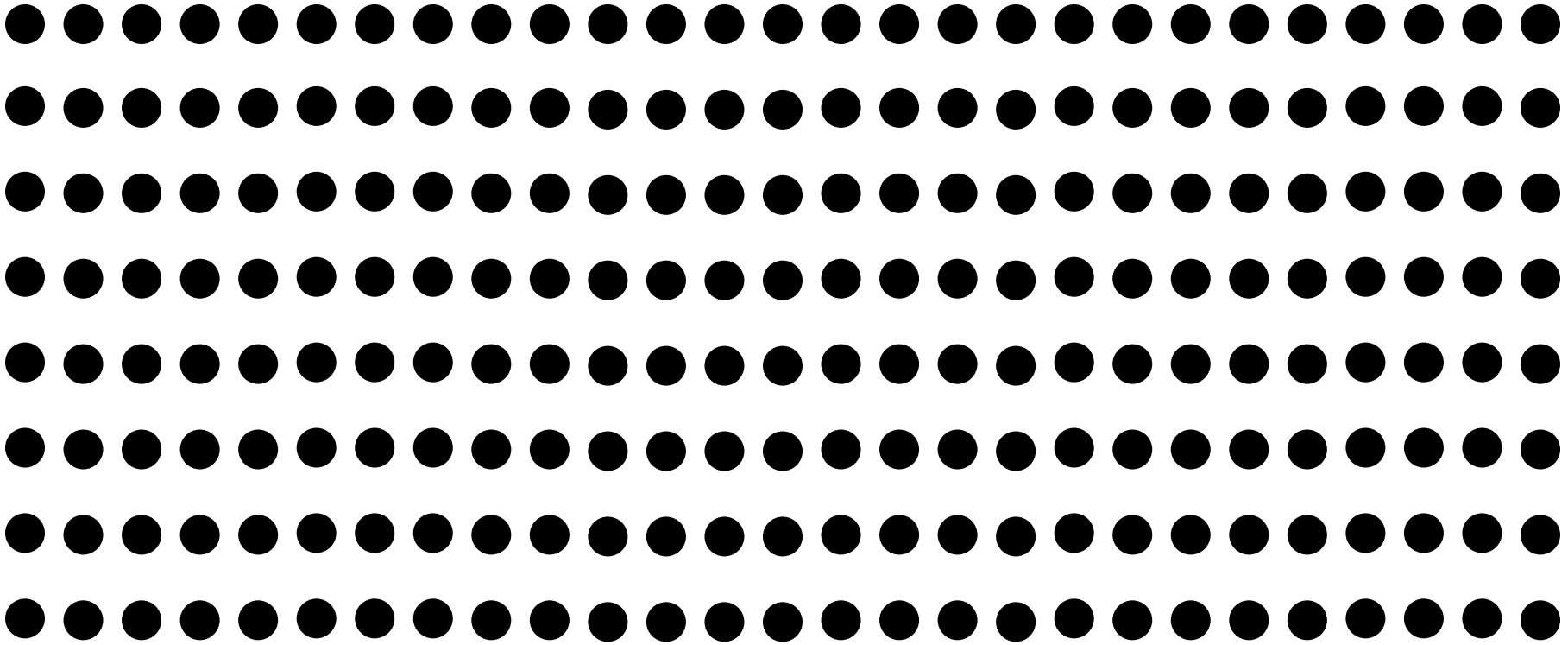
Finden unsere Tests eigentlich Fehler?

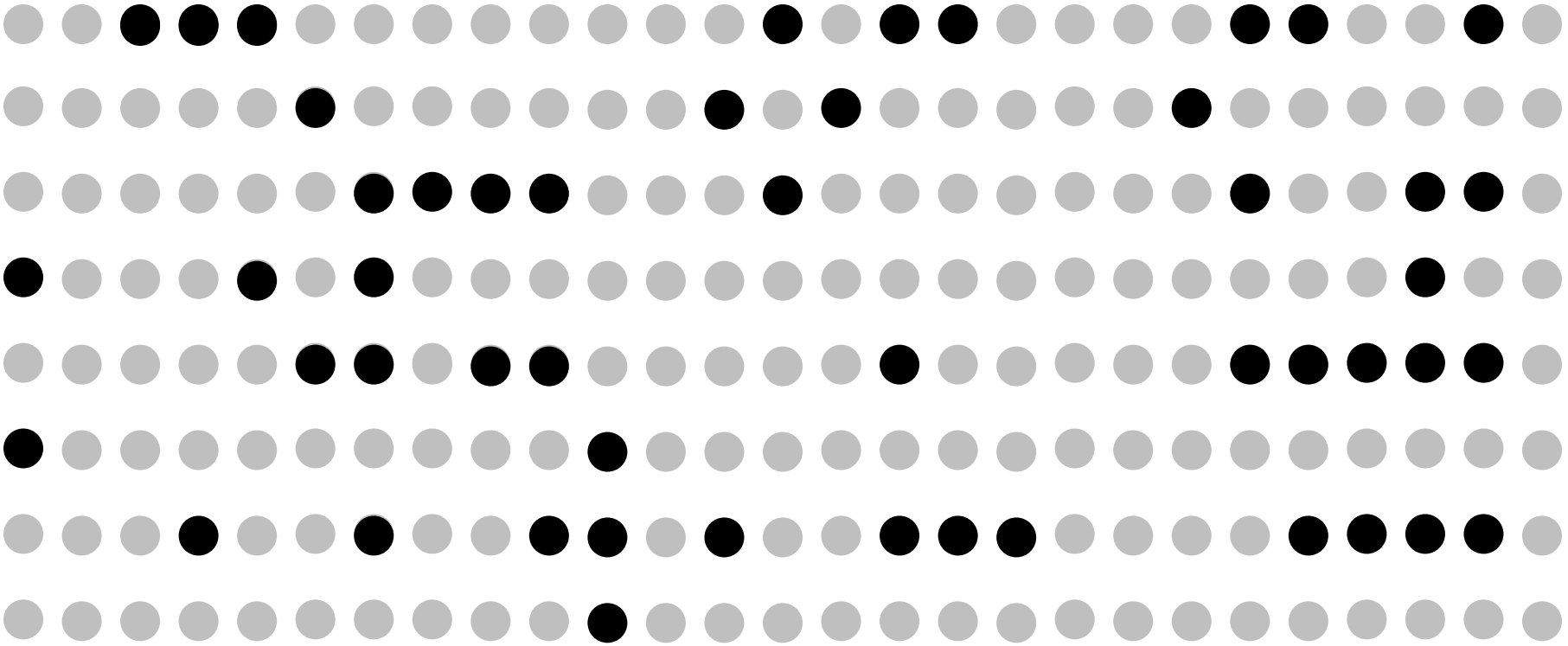


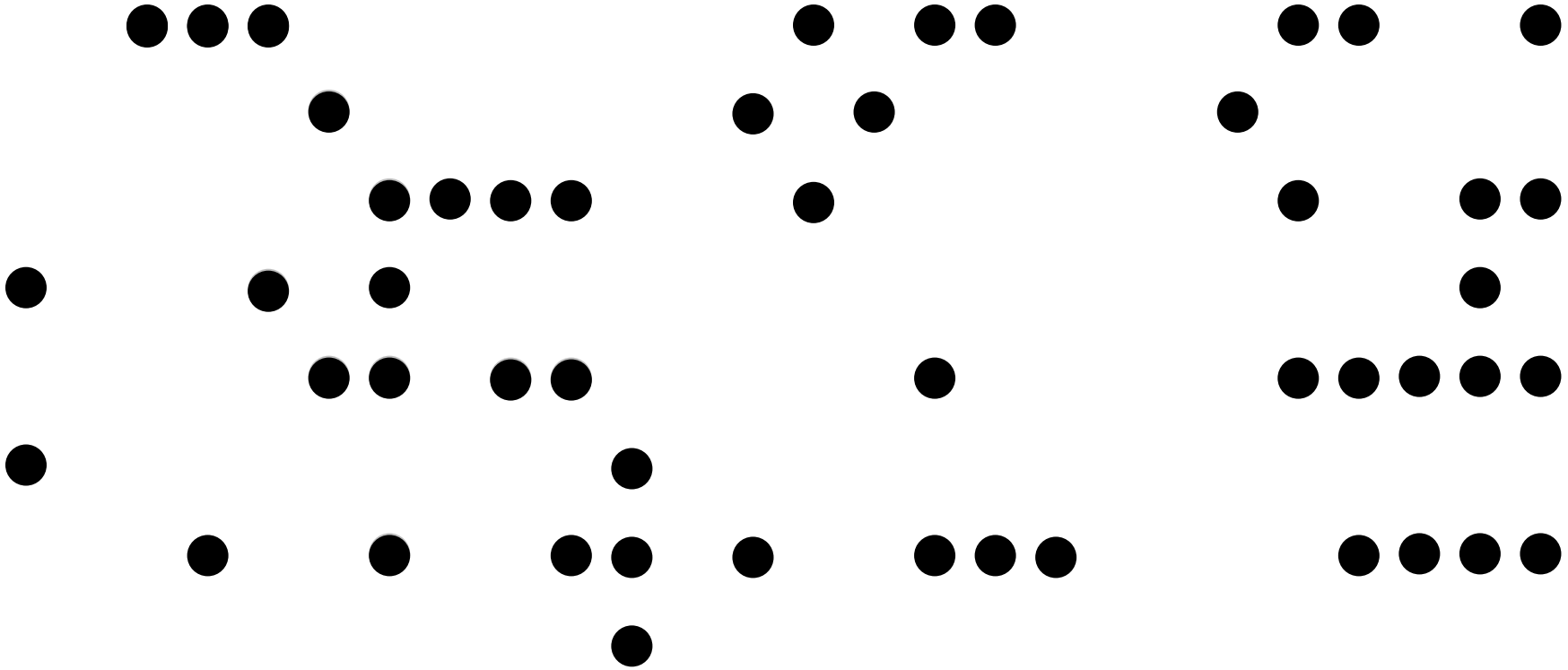
Test Gap Analyse: Werden Systemänderungen auch wirklich von Tests erfasst?

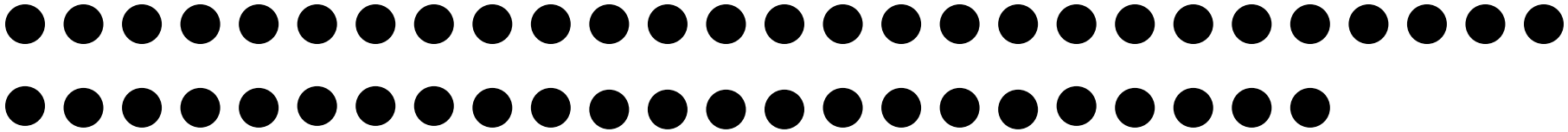

• = Modifiziert & ungetestet
• = Neu & ungetestet
• = Unverändert
• = Geändert & ausgeführt
im Test

Test Impact Analyse: “Muss ich wirklich wieder *alle* Tests ausführen?”

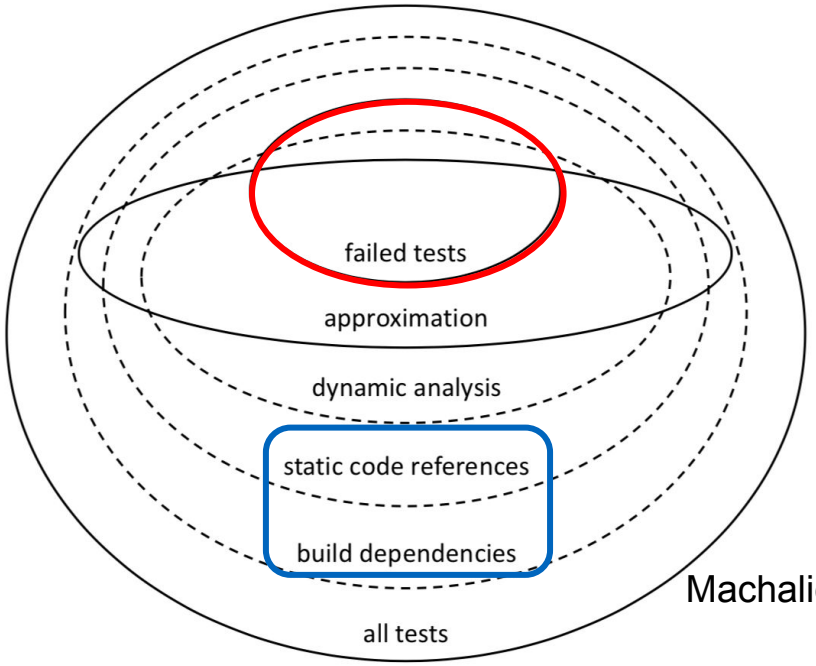








Smarter Testing: Auswahl der relevanten Testfälle durch statische Analyse



Machalica et al. 2019

Statische Analyse der Code-Abhängigkeiten für
änderungsbasierte Testfall-Selektion

Fragen? Anmerkungen?