
sidion
SIDION
Zuhören. Analysieren. Beraten.

MUTATION TESTING IN THEORIE UND PRAXIS

Alexander John-Anacker | JFS 2022

sidion

ERKENNUNGSDIENSTLICHE HINWEISE

Wer bin ich
Alexander John-Anacker
Senior Software Entwickler
sidion GmbH

Aktuelle Schwerpunkte
Fullstack Java
Cloud-native Microservices
DevSecOps
CI/CD
Testing

Kontakt
alexander.john-anacker@sidion.de
XING/LinkedIn

sidion

AGENDA

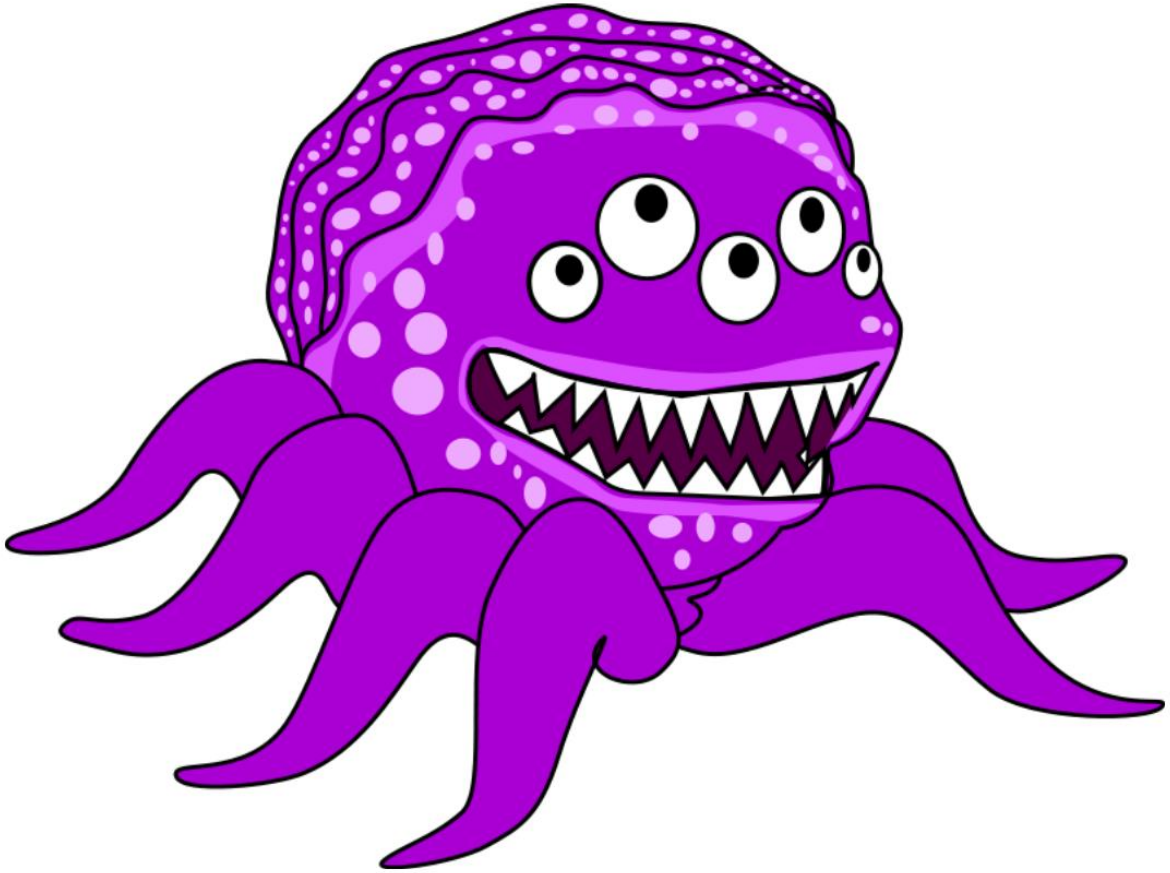
- Einführung
- Theorie
- Praxis
- Fazit

sidion

EINFÜHRUNG

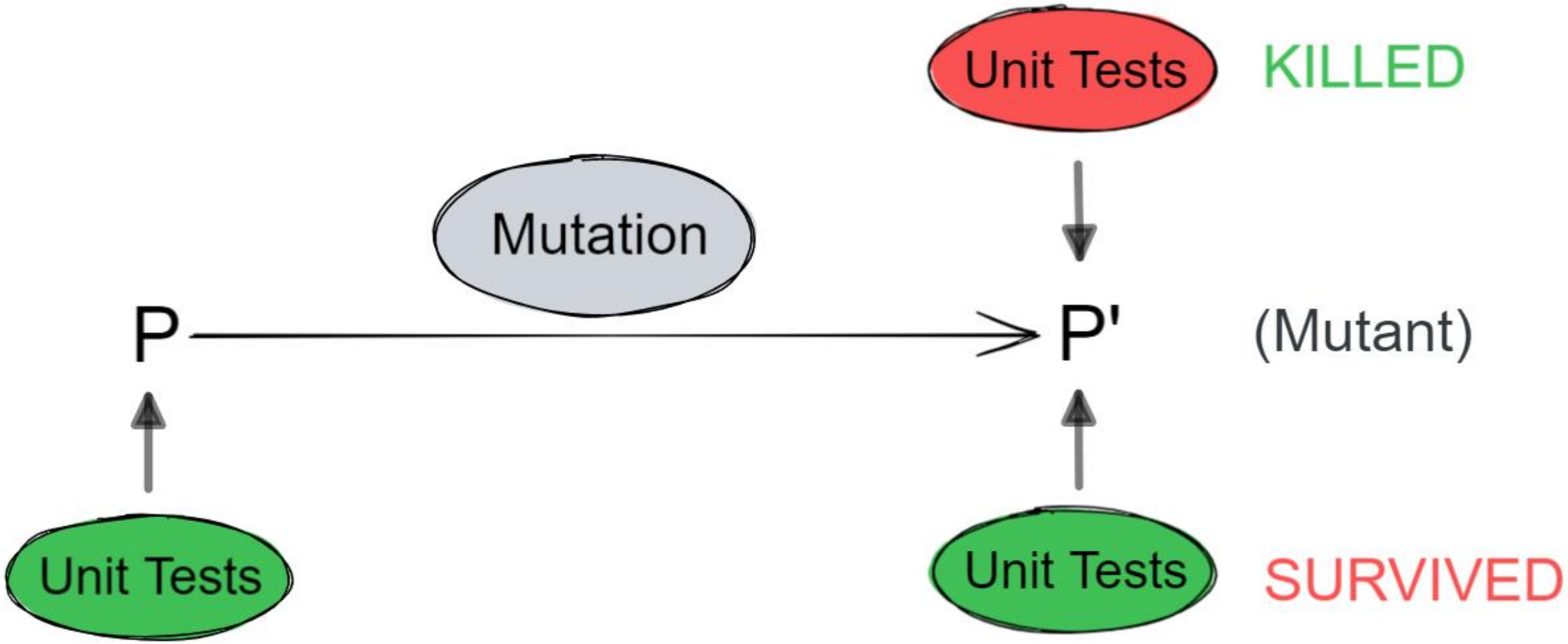
sidion

MUTANTEN



sidion

PRINZIP HINTER DER VORGEHENSWEISE



sidion

MÖGLICHE MUTATIONEN (AUSWAHL)

- **Operatoren**

  Operator durch anderen ersetzen:
  - \+ durch –
  - > durch >= oder durch <
  - ++i durch --i

  Operator durch Ergebnis ersetzen
  - *a == b* ersetzen durch *true*

- **Variable**

  Verändern durch
  - Inversion
  - Zuweisung durch default Wert ersetzen

sidion

MÖGLICHE MUTATIONEN (AUSWAHL)

- **Methoden**

  return value ersetzen durch:
  - Konstante
  - Inverses Element
  - Null Element oder Leeres Element

  Aufruf weiterer Methoden
  - Void Methoden Aufruf entfernen
  - Nicht Void Methoden Rückgabe ersetzen durch z.B. festen Wert

- **Konstruktoren**

  Aufruf *new Object()* durch *null* ersetzen

PITEST

- Mutation Test Tool for Java
- Open Source
- Wird aktiv weiterentwickelt
- Hochoptimiert

https://Pitest.org

sidion

MUTATION ANALYSE

SUPER !!!

DAS IST

BESTIMMT

BRANNTNEU

GESCHICHTE

- **1971** Richard Lipton schlägt initiales Konzept in Seminararbeit vor
- **1978** DeMillo et al.: wegweisendes paper publiziert
- **1980** PIMS: erstes Mutation Testing Tool für FORTRAN
- **1987** Mothra: Standard Tool in Forschung und Universitäten
- **2007** Certitude: erster industrieller Einsatz beim Chip Design
- **2011** Pitest

TOOLING (AUSWAHL)

Java	**Pitest**	https://pitest.org/
JavaScript	**StrykerJS**	stryker-mutator.io/docs/stryker-js/introduction
C#	**Stryker.NET**	stryker-mutator.io/docs/stryker-net/introduction
Scala	**Stryker4s**	stryker-mutator.io/docs/stryker4s/getting-started
Python	**CosmicRay**	sixty-north/cosmic-ray
Ruby	**Mutant**	https://github.com/mbj/mutant

sidion

THEORIE

sidion

PREISFRAGE

Wie **relevant** sind die **künstlichen bugs**

der Mutationsoperatoren **als Ersatz für reale bugs?**

sidion

RELEVANZ DER KÜNSTLICHEN DEFECTS

73% of real faults are coupled to the mutants generated by commonly used mutation operators

Quelle: Are Mutants a Valid Substitute for Real Faults in Software Testing?
René Just et al.
https://people.cs.umass.edu/~rjust/publ/mutants_real_faults_fse_2014.pdf

sidion

RELEVANZ DER KÜNSTLICHEN DEFECTS

faults produced by typical mutation operators

are not representative of real faults

Quelle: Mutations: How close are they to real faults?
Rahul Gopinath et al.
https://agroce.github.io/issre14.pdf

sidion

RELEVANZ DER KÜNSTLICHEN DEFECTS

Im Mittel haben wir wohl eine

schwache bis **mittlere** Kopplung

… ohne den **menschlichen** Faktor!

sidion

MUTATION TESTING AT GOOGLE

- Eigener Ansatz / proprietäre Engine
- Anwendbar auf große Codebases
- Verwendung von nur 5 Mutations Operatoren
- Maximal ein Mutant pro Code-Zeile
- Filterung von Mutanten, die als unproduktiv angesehen werden
- Ausschließlich für Code Reviews

Quelle: State of Mutation Testing at Google
Goran Petrovic et al.
https://storage.googleapis.com/pub-tools-public-publication-data/pdf/abdfdbac520614dc87bc0b3fdf10bdc2869fce41.pdf

sidion

MUTATION TESTING AT GOOGLE

Developers exposed to mutants **write more, effective tests** in response to them.

In **70% of cases, a bug is coupled with a mutant** that, had it been reported during code review, could have prevented the introduction of that bug

Quelle: Does mutation testing improve testing practices?
Goran Petrovic et al.
https://homes.cs.washington.edu/~rjust/publ/mutation_testing_practices_icse_2021.pdf

sidion

PRAXIS

sidion

HERAUSFORDERUNGEN

- Äquivalente Mutanten
- Performance/Laufzeit
- Einbettung in DEV Workflow

ÄQUIVALENTE MUTANTEN

- Semantisch identisch bez. Mutations-Operation
- Manchmal schwierig zu erkennen
- Manchmal können Sie nicht erledigt werden

ÄQUIVALENTE MUTANTEN

Beispiel Logging

```
log.info("{}% of the tasks are complete.", 100 * complete / base);
```

Lösung

- Lösung: in Pitest können Aufrufe exkludiert werden mit ‚avoidCallsTo‘

ÄQUIVALENTE MUTANTEN

Beispiel Performance Optimierung

```
Object getImportantObject(int id) {
    Object iAmImportant = null; // originally: iAmImportant = lookupInCache(id);
    if(iAmImportant != null) {
        return iAmImportant;
    }
    return getObjectUsingVeryExpensiveCalculation();
}
```

- Entweder ignorieren
- Oder verify verwenden

ÄQUIVALENTE MUTANTEN

Beispiel redundanter/toter code

```
int someFancyLogic(int id) {
    int result = id / 42; // originally id * 42
    return id * 42;
}
```

ÄQUIVALENTE MUTANTEN

Lösung redundanter/toter code

```
int someFancyLogic(int id) {
    return id * 42;
}
```

- Produktivcode immer mit ins Kalkül ziehen

sidion

ÄQUIVALENTE MUTANTEN

- Man hört oft: sie sind größtes Problem
- Warum? Wenn nichts hilft: einfach ignorieren
- Pitest: DEFAULT Mutator-Set versucht sie zu vermeiden

PERFORMANCE/LAUFZEIT

Wie lange dauert Mutation Testing für eine kleine/mittelgroße Library?

Beispiel: apache/commons-collections

Lines of code:	~13500
Anzahl Tests:	~22300

https://github.com/apache/commons-collections

PERFORMANCE/LAUFZEIT

Wie lange dauert Mutation Testing für eine kleine/mittelgroße Library?

Kompilierzeit:	4 Sekunden
Zeit für Ausführung der Unittests:	34 Sekunden
Anzahl Mutanten:	8700 (DEFAULT Mutator Set)
	91 Stunden / 3 ¾Tage
Mit PITEST dauert es nur noch	**2 Minuten 45 Sekunden**

sidion

UNTER DER MOTORHAUBE VON PITEST

- Parallele Ausführung
- Bytecode Modifikation
- Nur Tests ausführen, die den Mutanten erreichen können
- Tests in aufsteigender Reihenfolge ihrer Laufzeit ausführen
- Abbruch, wenn der erste Test fehlschlägt
- Inkrementelle Analyse

EINBETTUNG IN DEV WORKFLOW

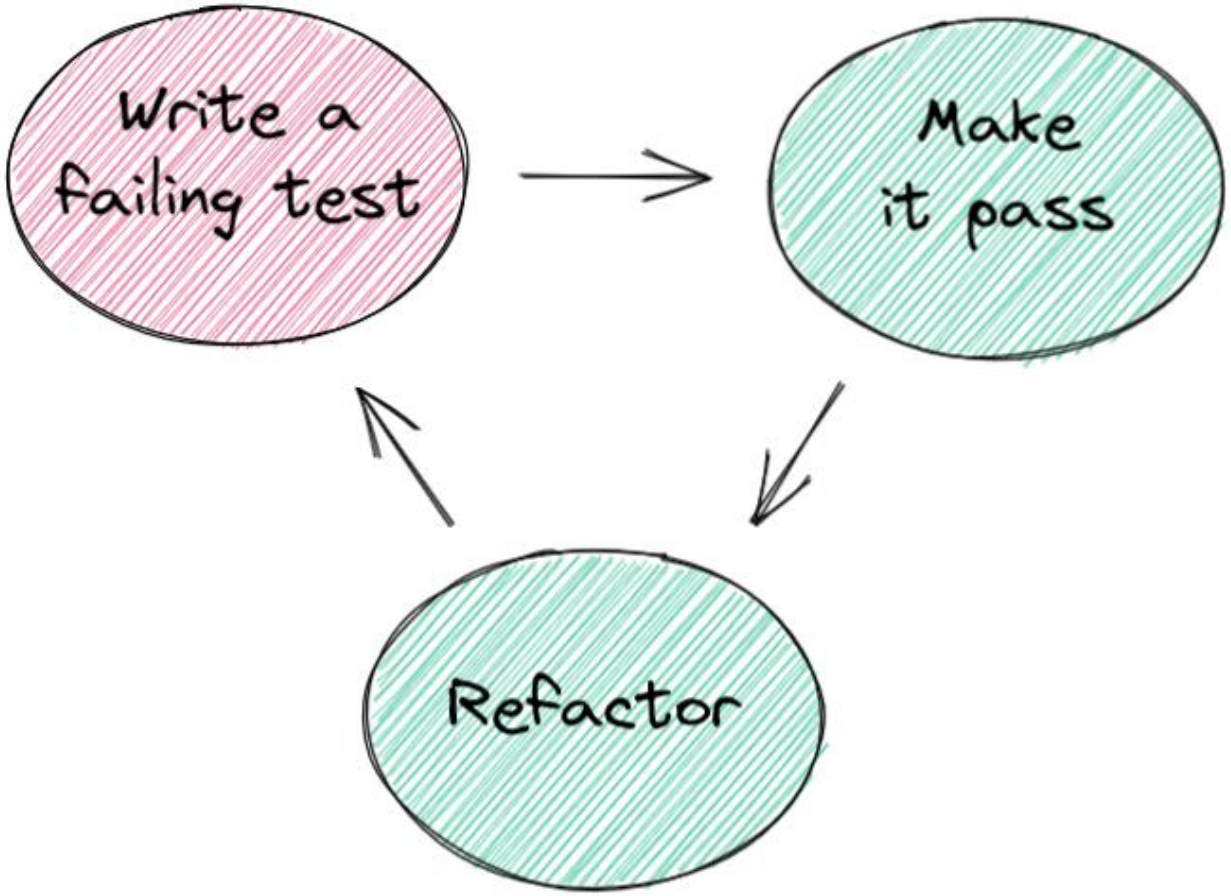
- Bottom up – auf der Entwickler Maschine
- Top-down – auf der CI-Platform
- Beides ?

sidion

BOTTOM-UP (ANFÄNGER)

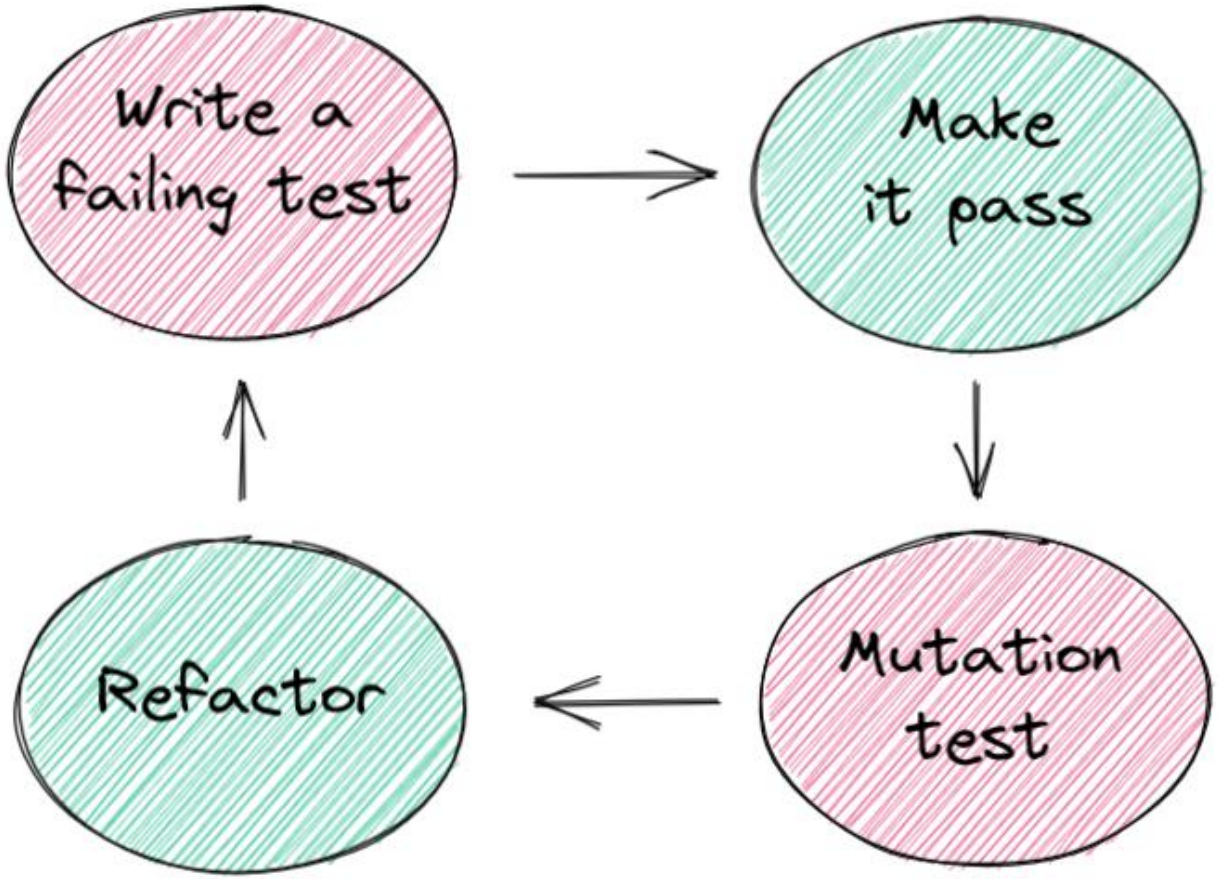
- Einmalige Ausführung für Assessments
- Bughunting Sessions
- Quality Sprints

TDD



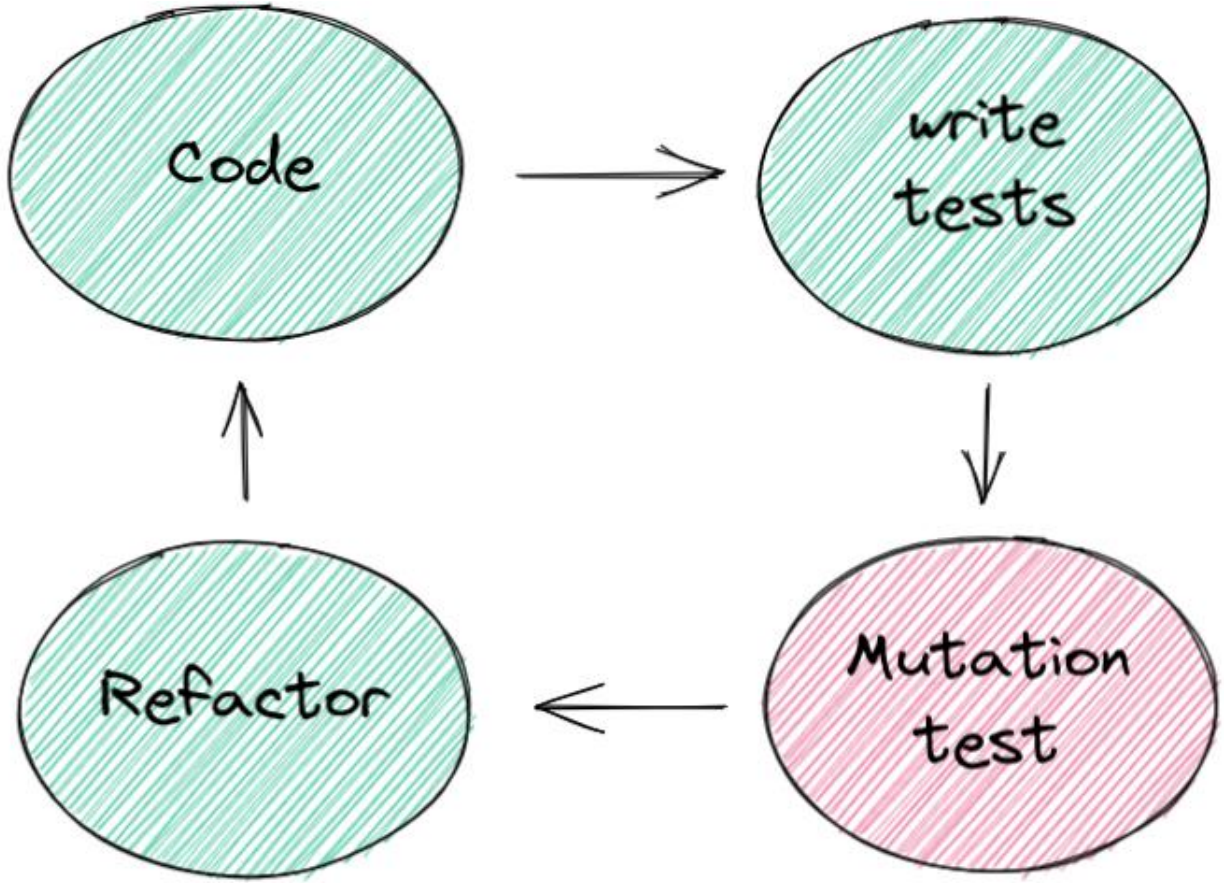
sidion

TDD RELOADED



sidion

… ODER WIE ES WOHL IN DER PRAXIS LÄUFT



sidion

BOTTOM-UP (FORTGESCHRITTEN)

- Sehr selektive Mutationstests regelmäßig
- Ultra schnell – unabhängig von Projektgröße
- Fast Feedback Loop
- Training um bessere Tests zu schreiben

TOP-DOWN

- Inkrementelle Analyse sehr sinnvoll
- Sichtbarkeit gewährleisten: z.B. Sonarqube
- Integration in Code Review Prozess
- Wissenstransfer im Team

SONARQUBE PLUGIN

- Surviving Mutants als Bugs oder Code Smells
- Ggf. in weiteren Anwendungen nutzbar
- Ab Sonarqube 9.x benotigt man Version >= 1.6

https://www.devcon5.ch/products_and_services/products/sonarqube-mutation-plugin/
https://github.com/devcon5io/mutation-analysis-plugin/releases/

sidion

ODER DOCH BEIDES ?

- Bottom-up anfangen
- Strategie *im Team* überlegen
- Ggf. Top-Down im Anschluss

FAZIT

sidion

MUTATION ANALYSE ...

... FINDET

- Unvollständige Tests
- Ineffektive Tests
- Toten Code
- Redundanten Code
- Bugs

... SCHÜTZT

- Härtet Testsuite
- Gibt Vertrauen für CICD
- Shift Left
- Vermeidet Regressionen
- Mut zur Refaktorisierung

VORSICHT IST GEBOTEN …

… BEI

- Initiales Setup
- Mutation Score
- Implementierung der Tests
- Integrationstests
- Akzeptanz

BITTE BEACHTEN:

- Weniger ist mehr
- Keine falschen Ziele setzen
- Nicht zu ‚whiteboxige' Tests
- Nur für Junit sinnvoll
- Nicht entmutigen lassen

sidion

DO‘S AND DON‘T DO‘S

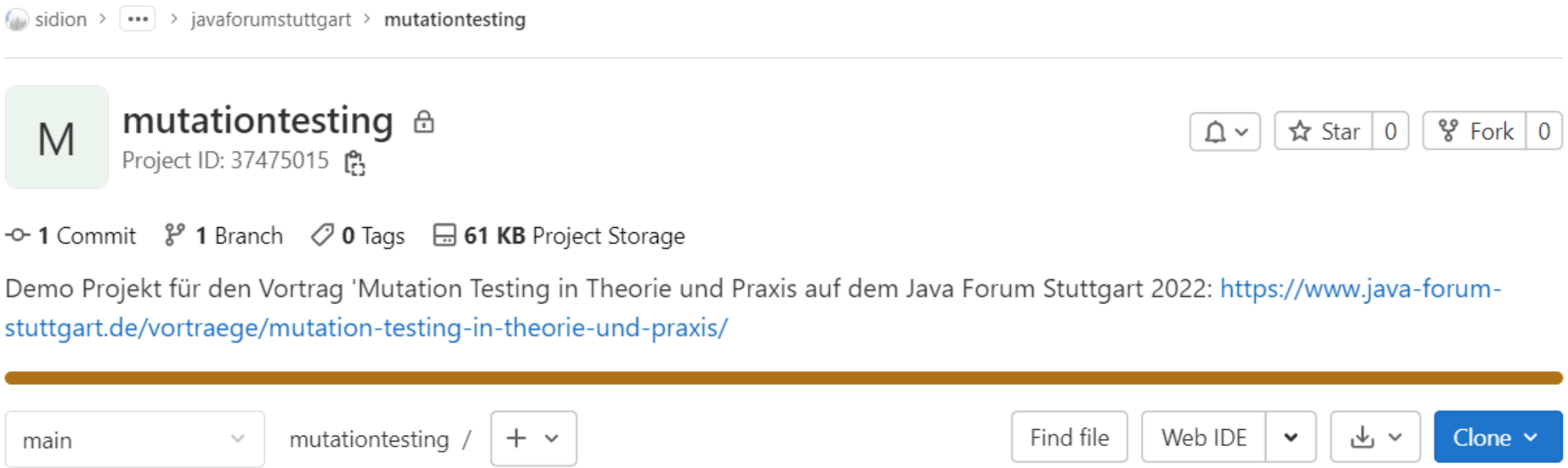
Do it!

Don‘t do it not!

sidion

DEMO CODE

https://gitlab.com/sidion/vortraege/2022/javaforumstuttgart/mutationtesting



sidion

NOCH FRAGEN ?

... oder an unserem Stand
in der Ausstellung

sidion

sidion

sidion

VIELEN DANK