ABB Technikerschule

Technik. Informatik. Wirtschaft. Management →

Dominik Meyer
Software Engineering

JUnit

Ziel

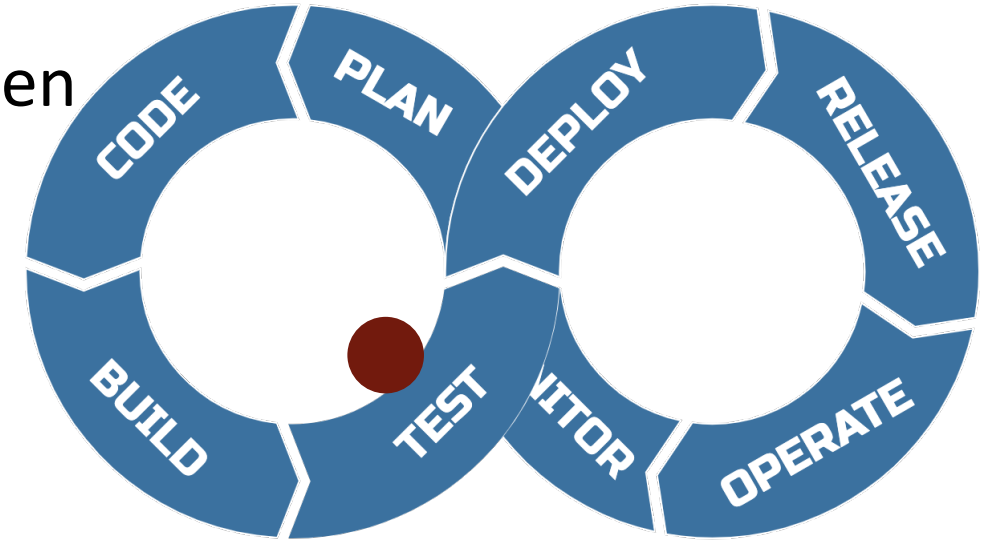
Nach der Lektion können die Studierenden Test Tools beschreiben und haben einen Unit Test für die Starter App geschrieben.

JUnit

– Bei jedem PR laufen Tests, aber welche?
– Lektion 16 GitHub Actions schlagen fehl, schlecht, warum?
– Verschiedene «Architektur» Level

– Verschiedene «Flughöhen» aufteilen und separat testen
– Auf dem niedrigsten Level starten

– Unit Tests ersetzen laufendes Testen mit Klicken durch repetitive Tests
– Fertig mit Klicken und Starten zum Testen



Agenda

– Ziel
– Unit Tests
– 1 Test schreiben
– Zielkontrolle

Test L16 schlägt fehl

– https://github.com/nds-swe/jinder/pull/2/checks

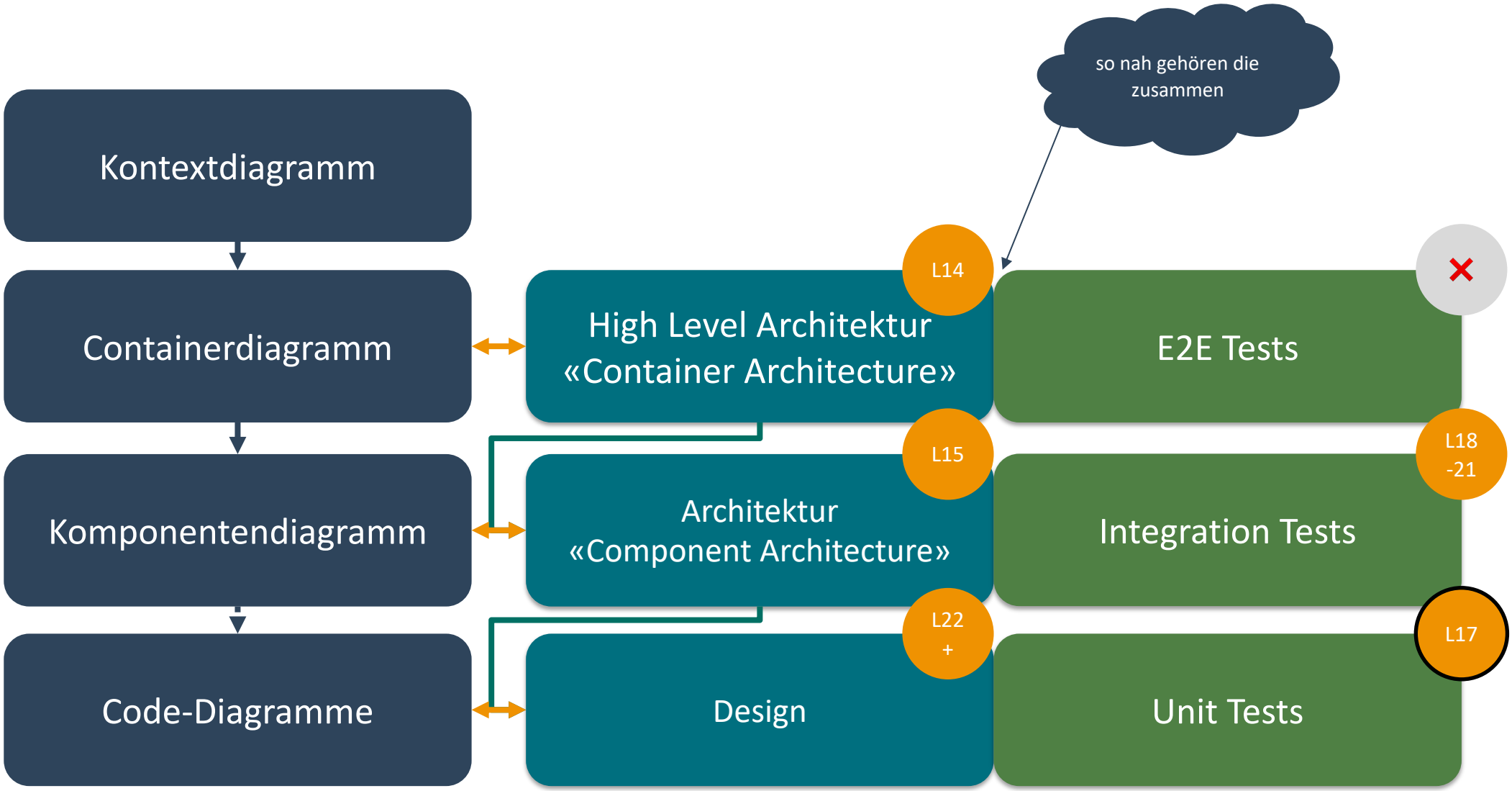
```
39  MatcherTest > If no skills match, the result must be empty FAILED
40      org.opentest4j.AssertionFailedError at MatcherTest.java:26
41  4 actionable tasks: 4 executed
42
43  3 tests completed, 1 failed
44
45  FAILURE: Build failed with an exception.
46
47  * What went wrong:
48  Execution failed for task ':test'.
49  > There were failing tests. See the report at: file:///home/runner/work/jinder/jinder/build/reports/tests/test/index.html
50
51  * Try:
52  Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.
53
54  * Get more help at https://help.gradle.org
55
56  BUILD FAILED in 28s
57  Error: Gradle process exited with status 1
```

Test L16 schlägt fehl

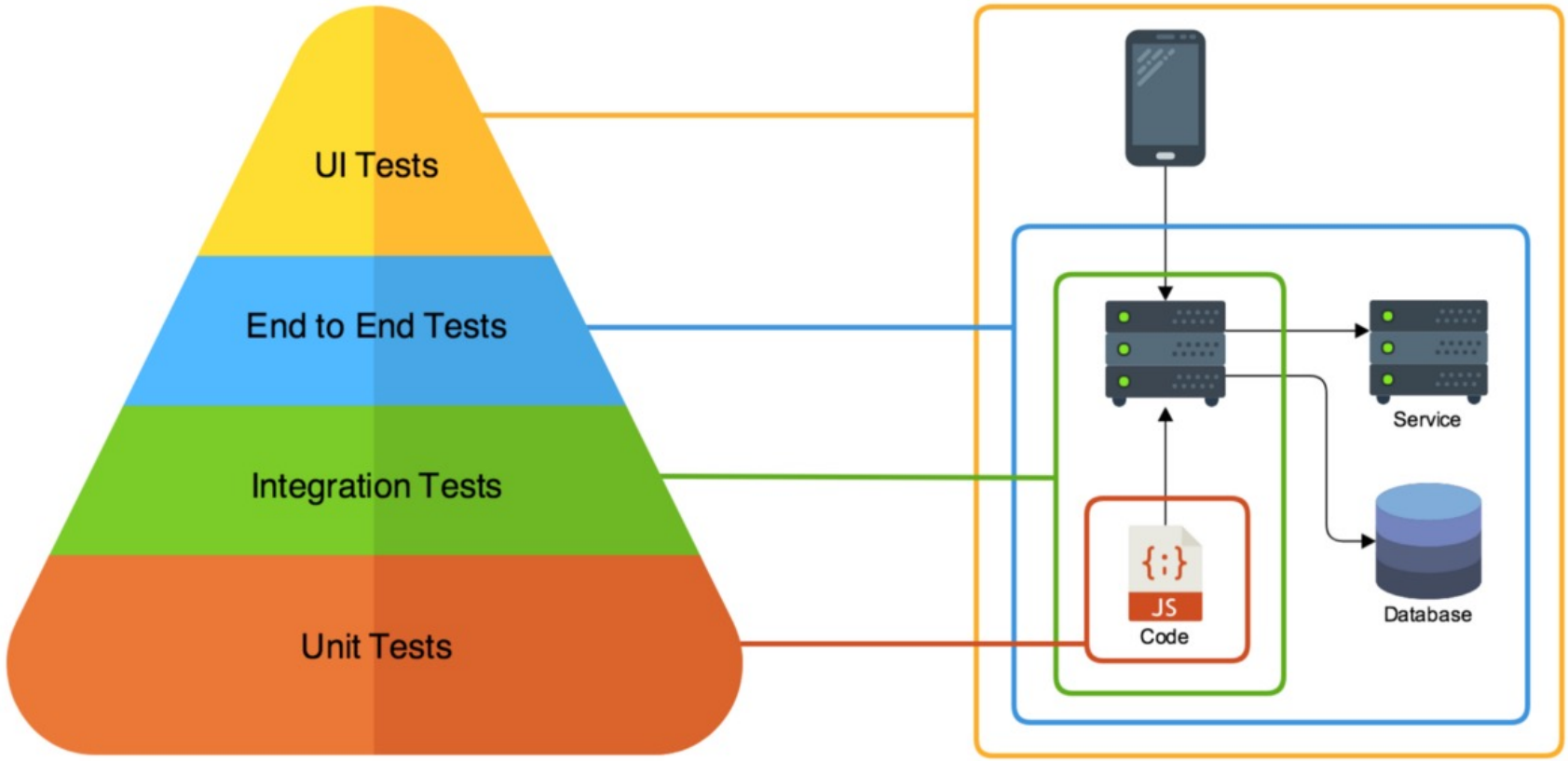
– ✓ Unit Tests laufen
– 1 Test schlägt aber fehl
– Test testet ob `false===true` logischerweise kreuzfalsch

```
@Test
@DisplayName("If no skills match, the result must be empty")
public void matchNone() {
  assertFalse( condition: true);
}
```

Erinnerung Flughöhen (L14 Architektur)



Unit Tests


UI Tests
End to End Tests
Integration Tests
Unit Tests
Service
Database
JS
Code

Warum zuerst die Tests?

- Unit Testing bevor wir «Units» haben
- Vorher
  - Alles in die main Methode und rumklicken zum Testen
- Nachher
  - Den neuen «Unit» Code direkt mit 1-n Testmethoden abdecken

- «Einfachste» Tests
  - da nichts «anderes» getestet wird ausser **1 METHODE, 1 UNIT**

- Für Java gibt es JUnit
  - diese Beispiele verwenden JUnit
  - weitere Werkzeuge sehen wir in L18

Unit: Matcher

```java
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

public class Matcher {
    public String[] isMatch(Candidate candidate, Profile profile){
        Set<String> s1 = new HashSet<~>(Arrays.asList(candidate.getSkills()));
        Set<String> s2 = new HashSet<~>(Arrays.asList(profile.getSkills()));
        s1.retainAll(s2);
        return s1.toArray(new String[s1.size()]);
    }
}
```

```java
public interface Candidate {
    public String[] getSkills();
}
```

```java
public interface Profile {
    public String[] getSkills();
}
```

- Unit Tests testen kleinste Einheiten
  - Eigenschaft: Isoliert, keine Interaktion mit anderen Units (sonst Integration)
  - Alles, was in die Unit rein geht wird statisch (fix-ture) codiert
    - z.B. Matcher, Profile und Kandidaten hardcodiert
    - Resultat `isMatch()` wird getestet

Unit Test Matches 1

```
42      @Test
43      @DisplayName("Matching candidate to profile should work if skills match")
44      public void matchOne() {
45          Candidate c = () -> new String[]{"maths", "software-design"};
46          Profile p = () -> new String[]{"maths", "algebra"};
47          String[] overlap = new String[]{"maths"};
48          assertArrayEquals(overlap, matcher.isMatch(c, p),
49                  message: "Intersection of arrays should yield 'maths'");
50      }
```

Struktur Unit Test

```
42      @Test
43      @DisplayName("Matching candidate to profile should work if skills match")
44      public void matchOne() {
45          Candidate c = () -> new String[]{"maths", "software-design"};
46          Profile p = () -> new String[]{"maths", "algebra"};
47          String[] overlap = new String[]{"maths"};
48          assertArrayEquals(overlap, matcher.isMatch(c, p),
49                  message: "Intersection of arrays should yield 'maths'");
50      }
```

```
@Test
@DisplayName("If no skills match, the result must be empty")
public void matchNone() {
    assertFalse( condition: true);
}
```

- @Test
  1 Testfall
- @DisplayName
  TestName
- Methodenname
  passend, aber egal da
  @DisplayName gesetzt
- Testaufbau
- **Assertion**
  - **Erfolg = Test bestanden**
  - **Fehler = Test fehlgeschlagen**

Task

Unit Test reparieren

AUFTRAG
⏰ 15min

Reparieren Sie den Unit Test in der Methode matchNone

/docs/tasks/bites/junit

```
@Test
@DisplayName("If no skills match, the result must be empty")
public void matchNone() {
  // 1. Create new Candidate
  // Java can run anonymous classes like
  /* Class c = new Class() {
    @Override
    public String[] getSkills() {
      return new String[]{"maths", "software-design"};
    };
  };*/

  // 2. Create a Profile
  // Profile p = ?

  // 3. Calculate the actual result
  // matcher.isMatch(candidate, profile);

  // 4. Make an assertion - an expectations
  // use assertFalse from the package
  // import static org.junit.jupiter.api.Assertions.assertFalse;
  // it can be used like assertFalse(Arrays.asList(actualResult).isEmpty(),
  // "Array should be empty");
  assertFalse(true);
}
```

Zielkontrolle

AUFTRAG
⏰ 5min

https://forms.office.com/r/tA3nnd3ff0

🔐 ABB Login

WEITER WISSEN.

Wir begleiten Sie!