DIY
JUnit-5-Test-Engine

Entwicklertag Karlsruhe 2017

@johanneslink

johanneslink.net

Softwaretherapeut

"In Deutschland ist die Bezeichnung Therapeut allein oder ergänzt mit bestimmten Begriffen gesetzlich nicht geschützt und daher kein Hinweis auf ein erfolgreich abgeschlossenes Studium oder auch nur fachliche Kompetenz." Quelle: Wikipedia

Matthias Merdes

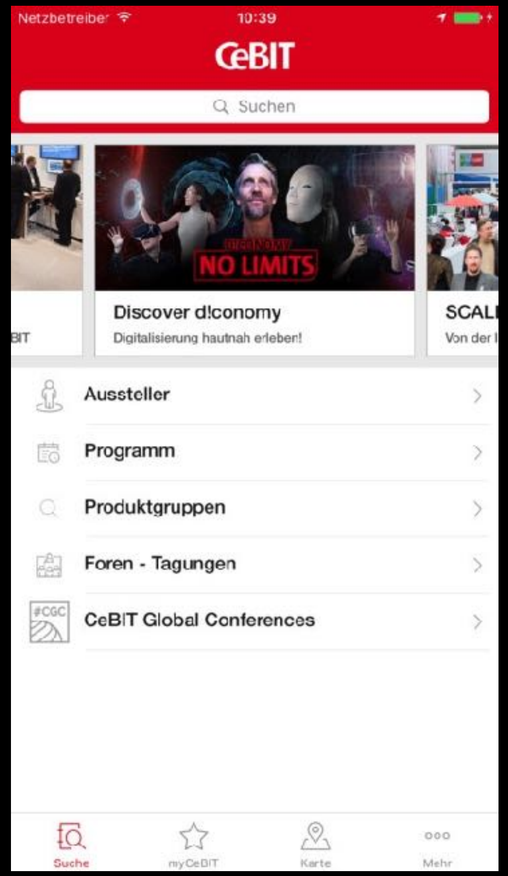
github.com/mmerdes

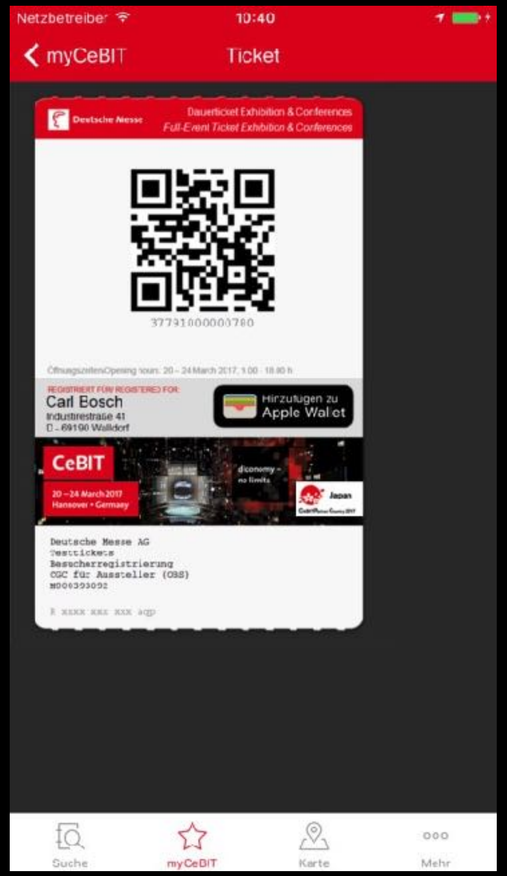
- Committer im JUnit 5-Team

- Lead Developer
  Architektur & Services
  bei Heidelberg Mobil

- Heidelberg Mobil: Sponsor für JUnit 5

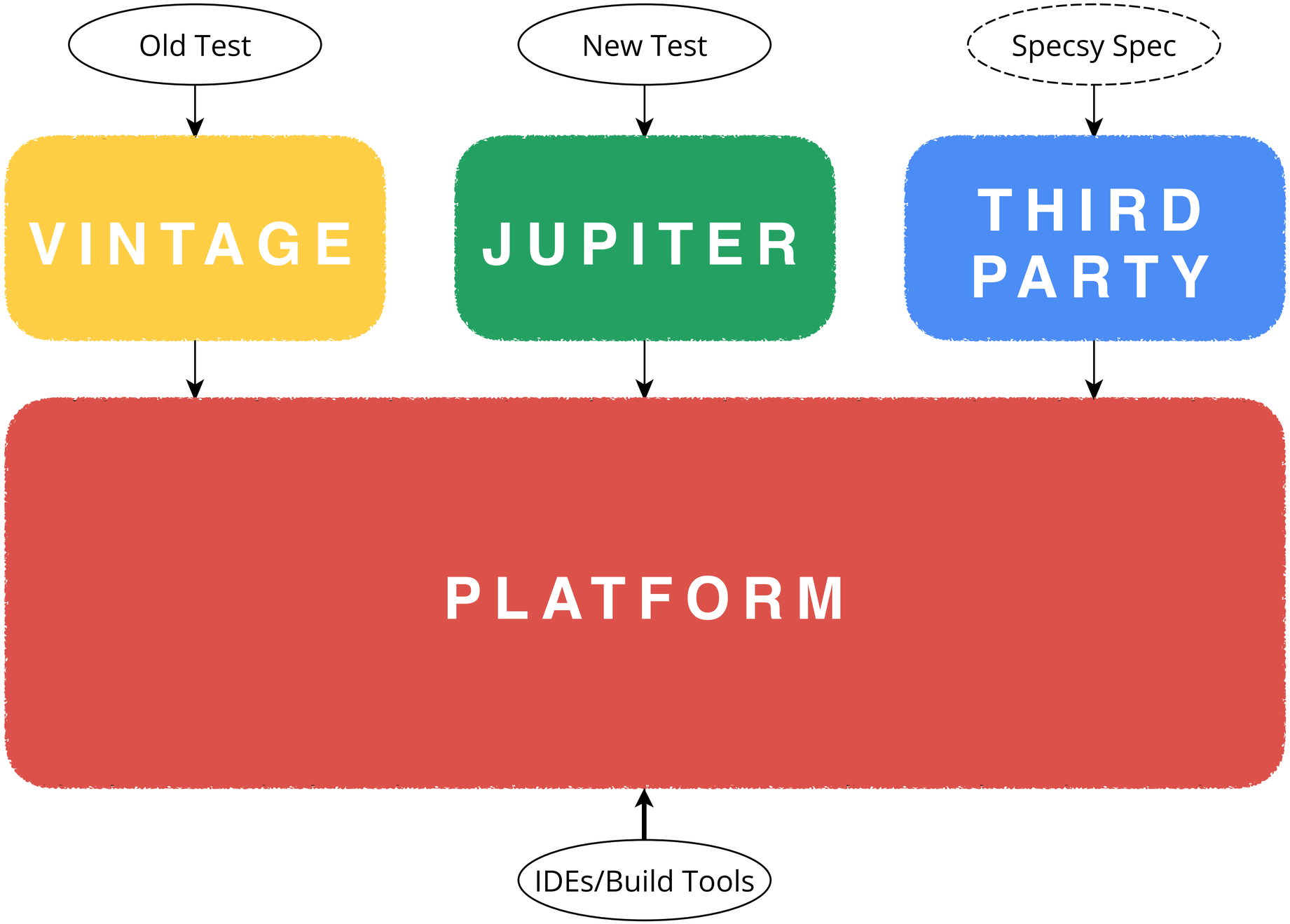
Heidelberg Mobil

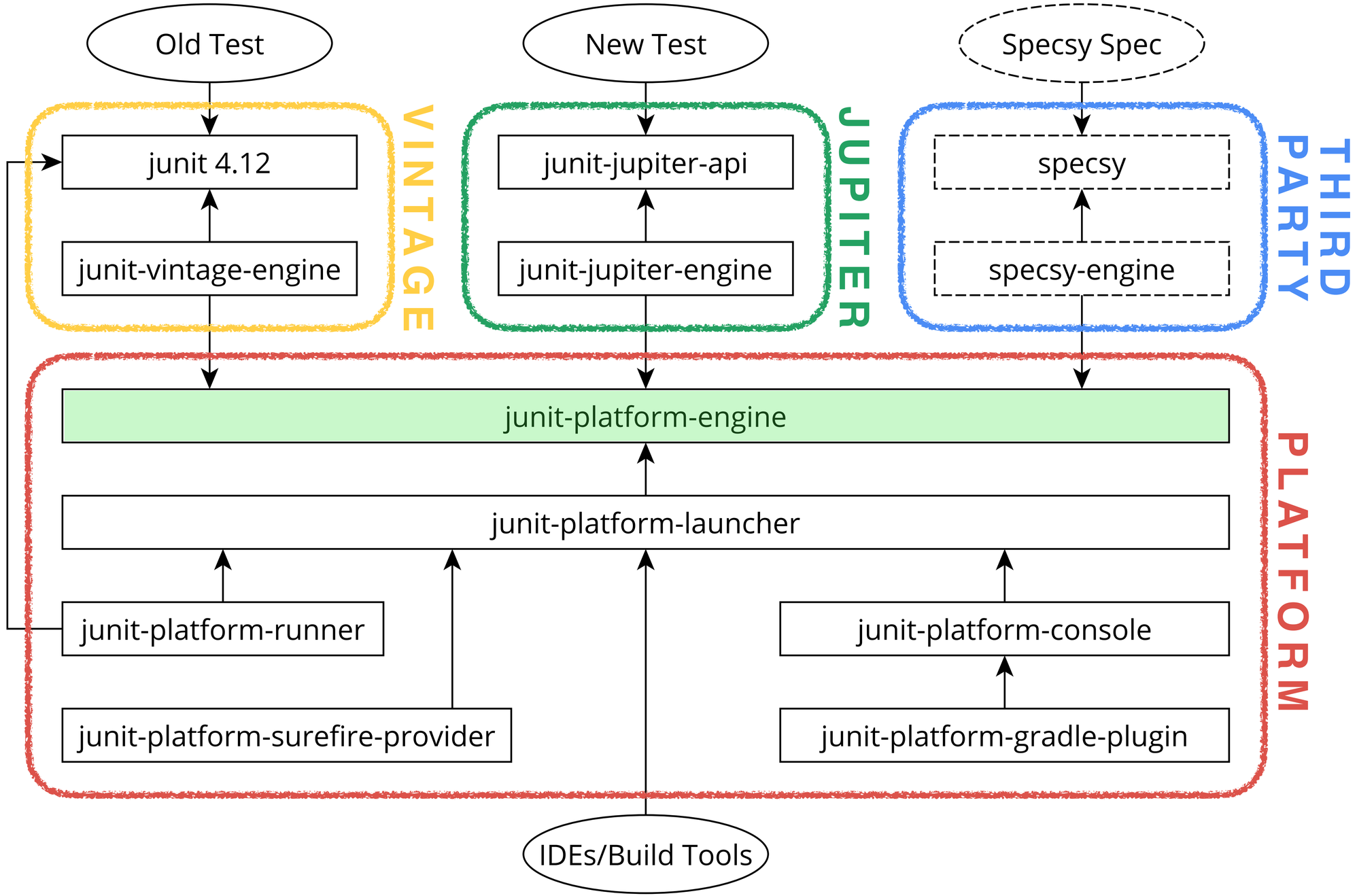


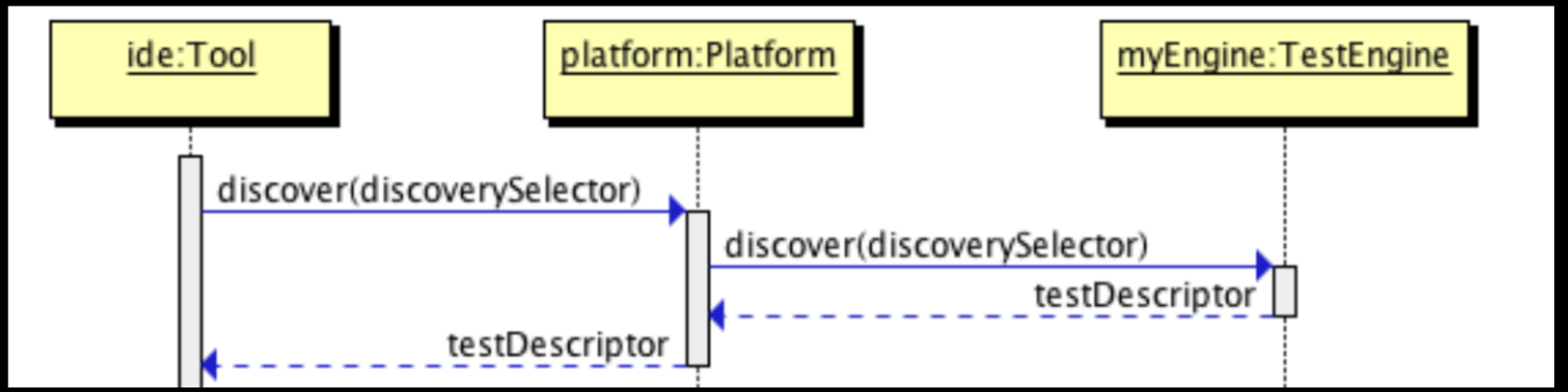


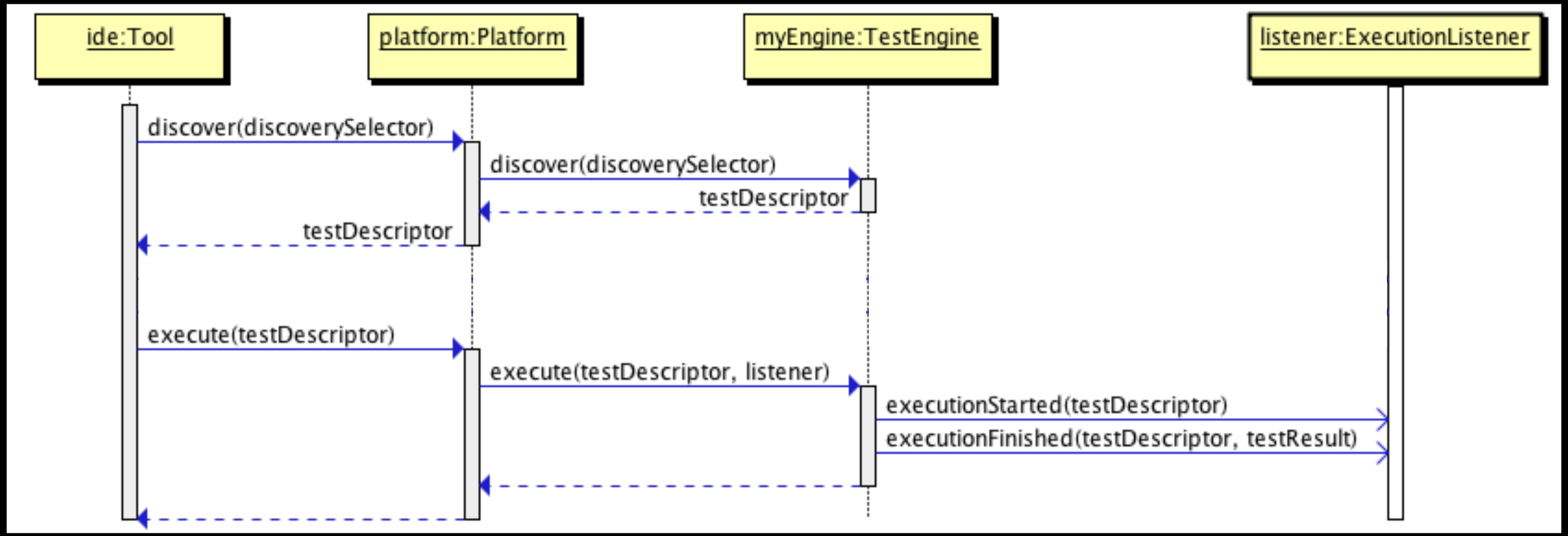
JUnit-5-Plattform Ziel

- Wunsch: JUnit 4 und 5 nebeneinander, um Adaption und Migration zu erleichtern
- Ergebnis: Leichte IDE- und Tool-Integration für *beliebige* Test-Frameworks


Old Test
New Test
Specsy Spec
VINTAGE
JUPITER
THIRD PARTY
PLATFORM
IDEs/Build Tools


Old Test
New Test
Specsy Spec
junit 4.12
junit-vintage-engine
junit-jupiter-api
junit-jupiter-engine
specsy
specsy-engine
VINTAGE
JUPITER
THIRD PARTY
junit-platform-engine
junit-platform-launcher
junit-platform-runner
junit-platform-surefire-provider
junit-platform-console
junit-platform-gradle-plugin
PLATFORM
IDEs/Build Tools


ide:Tool
platform:Platform
myEngine:TestEngine
discover(discoverySelector)
discover(discoverySelector)
testDescriptor
testDescriptor


ide:Tool
platform:Platform
myEngine:TestEngine
listener:ExecutionListener
discover(discoverySelector)
discover(discoverySelector)
testDescriptor
testDescriptor
execute(testDescriptor)
execute(testDescriptor, listener)
executionStarted(testDescriptor)
executionFinished(testDescriptor, testResult)

Test-Engine benutzen

1. TestCompile-Dependency hinzufügen:

   `org.myorg:my-engine:x.y.z`

2. Tests schreiben

Test-Engine benutzen

1. TestCompile-Dependency hinzufügen:

   ```
   org.myorg:my-engine:x.y.z
   org.junit.jupiter:junit-jupiter-engine:5.0.0-M4
   ```

2. Tests schreiben

Warum braucht die Welt mehr als eine Test-Engine?

- Andere JVM-Sprache
- Anderes Spezifikationsmodell
- Anderes Ausführungsmodell

- Manchmal genügt eine Jupiter-Extension

Test-Engine Kochrezept

1. Compile-Dependencies hinzufügen
2. `TestEngine`-Interface implementieren
3. Engine registrieren
4. Tests in IDE starten

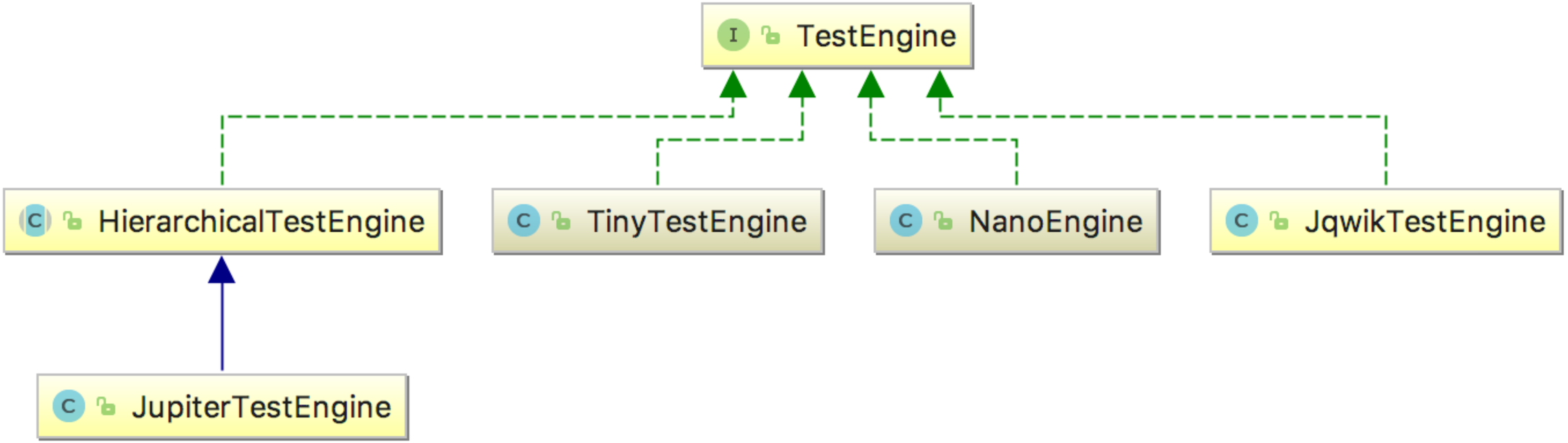
DEMO

```
dependencies {
  compile("org.junit.platform:junit-platform-engine:1.0.0-M4“)
  compile("org.junit.platform:junit-platform-commons:1.0.0-M4")

  // For writing integration tests
  testCompile("org.junit.platform:junit-platform-launcher:1.0.0-M4")

  // Only necessary to enable IntelliJ support
  testRuntime("org.junit.jupiter:junit-jupiter-engine:5.0.0-M4")
}
```

Test-Engine registrieren

`/META-INF/services/`
`org.junit.platform.engine.TestEngine`

```
nano.NanoEngine
```


TestEngine
HierarchicalTestEngine
TinyTestEngine
NanoEngine
JqwikTestEngine
JupiterTestEngine

TestEngine-Interface implementieren

1. Create TestEngine class
2. Implement discover()
3. Implement execute()

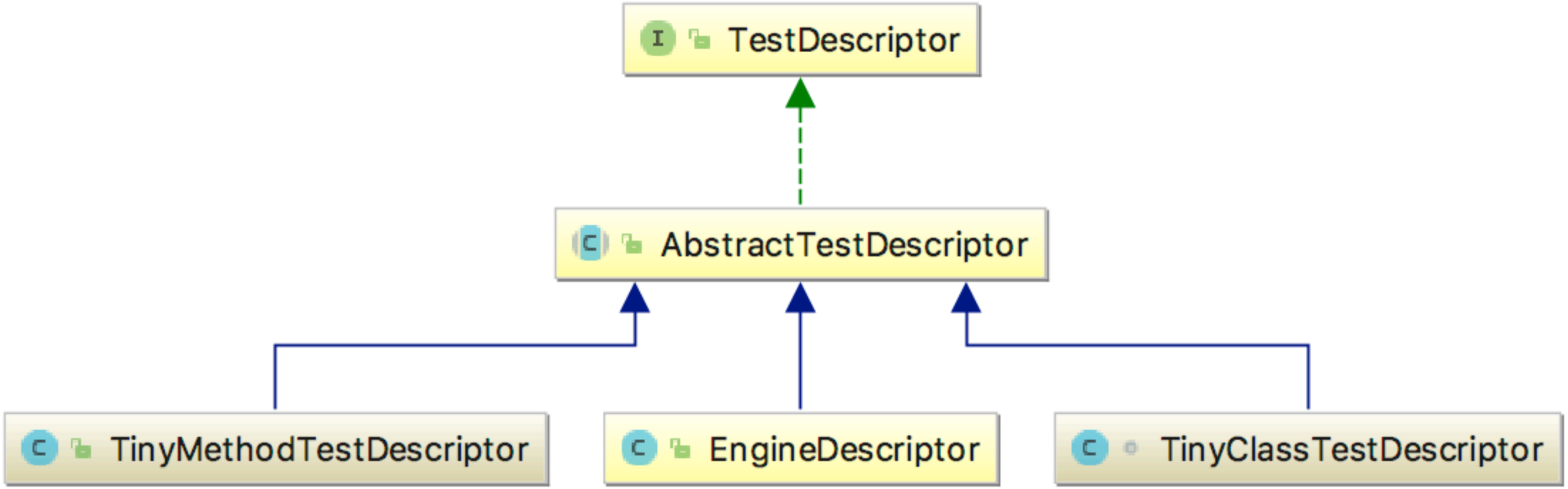
```
public interface EngineExecutionListener

 void executionStarted(TestDescriptor testDescriptor);

 void executionFinished(
     TestDescriptor testDescriptor,
     TestExecutionResult testExecutionResult
 );
```

Eine „richtige“ Testengine

- Test-Spezifikation in Klassen und Methoden
- Navigierbarkeit
- Lesbare Testnamen

TinyTest

```
@TinyTest
public class A_tiny_test {

  final int theAnswer = 42;

  public boolean this_should_return_true() {
    return theAnswer == 42;
  }

  public boolean this_returns_FALSE() {
    return theAnswer == 43;
  }
}
```


TestDescriptor
AbstractTestDescriptor
TinyMethodTestDescriptor
EngineDescriptor
TinyClassTestDescriptor

Was geht noch?

- Ausführen von Dateien, URLs, und anderen Ressourcen
- jqwik.net: Property Testen in Java
- Specsy: Alles mit Lambdas
- Wrapper für Cucumber, Spock, Fitnesse etc

Tool-Support: State of the Union

- IntelliJ
  - @Testable erforderlich für Run-Symbol am Source-Code
  - Gezielter Teststart nur für Klasse/Methode/Package
  - und andere Unstimmigkeiten…
- Eclipse
  - Geplant für 4.7.1
- Gradle und Maven vom JUnit-Team gepflegt

http://github.com/jlink/jax2017

Fragen?