Testautomatisierung mit Behaviour Driven Development

30 März 2020 | **Software Engineering** | **Marcel Stalder, Romano Roth**
Lesezeit: 4 Minutes

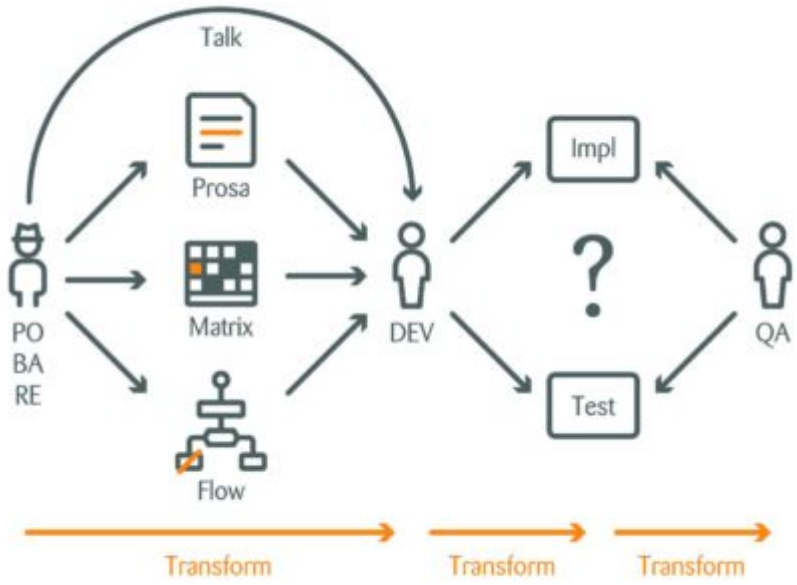
Die Automatisierung von Tests ist eine aufwändige und anspruchsvolle Aufgabe. Durch das iterative Vorgehen in der Entwicklung müssen die automatisierten Tests fortlaufend angepasst werden. Mit Behaviour Driven Development (BDD) lässt sich die Testautomatisierung vereinfachen und beschleunigen.

Eine agile und iterative Softwareentwicklung mit DevOps inklusive einer automatisierten Build- und Deployment-Pipeline verlangt nach einer guten, automatisierten Testabdeckung welche die Anforderungen an die Software abbilden.

Anforderungen für eine Business-Software zu definieren ist nicht trivial. Die grundlegende Vorstellung einer neuen Funktionalität ist meist schnell vermittelt, die Herausforderungen liegen aber in den Details. So verwenden Product Owners (PO), Business Analysts (BA) und Requirement Engineers (RE) verschiedene Formen zur Definition von Anforderungen wie Prosa-Text, Entscheidungs-Tabellen oder Sequenz-Diagramme. Auch mündlich werden Anforderungen an die Entwickler (DEV) kommuniziert, beispielsweise bei Rückfragen und Unklarheiten, oder wenn Anforderungen nicht anderweitig dokumentiert wurden.

Dies führt zu mehreren Transformations-Schritten:

1. Die Anforderungen aus verschiedenen Quellen und in verschiedenen Formaten müssen den Entwicklern vermittelt werden.
2. Die Entwickler transformieren die Anforderungen in Quellcode, einerseits für die Implementation an sich und andererseits für die automatisierten Tests.
3. Die Qualitätsverantwortlichen (QA) wiederum müssen sowohl die Anforderungen kennen und sie müssen wissen wie diese konkret umgesetzt wurden.
   Nun bleiben Anforderungen in der Regel nicht fix, sondern ändern sich. Dadurch beginnt die Transformation erneut und das Delta zur bestehenden Umsetzung muss identifiziert werden. Dieser Schritt ist zeitintensiv und erschwert das Testing und die Qualitätssicherung bei Änderungen. Mit dem Blick auf eine automatisierte CI/CD-Pipeline mit kurzen Entwicklungs- und Test-Zyklen ist dies nicht ideal.



Grafik: Zühlke

Die ausführbare Spezifikation

Was wäre, wenn die Spezifikation ausführbar wäre? Dadurch würden die Transformations-Schritte vereinfacht, es gäbe weniger Missverständnisse und der Entwicklungsprozess würde beschleunigt. Als zusätzlichen Benefit resultiert eine Dokumentation zur Funktionalität des Systems, die auch wirklich der aktuellen Implementation entspricht. Daraus sollten eine bessere Qualität und somit weniger Bugs resultieren.

Eine Möglichkeit zur ausführbaren Spezifikation ist Behavior Driven Development (BDD). BDD ist ein agiler Software-Entwicklungsprozess mit Wurzeln im Test Driven Development (TDD) und Ideen aus dem Domain Driven Design (DDD).
In BDD kann das gewünschte Verhalten einer Software beispielsweise direkt in einer User Story definiert werden:

```
Feature: Story title

As a role I want the feature so that the benefit.

Scenario 1: Scenario title
Given precondition
When action
Then postcondition

Scenario 2: Scenario title
Given precondition
When action
Then postcondition
```

Bei der Definition wird die Syntax Gherkin verwendet. Diese verwendet als Grundstruktur Given-When-Then zur Beschreibung der Vorbedingung, der Aktion und des erwarteten Resultats. Da es sich um reinen Text handelt, kann die Beschreibung des Verhaltens mit verschiedenen Technologien genutzt werden, unter anderem:

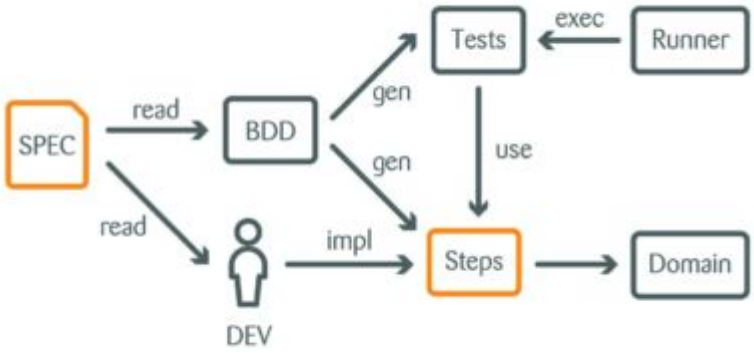
- SpecFlow für .NET
- Cucumber.js für JavaScript
- Cucumber-JVM für Java
- Behave für Phyton

Ein Beispiel aus der Praxis

Gestartet wird mit der Spezifikation des gewünschten Verhaltens. Als Beispiel fungiert eine (vereinfachte) Berechnung der Market Value einer Position eines Wertschriftenportfolios:

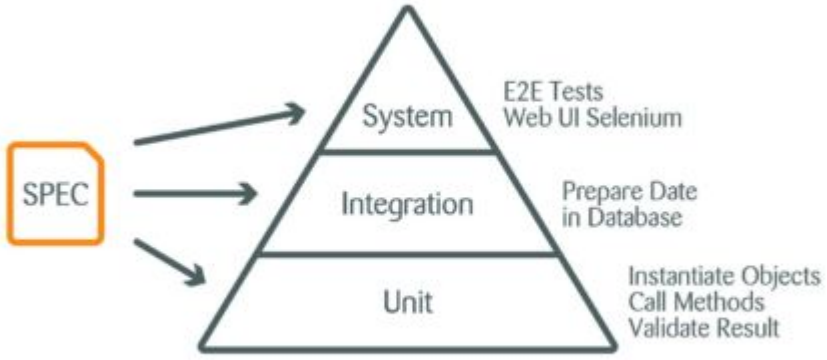
```
Feature: Market Value Calculation

Scenario: Market Value calculation for position
  Given Instrument with Rate 100
    And Position with Quantity 10
  When values are calculated
  Then Position MarketValue is 1000
```

Das BDD-Tool liest diese Spezifikation ein und generiert für jede Given/When/Then-Zeile eine Step-Definition. Step-Definitions sind Methoden im Test-Code und dienen als Bindeglied zu den Domänen-Klassen. Die Aufgaben der Entwickler besteht nun darin, die generierten Step-Definitions auf Basis der Spezifikation zu Implementieren. Anschliessend wird die Spezifikation über den Runner des eingesetzten Test-Frameworks ausgeführt.



Grafik: Zühlke

Wie die Domain-Klassen vermuten lassen, wird hier BDD für Unit-Tests genutzt, um die Businesslogik zu testen. Die automatisierte Ausführung der Spezifikation ist aber nicht auf Unit-Test begrenzt, sondern kann auf allen Ebenen der Testpyramide verwendet werden. Viele Beispiele für BDD arbeiten auf System-Ebene und automatisieren UI-Tests mittels Selenium. Aber auch Integration-Tests sind möglich, um beispielsweise Daten in einer Datenbank zu erstellen oder zu verifizieren.

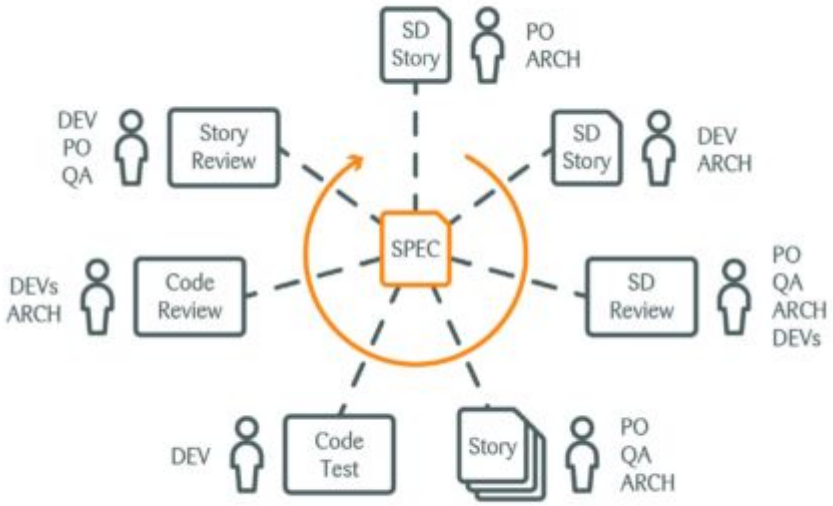


Grafik: Zühlke

Der Entwicklungsprozess

Die Spezifikationen sind meist einiges komplexer als das vereinfachte Beispiel aus der Praxis. Aus diesem Grund wird vor der eigentlichen Umsetzung ein Solution Design (SD) erstellt. Dieses SD beschreibt unter anderem Ist- und Soll-Situation sowie die Implementations-Schritte (Stories). Zentrales Element ist dabei die Spezifikation in Gherkin. Diese Spec wird beginnend mit der User-Story für das Solution Design stetig erweitert und präzisiert. Auch in

den Reviews ist die Spec ein zentrales Element, um die Anforderungen und die aktuelle Implementation zu verifizieren.



Grafik: Zühlke

Behaviour Driven Development (BDD) mit der Spezifikation des Applikations-Verhaltens in Gherkin ist ein empfehlenswerter Schritt, um komplexe Fachanforderungen automatisiert zu testen. Da die ausführbare Spezifikation bereits zu Beginn der Entwicklungsarbeiten verfügbar ist, ist Test Driven Development (TDD) einfach umzusetzen. Voraussetzung für eine konsistente Spezifikation ist eine gemeinsame Sprache der am Entwicklungsprozess beteiligten Personen. Aus diesem Grund ist Domain Driven Design (DDD) mit einem zentralen und dokumentierten Domain-Model eine gute Grundlage.

Dank Behaviour Driven Development (BDD) und einer automatisierten Build- und Deployment-Pipeline können Unternehmen die Feedback-Zyklen sowie die Durchlaufgeschwindigkeit steigern. Dadurch reagieren sie schneller und können so ihre Kunden durch stetige Innovation begeistern. Durch DevOps werden Prozesse automatisiert und optimiert: von der Entwicklung über das Release bis hin zum Betrieb. Dies sorgt für einen Effizienzgewinn und schliesslich tiefere Änderungskosten.