Specification by Example
Wie man den Kunden an Board holt

Markus Unterauer

Specification by Example bedeutet, Anforderungen durch konkrete Beispiele zu spezifizieren. Dazu wird ein fixes Satzschema verwendet, welches ein einfaches Andocken von Testautomatisierung ermöglicht. So wird aus einem wertlosen Write-Only Dokument eine wertbringende lebende Spezifikation.

Eine der schwierigsten Aufgaben in der Softwareentwicklung ist es, herauszufinden, was das System eigentlich tun und können soll. An dieser Anforderungserhebung sind viele Stakeholder beteiligt. Die Information über die Anforderungen des Kunden wandert dabei über zahlreiche Stationen. Der Kunde erzählt es dem Business Analysten, der gibt es an den Projektleiter wieder. Dieser bespricht es mit dem Architekten und vielleicht externen Beratern und gibt es dann an den Entwickler weiter. Der baut das System und gibt es gemeinsam mit den zu prüfenden Anforderungen an den Tester weiter. Schlussendlich landen System und Spezifikation bei der Wartungs- und Betriebsmannschaft. Bei jedem Schritt in dieser Kommunikationskette geht Information verloren, wie beim Spiel „Stille Post“. Erschwert wird die Anforderungsanalyse noch dadurch, dass es meist nicht nur einen Kunden gibt, sondern viele und somit das System viele Szenarien, Varianten und Ausprägungen unterstützen muss.

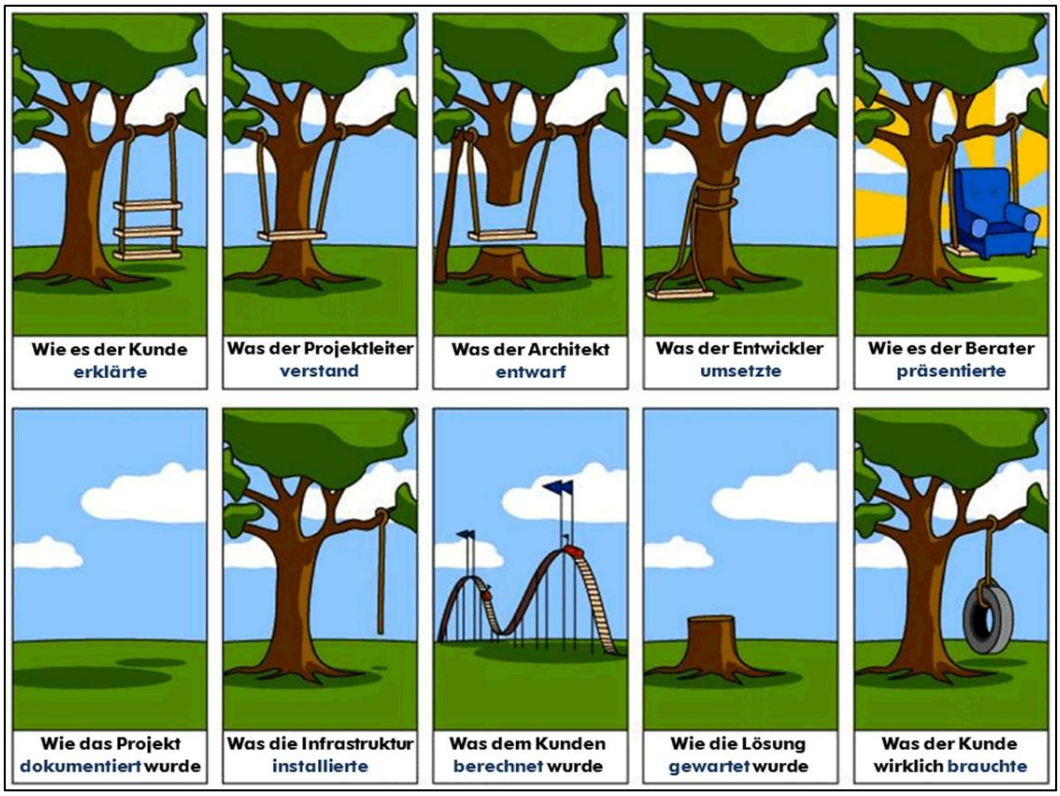

Abb. 1: Bei der Weitergabe von Anforderungen geht oft Information verloren (Quelle: http://www.connexin.net/de/humor-witze/projekt-management/)

Zusätzliche Verwirrung entsteht, wenn die Information nicht mündlich, sondern als schriftliche Spezifikation in Form von Text und Modellen weitergegeben wird. Was für den einen völlig klar und verständlich ist, ist für den anderen nur eine sinnlose Aneinanderreihung von Zeichen. Oder schlimmer noch: Der Empfänger der Spezifikation glaubt alles verstanden zu haben, in Wirklichkeit aber weicht sein Bild von den Anforderungen deutlich von dem des Kunden ab.

Das Ergebnis sind oft schlecht designte Systeme, die die Bedürfnisse der Kunden nicht erfüllen. Diese Systeme werden in vielen Fällen trotzdem in Betrieb genommen und erst im laufenden Betrieb richtiggestellt. Man spricht hier auch von „Bananensoftware". Wie eine Banane reift auch die Software erst beim Kunden.

Gesucht sind also Methoden, die uns helfen, Anforderungen vom Kunden zu erheben und diese so an alle Beteiligten zu kommunizieren, dass dabei keine Information verloren geht. Specification by Example ist so eine Methode.

Die Grundidee hinter Specification by Example ist, Anforderungen gemeinsam in Form von konkreten Beispiele zu spezifizieren und nach einem einfachen Schema zu dokumentieren. Das klare und immer gleiche Schema fördert die Verständlichkeit, die konkreten Beispiele erhöhen die Eindeutigkeit und machen es leichter, die Anforderungen auf Vollständigkeit zu prüfen.

Specification by Example basiert dazu auf 5 Grundprinzipien:

1. Scope von Zielen ableiten: Jede Anforderung wird auf ein Geschäftsziel und einen damit verbundenen Business Value zurückgeführt.
2. *Kollaboration*: Anforderungen und konkrete Beispiele werden gemeinsam erarbeitet. Entwicklungsteam, PO und Kunde sind beteiligt.
3. Konkrete Beispiele: Anforderungen werden in Form von konkreten Beispielen und nicht abstrakten Umschreibungen spezifiziert.
4. Lebende Dokumentation: Durch den Einsatz von Tools können so formulierte Anforderungen direkt automatisiert getestet werden.
5. Wiederkehrende Validierung: Durch eine hohe Abdeckung der Anforderungen durch automatisierte Tests ist eine erneute Prüfung z.B. nach Änderungen leicht möglich.

Der erste Schritt ist also immer, Geschäftsziele und den zu erreichenden Nutzen des Systems zu definieren. Teil dieser Zieldefinition ist die Überlegung, für wen das Ziel wichtig ist. Dieser Aspekt des „für wen" wird oft in Form von Personas, also erfundenen Beispiel-Anwendern und -Kunden beschrieben. Im nächsten Schritt werden Geschichten erzählt, wie diese Personas das System nutzen werden. Für jede dieser Geschichten wird als Merker ein Satz mittels der User Story Satzschablone geschrieben:

„Als <Rolle> möchte ich <Aktion>, damit <Nutzen>"

Zu jeder User Story werden nun Bedingungen definiert, die erfüllt sein müssen, damit die User Story abgenommen wird und Wert für den Kunden liefert. Diese

Abnahmebedingungen, auch Akzeptanzkriterien genannt, werden wieder nach einer definierten Struktur, dem „Gherkin“ Schema beschrieben:

„Gegeben <Ausgangszustand>, wenn <Aktion>, dann <Ergebnis>“

Inhaltlich werden dabei keine abstrakten Sachverhalte beschrieben, sondern konkrete Beispiele gegeben. Ziel ist es, für jede User Story so viele Beispiele zu spezifizieren, dass alle möglichen Szenarien, Varianten und Sachverhalte der Story abgedeckt sind. Oft gibt man den einzelnen Szenarien noch einen griffigen Titel.

Beispiel:
Ziel:
Bauen eines Smart Home, das einer kleinen Familie das Leben erleichtert.

User Story:
Als arbeitender Papa möchte ich rechtzeitig geweckt werden, damit ich pünktlich in der Arbeit bin.

Akzeptanzkriterien in Form von Beispielen:

- *„Szenario ‚normaler Arbeitstag‘:*
  Gegeben es gibt keine Verkehrsbehinderungen, wenn ich ‚Arbeitstag‘ aktiviere, dann ist die Weckzeit 8:15“
- *„Szenario ‚Stau‘:*
  Gegeben ich brauche normalerweise 30 Minuten zur Arbeit und ich habe um 9 Uhr das Daily Scrum Meeting und es gibt 20 Minuten Stau, wenn ich ‚gemütlich Frühstücken‘ aktiviere, dann ist die eingestellte Weckzeit 6:10.“
- *„Szenario ‚Party am Vorabend‘:*
  Gegeben ich brauche 30 Minuten zur Arbeit und ich habe um 9 Uhr das Daily Scrum Meeting und es gibt keine Verkehrsbehinderungen, wenn ich ‚lange schlafen‘ aktiviere, dann ist die eingestellte Weckzeit 8:15.“

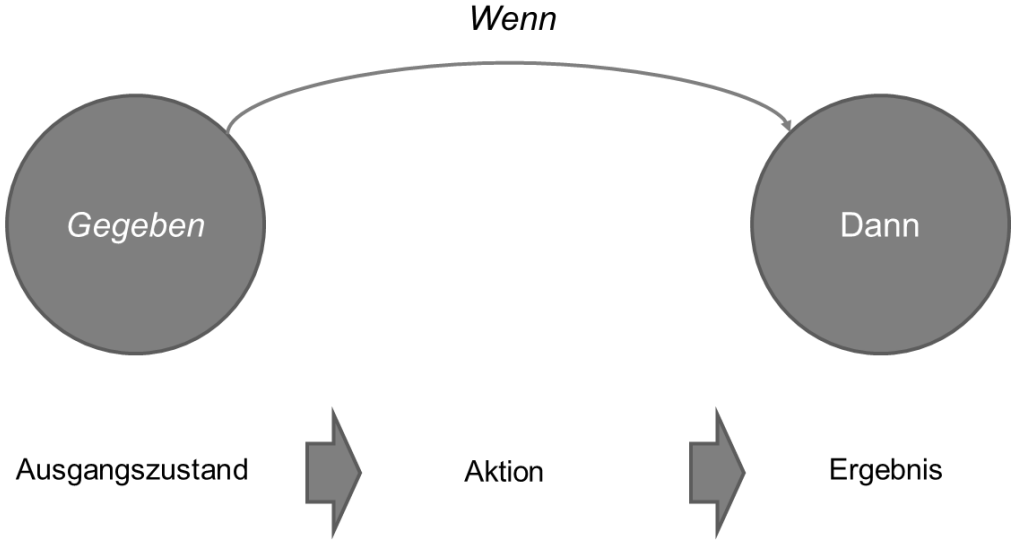

Abb. 2 Akzeptanzkriterien werden als konkrete Beispiele nach dem Gherkin-Schema („Gegeben … wenn … dann“) spezifiziert.

Die konkreten Beispiele in den Akzeptanzkriterien sollen das Verhalten des Systems in allen Varianten beschreiben. Man spricht deshalb oft auch von „Behaviour Driven Development“. Beim Schreiben von User Stories und Akzeptanzkriterien nach diesem Ansatz haben sich einige Best Practices herauskristallisiert:

- Halte die Szenarien kurz.
- Teile ein komplexes Szenario in mehrere einfache auf.
- Beschriebe Szenarien aus Sicht des Anwenders.
- Halte alle Szenarien auf der gleichen Abstraktionsebene.
- Verwende Keine technischen Begriffe (u.a. UI, REST).
- Verwende Begriffe aus der Fachdomäne.
- Verwende konkrete Zahlen, Daten, Fakten.
  (also statt „Wert eingeben“ die konkrete Zahl, z.B. „5.212,65 eingeben“)

Durch die Verwendung einer standardisierten Satzstruktur können User Story und Akzeptanzkriterien als Andockpunkt für automatisierte Tests verwendet werden. Tools wie Cucumber und Fitnesse ermöglichen es User Stories und Akzeptanzkriterien zu verwalten und sie um Testcode zu ergänzen.

```
Feature: User authentication
User Story: In order to protect access to the HMI
As a system administrator
I want that all users need to be authorized

Scenario: Autologin after 60 seconds
Given HMI started with default configuration
When user waits for 60 seconds at the login screen
Then Default user should be logged in automatically

[Given("HMI started with default configuration")]
public void GivenTheStartedHMIWithMachineConfiguration()
{
   TestBase.loginForm = TestBase.app.Start();
}

[Then("(.*) user should be logged in automatically")]
public void ThenLoggedInUser(string user)
{
  Assert.That(frameFunctions.UserGroup,Is.EqualTo(user));
}
```

Manche Tools erlauben es auch, die konkreten Werte für die Szenarien in einer Tabelle darzustellen. Das Szenario selbst beinhaltet dann Platzhalter. Dies ist kürzer und übersichtlicher, als würde man für jedes Szenario den vollen Satz nach dem Gherkin-Schema schreiben. Durch das Auslagern der Werte in eine Tabelle geht allerdings auch eine Spur Konkretheit in der Formulierung der Akzeptanzkriterien verloren.

```
Szenariogrundriss: Plausbilitätsprüfung im Handbetrieb
  Angenommen die Werte '<Werkstoff>', <Blechdicke>,
     <Biegewinkel> im Handbetrieb
  Wenn der Bediener die Vorberechnung startet
  Dann ist 'UpperDeadPoint' <= 'MutePoint1'
  Und 'ClampingPoint' <= 'LowerDeadPoint'
  Und 'DecompressionPoint' <= 'LowerDeadPoint'
  Und 'ClampingPoint' - 'MutePoint1' = 'MuteDistance1'

  Beispiele:
    | Werkstoff | Blechdicke | Biegewinkel |
    | 1.0038    | 0,1        |  90,0       |
    | 1.0038    | 1,0        |  90,0       |
    | 1.0038    | 3,0        |  90,0       |
```

Sind Anforderungen auf diese Weise mit automatisiert ausführbaren Testfällen verbunden, spricht man von einer „lebenden Spezifikation“. Der gesamte Ablauf der Erstellung einer solchen von der Anforderung bis zum Test besteht aus 6 Schritten:

1. Spezifizieren
2. Test-Schritte implementieren
3. Test ausführen
4. Programmcode schreiben
5. Kontinuierliches Feedback von den Tests einholen
6. Fertig!

Dieses Vorgehen unterstützt also einen „Test-First“ Ansatz, bei dem zuerst ein vollständiges Set an automatisierten Tests geschrieben wird, welche die Anforderung vollständig abdecken. Danach erst wird solange Programmcode geschrieben, bis alle Tests grün sind.

Richtig angewandt ermöglicht es Specification by Example Sonderfälle früher und vollständiger zu beachten und dadurch Fehler zu vermeiden. Durch die Beteiligung von Kunde, PO und Team bei der Erstellung wird ein gemeinsames Verständnis der Anforderungen gefördert. Die Automatisierung der Tests schließlich ermöglicht es, auch bei Änderungen an Anforderungen jederzeit sicher zu sein, dass noch alle Szenarien einwandfrei funktionieren.

Quellen

Gojko Adzic, „Specification by Example“, Manning Verlag, 2011

Johannes Bergsmann, Markus Unterauer, “Requirements Engineering für die agile Softwareentwicklung”, dpunkt.Verlag, 2014

Markus Unterauer, „Workshops im Requirements Engineering“, dpunkt.Verlag, 2014

Autor

Markus Unterauer hat Wirtschaftsinformatik studiert. In seiner Berufspraxis war er in vielen Bereichen der Softwareentwicklung, wie Architektur, Entwurf, Entwicklung, Testen und Testautomatisierung, tätig. Er lernte dabei sowohl klassische als auch agile Methode intensiv kennen. Seit 2012 arbeitet Markus Unterauer bei Software Quality Lab und leitet dort das Beraterteam. Er ist zertifizierter Scrum Master und hat sich auf die Bereiche Softwareprozesse und Anforderungsmanagement spezialisiert. Markus Unterauer ist als Vortragender in diesen Themenbereichen immer wieder auf Konferenzen tätig.

Kontakt

Internet: http://www.software-quality-lab.com
E-Mail: markus.unterauer@software-quality-lab.com