Konstantin Diener

(konstantin.diener@cofinpro.de)
ist Leading Consultant bei der Cofinpro AG. Er beschäftigt sich seit über zehn Jahren mit Softwarearchitektur und sein Interesse gilt allem, was IT und Fachabteilungen flexibel und produktiv macht (Groovy, Drools, Scrum, Kanban, DevOps, fachliche Akzeptanztestverfahren).

Modellbasierte Fachtests
Test-Driven Development für Domänenexperten

In vielen Projekten sind manuelle Akzeptanztests durch die Fachexperten Realität. Aufgrund des hohen Aufwands werden diese Tests sehr selten und eher spät im Entwicklungszyklus ausgeführt. Spätes und seltenes Feedback führt jedoch unweigerlich zu Abweichungen gegenüber den Anforderungen. Ein modellbasiertes Werkzeug unterstützt demgegenüber den Fachexperten bei der Beschreibung automatisierbarer Testfälle in „seiner Sprache". Die Entwickler können diese Testfälle kontinuierlich ausführen und auf dieser Basis akzeptanztestgetrieben entwickeln.

Software entsteht durch einen Übersetzungsprozess funktionaler Anforderungen in eine technische Implementierung. Während der Entstehung versuchen die Softwareentwickler, die Fachdomäne der zukünftigen Anwender und die aus dieser Domäne entstehenden Anforderungen möglichst gut zu verstehen. Ob die Domäne und die Anforderungen richtig verstanden worden sind, lässt sich am effektivsten ermitteln, indem die Softwareentwickler „Was wir verstanden haben" in Form von funktionsfähiger Software vorstellen und die zukünftigen Anwender ihnen anhand dieser Software Feedback geben.

Warum lohnen sich häufige Tests?

Die Situation der Softwareentwickler ist vergleichbar mit der von Schülern oder Erwachsenen, die beispielsweise eine neue Fremdsprache lernen. Auch diese Personen sind auf Feedback derjenigen angewiesen, die ihnen die Sprache beibringen. Damit das Feedback objektiv und nachvollziehbar ist, wird der Lernerfolg in Form von Tests ermittelt. Die Schüler sehen an den Ergebnissen der Tests sofort Defizite und können die Tests im Idealfall auch im Selbststudium wiederholen („Übungsaufgaben").

In der Softwareentwicklung verhält es sich analog. Ob die Domäne und die Anforderungen richtig verstanden und umgesetzt wurden, lässt sich am Besten durch Tests ermitteln. Diese „Übungsaufgaben für die Software" werden durch die Domänenexperten (also die späteren Anwender) definiert und konfrontieren die Software mit realen Problemstellungen.

Beim Erlernen einer Sprache ist es nützlich, wenn der Lernerfolg kontinuierlich in Form von Tests beziehungsweise Übungsaufgaben geprüft wird. Je länger sich etwas Falsches oder falsch Verstandenes beim Schüler manifestiert, umso schwerer wird die Korrektur des Wissens. Continuous Integration (CI) [Fowl06] überträgt diese Erkenntnis auf die Softwareentwicklung, indem die Softwareentwickler Tests für eine Anwendung definieren, die im Idealfall bei jeder Änderung am Quellcode ausgeführt werden. Die Metapher der kontinuierlichen

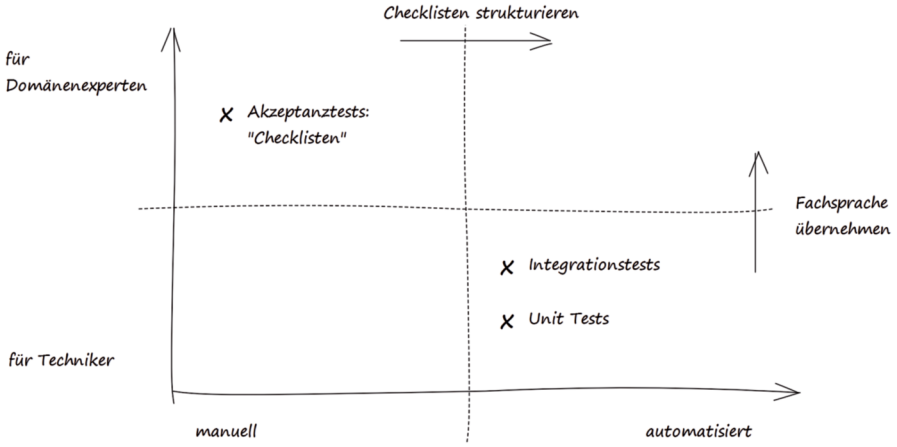

Abb. 1: Einordnung von Softwaretests

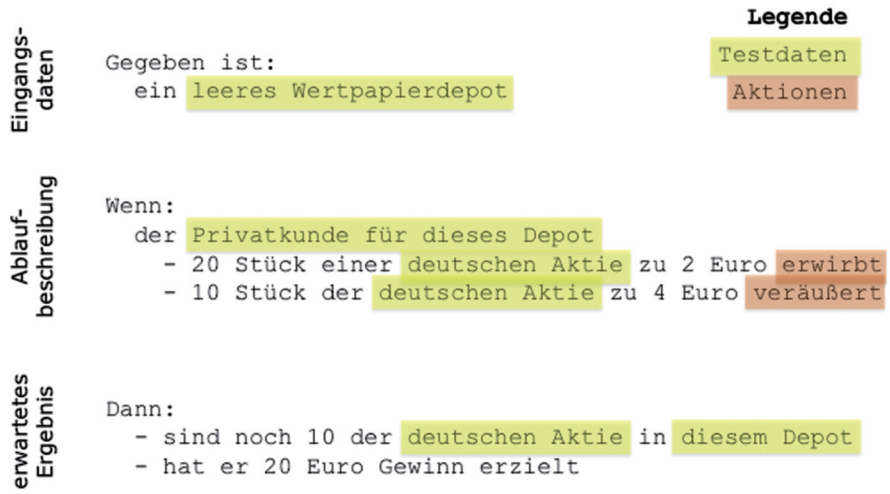

Abb. 2: Strukturiertere Testfallbeschreibung

Übungsaufgaben führt noch zu einer weiteren Erkenntnis im Kontext der Softwareentwicklung. Je länger der Zeitraum ist, in dem eine Software entsteht und keine Tests vorliegen, desto schwieriger und teurer wird eine spätere Korrektur. Test-Driven Development (TDD) [Beck02] greift diesen Punkt auf. Beim TDD werden erst die Tests und dann die Software erstellt. Die beiden Ansätze CI und TDD haben allerdings den Nachteil, dass die Testfälle nicht durch die Domänenexperten selbst, sondern durch die Softwareentwickler definiert werden.

Warum testen die Domänenexperten selten?

Übertragen auf die Metapher des Sprachschülers bedeutet dieses Vorgehen, dass die Schüler selbst die Übungsaufgaben definieren. Da sich das Erlernte aller Schüler ergänzt, entstehen bessere Resultate, als wenn nur ein Schüler sich selbst testen würde. Sie ersetzen aber nicht die Tests der Experten, also der Lehrer.

Die späteren Nutzer einer Anwendung sind Experten in der Domäne und können am besten entscheiden, ob die umgesetzten Anforderungen in Form von Software korrekt umgesetzt wurden. Tests aus der Anwenderperspektive werden in der Regel allerdings erst sehr spät im Entwicklungszyklus einer Software definiert und zudem nur ein- bis zweimal ausgeführt. Diese Phase wird oft als Akzeptanz- oder Abnahmetest bezeichnet.

Warum erfolgt die Ausführung dieser offensichtlich wertvollen Testfälle so spät und so selten? Zum Verständnis hilft es, sich die Tests etwas genauer anzusehen.

In den meisten Fällen handelt es sich bei den Testfällen um Checklisten, die von Domänenexperten für Domänenexperten in deren Sprache erstellt und manuell durchgeführt werden. Diese manuellen Tätigkeiten sind sehr aufwendig und passieren deshalb möglichst selten. Erschwerend kommt oft hinzu, dass die Software den Experten erst recht spät und „vollständig“ zum Test zur Verfügung gestellt wird.

Abbildung 1 zeigt die zwei Welten der Softwaretests. Um kontinuierliches, qualitativ hochwertiges Feedback für die Softwareentwicklung zu erhalten, müsste ein Testvorgehen im Quadranten rechts oben angesiedelt sein und die Inhalte der Domänenexperten sowie die Automatisierungstechnologie zusammenbringen. Zu diesem Zweck müssen zum einen die Checklisten strukturiert werden und zum anderen muss die Automatisierungstechnologie die Fachsprache übernehmen.

Checklisten strukturieren

Die Checklisten der Domänenexperten bestehen aus einzelnen Ausführungs- und Prüfschritten. Diese Schritte sind in der Regel in Form von Fließtext beschrieben:

- Lege ein leeres Wertpapierdepot an
- Erwirb 20 deutsche Aktien zu je 2,00 Euro
- Verkaufe 10 Stück zu je 4,00 Euro
- Prüfe, ob sich noch 10 Stück im Depot befinden
- Prüfe, ob 20 Euro Gewinn entstanden sind

Für einen Menschen, der über das entsprechende Domänenwissen verfügt, ist diese Beschreibung verständlich, für eine maschinelle Ausführung jedoch zu wenig strukturiert.

Durch Methoden aus dem Behaviour-Driven Development [DanN] lässt sich das Beispiel in einem ersten Schritt wie in Abbildung 2 strukturieren.

Damit gliedert sich die Beschreibung in drei Blöcke: „Eingangsdaten und Vorbedingungen“, „Ablauf“ und „erwartetes Ergebnis“ beziehungsweise „Nachbedingung“. In allen drei Blöcken werden Begriffe verwendet, die dem Domänenexperten geläufig sind und für die maschinelle Ausführung näher definiert werden müssen. Bei den Begriffen handelt es sich zum einen um Testdaten (wie leeres Wertpapierdepot, deutsche Aktie usw.) und zum anderen um Aktionen (wie Erwerben oder Veräußern).

Testdaten

Damit ein Computerprogramm mit den Testfällen umgehen kann, müssen vorab die Ebenen Struktur (Syntax) und Inhalt (Semantik) dieser Daten definiert sein. Auf

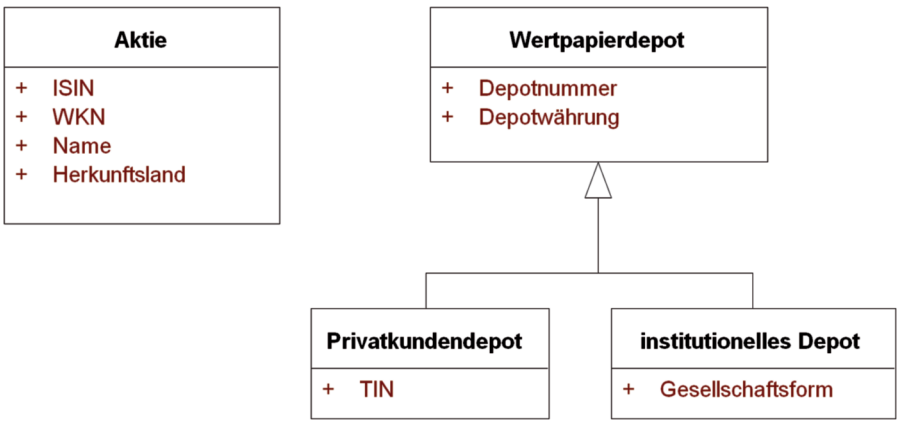

Abb. 3: Modelle für Testdaten

	deutsche Aktie	US-Aktie	sonstige Aktie
ISIN	DE000CBK1001	US0378331005	CH0038863350
WKN	CBK100	865985	A0Q4DC
Name	dt. Beispielaktie	US-Beispielaktie	sonst. Beispielaktie
Herkunftsland	Bundesrepublik Deutschland	USA	Schweiz

Tab. 1: Testdatenvorlagen für die drei Äquivalenzklassen

der semantischen Ebene wird das Prinzip der Äquivalenzklassen angewendet. Als Beispiel soll die in der Checkliste verwendete „deutsche Aktie“ dienen.

Das Substantiv „Aktie“ ist der Einstieg in die strukturelle Beschreibung der Testdaten. Die Domänenexperten definieren, welche Datenfelder im Bezug auf eine Aktie für ihren Test relevant sind. In diesem Beispiel sollen alle Aktien-Testdaten gleich aufgebaut sein und bilden somit das Schema von Aktien durch Festlegung in Form von „Eine Aktie besteht immer aus ...“:

- ISIN/ WKN
- Name
- Herkunftsland der Aktie

Sollte es im Hinblick auf den Test verschiedene Arten von Aktien geben, bilden die Experten entsprechende Äquivalenzklassen. So könnte es sein, dass Wertpapierdepots von Privatpersonen und institutionellen Anlegern gemeinsame Merkmale besitzen, aber auch jeweils Spezifika.

Die Strukturdefinition für Testdaten lässt sich mit der Definition von Klassen in objektorientierten Programmiersprachen (oder bspw. struct in C) vergleichen. Die verschiedenen Arten von Depots lassen sich über einen Vererbungsmechanismus modellieren (siehe **Abbildung 3**).

Zur Verwendung der Testdaten in einem automatisierten Test fehlen noch die eigentlichen Inhalte. Für deren Festlegung bilden die Domänenexperten entsprechende Äquivalenzklassen. Am Adjektiv „deutsche“ ist erkennbar, dass im vorgestellten fiktiven Beispiel das Herkunftsland der Aktie eine Auswirkung auf die Abrechnung der beiden Aktionen (Erwerb, Veräußerung) haben soll; für deutsche Aktien und US-Aktien sollen spezielle Abrechnungsverfahren gelten und für alle anderen ein Standardverfahren.

Die Experten definieren „deutsche Aktien“, „US-Aktien“ und „sonstige Aktien“ als Äquivalenzklassen. Alle übrigen Herkunftsländer werden in der Kategorie „sonstige Aktien“ zusammengefasst, da in diesen Fällen das konkrete Land keinen Einfluss auf die Testergebnisse hätte. Für jede Äquivalenzklasse legt der Experte eine Testdatenvorlage an (siehe **Tabelle 1**).

Übertragen auf die objektorientierten Programmiersprachen sind die inhaltlichen Äquivalenzklassen mit Objekten - also Instanzen von Klassen - vergleichbar.

Aktionen

Die strukturierte Form der Testfallbeschreibung aus Abbildung 2 enthält die zwei Aktionen „Erwerben“ und „Veräußern“. Beide Aktionen sind mit jeweils vier Parametern versehen: Depot, Aktie, Stücke und Kurs.

Damit die fachlichen Testfälle automatisiert ausgeführt werden können, definiert der Domänenexperte die notwendigen Aktionen nach dem folgenden Schema:

- Name der Aktion
- für jeden Parameter: Name und Schema (Aktie, Wertpapierdepot, ...)

Der Umstand, dass die Parameter der Aktionen fachlich „typisiert“ sind und als Typ ausschließlich zuvor definierte Schemata verwendet werden können, ermöglicht eine strikte syntaktische Konsistenzprüfung. Diese Prüfung ist für eine automatisierte Ausführung unerlässlich. Die Software kann nur verarbeiten, was ihr vorab strukturiert bekannt gemacht wurde.

Die Aktionsdefinitionen sind vergleichbar mit Funktionsprototypen in C oder den Methoden eines Java Interfaces. Es handelt sich lediglich um die Signatur einer Funktion, deren Implementierung später behandelt wird. Für die Testdaten wurde jeweils der Bezug zu Konstrukten aus der objektorientierten Programmierung hergestellt. Um diese Analogie für die Aktionen fortzusetzen, würde der erste Parameter aus dem Beispiel ausgelassen und die Aktionen stattdessen „Methoden“ der Aktie. Diese Art der Modellierung hat sich aber für die Domänenexperten als weniger intuitiv herausgestellt, weshalb die eher „prozedurale“ Form beibehalten wurde.

Testfälle

Wenn die Testdatenvorlagen und die Aktionsdefinitionen vorliegen, könnte der Domänenexperte nach dem Baukastenprinzip die konkreten Testfälle zusammenstellen.

Der Ablauf des Testfalls (der ursprünglichen Checkliste) wird Schritt für Schritt aus den zur Verfügung stehenden Aktionen aufgebaut und mit den Testdatenvorlagen parametrisiert. Für das Beispiel sieht das wie folgt aus:

1. Wertpapierdepot anlegen (Privatkundendepot)
2. Erwerben (Privatkundendepot, deutsche Aktie, 20 Stück, 2 Euro/Stück)
3. Veräußern (Privatkundendepot, deutsche Aktie, 10 Stück, 4 Euro/Stück)
4. Aktienbestand prüfen (Privatkundendepot, deutsche Aktie, 10 Stück)

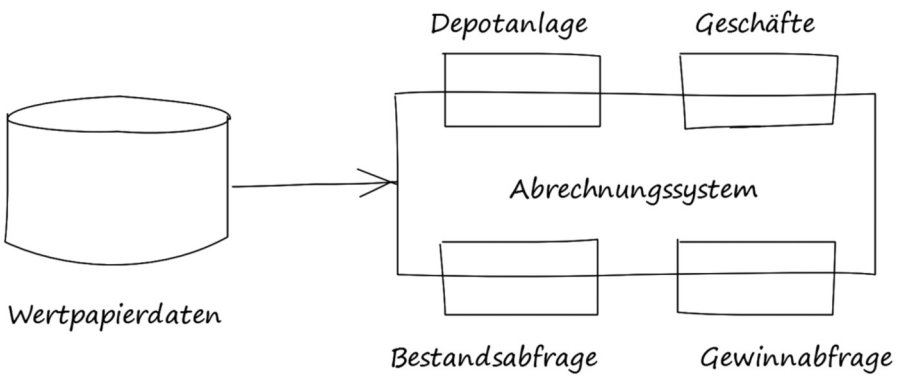

Abb. 4: Aufbau des fiktiven Softwaresystems

Ziel	Ausdruck
security_isin	Aktie.ISIN
security_code	Aktie.WKN
security_shortname	Aktie.Name
security_country_code	lookup(Aktie.Herkunftsland, COUNTRY_TABLE)
delivery_status	1
delivery_date	NOW

Tab. 2: Beispiel für ein Testdaten-Mapping

5. Gewinn prüfen (Privatkundendepot, 20 Euro)

In den Schritten 4 und 5 ist zu sehen, dass die Definition und Prüfung von erwarteten Ergebnissen (Struktur, Inhalte) analog zur Definition der Testdaten erfolgt. Ebenso wird mit der Anlage der Eingangsdaten verfahren. Allerdings hat das Erstellen von Testdatenvorlagen aktuell eine sehr technische Form, die sich bei den Domänenexperten in der praktischen Nutzung nicht bewährt hat. Im Folgenden wird an einem durchgängigen Beispiel dargestellt, wie ein modellbasiertes Testwerkzeug zur Erstellung und Ausführung von Testfällen aussehen kann.

Fachsprache übernehmen

Im ersten Schritt wurden die Checklisten in eine strukturierte Form überführt, die sich für eine automatisierte Ausführung eignet. Bislang handelt es sich aber noch lediglich um eine Beschreibung der Testfälle. Im zweiten Schritt wird dieser Beschreibung Leben eingehaucht. Dieser Vorgang wird anhand des in Abbildung 4 dargestellten fiktiven Softwaresystems beschrieben; einem System zur Verarbeitung von Wertpapiergeschäften.

Das System hat eine datenbankbasierte Lieferschnittstelle für die (Stamm-)Daten der Wertpapiere und Webservice-Schnittstellen zum Anlegen von Wertpapierdepots, zum Tätigen von Geschäften sowie zur Abfrage von Beständen und Gewinnen.

Testdaten

Ein technisches Mapping bringt die Testdatenvorlagen und die Datenstrukturen der Anwendung zusammen.

Tabelle 2 ist ein Beispiel für ein solches Mapping zu entnehmen. Das Mapping leitet die Informationen für einen gültigen Datensatz in der Eingangsdatenbank aus der Struktur der Aktie ab. Fixe Werte - wie für den technischen Lieferstatus - oder komplexere Mapping-Funktionen - wie zur Auflösung des Länderkürzels - sind ebenfalls möglich. In der Regel entsteht zu diesem Zweck eine passende domänenspezifische Sprache (DSL).

Aktionen

Das Mapping der Daten bildet die Vorbereitung für den Schritt, den Testfällen Leben einzuhauchen. Die Ablaufbeschreibung des Testfalls ist aus einer Reihe von Aktionen zusammengestellt, die bislang ausschließlich Signaturen der „Funktionen“ sind. Für die Testausführung werden sie „implementiert“.

Im vorgestellten Beispiel bietet es sich an, nicht jede der Aktionen separat zu implementieren, sondern über zwei beziehungsweise drei generische Implementierungen:

- Schreiben in eine Datenbanktabelle
- Aufruf eines Webservice zum Schreiben von Daten
- Aufruf eines Webservice zum Lesen von Daten

Die fachlichen Aktionsdefinitionen werden mit diesen generischen Implementierungen über eine Konfiguration verknüpft. Jeder dieser Schritte beinhaltet die Ausführung des Mappings.

Modellbasierter Ansatz

Der vorgestellte Ansatz zur Definition fachlicher Akzeptanztests ist zunächst einmal nichts grundlegend Neues. Bestehende Frameworks wie Fitness [Fitn] oder Robot [Robo] basieren auf ähnlichen Prinzipien. Alle diese Frameworks benutzen allerdings einen text- oder Skript-basierten Ansatz für die Erstellung von Testfällen. Die Domänenexperten müssen zum einen die passende „Sprache“ erlernen. Zum ande-

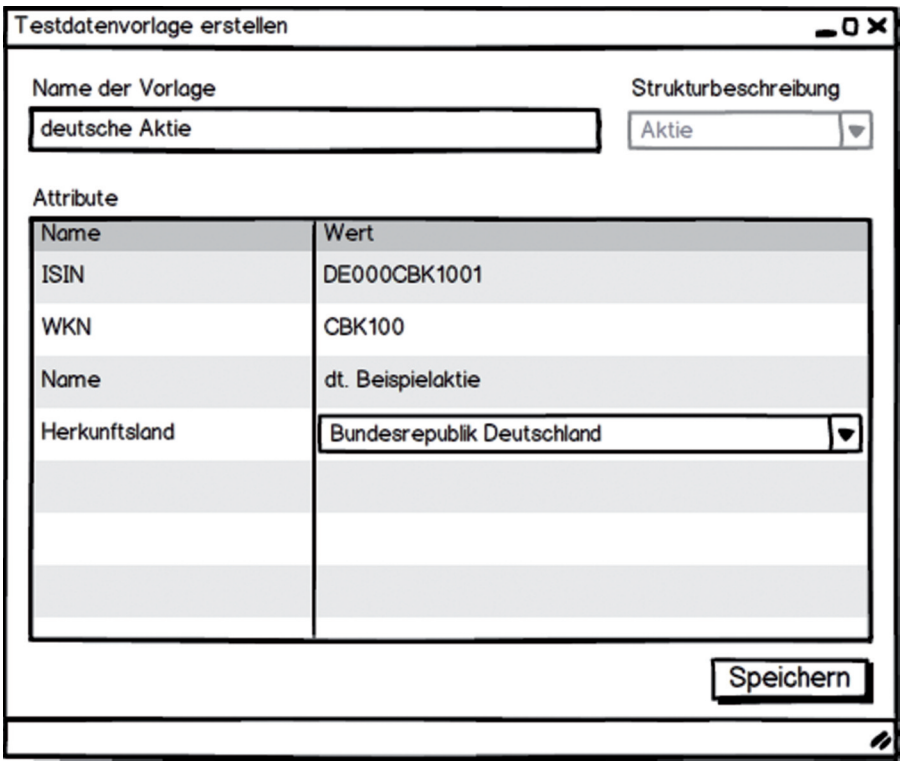

Abb. 5: Pflegemaske für Testdaten

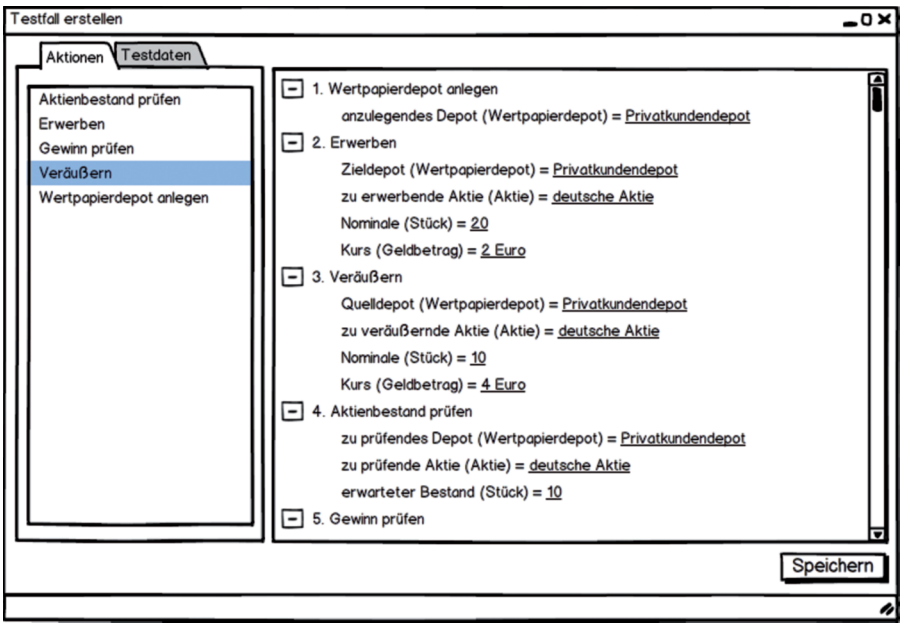

Abb. 6: Pflegemaske für Testfälle

ren führen textbasierte Ansätze immer wieder zu syntaktischen Fehlern in den Testfallbeschreibungen.

Syntaktische Fehler lassen sich weitgehend reduzieren, wenn die Erstellung der Testdaten, Aktionen und auch der Testfälle komplett modellbasiert erfolgt und die Pflege über entsprechende Benutzermasken durchgeführt wird.

In Abbildung 5 ist dieses Prinzip anhand der Pflegemaske für Testdaten dargestellt. Die Zeilen der Pflegetabelle werden aus der Struktur der Aktie abgeleitet. Der Domänenexperte sieht alle relevanten Attribute. Beim Speichern wird überprüft, ob der Benutzer alle Attribute mit Werten belegt hat.

Noch deutlicher wird das Vorgehen an der Maske für die Erstellung von Testfällen (siehe Abbildung 6). Der Benutzer kann seinen Testfall aus einem fachlichen Baukastensystem zusammenstellen. Auf der linken Seite der Maske findet er alle zur Verfügung stehenden Aktionen und Testdaten. Der Testfall wird bequem im Entwurfsbereich in der Mitte per Drag& Drop zusammengestellt und konfiguriert.

Da die Aktionsparameter typisiert sind, kann die Pflegemaske Falscheingaben verhindern. Es ist beispielsweise nicht möglich, eine Aktie an einer Stelle zu verwenden, an der ein Wertpapierdepot sinnvoll wäre.

Fazit

In wenigen Projekten existieren automatisiert ausführbare fachliche Akzeptanztests, wodurch wertvolles Feedback für die Softwareentwickler verloren geht. Die Domänenexperten nutzen die Möglichkeiten der Testautomatisierung nicht, sondern führen ihre Tests in der Regel mit hohem Aufwand manuell aus, da ihnen die gängigen Automatisierungswerkzeuge nicht eingängig genug sind und der Aufwand zum Erlernen einer „neuen Sprache“ zu fehleranfällig und zeitaufwendig ist.

Durch die Verwendung eines modellbasierten Testansatzes erhalten die Experten ein Werkzeug, dem sie ihre eigene Domänensprache „beibringen“ können; sie müssen keine fremde, technische Sprache erlernen, sondern können tool-gestützt Testfälle und -läufe definieren und ausführen. Gleichzeitig sind die Modelle im Gegensatz zu den gängigen fachlichen Testbeschreibungen strukturiert. Diese Eigenschaft erlaubt zum einen eine Automatisierung der Tests. Zum anderen wird der fachliche Benutzer bei der Erstellung von Testfällen unterstützt, sodass syntaktische Fehler direkt vermieden werden können.

Cofinpro hat für einen Kunden aus der Finanzdienstleistungsbranche bereits eine solche Testplattform entwickelt, mit der Tests für verschiedene Wertpapiersysteme durchgeführt werden. Hierbei zeigte sich zum einen der Nutzen der Automatisierung durch schnelle Reproduzierbarkeit der Testergebnisse. Zum anderen stellte sich heraus, dass durch die Erstellung der Testdatenvorlagen auch eine Arbeitsteilung zwischen den verschiedenen Fachtestern möglich ist. So entstanden Experten für Teildomänen wie Kundenstammdaten, deren Testdatenvorlagen dann von ihren Kollegen verwendet werden können - ohne, dass sie über dasselbe Detailwissen verfügen. ■

Literatur & Links

[Beck02] K. Beck, Test-Driven Development. ByExample, Addison-Wesley, 2002

[DanN] Dan North & Associates, Introducion BDD, siehe: http://dannorth.net/introducing-bdd/

[Fitn] http://www.fitnesse.org/

[Fowl06] http://www.martinfowler.com/articles/continuousIntegration.html

[Robo] http://robotframework.org/