Von fachlogischen Testfällenzu physikalischen Testdaten - ein werkzeuggestützter Ansatz zur Überbrückung der semantischen Lücke zwischen Requirements und Test

Beitrag zur TAV 28, 12./13. Februar in Dortmund
Harry M. Sneed, ANECON GmbH, Wien

Abstrakt: Nach dem anforderungsbasierten Test werden Testfälle aus dem Anforderungsdokument abgeleitet. In den Anforderungen wird allerdings ein fachlogischer Wortschatz verwendet. Dort ist von Geschäftsobjekten, Geschäftsprozessen und Geschäftsregeln die Rede. Die Testfälle, die aus dem Anforderungstext abgeleitet werden, sind naturgemäß fachlogische Testfälle ohne Bezug zu den physikalischen Testobjekten – die GUIs, Schnittstellen, Dateien und Datenbanktabellen. In diesem Beitrag wird ein toolgestützter Ansatz zur Überbrückung jener semantischen Lücke zwischen Anforderungen und Test basierend auf einer Zuordnungstabelle vorgestellt.

Schlüsselwörter: Anforderungsbasierter Test, Objektontologie, fachlogische Testfälle, Testdatengenerierung, Testergebnisvalidierung, Systemtest

1 Hintergrund

Für den Systemtest wird erwartet, dass das System gegen die Anforderungen getestet wird. Software Validation ist der Versuch nachzuweisen, dass ein System seiner Anforderung entspricht, d.h. das richtige System wurde gebaut [IEEE90]. Dieser Nachweis wird heute gebracht, in dem die Systemtester Testfälle aus den Anforderungen ableiten und das betreffende System damit ausführen. Für jede Anforderung soll es mehrere Testfälle geben, die belegen, dass die Anforderung erfüllt ist. Dabei werden sowohl funktionale als auch nicht-funktionale Anforderungen berücksichtigt.

In dem V-Model-XT wird betont, dass nur das getestet werden darf, was in dem Lastenheft als Anforderung niedergeschrieben ist [Höhn/Höppner 08]. Zu diesem Zweck werden die Anforderungen in Prüffällen umgesetzt und die Prüffälle in einer Prüfspezifikation festgehalten. Beides, Lastenheft und Prüfspezifikation, sind Beilagen zum Vertrag mit dem Auftragnehmer. Damit stehen noch vor dem Beginn der Entwicklung sämtliche Test- bzw. Prüffälle fest. Während der Auftragnehmer das Zielsystem entwickelt, obliegt es dem Auftraggeber den Test des Systems vorzubereiten, d.h. er muss u. a. in dieser Zeit die fachlogischen Testfälle in konkrete Testdaten auf physikalischen Datenträgern umsetzen. Da er aber nicht wissen kann, welche Daten auf welchen Datenträgern implementiert werden, ist diese Aufgabe kaum zu bewältigen. Er ist gezwungen zu warten, bis mindestens das Datenmodell endgültig feststeht. Allerdings könnte er schon damit anfangen, sobald Teile des Datenmodells fertig sind.

2 Fachlogische Testfälle

Testfälle, die aus den Anforderungen gewonnen werden, beziehen sich auf die Begriffe der Anforderungen. Diese beschrieben fachliche Vorgänge, Objekte, Prozesse, Regel und Anwendungsfälle. Die fachlichen Objekte sind die Hauptwörter in dem Text. Zeitwörter deuten auf Aktionen, Eigenschaftswörter auf Zustände. Bedingungswörter implizieren Regel [Sneed06]. In dem Satz „Falls der von dem Kunden bestellte Artikel vorhanden ist und die Artikelmenge ausreichend ist, wird die Artikelmenge um die Bestellmenge reduziert und ein Lieferposten sowie eine Rechnung erstellt…“ gibt es die Aktionen „reduziere Artikelmenge“ und „erstelle Lieferposten und Rechnung“, die Zustände “Artikel nicht vorhanden“, “Artikel vorhanden“ und “Artikelmenge ausreichend“, und die Regel “falls der vom Kunden bestellte Artikel vorhanden ist und die Artikelmenge ausreichend ist“.

Die Objekte in dieser Anforderung sind:

- Kunden
- Artikel
- Artikelmenge
- Bestellmenge
- Lieferposten
- Rechnung

Implizit in diesem Satz sind drei fachlogische Testfälle entsprechend den drei Vorbedingungen:

1. der bestellte Artikel ist nicht vorhanden = TF_1
2. der bestellte Artikel ist vorhanden, aber die Artikelmenge ist nicht ausreichend = TF_2
3. der bestellte Artikel ist vorhanden und die Artikelmenge ist ausreichend = TF_3

Daraus folgen zwei Nachbedingungen

1. es hat sich an dem Artikel nichts geändert
2. die Artikelmenge ist um die Bestellmenge reduziert worden und es gibt einen Lieferposten und eine Rechnung.

Es ist für einen ausgebildeten Tester nicht schwer, die drei Testfälle samt Vor- und Nachbedingungen, Auslöser und Objekte zu erkennen und aufzulisten. Schwierig wird es erst, wenn der Tester versuchen muss, für diese drei Testfälle konkrete Testdaten herbeizuschaffen. Denn spätestens zu diesem Zeitpunkt muss er sich mit der Datenarchitektur auseinandersetzen.

3 Physikalische Testdaten

Um die richtigen Testdaten zu beschaffen, muss der Tester wissen, wo und wie die Daten gespeichert sind, bzw. über welchen Kanal sie in das System gelangen. Demzufolge muss er sich mit der Architektur des Systems auseinandersetzen, zumindest die Datenarchitektur. Die persistenten Daten sind auf Datenbanktabellen und Dateien verteilt. Die vorübergehenden Daten befinden sich in den Benutzeroberflächen – die GUIs, HTMLs, XSLs und sonstigen Maskendokumenten, sowie in den Systemschnittstellen – IDLs, XMLs, WSDLs, usw. Dort verstecken sie sich hinter ganz anderen Namen als die in den Anforderungstexten.

Hier können sie heißen „Feld_5_10_3“, was heißen könnte, das Feld in der 5. Zeile an der 10. Spalte mit der Länge 3.

In Anlehnung an unser Beispiel von oben, nehmen wir an, es gäbe eine Datenbanktabelle namens KUNDE mit dem Hauptschlüssel KUNDE_ID. Daneben gibt es eine zweite Datenbanktabelle namens ARTIKEL mit dem Hauptschlüssel ARTIKEL_ID und dem Attribut MENGE. Das Attribut Bestellmenge ist ein Feld in der Auftragsmaskendefinition namens FELD_5_20. Der Lieferposten ist eine XML Datei, ebenso die Rechnung. Über diese Namen kann der Tester die Begriffe im Anforderungstext mit den physikalischen Testobjekten in einer Zuordnungstabelle verknüpfen. Diese Tabelle muss für alle verwendeten Datennamen beretgestellt werden.

Fachobjekt	Testobjekt	Objekttyp
Kunden	KUNDE (KUNDE_ID)	sql
Artikel	ARTIKEL (ARTIKEL_ID)	sql
Artikelmenge	ARTIKEL.MENGE	sql
Bestellmenge	AUFTRAGSMASKE.FELD_5_20	html
Lieferposten	LIEFERPOSTEN (AUFTRAG_ID)	xml
Rechnung	RECHNUNG (KUNDEN_ID)	xml

Um den Vorgang zu ergänzen, muss der Tester einige implizierte Daten in die Tabelle hinzufügen, nämlich die Identifikationsfelder und Rückmeldungen der Auftragsmaske.

Auftragsnummer	AUFTRAGSMASKE.FELD_3_11	html
Kundennummer	AUFTRAGSMASKE.FELD_4_11	html
Artikelnummer	AUFTRAGSMASKE.FELD_5_11	html
Fehlermeldung	AUFTRAGSMASKE.FELD_20_11	html

4 Verfassung der Testskripts

Jetzt ist der Tester in der Lage, ein Testskript zu verfassen. Für den ersten Testfall TF_1 muss er dafür sorgen, dass die in der Auftragsmaske angegebene Artikelnummer nicht in der Artikeltabelle ist.

```
if (testcase = „TF_1“)
  assert pre AUFTRAGSMASKE.FELD_5_11 = “4711”;
   assert pre ARTIKEL.ARTIKEL_ID =
                    AUFTRAGSMASKE.FELD_5_11;
   assert post AUFTRAGSMASKE.FELD_20_11 =
                    “Artikel nicht vorhanden“;
endcase;
```

Für den zweiten Testfall TF_2 muss er dafür sorgen, dass der bestellte Artikel zwar vorhanden, aber dass seine Menge kleiner als die bestellte Menge ist.

```
if (testcase = „TF_2“)
  assert pre AUFTRAGSMASKE.FELD_5_11 = “4711”;
  assert pre AUFTRAGMASKE.FELD_5_20 = “10”;
   assert pre ARTIKEL.ARTIKEL_ID =
                    AUFTRAGSMASKE.FELD_5_11;
assert pre ARTIKEL.MENGE <
                    AUFTRAGSMASKE_5_20;
assert post AUFTRAGSMASKE.FELD_20_11 =
                    “Menge nicht ausreichend“;
endcase;
```

Für den dritten Testfall TF_3 muss er dafür sorgen, dass der bestellte Artikel vorhanden und seine Menge größer als die bestellte Menge ist. Zusätzlich muss er dafür sorgen, dass die Artikelmenge korrekt reduziert wurde und dass die Lieferposten und Rechnung erstellt sind.

```
if (testcase = „TF_3“)
  assert pre AUFTRAGSMASKE.FELD_5_11 = “4711”;
  assert pre AUFTRAGSMASKE.FELD_5_20 = “10”;
  assert pre AUFTRAGSMASKE.FELD_3_11 = “1111“;
  assert pre AUFTRAGSMASKE.FELD_4_11 = “2222“;
  assert pre ARTIKEL.ARTIKEL_ID =
                    AUFTRAGSMASKE.FELD_5_11;
  assert pre ARTIKEL.MENGE >
                    AUFTRAGSMASKE_5_20;
  assert post ARTIKEL.MENGE =old.ARTIKEL.MENGE
                - AUFTRAGSMASKE.FELD_5_20;
  assert post LIEFERPOSTEN.AUFTRAG_ID =
                AUFTRAGSMASKE.FELD_3_11;
          assert post RECHNUNG.KUNDEN_ID =
                AUFTRAGSMASKE.FELD_4_11;
endcase;
```

Aus dem Testskript werden pro Testfall die Datenbanktabellen KUNDE und ARTIKEL generiert und die Benutzeroberfläche AUFTRAGSMASKE simuliert. Nach dem dritten Testfall wird die Menge in der ARTIKEL Tabelle und die beiden XML Dateien LIEFERPOSTEN und RECHNUNG geprüft. Dieses Verfahren setzt voraus, dass es für jeden Testfall eigene Kopien der physikalischen Datenobjekte gibt, damit sie nicht durcheinander kommen. Damit wird ein vollständiger und konsistenter Test des Systemverhaltens gegen die Anforderungen gewährleistet.

5 Zusammenfassung

Ein systematischer Test gegen die Anforderung ist weitgehend automatisierbar. Voraussetzungen dafür sind allerdings

1) Ableitung der Testfälle aus dem Anforderungstext
2) Verknüpfung der fachlogischen Testfälle mit den physikalischen Testdaten
3) strikte Trennung der Testfallausführung.

Diese Voraussetzungen werden erfüllt in dem man

a) einen Anforderungsprozessor benutzt
b) Querverweise zwischen fachlichen Objekten und real existierenden Daten pflegt
c) die Testumgebungen der einzelnen Testfälle trennt.

Das Systemtestverfahren muss natürlich durch spezielle Werkzeuge unterstützt werden. Hierfür werden vier Werkzeuge eingesetzt: *Testanalysator*, um deutsche und englischsprachige Anforderungstexte zu analysieren, *Testbridge*, um die fachlogische Testfälle und Fachobjekte mit den physikalischen Testdaten zu verbinden. *DataTest*, um die Testskripten zu interpretieren und die Testdaten sowohl zu generieren als auch zu validieren. *TestDocumenter*, um die Auswirkung eines jeden Testfalls, seine Pfade, seinen Überdeckungsbereich und seine aufgedeckten Fehler zu dokumentieren.

Durch die Integration dieser Werkzeuge wird ein durchgängiger anforderungsbasierter Test, von der Anforderungsanalyse bis zur Testbewertung, angestrebt.