TAV

Von fachlogischen Testfällen zu physikalischen Testdaten

- ein werkzeuggestützter Ansatz zur Überbrückung der semantischen Lücke zwischen Requirements und Test

Harry M. Sneed
ANECON GmbH, Wien

Für die GI-TAV Workshop
Dortmund, Germany
im Februar, 2009

TAV-01

Der Systemtest ist aus folgenden Gründen nicht automatisiert:

- Der Systemtest sollte immer ein Test eines implementierten Systems gegen spezifizierte Anforderungen sein
- Entitäten der Anforderungen entsprechen nicht den Entitäten der Implementierung
- Anforderungsentitäten sind Geschäftsobjekte, fachliche Attribute, Geschäftsprozesse und Anwendungsfälle
- Entitäten der Implementierung sind HTML Seiten, XML Schnittstellen, SQL Tabellen und Transaktionen

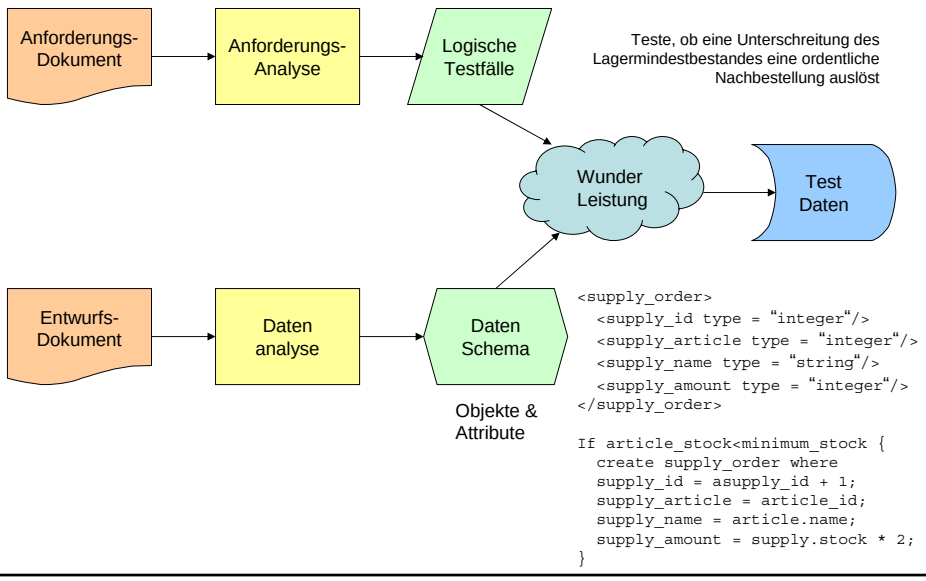
TAV-02

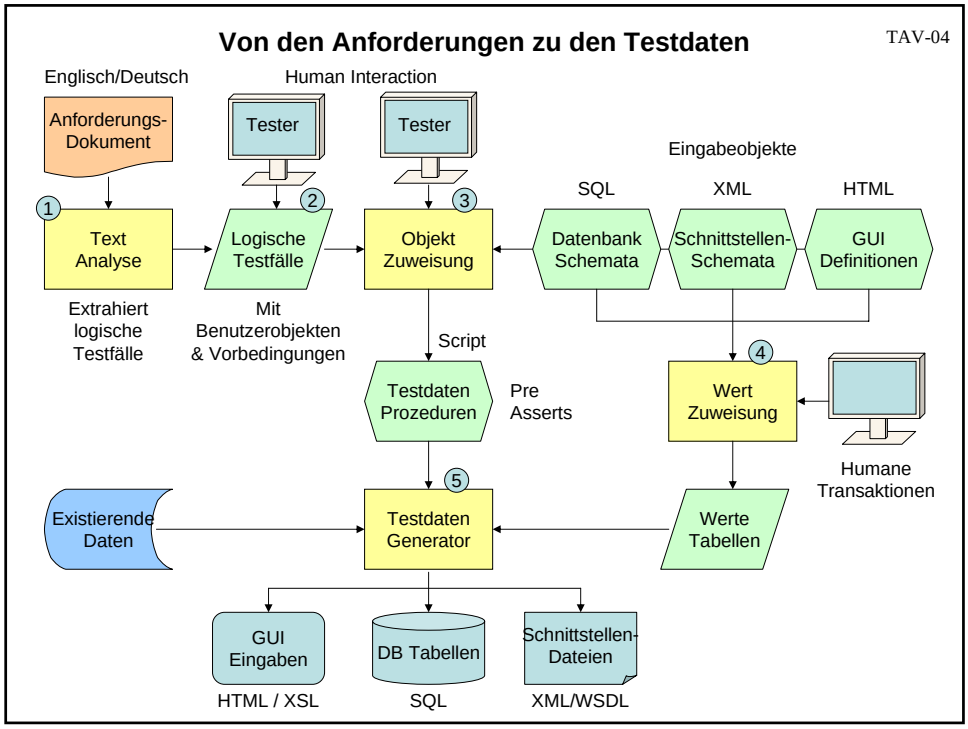
Aktueller Stand des System Tests:

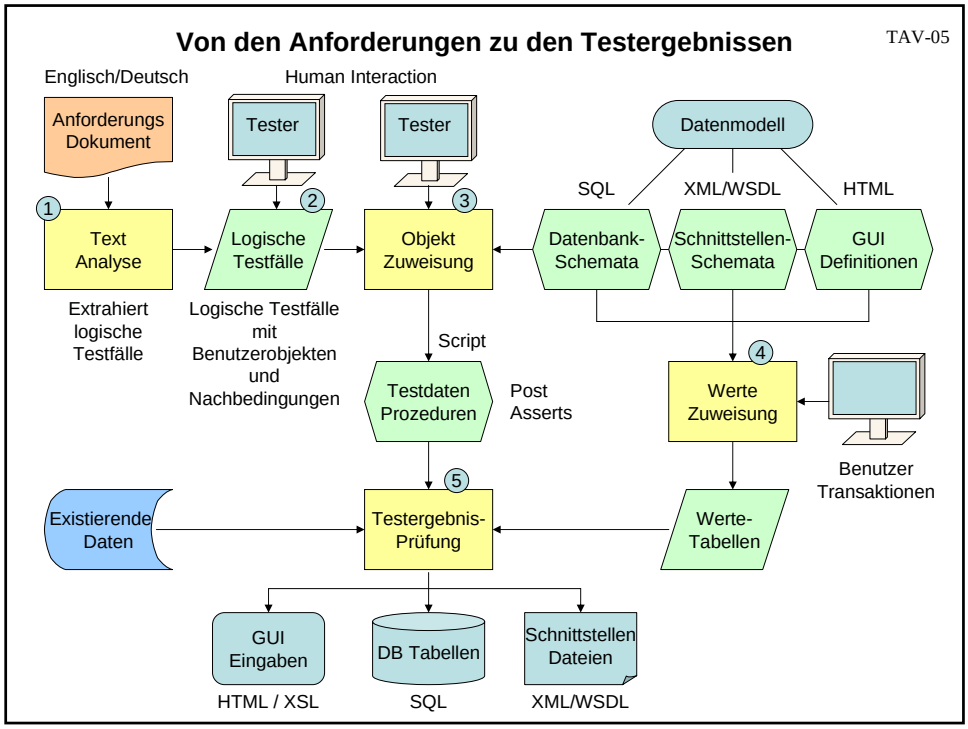
- Menschliche Tester leiten logische Testfälle manuell aus den Anforderungen ab
- Menschliche Tester erzeugen manuell Testdaten für ihre Testfälle
- Menschliche Tester schließen die Lücke zwischen den aus den Anforderungen gewonnenen logischen Testfällen und den aus den Implementierungsentitäten generierten Testdaten
- Darin besteht die Leistung des Systemtests

TAV-03

Die Wunderleistung des Systemtests




Von den Anforderungen zu den Testdaten
TAV-04
Englisch/Deutsch
Human Interaction
Anforderungs-Dokument
Tester
Tester
Eingabeobjekte
SQL
XML
HTML
1
2
3
Text Analyse
Logische Testfälle
Objekt Zuweisung
Datenbank Schemata
Schnittstellen-Schemata
GUI Definitionen
Extrahiert logische Testfälle
Mit Benutzerobjekten & Vorbedingungen
Script
4
Testdaten Prozeduren
Pre Asserts
Wert Zuweisung
Humane Transaktionen
5
Existierende Daten
Testdaten Generator
Werte Tabellen
GUI Eingaben
DB Tabellen
Schnittstellen-Dateien
HTML / XSL
SQL
XML/WSDL


Von den Anforderungen zu den Testergebnissen
TAV-05
Englisch/Deutsch
Human Interaction
Anforderungs Dokument
Tester
Tester
Datenmodell
SQL
XML/WSDL
HTML
1
2
3
Text Analyse
Logische Testfälle
Objekt Zuweisung
Datenbank-Schemata
Schnittstellen-Schemata
GUI Definitionen
Extrahiert logische Testfälle
Logische Testfälle mit Benutzerobjekten und Nachbedingungen
Script
4
Testdaten Prozeduren
Post Asserts
Werte Zuweisung
Benutzer Transaktionen
5
Existierende Daten
Testergebnis-Prüfung
Werte-Tabellen
GUI Eingaben
DB Tabellen
Schnittstellen Dateien
HTML / XSL
SQL
XML/WSDL

Anforderungsbasierte logische Testfälle

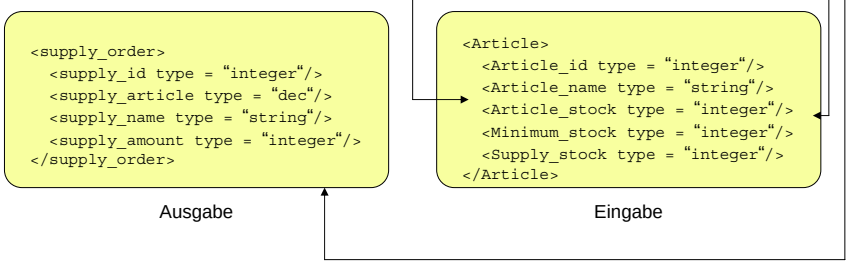
②

Testfall	Testzweck	Testobjekte	Vor-/Nach- Bedingungen
Bestellung01	Bestellung soll zurückgewiesen werden wenn Kunde unbekannt	Bestellung Kunde	zurückgewiesen (post) unbekannt(pre)
Bestellung02	Bestellung soll zurückgewiesen werden wenn Kundenbonität nicht ausreichend	Bestellung Kunde Bonität	zurückgewiesen (post) bekannt (pre) zu gering (pre)
Bestellung03	Bestellung soll zurückgewiesen werden wenn Artikel nicht verfügbar	Bestellung Artikel	zurückgewiesen (post) Nicht verfügbar (pre)
Bestellung04	Bestellung soll zurückgestellt werden, wenn verfügbare Menge kleiner als Bestellmenge	Bestellung Verf. Menge Bestellmenge	zurückgestellt (post) < Bestellmene (pre) => verf. Mange (pre)
Bestellung05	Bestellung soll ausgeführt werden, wenn Menge ausreichend	Bestellung Verf. Menge	ausgeführt (post) > Bestellmenge (pre)
Bestellung06	Menge auf Lager soll um Bestellmenge reduziert werden	Bestellmenge Verf. Menge	- Bestellmenge (post) < verf. Menge (pre)
Bestellung07	Wenn verfügbare Menge die definierte Mindestmenge unterschreitet, Nachbestellung	Verf. Menge Mindestmenge Bestellung	< Mindestmenge (pre) > verf. Menge (pre) existiert (post)

Zuweisung von logischen Objekten an physische Objekte

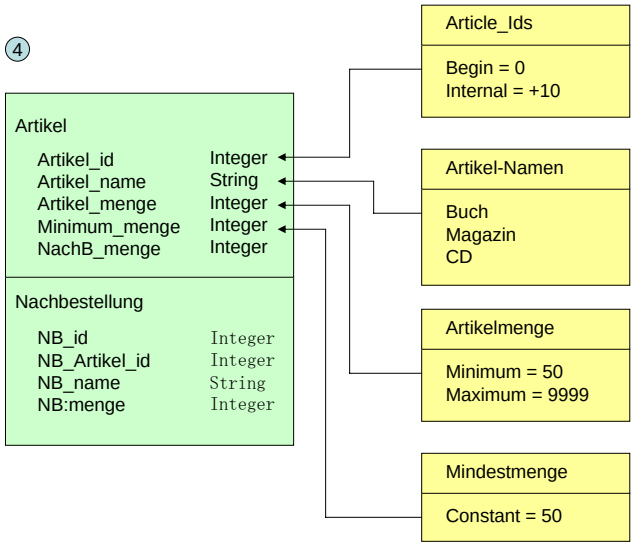
③

Testfall	Testzweck	Testobjekte	Typ
Bestellung09	Wenn die verf. Menge die definierte Mindestmenge unterschreitet, soll nachbestellt werden	Verf. Menge Mindestmenge Bestellung	Eingabe Eingabe Ausgabe



TAV-08

Zuweisung von Werten an physikalische Datenfelder



TAV-09

Analyse der Anforderungstexte

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.

TAV-1

Zuordnung der Fachobjekte zu den technischen Objekten

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

TAV-11

Zuweisung der Testdatenwerte

```
  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;
```

TAV-12

Aufbau der DataTest Testumgebung

```
ASSERTS
    TF_1.asrt          →    AUFTRAGSMASKE
                            ARTIKEL
    TF_2.asrt          →    AUFTRAGSMASKE
                            ARTIKEL
    TF_3.asrt          →    AUFTRAGSMASKE
                            ARTIKEL
                            LIEFERPOSTEN
                            RECHNUNG

OLDFILES                        NEWFILES
    TF_1                         TF_1
        ARTIKEL.sql                 ARTIKEL.sql
        AUFTRAGSMASKE.html          AUFTRAGSMASKE.html

    TF_2                        TF_2
        ARTIKEL.sql                 ARTIKEL.sql
        AUFTRAGSMASKE.html          AUFTRAGSMASKE.html

    TF_3                        TF_3
        ARTIKEL.sql                 ARTIKEL.sql
        AUFTRAGSMASKE.html          AUFTRAGSMASKE.html
        LIEFERPOSTEN.xml            LIEFERPOSTEN.xml
        RECHUNG.xml                 RECHNUNG.xml
```

TAV-13

Testskript zur Steuerung des Testprozesses

```
Test: Auftragsbearbeitung;

    if ( $Time   = " 200719021800 " );
            do for each TestCase (AUFTRAG001:AUFTRAG0012);

                    generate ARTIKEL;           // generiere Artikeldatenbank (SQL)
                    generate KUNDEN;            // generiere Kundendatenbank (SQL)
                    generate PREISE;            // generiere Preisdatenbank (SQL)
                    generate LIEFERAN;          // generiere Lieferantendatenbank (SQL)

                    generate AUFTRAG;           // generiere eine HTML Seite
                                                // mit der Auftragserteiling (HTML)

                    invoke process Auftragsbearbeitung;     // Starte Job um Auftrag zu bearbeiten

                    validate RECHPOST;          // validiere die Rechnungsposten (XML)
                    validate VERSAND;           // validiere die Versandposten (XML)
                    validate LIEFPOST;          // validiere die Lieferposten (XML)

                    validate ARTIKEL;           // validiere Artikeldatenbank (SQL)
                    validate LIEFERAN;          // validiere Liferantendatenbank (SQL)

            enddo;
    endif;
endTest;
```

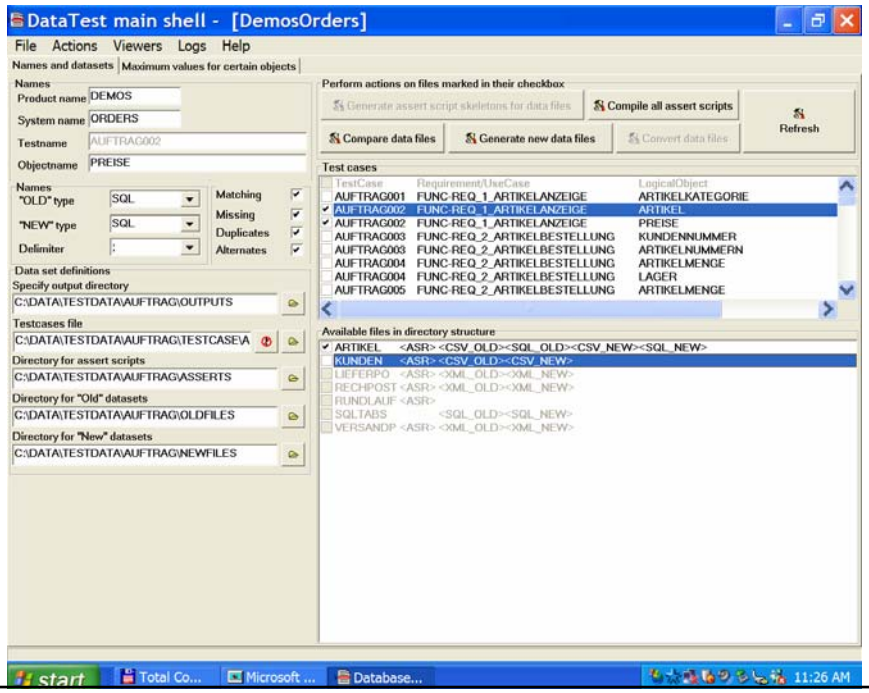
Generierung der Testfalltabelle aus den Anforderungstexte

TAV-14

```
TestCase;Requirement/UseCase;LogicalObject;TestObject;ObjectType
AUFTRAG001;FUNC-REQ_1_ARTIKELANZEIGE;ARTIKELKATEGORIE;;
AUFTRAG002;FUNC-REQ_1_ARTIKELANZEIGE;ARTIKEL;ARTIKEL;CSV
AUFTRAG002;FUNC-REQ_1_ARTIKELANZEIGE;PREISE;KUNDEN;CSV
AUFTRAG003;FUNC-REQ_2_ARTIKELBESTELLUNG;KUNDENNUMMER;;
AUFTRAG003;FUNC-REQ_2_ARTIKELBESTELLUNG;ARTIKELNUMMERN;;
AUFTRAG004;FUNC-REQ_2_ARTIKELBESTELLUNG;ARTIKELMENGE;;
AUFTRAG004;FUNC-REQ_2_ARTIKELBESTELLUNG;LAGER;;
AUFTRAG005;FUNC-REQ_2_ARTIKELBESTELLUNG;ARTIKELMENGE;;
AUFTRAG006;FUNC-REQ_2_ARTIKELBESTELLUNG;ARTIKEL;;
AUFTRAG007;FUNC-REQ_3_VERSANDAUFTRAG;ARTIKEL;;
AUFTRAG007;FUNC-REQ_3_VERSANDAUFTRAG;VERSANDAUFTRAG;;
AUFTRAG007;FUNC-REQ_3_VERSANDAUFTRAG;LAGERVERWALTER;;
AUFTRAG008;FUNC-REQ_4_ABRECHNUNG;RECHNUNGEN;;
AUFTRAG009;FUNC-REQ_4_ABRECHNUNG;BUCHHALTER;;
AUFTRAG010;FUNC-REQ_4_ABRECHNUNG;RECHNUNG;;
AUFTRAG010;FUNC-REQ_5_NACHLIEFERUNG;LIEFERAUFTRAG;;
AUFTRAG011;FUNC-REQ_5_NACHLIEFERUNG;ARTIKELMENGE;;
AUFTRAG012;FUNC-REQ_5_NACHLIEFERUNG;LAGER;;
AUFTRAG013;FUNC-REQ_5_NACHLIEFERUNG;ARTIKELMENGE;;
AUFTRAG013;FUNC-REQ_5_NACHLIEFERUNG;ARTIKEL;;
AUFTRAG014;FUNC-REQ_6_RUECKSTELLPOSTEN;AUFTRAGSVERWALTER;;
AUFTRAG015;NON-FUNC-REQ_11_REPONSEZEIT;KUNDENABFRAGE;;
AUFTRAG016;NON-FUNC-REQ_11_REPONSEZEIT;KUNDENAUFTRAEGE;;
AUFTRAG016;NON-FUNC-REQ_11_REPONSEZEIT;SEKUNDEN;;
AUFTRAG017;NON-FUNC-REQ_13_VERFUEGBARKEIT;STUNDEN;;
AUFTRAG018;NON-FUNC-REQ_14_SICHERHEIT;KUNDEN;;
AUFTRAG019;NON-FUNC-REQ_16_DATENSICHERUNG;KUNDEN;;
AUFTRAG019;NON-FUNC-REQ_16_DATENSICHERUNG;ARTIKELDATEN;;
```

Selektion der Testfälle

TAV-15



Schema einer SQL Datenbank als Eingabe zur Testdatengeneration

```
/*-- Auftragsdatenbank */
/*-- Database: ARTIKEL */

/*---------------------------------------------------------------*/
CREATE TABLE ARTIKEL (
        ARTIKELKATEGORIE NUMBER (2,0) NOT NULL WITH DEFAULT,
        ARTIKELNUMMER             NUMBER(6,0) NOT NULL WITH DEFAULT,
        ARTIKELNAME               CHAR(40) NOT NULL WITH DEFAULT,
        ARTIKELMENGE              NUMBER(6,0) NOT NULL WITH DEFAULT,
        ARTIKELPREIS              NUMBER(8,2) NOT NULL WITH DEFAULT,
        MINDESTMENGE              NUMBER(6,0) NOT NULL WITH DEFAULT,
        LIEFERMENGE               NUMBER(6,0) NOT NULL WITH DEFAULT,
        PRIMARY KEY               (ARTIKELNUMMER)
        ON DELETE                 RESTRICT
        )
GO
```


Assertionskript für die SQL Datenbank

```
// Generated Assertion Script
// Base File is ARTIKEL
// Date of Assertion Generation is  2008-08-30

file: ARTIKEL;
   if (testcase = “AUFTRAG001“);
     if (new.ARTIKELNUMMER = old.ARTIKELNUMMER);
        assert new.ARTIKELKATEGORIE = "01" ! "02" ! "03" ;
        assert new.ARTIKELNUMMER = "111111" ! "222222" ! "333333";
        assert new.ARTIKELNAME = old.ARTIKELNAME | "-" | old.ARTIKELKATEGORIE;
        assert new.ARTIKELMENGE = "15" ;
        assert new.ARTIKELPREIS = old.ARTIKELPREIS - 1;
        assert new.MINDESTMENGE = old.MINDESTMENGE + 1;
        assert new.LIEFERMENGE = {11:51};
   endCase;
end;
```

TAV-18

Generierte SQL Daten

```
ARTIKELKATEGORIE;ARTIKELNUMMER;ARTIKELNAME;ARTIKELMENGE;
ARTIKELPREIS;MINDESTMENGE;LIEFERMENGE;
01;111111;MIST-01;15;9.50;26;10;
02;222222;MIST-01;15;9.50;26;11;
03;333333;MIST-01;15;9.50;26;31;
01;111111;HEFT-02;15;14.50;51;10;
02;222222;HEFT-02;15;14.50;51;11;
03;333333;HEFT-02;15;14.50;51;31;
01;111111;STIFT-03;15;10.50;76;10;
02;222222;STIFT-03;15;10.50;76;11;
03;333333;STIFT-03;15;10.50;76;31;
```

TAV-19

Assertionskript für die XML Datei

```
// Generated Assertion Script
// Base File is RECHPOST
// Date of Assertion Generation is  2008-06-12

file: RECHPOST;
 if (testcase = "AUFTRAG001");
  if (object = "RECHNUNGSKOPF" &
       new.AUFTRAGNR = old.AUFTRAGNR &
       new.KUNDENNR   = old.KUNDENNR );
     assert new.KUNDENBONITAET = old.KUNDENBONITAET;
     assert new.KUNDENNAME = old.KUNDENNAME;
     assert new.KUNDEN-LAND = old.KUNDEN-LAND;
     assert new.KUNDEN-PLZ = old.KUNDEN-PLZ;
     assert new.KUNDEN-ORT = old.KUNDEN-ORT;
     assert new.KUNDEN-STRASSE = old.KUNDEN-STRASSE;
  endObject;
  if (object = "RECHNUNGSPOSTEN" &
      new.AUFTRAGNR-RP = old.AUFTRAGNR-RP &
      new.KUNDENNR-RP  = old.KUNDENNR-RP &
      new.BESTELLNR    = old.BESTELLNR );
//   assert count = "3";
     assert new.ARTIKELNR = old.ARTIKELNR ;
     assert new.ARTIKELNAME = old.ARTIKELNAME ;
     assert new.ARTIKELPREIS = old.ARTIKELPREIS + 10;
     assert new.BESTELLMENGE = old.BESTELLMENGE + 1;
     assert new.POSTENPREIS = old.POSTENPREIS - 2;
  endObject;
endCase;
if (testcase = "AUFTRAG002");
  if (object = "RECHNUNGSKOPF" &
       new.AUFTRAGSNR = old.AUFTRAGSNR &
       new.KUNDENNR   = old.KUNDENNR );
     assert new.KUNDENBONITAET = old.KUNDENBONITAET;
```

TAV-20

Generierte XML Datei

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<!--DOCTYPE rechpost SYSTEM "Rechpost.xsd"-->
<rechpost>
 <RECHNUNG>
  <RECHNUNGSKOPF>
   <AUFTRAGNR>000111</AUFTRAGNR>
   <KUNDENNR>000222</KUNDENNR>
   <KUNDENBONITAET>02</KUNDENBONITAET>
   <KUNDENDATEN>
    <KUNDENNAME>Franz Lechner</KUNDENNAME>
    <KUNDENANSCHRIFT>
     <KUNDEN-LAND>Bayern</KUNDEN-LAND>
     <KUNDEN-PLZ>82054</KUNDEN-PLZ>
     <KUNDEN-ORT>Arget/Sauerlach</KUNDEN-ORT>
     <KUNDEN-STRASSE>Prellerweg 7</KUNDEN-STRASSE>
    </KUNDENANSCHRIFT>
   </KUNDENDATEN>
  </RECHNUNGSKOPF>
  <RECHNUNGSPOSTEN>
   <AUFTRAGNR-RP>000111</AUFTRAGNR-RP>
   <KUNDENNR-RP>000222</KUNDENNR-RP>
   <BESTELLNR>02</BESTELLNR>
   <ARTIKELNR>444444</ARTIKELNR>
   <ARTIKELNAME>HEFT</ARTIKELNAME>
   <ARTIKELPREIS>020000</ARTIKELPREIS>
   <BESTELLMENGE>0012</BESTELLMENGE>
   <POSTENPREIS>000000</POSTENPREIS>
  </RECHNUNGSPOSTEN>
</RECHNUNG>
```

TAV-21

Log zur Validierung einer SQL Datenbank

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--DOCTYPE TMETRICS SYSTEM "tmetrics.xsd"-->
 <TMETRICS>
    <METRIC_FILE
    Product = "LAGER"
    System = "ORDERS"
    Date = "19.12.08"
    Source = "DATATEST"
    />
    <METRIC_ENTITY
     Entity = "Data"
     EName = "ARTIKEL"
     EType = "CSV" >
     <Data_Comparison_Metrics>
        <Old_Records_checked>3</Old_Records_checked>
        <Old_Records_found>1</Old_Records_found>
        <Old_Records_not_found>0</Old_Records_not_found>
        <Old_Record_Duplicates>2</Old_Record_Duplicates>
        <New_Records_checked>9</New_Records_checked>
        <New_Records_found>3</New_Records_found>
        <New_Records_not_found>0</New_Records_not_found>
        <New_Record_Alternates>6</New_Record_Alternates>
        <Data_Fields_checked>21</Data_Fields_checked>
        <Data_Fields_deviated>6</Data_Fields_deviated>
        <Data_Correction_Rate>0.71</Data_Correction_Rate>
        <Data_Coverage_Rate>0.67</Data_Coverage_Rate>
     </Data_Comparison_Metrics>
    </METRIC_ENTITY>
 </TMETRICS>
```

TAV-22

Abgleichsbericht zur Validierung einer SQL Datenbank

```
+-------------------------------------------------------------------------------------+
|                            File/Table Comparison Report                             |
| File:   FIL0001.rep                                          Params: Y Y Y Y        |
| Object: ARTIKEL                                                Date: 19.12.08       |
| Type  : CSV                  TestCase: AUFTRAG002             System: ORDERS         |
| Key Fields of Record(new,old)                                                       |
+-------------------------------------------------------------------------------------+
| New:ARTIKELNUMMER                                                                   |
| Old:ARTIKELNUMMER                                                                   |
+-------------------------------------------+-----------------------------------------+
| Non-Matching Fields                       | Non-Matching Values                     |
+-------------------------------------------+-----------------------------------------+
| RecKey:111111                             | missing from the old File/Table         |
+-------------------------------------------+-----------------------------------------+
| RecKey:222222                             | missing from the old File/Table         |
+-------------------------------------------+-----------------------------------------+
| RecKey:333333                             |                                         |
| New: ARTIKELNAME                          | HEFT-02                                 |
| Old: ConcatenatedString                   | MIST-01                                 |
+-------------------------------------------+-----------------------------------------+
| RecKey:333333                             |                                         |
| New: ARTIKELPREIS                         | 14.50                                   |
| Old: ARTIKELPREIS                         | 9.50                                    |
+-------------------------------------------+-----------------------------------------+
| RecKey:333333                             |                                         |
| New: MINDESTMENGE                         | 51.00                                   |
| Old: MINDESTMENGE                         | 26.00                                   |
+-------------------------------------------+-----------------------------------------+
| RecKey:333333                             |                                         |
| New: ARTIKELNAME                          | STIFT-03                                |
| Old: ConcatenatedString                   | MIST-01                                 |
+-------------------------------------------+-----------------------------------------+
```

TAV-23

Abgleichsbericht zur Validierung einer XML Datei

```
+-------------------------------------------------------------------------------------+
|                            File/Table Comparison Report                             |
| Object: RECHNUNGSKOPF                                          Date: 19.12.08       |
| Type  : XML                  TestCase:                        System: ORDERS         |
| Key Fields of Record(new,old)                                                       |
+-------------------------------------------------------------------------------------+
| New:AUFTRAGNR                                                                       |
| Old:AUFTRAGNR                                                                       |
+-------------------------------------------+-----------------------------------------+
| Non-Matching Fields                       | Non-Matching Values                     |
+-------------------------------------------+-----------------------------------------+
| RecKey:000222                             |                                         |
| New: KUNDENNAME                           | HARRY.SNEED                             |
| Old: KUNDENNAME                           | HARRY_M._SNEED                          |
+-------------------------------------------+-----------------------------------------+
|  Total Number of old Records checked:                02                             |
|  Number of old Records found in new File:            02                             |
|  Number of old Records not in new Table:             00                             |
|  Total Number of new Records checked:                02                             |
|  Number of new Records found in old File:            02                             |
|  Number of new Records not in old File:              00                             |
|  Total Number of Fields checked:                     16                             |
|  Total Number of non-Matching Fields:                01                             |
|  Percentage of matching Fields:                      94 %                           |
|  Percentage of matching Records:                    100 %                           |
+-------------------------------------------------------------------------------------+
```

Zusammenfassung

- Systemtests könnten prinzipiell automatisiert werden
- Dies kann nur durch die Überwindung der Lücke zwischen den logischen, anforderungsbasierten Testfällen auf der einen Seite und den physikalischen, codebasierten Testdaten auf der anderen Seite gelingen.
- Dazu ist eine Ontologie erforderlich, welche die Geschäftsentitäten auf die entsprechenden Implementierungsentitäten abbildet und eine Sprache für die Zuweisung von Wertbereichen zu Daten.
- Die Herausforderung liegt in der Überbrückung der Lücken zwischen semantischen Ebenen!