Wie Modelle das Testen effizienter machen

Modellbasiertes Testen mit dem UML Testing Profil V2 der OMG

Markus Schacher, KnowGravity Inc.

Die Aufwände für das Testen machen heute immer noch einen substantiellen Teil der Entwicklungskosten komplexer Systeme aus. Modellbasiertes Testen (MBT) verspricht eine deutliche Reduktion dieser Aufwände, vor allem beim Entwurf und der Pflege von Testspezifikationen. Einerseits vereinfacht die umfassende Wiederverwendung testrelevanter Konzepte deren Pflege und Weiterentwicklung. Andererseits ermöglicht der erhöhte Formalisierungsgrad von MBT sowohl den Testentwurf als auch die Testausführung weitgehend zu automatisieren, was insbesondere bei Regressionstests zu großen Kosteneinsparungen führt. Das UML Testing Profile V2 (UTP 2) ist eine neue Spezifikation der Object Management Group (OMG), welche ganz auf das modellbasierte Testen ausgerichtet ist.

Modelle

Bevor wir uns mit dem modellbasierten Testen befassen, ist es wichtig klar zu machen, was unter einem "Modell" eigentlich genau zu verstehen ist. Im Wesentlichen ist ein Modell etwas, das sich für einen bestimmten Zweck als (stark) vereinfachende Beschreibung eines Originals (dem Modellierungsgegenstand) nutzen lässt. Bei der Erstellung oder Nutzung eines Modells sind zwei Dinge essentiell:

- Es muss jederzeit klar sein, was das **Original** ist, d.h. was genau "modelliert" bzw. beschrieben wird.
- Es muss jederzeit klar sein, was der **Zweck des Modells** ist, denn daraus leitet sich einerseits ab, welche Aspekte des Originals weggelassen werden (die Vereinfachung) und welche Darstellungsform für das Modell verwendet wird bzw. zu verwenden ist.

Der Nutzen, der aus der Erstellung und Verwendung von Modellen entsteht, lässt sich in folgende Themen gruppieren:

- Modelle ermöglichen die **Kommunikation** sowohl zwischen Menschen (vom initialen Gedankenaustausch bis hin zur Auftragsformulierung) als auch zwischen Menschen und Maschinen (beispielweise zur automatisierten Auftragsausführung durch Maschinen).
- Modelle fördern das **Verständnis** über den Modellierungsgegenstand indem dieser untersucht wird und die daraus entstandenen Erkenntnisse in Form eines Modells festgehalten werden (beispielsweise Modelle über das menschliche Verhalten oder über die Abläufe in einem Unternehmen).
- Modelle einer gewissen Präzision ermöglichen eine **formale Analyse** und damit die Vorhersage gewisser Eigenschaften oder Verhalten des modellierten Originals (beispielsweise Wettermodelle oder Modelle aus der Konstruktions- und Festigkeitslehre).

Modelle gibt es in ganz verschiedenen Ausprägungen. Modelle unterscheiden sich beispielsweise durch ihre Ausdrucksformen (physische Objekte wie ein Modellschiff, textuelle Beschreibungen wie Formelsammlungen oder graphische Darstel-

lungen wie elektrische Schemata) und ihre Präzision (informale Modelle versus mathematisch analysierbare oder simulierbare Modelle). Welche Ausdrucksform und welche Präzision eines Modells die richtige ist, hängt von dessen Zweck, und damit verbunden, der Zielgruppe der Modelle ab (siehe dazu auch Abbildung 1).

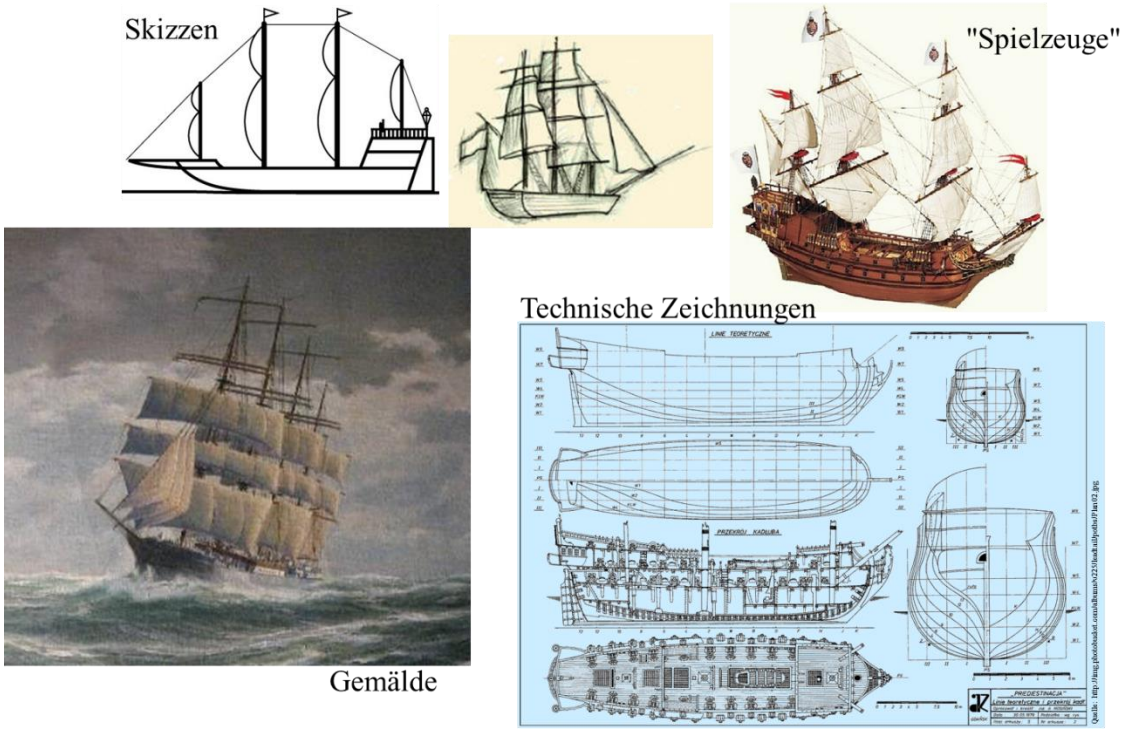

Abbildung 1: Verschiedene Ausprägungen von Modellen

Bei Modellen, welche Originale beschreiben, die sich permanent verändern oder bei Modellen, welche auf einem noch unvollständigen, sich stetig entwickelnden Verständnis des Originals basieren, ist es essentiell, dass die Modelle effizient dem jeweils neuesten Erkenntnisstand angepasst werden können. Eine wichtige Eigenschaft von gut wartbaren Modellen ist, dass sie mit Referenzen arbeiten: ein solches Modell ist aus vielen Modellelementen aufgebaut, die sich gegenseitig referenzieren.

Im Gegensatz zu einer Beschreibung in Form eines sequenziellen Prosatextes ist dabei die Beschreibung des gesamten Originals aus einer grossen Menge von hochgradig, durch Referenzierung wiederverwendeten Teilbeschreibungen (den Modellelementen) aufgebaut. Dies erlaubt einerseits die Produktion verschiedener (zielgruppenspezifischer) Sichten auf ein einziges Modell und andererseits Änderungen zentral an einzelnen Modellelementen vorzunehmen und daraus wieder ein konsistentes Set von Sichten zu reproduzieren (siehe Abbildung 2).

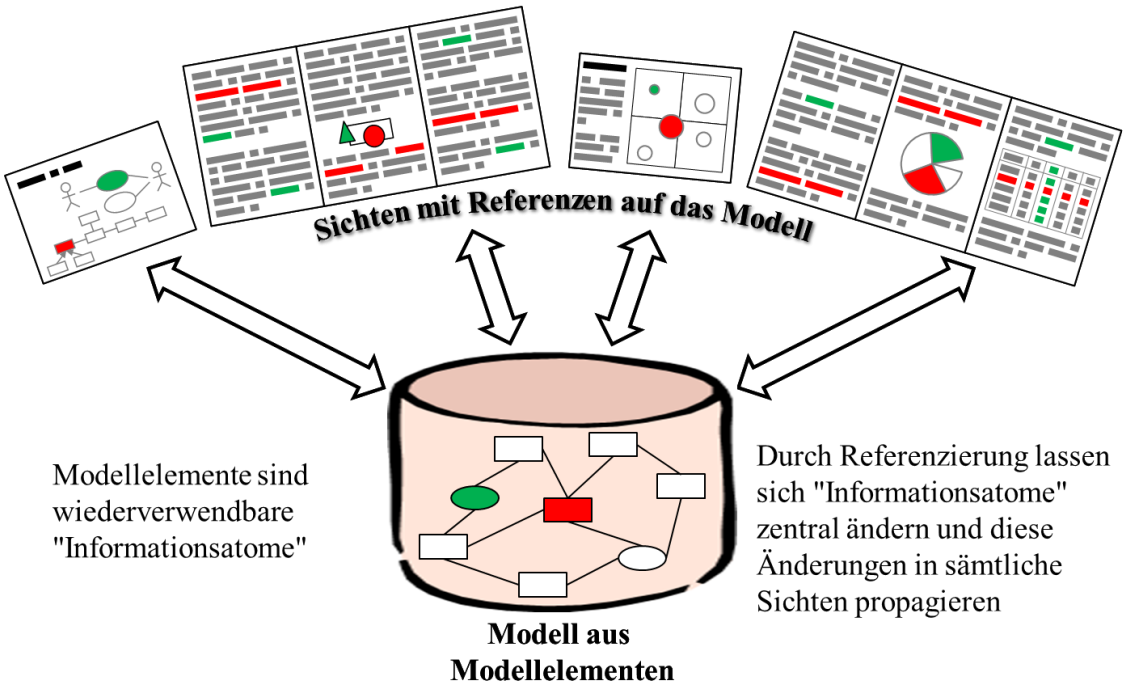

Abbildung 2: Konsistente Sichten auf ein Modell durch Modellelement-Referenzen

Modellbasiertes Testen
Sind technische oder soziotechnische Systeme zu testen, so werden oft Testspezifikationen erstellt. Solche Testspezifikationen sind komplexe Beschreibungen verschiedener **Modellierungsgegenstände**, die im Zusammenhang mit dem Testen relevant sind:

1. Das zu testende Objekt sowie dessen zu testende Eigenschaften, Verhalten und Konfigurationen
2. Die hinter dem Testen stehende Absicht, d.h. was, warum und mit welcher Intensität zu testen ist
3. Die für das Testen anzuwendenden Strategien, Verfahren und Techniken
4. Die für das Testen eingeplanten Ressourcen und Tätigkeiten sowie die sich daraus ergebende Zeitplanung
5. Die konkreten Ausführungsanweisungen für die Durchführung der einzelnen Tests
6. Die für das Testen benötigten Hilfsmittel wie Umsysteme, Testkomponenten aber auch Testdaten
7. Die erwarteten Messwerte oder Reaktionen des zu testenden Objekts sowie die Auswertungsregeln mit denen entschieden wird, ob ein Test erfüllt oder nicht erfüllt ist
8. Die Testabdeckung gegenüber den Eigenschaften, Verhalten und Konfigurationen des zu testenden Objekts.

All diese oben aufgezählten Elemente lassen sich nun als Modellierungsgegenstände betrachten, die in einer Testspezifikation beschrieben werden können. Erfolgt diese Beschreibung modellbasiert, so ergeben sich daraus die folgenden **Vorteile**:

A. Durch das gezielte Weglassen von Detailinformationen wird ein **höheres Abstraktionsniveau** der Testspezifikation erreicht wodurch sie kompakter und besser verständlich wird.
B. Durch die Referenzierung häufig **wiederverwendeter Elemente** (bis hin zu Standard-Bibliotheken) werden Testspezifikation viel änderungsfreundlicher, falls das zu testende Objekt sich verändert oder weiterentwickelt (z.B. bei Regressionstests).
C. Werden die Modelle in einer entsprechenden **Präzision** erstellt, so lassen sie sich automatisch analysieren (z.B. Impact-Analyse), automatisch in weitere Modelle transformieren oder die Tests können gar automatisch ausgeführt werden.

Kommt für die Beschreibung dieser Aspekte eine standardisierte Modellierungssprache zum Einsatz, so ergeben sich weitere Vorteile:

D. Die standardisierte Modellierungssprache gibt eine gewisse **Strukturierung der relevanten Information** vor, an der sich der Tester orientieren kann.
E. Die Kommunikation der verschiedenen Testaspekte (z.B. Testziele, Teststrategien, Testprozeduren, etc.) wird vereinfacht und effizienter, da alle Beteiligten eine **einheitliche Sprache** sprechen.
F. Eine Standardsprache für die Formulierung von Testspezifikationen fördert ein **Ökosystem verschiedener Werkzeuge** und Hilfsmittel, welche bei der Erstellung und Verwaltung, aber auch für die automatische Ausführung von Testspezifikationen genutzt werden können.

Das UML Testing Profile V2 (UTP 2)

Das UML Testing Profile ist eine Spezifikation der Object Management Group (OMG), welche in ihrer ersten Version bereits 2005 veröffentlicht wurde. Sie hatte bereits damals das Ziel, einen Standard für das modellbasierte Testen zu schaffen. 2012 und 2013 wurden zwei kleinere Updates (1.1 und 1.2) veröffentlicht und parallel dazu die neue Hauptversion UTP 2 in Angriff genommen. UTP 2 ist eine standardisierte Modellierungssprache für das Testen mit der die folgenden Ziele angestrebt werden:

- UTP 2 soll **Tests jeglicher Art** für jegliche Art von Systemen unterstützen (nicht bloss Software-Tests)
- UTP 2 soll **den gesamten Testprozess** unterstützen, von der Testkonzeption bis hin zur Testausführung und -auswertung
- UTP 2 soll die **Vorteile modellbasierter Ansätze** im Bereich des Testings zum Tragen bringen
- UTP 2 soll **ein Standard** für modellbasiertes Testen (MBT) werden
- UTP 2 soll (bei Bedarf) so **formal** sein, dass sich aus Testspezifikationen weitere Artefakte erzeugen lassen können
- UTP 2 soll auf der Unified Modeling Language (UML) basieren, sodass UTP sich mittels **gängiger UML-Werkzeuge** und mit üblichen UML-Kenntnissen nutzen lässt
- UTP 2 soll andere auf der UML basierende **Modellierungssprachen ergänzen** (beispielsweise SysML, MARTE, UPDM, oder BMM)
- UTP 2 soll **kompatibel zu anderen Standards** im Bereich des Testings sein, wie beispielsweise ISO/IEC/IEEE 29119, ISTQB, ETSI TTCN-3 oder TDL

Die Entwicklung von UTP 2 befindet sich momentan im Abschluss und es ist vorgesehen, die Spezifikation per Dezember 2016 der Object Management Group zur Finalisierung und Veröffentlichung einzureichen.

UTP 2 besitzt verschiedene Eigenschaften und Mechanismen mit denen die oben genannten Vorteile erreicht werden:

- UTP 2 bietet **verschiedenste Abstraktions- bzw. Detaillierungsebenen** an, die es dem Anwender erlauben, sich auf das Wesentliche zu konzentrieren. Mit verschiedenen Abstraktionstechniken (Spezialisierung, Wiederverwendung, Parametrisierung, Instanziierung, etc.) bietet UTP 2 die Möglichkeit, wichtige Zusammenhänge sehr kompakt und trotzdem präzise zu spezifizieren. (Vorteil A)
- UTP 2 **definiert ein Metamodell** aus standardisierten Modellelementtypen, welche in verschiedenen Sichten referenziert und damit konsistent wiederverwendet werden können. (Vorteil B)
- UTP 2 bietet bei Bedarf präzis definierte **Standard-Testaktionen, Datenoperationen sowie Testauswertungsregeln**, sodass sich z.B. Testfälle aus anderen Modellen generieren oder automatisch ausführen lassen. (Vorteil C)
- UTP 2 bietet eine **umfassende Abdeckung über Themen aus dem Bereich des Testing**: Test Analyse, Test Design, Test Architektur, Testverhalten, Testdaten und Testauswertung. UTP 2 bietet dem Anwender eine ganz konkrete, strukturierte Hilfestellung an, wie diese Themen auszuarbeiten sind und wie sie untereinander zusammenhängen. (Vorteil D)
- UTP 2 ist eine domänenspezifische, diagrammorientierte Sprache, welche die **Formulierung Test-relevanter Aspekte vereinheitlicht** und damit eine zentrale Grundlage für unmissverständliche Kommunikation darstellt. (Vorteil E)
- Da UTP 2 eine Erweiterung der Unified Modeling Language (UML) ist, lässt sich UTP 2 **in gängigen UML-Werkzeugen nutzen** und es kann auf bereits bestehendem Know-how aufgebaut werden. (Vorteil F)

Ein Beispiel

Zur Illustration, wie die durch UTP 2 abgedeckten Themen in der Praxis angewandt werden können, sind im Folgenden einige Besipiele aus der Entwicklung eines Personenaufzugs zusammengestellt.

Eine der ersten und wichtigsten Überlegungen beim Testen ist die Festlegung, was eigentlich der zu testende Prüfling ist (in UTP 2 «Test Item» genannt), welche Hilfsmittel dazu erforderlich sind (in UTP 2 «Test Component» genannt) und welche Konfiguration dieser Elemente für die Durchführung von Tests erforderlich ist. Abbildung 3 zeigt vier verschiedene solche «Test Configurations» welche verschiedene Elemente eines Personenaufzugs mit verschiedenen Hilfsmitteln und verschiedenen Zwecken beschreiben. (Modellierungsgegenstände 1 und 6).

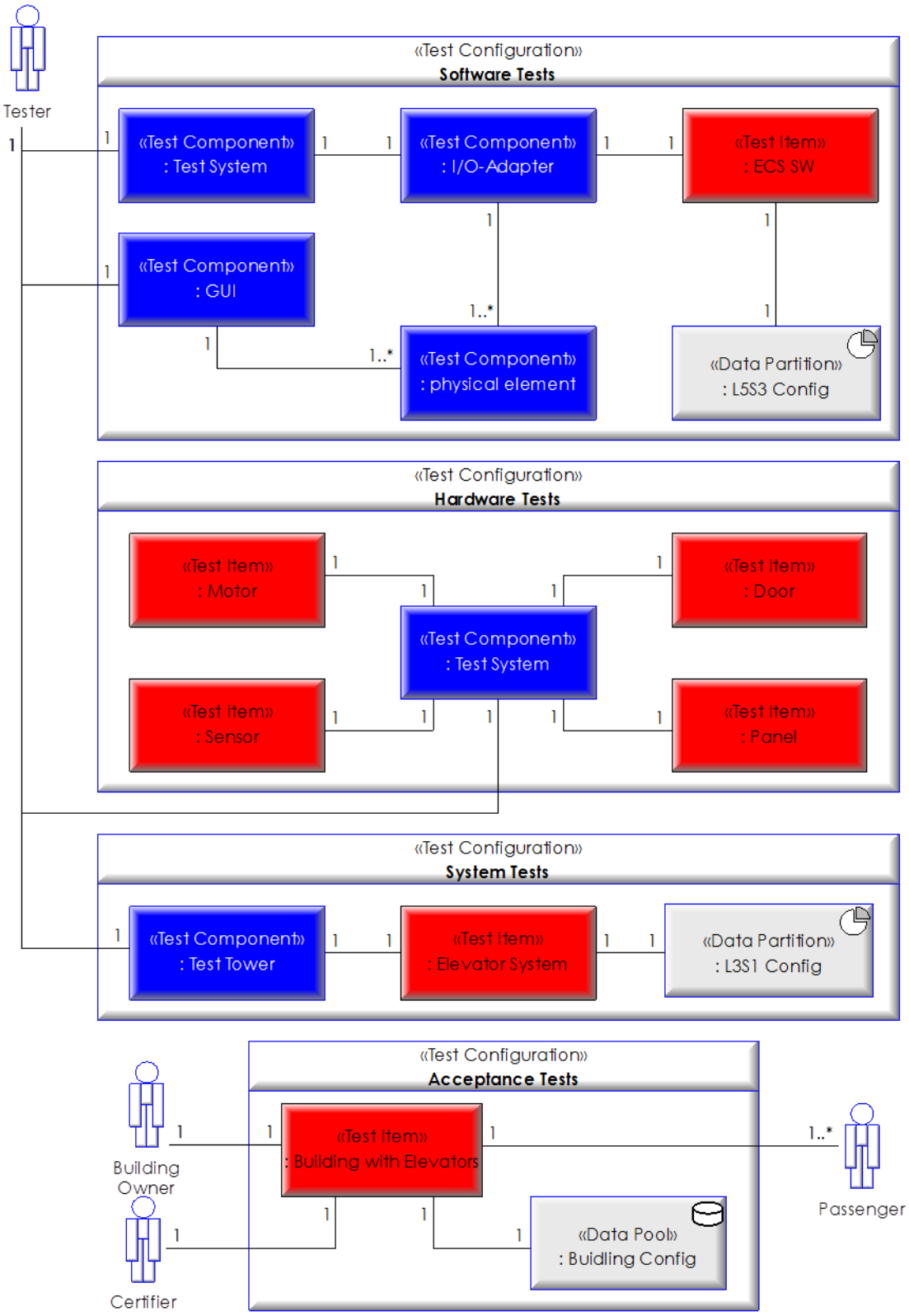

Abbildung 3: UTP 2 Test Konfigurationen

Demgegenüber beschreibt Abbildung 4 ganz andere Zusammenhänge: Welche Ziele («Objective») und Anforderungen («Requirement») an das «Test Item» mit welchen Testfällen («Test Case») validiert («validate» Beziehungen) bzw. verifiziert («verify» Beziehungen) werden sollen (Modellierungsgegenstände 2, 3 und 8).

Es zeigt aber auch, welche Testfälle, welche Testanforderungen («Test Requirement») erfüllen («satisfy» Beziehungen). Durch diese Art von Beziehungen wird eine detaillierte Nachvollziehbarkeit (engl. "Traceability") ermöglicht, falls diese erforderlich ist. Zudem ist in diesem Diagramm ersichtlich, dass gewisse Testfälle Spezialisierungen anderer Testfälle sind oder andere Testfälle um spezifische Aspekte erweitern.

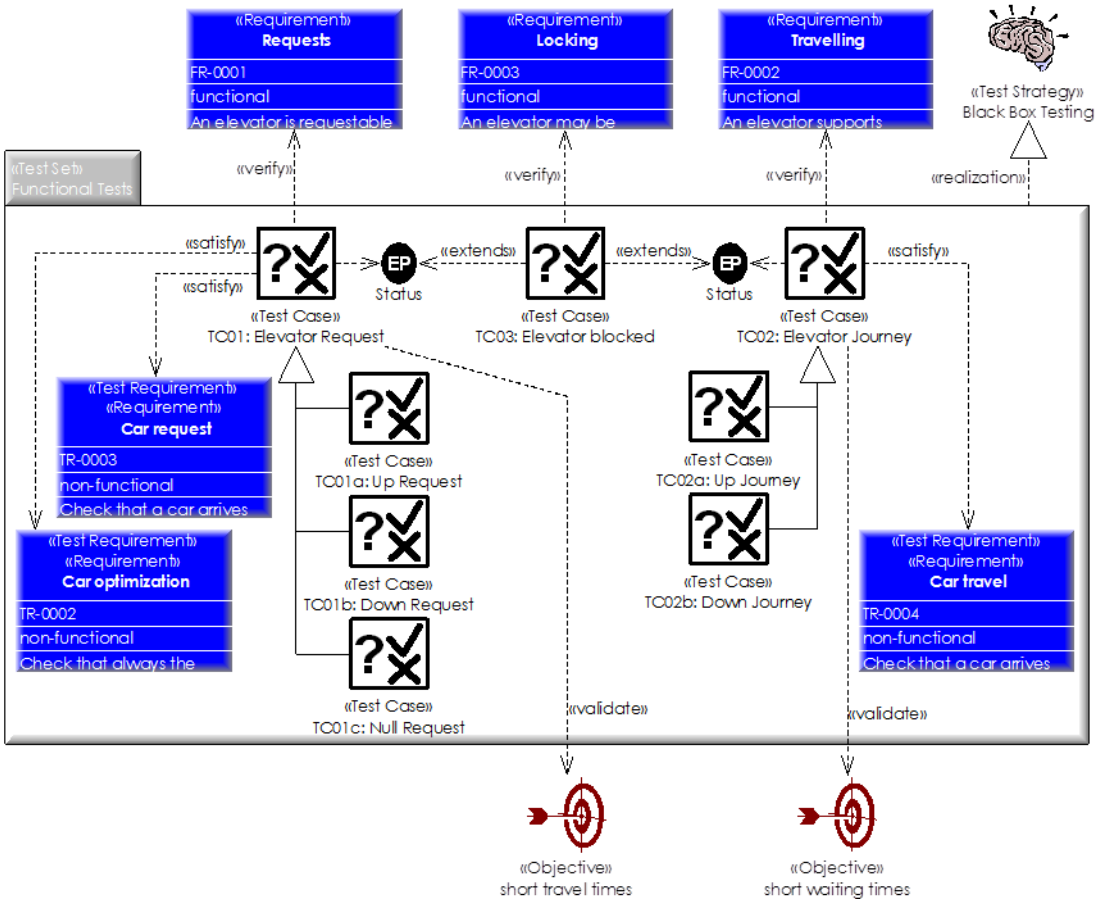

Abbildung 4: Test Sets und Testfälle mit UTP 2

Schliesslich zeigt Abbildung 5 ein Beispiel einer konkreten Testprozedur (diejenige des Testfalls "TC01: Elevator Request" in Abbildung 4), die aus standardisierten Testaktionen ("create stimulus", "expect response", "check property", etc.) aufgebaut ist (Modellierungsgegenstände 5 und 7).

Auf diesem Diagramm ist derselbe Tester aus Abbildung 3 als Testkomponente gezeigt und das «Test Item» ist dasjenige aus der «Test Configuration» "Acceptance Tests". Dies sind Beispiele, wie Diagramme eben lediglich Referenzen auf nur einmal definierte Modellelemente zeigen. Müssen Änderungen vorgenommen werden, so können diese Änderungen an einer Stelle erfolgen und sämtliche darauf aufbauenden Sichten werden automatisch aktualisiert.

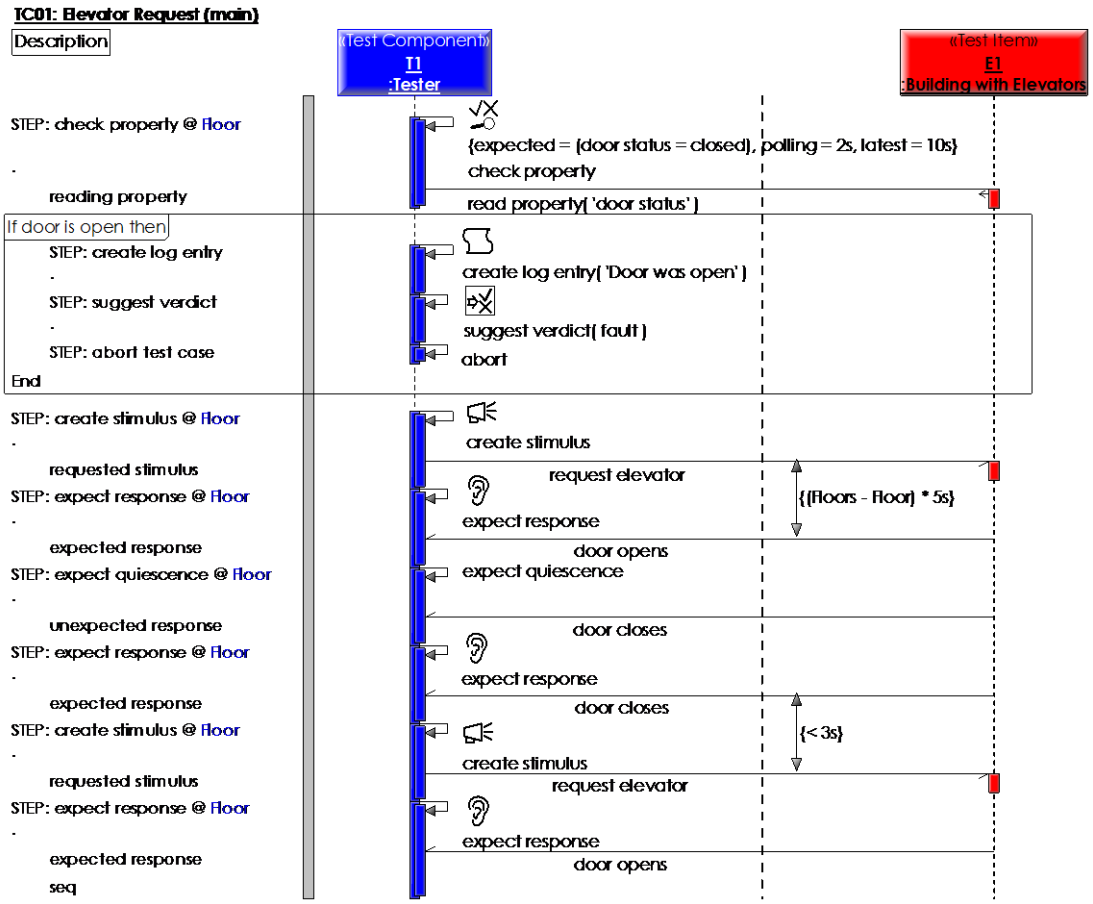

Abbildung 5: Eine UTP 2 Testprozedur

Zusammenfassung

Modellbasiertes Testen verspricht eine Effizienzsteigerung beim Testen, welche aus der Nutzung der Vorteile der Modellierung im Allgemeinen resultieren. Die Nutzung wiederverwendbarer Modellelemente erhöht die Konsistenz einer Testspezifikation und vereinfacht deren Wartung bei Änderungen. Die Präzision einer Modellierungssprache wie dem UTP 2 erlaubt zudem die Automatisierung verschiedener Aufgaben rund um das Testen.

Literaturverzeichnis

[ETSI] European Telecommunications Standards Institute (ETSI): ES 202 951: Requirements for Modeling Notations. ETSI Standard, Methods for Testing and Specification (MTS); Model-Based Testing (MBT), V1.1.1 (2011-07)

[ISO] International Organization for Standardization: Software and systems engineering – Software testing – Parts 1-5, ISO 29119-1:2013(E), 2013-09-01, September 2013

[TTCN3] ETSI: Methods for Testing and Specification (MTS) – The Testing and Test Control Notation Version 3 – Part 1: TTCN-3 Core Language (Version 4.7.1), ETSI ES 201 873-1, June 2015, www.ttcn-3.org/index.php/downloads/standards/

[UTP] Object Management Group: UML Testing Profile (UTP) – version 1.2, ptc/2012-09-13, September 2012

Autor

Markus Schacher ist Mitbegründer und KnowBody von KnowGravity Inc., einem kleinen aber feinem Beratungsunternehmen mit Sitz in Zürich (Schweiz), welches sich auf modellbasiertes Engineering spezialisiert hat. Als Trainer hat Markus bereits 1997 die ersten öffentlichen UML-Kurse in der Schweiz durchgeführt und hat als Berater vielen grossen Projekten geholfen, modellbasierte Techniken einzuführen und nutzbringend anzuwenden. Heute ist er als aktives Mitglied der Object Management Group (OMG) in die Entwicklung verschiedener Modellierungssprachen involviert und ist Ko-Autor dreier Bücher zu den Themen Geschäftsregeln, SysML sowie operationellen Risiken.

Kontakt

Internet: www.knowgravity.com
Email: markus.schacher@knowgravity.com