Marc-Florian Wendland

(E-Mail: marc-florian.wendland@fokus.fraunhofer.de) arbeitet seit 2008 im Kompetenzzentrum MOTION des Fraunhofer Institut FOKUS in Berlin. Seine aktuellen Forschungsschwerpunkte liegen im Bereich der modellbasierten/anforderungsgetriebenen Tests unter Zuhilfenahme der anerkannten Standards UML, UTP, SysML und TTCN-3. Als Mitglied der OMG leitet er darüber hinaus die Revision des UML Testing Profils bereits im zweiten Jahr und ist iSQI-zertifizierter TTCN-3-Experte. Er ist verantwortlich für die Entwicklung des MBT-Werkzeugs Fokus!MBT, welches in zahlreichen Fallstudien internationaler Projekte eingesetzt wird.

Prof. Dr. Ina Schieferdecker

(E-Mail: ina.schieferdecker@fokus.fraunhofer.de) beschäftigt sich mit Fragen der modellbasierten Softwareentwicklung, der Analyse, des Testens und Bewertung softwareintensiver Systeme und der Automatisierung und Optimierung von Prozessen zur Software-(Weiter-)-Entwicklung und Qualitätssicherung. An der Freien Universität Berlin leitet sie das Fachgebiet „Modellbasierte Entwicklung und Qualitätssicherung Software-basierter Systeme" und ist am Fraunhofer Institut FOKUS, Berlin, Leiterin des Kompetenzzentrums MOTION, das sich mit Modellierungs- und Testansätzen und der Automatisierung und Optimierung von Softwareentwicklungsprozessen beschäftigt.

Markus Schacher

(E-Mail: markus.schacher@knowgravity.com) Mitbegründer von KnowGravity Inc., einem kleinen High-End-Beratungsunternehmen in Zürich. Unterrichtet seit über 20 Jahren verschiedene Spezifikationstechniken und führte bereits Anfang 1997 die ersten öffentlichen Schulungen in der Schweiz zur Unified Modeling Language (UML) durch. Als KnowBody in verschiedene nationale und internationale Projekte im sicherheitskritischen Umfeld involviert, unterstützt aber auch Unternehmen bei der Anwendung der Model Driven Architecture (MDA) sowie der Erstellung lösungsneutraler Spezifikationen auf der Basis der executable UML (xUML). Als Mitglied der OMG zurzeit insbesondere aktiv in die Weiterentwicklung des UML Testing Profiles (UTP) involviert.

Testen mit dem UML Testing Profile

Das UML Testing Profile [UTP] ist eine grafische Modellierungssprache zur Definition, Dokumentation, Visualisierung, Implementierung und zum Austausch von modellbasierten Testspezifikationen auf Basis der UML [UML]. Der normative Teil der Spezifikation definiert das native UML-Profil des UTP[1]. Die Spezifikation des UTP als UML-Profil ist im Vergleich zu anderen Sprachen der UML-Familie [SysML, MARTE, SoaML] deutlich schlanker und kompakter, erfordert aber gleichzeitig profundes UML-Wissen, um die Potenziale von UTP voll ausschöpfen zu können. Ein Umstand, der die Akzeptanz des UTP bis heute behindert, da nur wenige Tester in den Konzepten der Modellierung im Allgemeinen ausreichend geschult und mit der UML im Besonderen vertraut sind. Dieser Artikel widmet sich daher neben einer kurzen Einführung in das UTP vor allem Fragen seiner Anwendbarkeit. Dabei stehen weniger spezifische Domänen, sondern eher grundsätzliche, testmodellspezifische Fragestellungen im Vordergrund.

Das UTP im Überblick

Strukturell gliedert sich das UTP in *vier* logische Konzeptpakete:

Testarchitektur: Unter dem Begriff *Testarchitektur* (test architecture) sind Konzepte zusammengefasst, die für die Definition struktureller Aspekte des Testmodells vonnöten sind. Hierzu zählen die Konzepte *TestContext*, *TestComponent*, *System Under Test (SUT)* und *Arbiter*.

Testverhalten: Testverhalten (test behavior) wird mit Konzepten wie dem zentralen Stereotyp TestCase, testspezifischen Aktionen (etwa *ValidationAction*) sowie der Definition und Anbindung von Default-Verhalten beschrieben.

Testdaten: Konzepte zur modellbasierten Definition von Testdaten finden sich in dem Paket *test data*. Aus TTCN-3 [TTCN3] wurden Wildcards (Platzhalter) entliehen, mit denen sich Modellierer bei der Spezifikation konkreter Datenpartitionsrepräsentanten ausschließlich auf die relevanten Wertebelegungen konzentrieren und im jeweiligen Kontext uninteressante Werte vernachlässigen können.

Timerkonzepte: Die Zeitkonzepte der UML sind vielfach nicht ausreichend, um Testfälle flexibel genug zu spezifizieren. Das UTP führt zu diesem Zweck die vordefinierte Schnittstelle Timer ein. Sie repräsentiert den Zugang zum Testausführungssystem auf abstrakter Ebene und lässt sich über wohl definierte Timer-Aktionen wie *StartTimerAction* und *StopTimerAction* innerhalb eines Testverhaltens aufrufen.

UTP v1.1 beinhaltet zudem Ergänzungen zum Thema Testmanagement, wobei für zukünftige Versionen geplant ist, diesen wesentlichen Teilbereich zu erweitern und als eigenständiges Paket zu konstituieren.

Für einen tieferen Einstieg empfehlen wir das umfassende Beispiel-Kapitel des OMG-Standards sowie das Sekundärwerk [Baker]. Zudem sind einige Papiere und Tutorials frei im Internet zugänglich, die das Verständnis erleichtern und Ansätze von UTP-basierten Testmethoden diskutieren. Berichte über Success Stories oder umfassende Testmethodiken (inklusive Guidelines, Do's and Dont's. etc.) auf Basis des UTP sind leider nach wie vor Mangelware, was sich aber beispielsweise mit der [MBTUC] im Oktober in Berlin ändern sollte.

Testspezifikationen mit UTP

Wie alle Technologien der OMG ist auch das UTP grundsätzlich methodenunabhän-

[1]) *Im informativen Anhang ist darüber hinaus ein MOF-basiertes, autarkes UPT-Metamodell beschrieben, welches hier aber nicht weiter erläutert wird.*

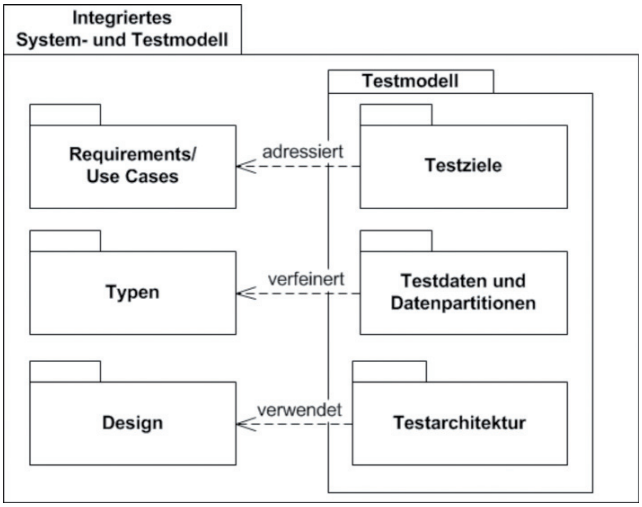

Abb. 1: Wiederverwendung von Systemmodellartefakten

gig, um eine möglichst umfassende Anwendbarkeit zu garantieren. Auf der einen Seite gestattet dies die Integration des UTP in jede beliebige auf UML aufsetzende (Test-)Methodik. Dieser Flexibilität steht jedoch die Notwendigkeit gegenüber, eigene Methoden, Richtlinien und Regeln für den Einsatz des UTP (und der genutzten UML-Konzepte) zu erarbeiten.

UTP bringt das nötige Rüstzeug mit, um angemessene, aussagekräftige und zielführende Testmodelle zu entwerfen. Die Komplexität und Fülle der zugrundeliegenden UML kann bei Testern rasch zu einer prinzipiellen Ablehnung des UTP führen – ein nicht ganz unverständliches, aber leider unnötiges Verhalten: so nutzen Entwickler ebenso nicht alle Merkmale einer genutzten Programmiersprache, sondern beschränken sich auf das Nötige und Passende – gleiches ist bei der Modellierung mit der UML und ergo UTP möglich.

Nachfolgend diskutieren wir, wie UTP-Testmodelle für verschiedene Ansätze genutzt werden können und geben Hinweise für eine UTP-basierte Testmethodik.

Eigenständige und integrierte Testmodelle

In [Schieferdecker] wurden mögliche MBT-Ansätze diskutiert, bei denen Testmodelle entweder eigenständig oder unter Zuhilfenahme eines existierenden Systemmodells eingesetzt werden. Das UTP ist flexibel genug, um beide Varianten zu realisieren. Ist ein Systemmodell nicht mit der UML entworfen oder nur für (abstrakte) Dokumentationszwecke definiert, lässt sich der Testprozess trotz allem modellbasiert umsetzen, allerdings muss der Tester sich in diesem Fall um die Entwicklung des Testmodells in seiner Gesamtheit kümmern.

Wird hingegen ein UTP-Testmodell in Anlehnung an ein UML-basiertes Systemmodell realisiert, lassen sich Informationen aus dem Systemmodell wiederverwenden (**Abbildung 1**). Dies erspart insbesondere den Entwurf grundlegender Strukturinformationen und erlaubt dem Tester, sich dediziert den testrelevanten Artefakten zu widmen. Diese Variante ist jedoch mit Bedacht einzusetzen. Systemmodellartefakte sollten beispielsweise nicht von testspezifischen Artefakten abhängig sein. Wird bei einem Review verlangt, die testspezifischen Informationen zu entfernen, darf weder die syntaktische noch die semantische Integrität des Systemmodells verletzt werden. Andererseits dürfen Informationen aus dem Systemmodell nur dann übernommen werden, wenn sie die Aussagekraft der Testfälle nicht negativ beeinflussen. Dazu zählt, dass das erwartete Verhalten des SUT innerhalb des Testmodells aus Sicht der Tester entworfen werden sollte, welches dann zur automatisierten Ableitung und Ausführung von Testfällen verwendet werden kann. Eine bedenkenlose Übernahme verhaltensspezifischer Beschreibungen aus dem Systemmodell zur Testfallableitung ist nur bedingt aussagefähig bezüglich der Anforderungsvalidation.

Testdaten als Partitionen und Instanzen

Testdaten sind essentiell für die Definition von Tests. Zur logischen Klassifikation von Testdaten unterstützt UTP Datenpartitionen und darauf basierende Instanzen, die flexibel im Testmodell genutzt werden können. Datenpartitionen lassen sich schachteln, vererben, redefinieren oder als abstrakt deklarieren. Diese Variabilität gestaltet ihre widerspruchsfreie Nutzung schwierig, schränkt aber andererseits die Kreativität des Testers und die Vielfalt der Anwendungsszenarien für UTP nicht ein.

Ein mögliches Szenario ist es, durch Vererbung bereits existierende Typen in das Testmodell zu importieren. Der generelle Typ fungiert in diesem Fall als Basisdatentyp der logischen Datenpartition. Instanzen der Datenpartitionen bleiben dadurch typkonform zum Basisdatentyp und können überall dort genutzt werden, wo Instanzen des Basisdatentyps erwartet werden. Weiterhin lassen sich Eigenschaften des Basisdatentypen redefinieren oder per Constraint einschränken. Dies stellt einen einfachen und intuitiven Weg dar, um beispielsweise Klassifikationsbäume aufzubauen, ohne das Typsystem für die Testspezifikation ändern zu müssen (**siehe Abbildung 2**).

Logische und technische Testfallmodellierung

Testfälle werden gemeinhin in *abstrakte* (auch *logische*) oder *konkrete* Testfälle unterschieden, wobei der Term abstrakt das gezielte Aussparen von Informationen (meistens konkreter Datenwerte) meint ([Roßner], [Utting], [Spillner]). *Konkret* hingegen bezeichnet die Vollständigkeit eines Testfalls bezüglich konkreter Datenwerte. Diese Terminologie ist nachvollziehbar, sollte nach dem Verständnis der Autoren jedoch differenziert werden.

Die Begriffe *logisch* und *abstrakt* sind nicht synonym zu verwenden, sondern klar voneinander abzugrenzen. Logische Testfälle beschreiben das fachliche Testverhalten auf einer hohen, technologieunabhängigen Ebene. Im Sinne einer Ablaufbeschreibung zielen logische Verhaltensbeschreibungen auf die Intention eines Testfalls, also was ein Testfall (*fachlich*) zu erbringen hat, nicht wie er es (*technisch*) realisiert. Ein Domänenexperte ist in der Lage, logische bzw. fachliche Testfälle für ein System (beispielsweise als Aktivitätsdiagramm) zu entwerfen. Logische Testfälle können jedoch abstrakt oder konkret bezüglich Datenwerten sein, was mit der technischen Realisierung respektive

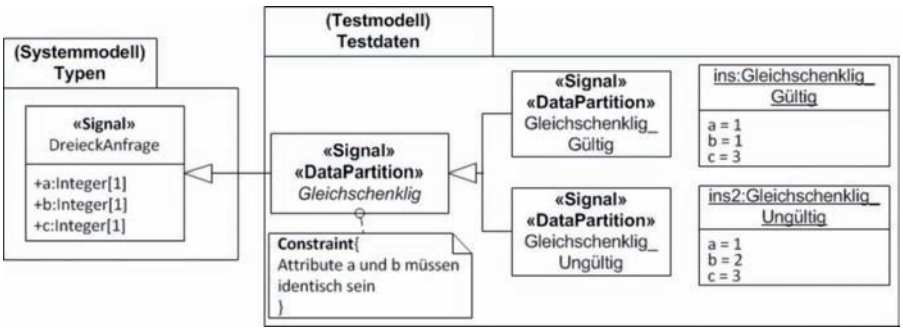

Abb. 2: Testdatenableitung per Generalisierung

Ausführung des Testfalls nichts gemein hat. Daher führen wir zudem den Begriff *technischer Testfall* ein, der ebenso konkret oder abstrakt sein kann. Ein konkreter, technischer Testfall beinhaltet ausreichend Informationen, um den Testfall gegen das System auszuführen und dessen Reaktionen auszuwerten. In Abbildung 3 sind die verschiedenen Kombinationen von logisch-abstrakt bis technisch-konkret exemplarisch vorgestellt ist.

Integration mit anderen UML-Profilen

Das UTP stellt eine domänenspezifische Erweiterung von UML für die Testmodellierung dar. Im Kielwasser der UML findet sich eine Vielzahl domänenspezifischer Profile, etwa die SysML, die SoaML oder MARTE. Jedes dieser Profile verfolgt einen bestimmten, domänenspezifischen Zweck und beschränkt, respektive erweitert die UML zielgerichtet. Das UTP ist prinzipiell mit jedem anderen UML-Profil kombinierbar, wobei mit Umsicht agiert werden muss, da es mitunter zu technischen Inkonsistenzen und Redundanzen kommen kann. Der Vorteil von Profilkombinationen liegt auf der Hand: es ist beispielsweise möglich, das UTP mit SoaML zu kombinieren, um UTP-basierte Testspezifikationen für service-orientierten Architekturen zu definieren. Dieser kombinatorische Ansatz steht dem Entwurf dedizierter Testprofile für jede domänenspezifische Spezifikation gegenüber, der neben einer regelrechten Profilinflation mit einem hohen Maß an Redundanz verbunden wäre.

Rückblick und Ausblick

In der seit 2005 verfügbaren UTP-Version fanden sich technische Mehrdeutigkeiten,

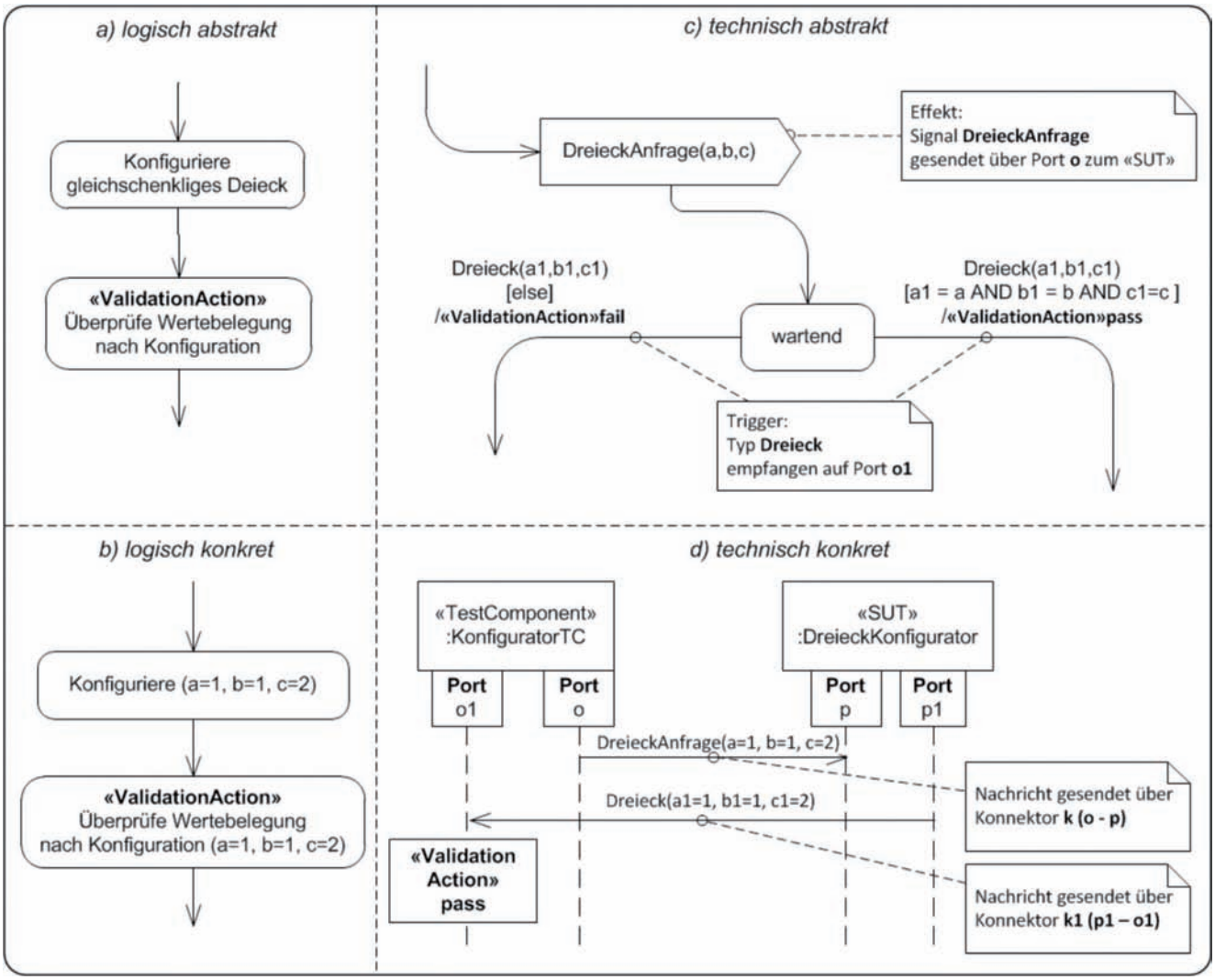

Abb. 3: Logische und technische Testfallspezifikation – abstrakt und konkret

die eine vollständige Implementierung des Standards erschwerten. Dennoch stieß die UTP-Spezifikation auf reges Interesse und vielfache Umsetzungen, da es eine wichtige Lücke in der modellbasierten Qualitätssicherung schloss. Dieses Wissen und die Erfahrungen der letzten Jahre waren der Anlass, 2010 eine Revision Task Force für das UTP bei der OMG zu konstituieren [UTP_RTF]. Insgesamt wurden 57 Issues eingereicht, von denen 21 gelöst wurden. Neben der Lösung technischer Probleme wurde vor allem eine besser zugängliche Dokumentenstruktur entworfen, was die Lesbarkeit des Standards deutlich erhöht. Die konzeptionell signifikantesten und richtungsweisenden Änderungen waren die stärkere Adressierung von Testmanagement-Aspekten und die Abkehr vom „ungeliebten" MOF-basierten Metamodell. Im Juli 2011 wurde UTP v1.1 erfolgreich bei der OMG verabschiedet und sollte noch in diesem Jahr offiziell als OMG-Standard verfügbar sein.

Mit Abschluss des UTP v1.1 ist die Arbeit am Standard noch lange nicht abgeschlossen. Während dieser Artikel verfasst wird, laufen die Planungen für ein UTP v2.1 bei der OMG. Auch wenn UTP in seiner aktuellen Version komfortabel anzuwenden ist, lautet das langfristige Ziel, den Standard kontinuierlich zu verbessern. Dies betrifft vor allem die Stärkung der Testdatenkonzepte, wie auch eine präzisere und umfangreiche Ausarbeitung der Testmanagement-Konzepte. Zudem sind Testanforderungen ein zentrales Thema der anstehenden Revision, um eine frühe Testanbindung zu adressieren.

Ein praktisches Anwendungsbeispiel des UTP

KnowGravity hat im vergangenen Jahr im Auftrag der Schweizerischen Bundesbahnen SBB in Zusammenarbeit mit einer Partnerfirma sowie Siemens und Thales einen neuen Standard für Schnittstellen zur Koordination von Stellwerksystemen verschiedener Hersteller entwickelt. Dabei ging es nicht um die Entwicklung der Schnittstelle selber, sondern um die Entwicklung einer funktionalen Spezifikation als Standard für diese Schnittstelle. Da bei solchen sicherheitskritischen Systemen besonderer Wert auf die Korrektheit der funktionalen Spezifikation gelegt wird, wurde unter Verwendung der eXecutable UML (xUML) ein ausführbares Modell dieser Spezifikation entwickelt. Parallel dazu wurden auf der Basis der UTP-Testfälle erarbeitet, um die Korrektheit der ausführbaren funktionalen Spezifikation zu verifizieren.

Konkret kam in diesem Projekt UTP 1.0 mit geringfügigen Erweiterungen zur Anwendung. Zusammen mit der Verwendung der SysML für die Anforderungsmodellierung und der UML für die umfassende Funktionsmodellierung kann zu Recht von einem modellbasierten Testansatz gesprochen werden. Nachdem die Anforderungen erhoben und mittels SysML modelliert worden sind, wurden parallel und von unterschiedlichen Personen die funktionale Spezifikation in xUML und die Testspezifikationen mittels UTP ausgearbeitet. Beim Aufbau der Testspezifikation wurden als erstes thematisch ähnliche Gruppen von Testfällen identifiziert und in Testkontexten gruppiert: System-Initialisierung, System-Terminierung, normale Szenarien, Sonderfälle, etc. Als Erweiterung gegenüber UTP 1.0 wurden zudem für diese Themen sogenannte "Test Maps" erstellt – Übersichtsdiagramme, die eine Gruppe von Testfällen sowie deren Zusammenhänge und Abhängigkeiten untereinander aufzeigen.

Die einzelnen Testfälle wurden mittels Sequenzdiagrammen als Blackbox-Tests spezifiziert, wobei jeweils für jeden einzelnen Testfall explizit der Testgegenstand (in UTP das "SUT", "System Under Test" genannt) identifiziert wurde. Zudem wurden für jeden Testfall das zu erreichende Ziel sowie Vor- und Nachbedingungen definiert. Schließlich wurden über die SysML-Beziehung «verify» die UTP-Testfälle mit entsprechenden SysML-Anforderungen verknüpft, wodurch sich jederzeit der Testabdeckungsgrad nachweisen lässt. Das Test-Setup und die darin enthaltenen Testkomponenten wurden mittels UML-Klassen- und Kompositionsstrukturdiagrammen modelliert. Sowohl für die funktionale Spezifikation als auch für die Testspezifikation wurden mit einem Dokumentgenerator aus dem gemeinsamen Projektmodell vollständige Dokumente generiert.

Die Ausführung der Testfälle gegenüber der ausführbaren funktionalen Spezifikation erfolgte in der xUML-Laufzeitumgebung, in welcher sämtliche spezifizierten Testfälle manuell ausgeführt, protokolliert und für die spätere automatische Ausführung aufgezeichnet wurden. Dies wiederum erlaubte, die sich kontinuierlich entwickelnde funktionale Spezifikation

Referenzen

[UTP] Object Management Group (OMG): UML Testing Profile, Version 1.1 – Beta 1. URL: http://www.omg.org/cgi-bin/doc?ptc/2011-07-19

[UML] Object Management Group (OMG): OMG Unified Modeling Language (OMG UML) Superstructure, Version 2.3. URL: http://www.omg.org/spec/UML/2.3/

[SysML] Object Management Group (OMG): OMG Systems Modeling Language (OMG SysML), Version 1.2. URL: http://www.omg.org/spec/SysML/1.2/

[MARTE] Object Management Group (OMG): Modeling and Analysis of Real-Time Embedded Systems (OMG MARTE), Version 1.1. URL: http://www.omg.org/spec/MARTE/1.1/

[SoaML] Object Management Group (OMG): OMG Service-oriented Architecture Modeling Language (OMG SoaML), Version 1.0 – Beta 2. URL: http://www.omg.org/spec/SoaML/1.0/Beta2/

[Baker] Baker, P. et al: Model-driven testing – using the UML testing profile. Springer (2007)

[TTCN3] ETSI: http://www.ttcn.org, (Letzter Besuch: Juni 2011)

[Schieferdecker] Schieferdecker, I.: Modellbasiertes Testen. OBJEKTSpektrum 3/07, S. 39-45, 2007.

[Roßner] Roßner et al.: Basiswissen Modellbasierter Test. Dpunkt-Verlag, Heidelberg. ISBN: 978-3-89864-589-8, 2010.

[Utting] Utting, M.; Pretschner, A., Legeard, B.: A Taxonomy of Model-Based Testing. ISSN 1170-487X, 2006. URL: http://www.cs.waikato.ac.nz/pubs/wp/2006/uow-cs-wp-2006-04.pdf.

[Spillner] Spillner, A.; Linz, T.: Basiswissen Softwaretest. Dpunkt-Verlag, Heidelberg; ISBN: 978-3-89864-642-0, 2010.

[UTP_RTF] Object Management Group (OMG): OMG UML Testing Profile 1.1 (UTP) RTF. URL: http://www.omg.org/techprocess/meetings/schedule/UTP.html

[MBTUC] 1st ETSI Model-Based User Conference MBTUC, Organisiert von Fraunhofer FIRST und FOKUS, 18-20. Oktober 2011, Berlin, http://www.model-based-testing.de/mbtuc11/

permanent mittels Regressionstests auf fehlerhafte Erweiterungen zu überprüfen.

Abschließend seien hier noch einige Kennzahlen zum Projekt zusammengestellt: Für die Stellwerk-Stellwerk-Schnittstelle wurden rund 150 einzelne, teilweise recht komplexe Testfälle modelliert. Die generierte Testspezifikation weist etwa den eineinhalbfachen Umfang der funktionalen Spezifikation auf. Die automatische Ausführung aller Testfälle (Regressionstest) dauert circa 80 Minuten. Die ursprünglich für die Validierung der ausführbaren Spezifikation erstellten Testspezifikationen bilden eine exzellente Grundlage für die Tests bzw. Zertifizierung der nach diesem Standard realisierten Systeme. Nach Abschluss der Standardisierung der Stellwerk-Stellwerk-Schnittstelle sollen weitere Schnittstellen rund um Stellwerke nach demselben Verfahren standardisiert werden. Zudem wird KnowGravity als Co-Autor des UTP versuchen, die praktischen Erfahrungen aus diesem und ähnlichen Projekten in die neueren UTP-Revisionen einfließen zu lassen.

Zusammenfassung

Mit dem UML Testing Profil existiert seit mehreren Jahren ein überaus vielsprechender Standard, der bereits zu einem frühen (vielleicht in 2005 zu frühen) Zeitpunkt Konzepte an die Hand gab, um auf Basis der UML eigene, maßgeschneiderte modellbasierte Testmethoden und -modelle zu entwerfen. Die Attraktivität des UTP liegt neben seiner Kompaktheit unter anderem darin begründet, dass jedes UML-konforme Werkzeug in der Lage ist, das Profil einzubinden. Als OMG-Standard ist das UTP zudem resistent gegenüber Launen oder finanziellen Engpässen eines einzelnen Herstellers. Anwender können sich darauf verlassen, dass das UTP auch in Zukunft von einem unabhängigen und kompetenten Konsortium gepflegt und weiter entwickelt wird. Wer mehr über den effizienten Einsatz von UTP zur Testmodellierung erfahren möchte, der sei erneut auf [Baker] oder [MBTUC] verwiesen. Gerne nehmen wir auch Anfragen per Mail entgegen.