Philip Makedonski
(makedonski@informatik.uni-goettingen.de)
ist Doktorand am Institut für Informatik der Georg-August-Universität Göttingen. Er beschäftigt sich im Rahmen seiner Forschungsarbeiten unter anderem mit Testsprachen, Qualitätssicherung für Testspezifikationen und Softwareevolution.

Jens Grabowski
(grabowski@informatik.uni-goettingen.de)
ist Professor für Softwaretechnik am Institut für Informatik der Georg-August-Universität Göttingen. Seine Forschungsarbeiten konzentrieren sich zurzeit auf Fragestellungen zur Qualitätssicherung und Evolution von Software.

Testbeschreibung mit TDL: Konzepte und Notationen der ETSI Test Description Language

Die zunehmende Software- und Systemkomplexität aufgrund der Integration verschiedener Subsysteme sowie die Standardisierung und Zertifizierung in bestimmten Einsatzgebieten, wie Telekommunikation, Medizintechnik, Fahrzeug- und Luftfahrtelektronik, stellen neue Herausforderungen beim Testen dar. Neue Technologien und Ansätze werden entwickelt, um den sich stets ändernden Anforderungen nachkommen zu können. Wir stellen im Folgenden einen dieser Ansätze, die ETSI-Testbeschreibungssprache TDL, vor.

Herkunft

Das Standardisierungsinstitut ETSI (European Telecommunications Standards Institute) hat langjährige Erfahrung mit der Entwicklung von Testspezifikationen für standardisierte Technologien. Um die effiziente Entwicklung von standardisierten Testspezifikationen zu unterstützen, arbeitet das ETSI TC-MTS (ETSI Technical Committee - Methods for Testing and Specification) hierfür an neuen Methoden, Prozessen und Sprachen.

Der ETSI-Testentwicklungsprozess [ETSI203130] definiert dabei zum Beispiel die schrittweise Verfeinerung vom Basisstandard bis zu ausführbaren Testimplementierungen basierend auf ISO/IEC 9646-1 [ISO9646]. Nach jedem Verfeinerungsschritt entstehen zwischenliegende Artefakte auf verschiedenen Abstraktionsebenen (Anforderungen, Testzwecke, Testbeschreibungen, Testimplementierungen), die auf bestimmte Stakeholder abzielen, wie Standardisierungsexperten, Technologieexperten, Testingenieure usw.

In den vergangenen 15 Jahren hat ETSI TC-MTS die Entwicklung und Wartung der Testsprache TTCN-3 (Testing and Test Control Notation) geleitet. Auch wenn sich TTCN-3 als die Sprache für die Implementierung von Tests bei ETSI durchgesetzt hat, fehlen auf den höheren Abstraktionsebenen immer noch etablierte Notationen. Selbst wenn die Struktur und die allgemeinen Inhalte von Testzwecken und Testbeschreibungen schon gut etabliert sind, gibt es eine Vielzahl an Dialekten und benutzergruppenspezifischen Notationen für die verschiedenen Standards. So war TPLan (Test Purpose Notation, [ETSI202553]) ein erster Ansatz, eine standardisierte Notation für die Spezifikation von Testzwecken zu entwickeln. Mit Blick auf TPLan und TTCN-3 entstand eine Lücke zwischen den deklarativen Testzweckbeschreibungen und den imperativen Testfallimplementierungen.

Um diese Lücke zu schließen, wurde in den letzten drei Jahren die Testbeschreibungssprache TDL (Test Description Lan-

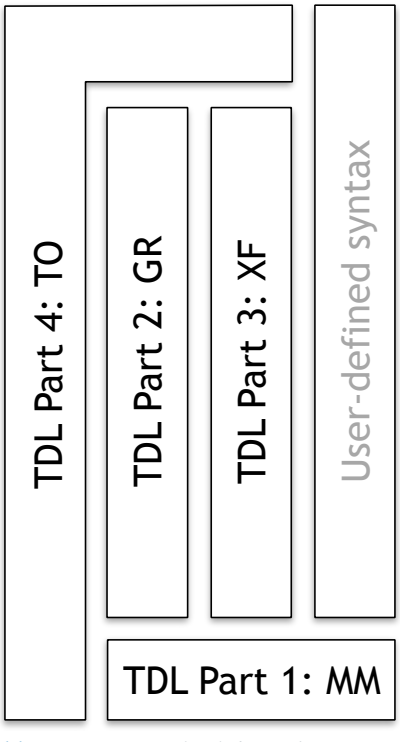

Abb. 1: TDL-Standardübersicht.

guage) beim Standardisierungsinstitut ETSI durch Zusammenarbeit von Industrie, Standardisierungsgremien und akademischen Einrichtungen entwickelt. TDL bietet eine formalisierte modellbasierte Lösung für die Spezifikation von Testbeschreibungen, die als Brücke zwischen den deklarativen Testzweckbeschreibungen und den imperativen Testfallimplementierungen dient.

Einsatzgebiet

Das vorgesehene Einsatzgebiet für TDL ist das Design, die Dokumentation und die Darstellung formalisierter Testbeschreibungen für Black-Box-Tests nach dem Ansatz des sogenannten „Szenario-basierten Testen". Nach diesem Ansatz werden die abstrakten Interaktionen mit dem SUT (System Under Test) in den Vordergrund gestellt, sodass nur die wesentlichen Informationen für den Testablauf abgebildet werden müssen. Alle weiteren Einzelheiten, die notwendig für die Testdurchführung, aber nicht explizit für den Testzweck relevant sind, werden anschließenden Verfeinerungsschritten überlassen.

Testzwecke (und allgemein Testziele), abgeleitet von den Anforderungen, können dann den einzelnen Testszenarien oder deren Teilen zugeordnet werden. Die Testszenarien dienen als Basis für die Ableitung und Automatisierung von Tests.

Weiterhin kann TDL auch für die Darstellung von Testabläufen, die von MBT (Model-based Testing)-Werkzeugen, System-Simulatoren oder Testausführungsprotokollen abgeleitet sind, eingesetzt werden.

Übersicht über den TDL-Standard

Der TDL-Standard besteht aktuell aus vier Teilen:

- Teil 1 des TDL-Standards beschreibt das Meta-Modell (MM) sowie die zugehörige Semantik für die einzelnen Konzepte der Sprache. Das TDL-Meta-Modell beschreibt die abstrakte Syntax und dient als der Kern der Sprache. Es beinhaltet alle Konzepte der Sprache sowie deren Eigenschaften und deren Beziehungen untereinander.
- Teil 2 definiert die standardisierte grafische Darstellung (GR) der TDL-Konzepte als eine allgemeine konkrete Syntax für TDL mit den entsprechenden Abbildungen der Konzepte auf die Syntax. Weitere konkrete Syntaxdefinitionen können in ähnlicher Weise für spezielle Einsatzzwecke und Nutzer definiert werden, um die relevanten Konzepte des Meta-Modells in besonderer Art und Weise und auf der notwendigen Abstraktionsebene zugänglich zu machen, wie grafisch, textuell, tabellarisch oder baumbasiert.
- Teil 3 beschreibt das TDL-Austauschformat (XF), welches als Grundlage für die Interoperabilität der TDL-Werkzeuge dienen soll.
- Teil 4 umfasst die erste standardisierte Erweiterung von TDL für die strukturierte Spezifikation von Testzwecken (TO). Zusätzlich wurde ein UML-Profil für TDL standardisiert, um der Einsatz von TDL in UML-basierten Arbeitsumgebungen zu erleichtern.

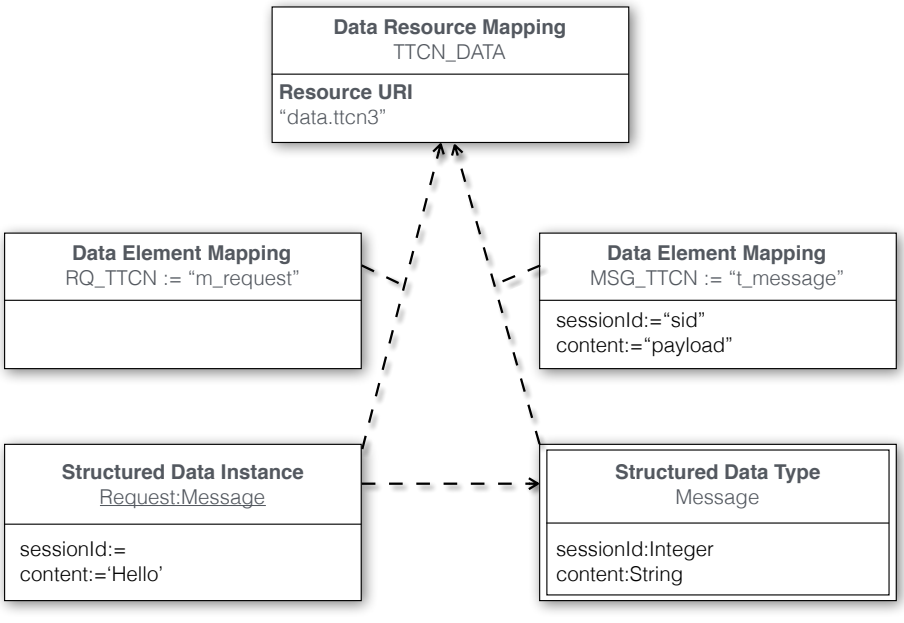

Abb. 2: Datendefinitionen mit TDL.

Abbildung 1 gibt eine Übersicht der Beziehungen zwischen den einzelnen Teilen des TDL-Standards. Die GR und XF, sowie potenzielle benutzerdefinierte Syntaxdarstellungen, bauen auf dem MM auf. Die TO erweitert das MM mit zusätzlichen Konzepten und stellt somit ein erweitertes Fundament bereit. Die zusätzlichen Konzepte werden durch erweiterte Syntaxelemente und die dazugehörigen Abbildungen für GR und XF entsprechend beschrieben. Benutzerdefinierte Syntaxnotationen können die zusätzlichen Konzepte des TO (oder auch Konzepte anderer Erweiterungen) optional ebenfalls abbilden.

Die Unterteilung von TDL in mehrere Teile erlaubt Werkzeugherstellern und Nutzern, je nach Einsatzgebiet zu entscheiden, welche Teile sie implementieren beziehungsweise unterstützen wollen. Zum Beispiel können Werkzeuge, die eine kundenspezifische Syntax implementieren, nur das MM und das XF unterstützen und brauchen somit die GR, wenn diese für die spezifische Kundengruppe nicht relevant ist, nicht zu implementieren.

TDL wurde auf der UCAAT 2015 (User Conference on Advanced Automated Testing) dem breiteren Publikum vorgestellt. Die darauf folgenden Diskussionen mit Stakeholdern aus verschiedenen Einsatzgebieten haben die Weiterentwicklung der Sprache beeinflusst. In näherer Zukunft wird eine Referenzimplementierung von TDL, die als gemeinsame Plattform für die Einführung beim Endnutzer sowie als Baustein für Werkzeughersteller dient, veröffentlicht.

In den folgenden Abschnitten stellen wir die Hauptbestandteile von TDL vor. Für die Definition einer TDL-Spezifikation werden grundsätzlich drei Teile benötigt: Datenspezifikation, Testkonfiguration und Testverhalten. Die Gliederung spiegelt den üblichen Vorgang bei der Definition einer TDL-Spezifikation wider. Abschließend stellen wir kurz die Erweiterung TDL TO vor. Wir verwenden für die Veranschaulichung von Beispielen die TDL GR.

Datenspezifikation

Datenbezogene Konzepte in TDL werden in Datendefinition und Datennutzung untergliedert. Abstrakte Datendefinitionen innerhalb von TDL werden mithilfe von Datentypen und Dateninstanzen definiert. Die interne Struktur der Datentypen und

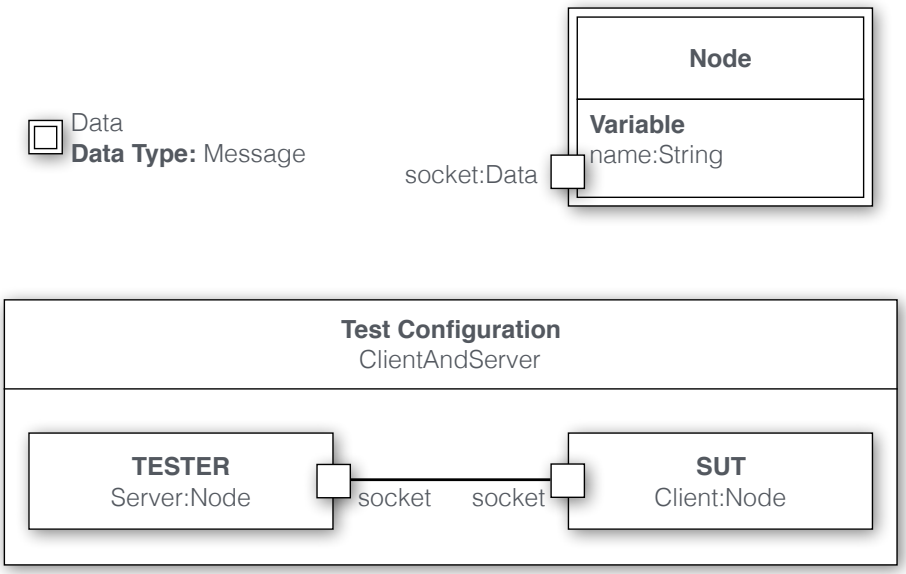

Abb. 3: Testkonfigurationsspezifikation in TDL.

Dateninstanzen kann innerhalb von TDL mittels „Members" und „MemberAssignments" weiter spezifiziert werden.

Datentypen und Dateninstanzen innerhalb von TDL sind abstrakte Symbole, denen dann konkrete Daten außerhalb von TDL mittels sogenannter „DataElementMappings" zugeordnet werden können. Konkrete Daten können in Form von TTCN-3, XML oder anderen Formaten vorliegen, die dann mittels „DataResourceMappings" in TDL eingebunden werden.

Abbildung 2 zeigt ein Beispiel für Datendefinitionen und Data-Mappings. Wir betrachten einen Datentyp ***Message*** und eine Dateninstanz dieses Datentyps namens ***Request***. Typendefinitionen in TDL werden mithilfe doppelter Rahmenlinien dargestellt. Der ***Message***-Datentyp beinhaltet zwei Felder: ***sessionId*** vom Typ Integer und ***content*** vom Typ ***String***. Die Dateninstanz Request definiert eine Zuweisung für das Feld ***content***. Das Feld ***sessionId*** bleibt in diesem Fall unspezifiziert.

Wie bereits erwähnt, stellt TDL nur abstrakte Symbole bereit, denen konkrete Daten außerhalb von TDL zugeordnet werden können. In diesem Fall nehmen wir an, dass eine Datei „data.ttcn3" bereits vorhanden ist, die die konkrete und vollständige Datenspezifikationen in der Sprache TTCN-3 beinhaltet. Diese Datei ist mittels eines „DataResourceMapping" in die TDL-Spezifikation eingebunden.

Die entsprechenden „DataElementMappings" definieren die Beziehungen zwischen den abstrakten Datenelementen innerhalb von TDL und der konkreten Datenspezifikation in der TTCN-3-Datei. In dem Beispiel wird der TDL-Datentyp ***Message*** dem konkreten TTCN-3-Datentypen ***t_message*** zugeordnet, wobei die Felder des abstrakten Datentyps auch den entsprechenden Feldern des konkreten Datentyps zugeordnet sind. In ähnlicher Weise wird die ***Request***-TDL-Dateninstanz dem konkreten TTCN-3-Template ***m_request*** zugeordnet. Die Auflösung und Validierung der Zuordnungen ist den Werkzeugimplementierungen überlassen. Weitere datenbezogene Konzepte in TDL sind Funktionsdefinitionen, Vorgangsbeschreibungen, Parameter und Variablen.

Die so definierten Datenelemente können in verschiedenen Kontexten, wie Interaktionsargumenten und Parameterzuweisungen, referenziert werden. Weiterhin bietet TDL auch die Möglichkeit, teil-spezifizierte Daten mittels sogenannter „Wildcards", zum Beispiel „beliebiger Wert eines Datentyps" (bezeichnet mit Fragezeichen), zu beschreiben.

Testkonfiguration

Testkonfigurationen in TDL beschreiben die notwendige Testarchitektur für die Durchführung eines Tests. Eine Testkonfiguration besteht aus mindestens zwei Komponenten, wobei jeweils mindestens eine Komponente die Rolle „Tester" und mindestens eine Komponente die Rolle „SUT" annehmen muss. Zudem muss mindestens eine Verbindung zwischen einer ***Tester***-Komponente und einer ***SUT***-Komponente vorhanden sein.

Jede Komponente erbt die Anschlüsse, Timer und Variablen ihres Komponententyps. Weiterhin haben alle Anschlüsse einen Anschlusstyp. Der Anschlusstyp beschränkt die Datentypen, die für die Kommunikation über Anschlüsse dieses Typs erlaubt sind. Ein Beispiel für eine Testkonfiguration in TDL ist in Abbildung 3 zu sehen. Zunächst müssen ein oder mehrere Anschlusstypen definiert werden. In diesem Fall definieren wir den Anschlusstyp ***Data***, der die Übertragung von Dateninstanzen des Datentyps ***Message*** unterstützt. Danach definieren wir den Komponententyp ***Node***, der einen Anschluss ***socket*** vom Typ ***Data***, sowie eine Variable ***Name*** vom Typ ***String*** besitzt. Schließlich definieren wir die Testkonfiguration ***ClientAndServer***.

Die Testkonfiguration ***ClientAndServer*** besteht aus einer Komponente ***Server***, die als ***Tester*** dient, und einer Komponente ***Client***, die die Rolle des ***SUT*** annimmt. Beide Testkomponenten sind vom Typ ***Node***. Zusätzlich definieren wir eine Verbindung zwischen den Anschlüssen beider Komponenten.

Testverhalten

Das Testverhalten in TDL wird mittels atomarer Verhaltenskonzepte, wie Interaktionen, Vorgänge (lokal oder global), Aufrufe anderer Testbeschreibungen und expliziter Testurteil-Zuweisungen, spezifiziert. Atomare Verhaltenskonzepte lassen sich mithilfe von zusammengesetzten Verhaltenskonzepten wie Bedingungen, Alternativen, Schleifen und Ausnahmen kombinieren.

In Abbildung 4 ist ein Beispiel für die Spezifikation von Testverhalten in TDL zu sehen. Hierbei handelt es sich um einen einfachen Testablauf, wo zunächst die bereits definierte Testkomponente ***Server*** eine ***Request***-Nachricht an die ***Client***-Komponente schickt, wobei die ***Client***-Komponente die Rolle der ***SUT*** annimmt. Hierbei wird dem Feld ***sessionId*** der Wert 1 zugewiesen.

Die ***Client***-Komponente kann entweder mit einer ***Accept***-Nachricht mit gleicher ***sessionId*** oder mit einer ***Accept***-Nachricht mit beliebiger ***sessionId*** antworten. Die Alternativen werden hierbei in der definierten Reihenfolge ausgewertet und die Testurteile ***PASS*** oder ***FAIL*** entsprechend der ausgeführten Alternative gesetzt.

In diesem Beispiel sind auch zwei Arten von Ausnahmen definiert. Falls die ***Client***-Komponente eine oder mehrere ***Tick***-Nachrichten vor der ***Accept***-Nachricht schicken sollte, wird die Server-Komponente mit einer ***Tock***-Nachricht antworten und weiterhin auf die ***Accept***-Nachricht warten. In diesem Fall sprechen wir von einem „Interrupt"-Ausnahmeverhalten. Falls die ***Client***-Komponente anstatt einer

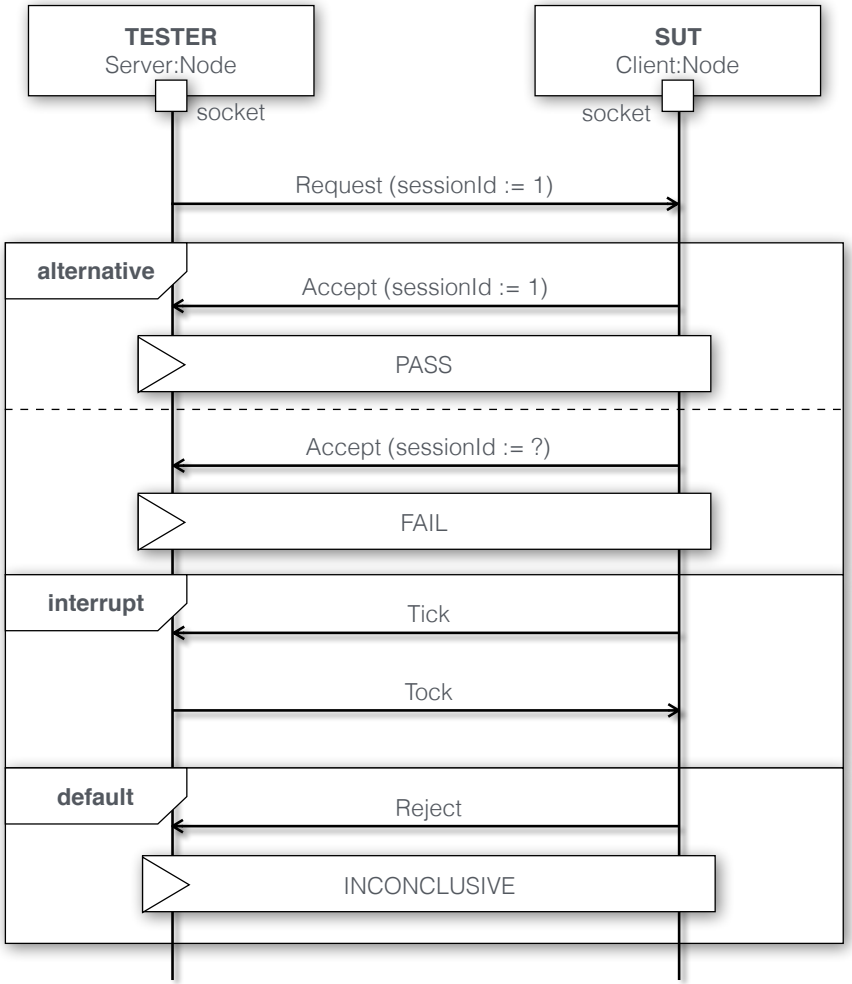

Abb. 4: Testverhaltensspezifikation in TDL.

Strukturierte Testzweckbeschreibungen

Teil 4 des TDL-Standards (TO) definiert eine Erweiterung von TDL, um die formalisierte Spezifikation von Testzwecken zu ermöglichen. Die Testzielbeschreibungen in Teil 1 ermöglichen den Bezug auf externe Dokumente mit Anforderungen und Testzwecken und/oder die Beschreibung des Testziels in Freitext. Teil 4 führt weitere Konzepte ein, die eine strukturierte und formalisierte Beschreibung von Testzwecken ermöglichen. Die strukturierte Testzweckbeschreibung ermöglicht die Validierung und eventuell auch Zuordnung zu Testverhaltensbeschreibungen.

Ein Beispiel für strukturierte Testzweckbeschreibungen ist in Abbildung 5 dargestellt. Dabei werden die Vor- und Nachbedingungen sowie das erwartete Verhalten für den Testzweck mittels strukturierten Texts beschrieben.

Weitere Konstrukte

Mithilfe von Paketen lassen sich zusammenhängende Teile einer TDL-Spezifikation gruppieren. Innerhalb von Paketen können Elemente, die in anderen Paketen definiert sind, importiert werden. Zusätzliche Informationen zu einzelnen Definitionen können im Freitext mittels Kommentaren oder teil-strukturiert mittels Annotationen hinzugefügt werden.

Accept-Nachricht eine ***Reject***-Nachricht schicken sollte, wird das Testurteil auf ***INCONCLUSIVE*** gesetzt und die Alternativen werden nicht weiter betrachtet. In diesem Fall sprechen wir von einem „Default"-Ausnahmeverhalten.

Ausnahmeverhalten beschreiben Situationen, die alternativ auftreten können – wenn eine ***Tester***-Komponente auf eine Nachricht warten muss. Sie können an Testbeschreibungen und zusammengesetzte Verhaltensbeschreibungen angehängt werden.

Zeitaspekte

TDL bietet verschiedene Konzepte zur Beschreibung von zeitlichen Abhängigkeiten zwischen einzelnen atomaren Ereignissen. „Timer" sind den Testkomponenten zugeordnet und können mit Befehlen wie Start, Stop und Timeout kontrolliert werden. Die Zeitoperation Wait beschreibt eine Verzögerung in der Ausführung weiterer Verhaltensspezifikationen und Quiescence beschreibt einen Kommunikations-Stillstand eines Anschlusses oder einer gesamten Testkomponente auf bestimmte Zeit.

TP Id	TP_1
Test Objective	Ensure that a request is accepted
Reference	TDL TO Examples
PICS Selection	

Initial Conditions

```
with {
  the Server entity connected to the Client entity
}
```

Expected Behaviour

```
ensure that {
  when {
    the Client entity receives a Request message
  }
  then {
    the Client entity sends an Accept message
  }
}
```

Final Conditions

Abb. 5: Strukturierte Testzweckbeschreibung in TDL.

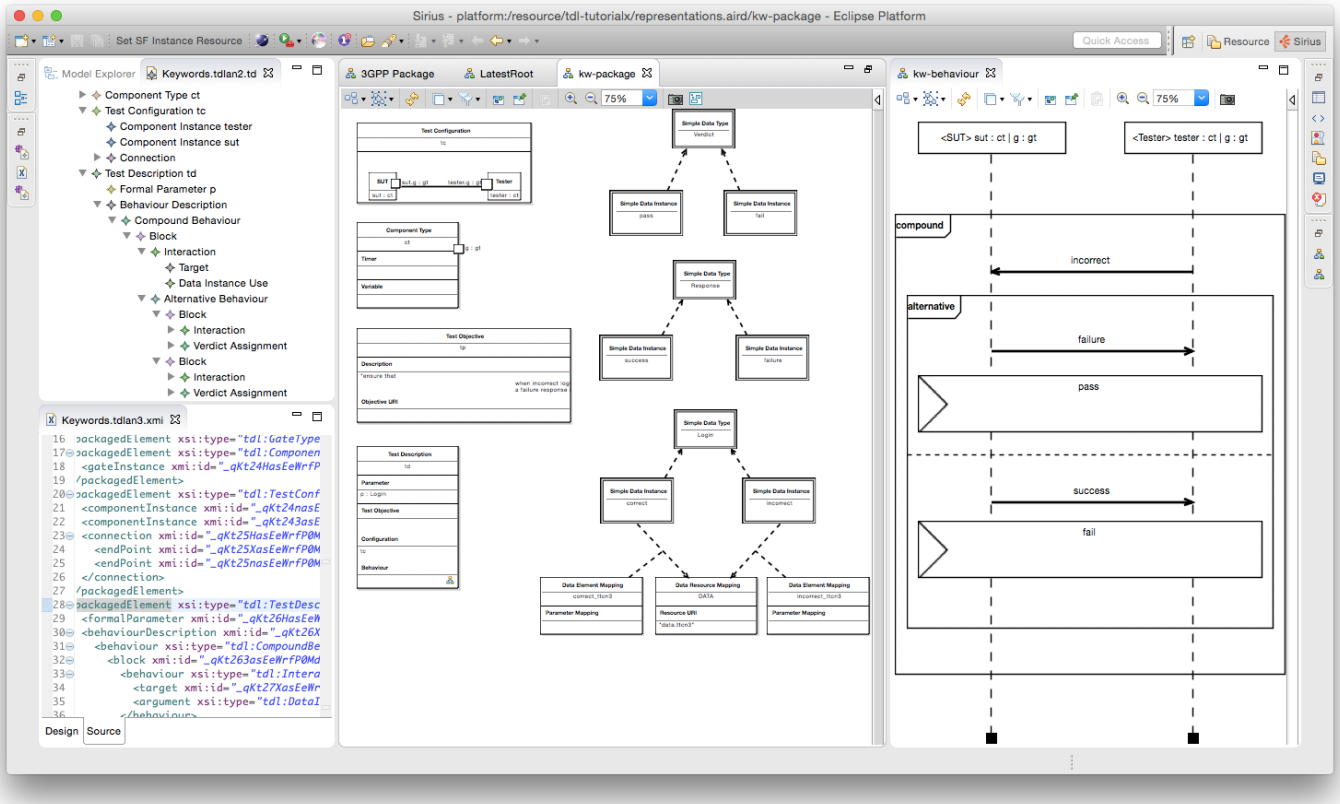

Abb. 6: Referenzimplementierung von TDL.

Fazit

TDL bietet eine formalisierte, modellbasierte Lösung für die Spezifikation von abstrakten Testbeschreibungen. Durch die Abstraktion können Testingenieure sich mehr auf die Definition von Szenarien, die die Testziele abdecken, fokussieren, anstatt sich mit Details zur Testausführung beschäftigen zu müssen. Durch die Unterstützung verschiedener konkreter Syntaxnotationen, die auf die verschiedenen Nutzergruppen zugeschnitten sind, wird das Review durch externe Experten erleichtert.

Aktuell werden bei ETSI erste Pilotversuche, TDL in die Standardisierungsprozesse zu integrieren, durchgeführt. Im Rahmen des Forschungsprojekts AVIO-605 CRIAQ im Bereich Luftfahrtsysteme [BAS2015] wurde TDL bereits für die Generierung von Tests aus Use Case Maps eingesetzt, um proprietäre Lösungen zu ersetzen.

Zukünftig wird unter anderem an einer standardisierten Abbildung zwischen TDL und TTCN-3 gearbeitet, um die normierte Generierung von ausführbaren Testfällen zu ermöglichen. Zusätzlich werden weitere Anpassungsmöglichkeiten und Erweiterungen untersucht und integriert, um den Einsatz von TDL in verschiedenen Domänen und Testarten (u. a. Sicherheits- und Performanztesten) zu erleichtern.

Die Standards sowie weitere Informationen und Materialien zu TDL sind auf der TDL-Webseite tdl.etsi.org zu finden. Die Referenzimplementierung wird zeitnah ebenfalls dort veröffentlicht (siehe Abbildung 6). ■

Literatur

[BAS2015] P. Boulet, D. Amyot, B. Stepien, Towards the Generation of Tests in the Test Description Language from Use Case Map Models, in: SDL 2015: Model-Driven Engineering for Smart Cities, J. Fischer, M. Scheidgen, I. Schieferdecker, R. Reed (Hrsg.), 193–201, LNCS 9369, Springer International Publishing, 2015

[ETSI202553] ETSI ES 202 553: Methods for Testing and Specification (MTS), TPLan: A notation for expressing Test Purposes, v1.2.1, ETSI, 2009

[ETSI203130] ETSI EG 203 130: Methods for Testing and Specification (MTS), Model-Based Testing (MBT), Methodology for Standardised Test Specification Development, v1.1.1, ETSI, 2013

[ISO9646] InformationTechnology – OpenSystemsInterconnection – Conformance testing methodology and framework – Part 1: General Concepts, International ISO/IEC multipart standard No. 9646-1, 1994-1998