TTCN-3

Andreas Schlegel
Hochschule Offenburg
schlegel.andreas@googlemail.com

19.12 .2010

Abstract: Das Testen komplexer Systeme, besonders im Bereich verteilter Systeme, gestaltet sich oft sehr schwierig. Dies ist bedingt durch das Zusammenwirken verschiedener Komponenten mit unterschiedlicher Hardware und Plattformen. Für diese Komponenten werden die selben Testfälle entwickelt, jedoch aufgrund der Unterschiede meistens für jede Komponente einzeln implementiert. Dies verursacht einen erheblichen Mehraufwand, der besonders auch beim Ändern von Testfällen sehr hoch ist. Um diesen Aufwand zu minimieren ist eine gemeinsame Testsprache notwendig, in der alle Testfälle erstellt werden können.

Dieses Paper stellt die Testsprache TTCN-3 vor. TTCN-3 ist eine einheitliche technologieunabhängige Testsprache für kommunikationsbasierte, verteilte Systeme und bietet dadurch eine Lösung für diese Probleme.

1 Einleitung

Die rasante Weiterentwicklung im IT-Bereich, besonders im Bereich der Hardware- und Plattformentwicklung, führt immer mehr zu Schwierigkeiten beim Testen verteilter Systeme. Hierdurch kommt es immer häufiger zum Einsatz von verteilten Systemen mit Komponenten, die sich in Hardware oder Plattform unterscheiden. Auch der Austausch bestehender Komponenten verursacht zusätzlichen Testaufwand, weil meistens aufgrund der schnellen Entwicklung nicht unbedingt die gleichen Komponenten beschafft werden können. Besonders das Testen von Protokollen ist für Gerätehersteller aufgrund der schnell wechselnden Hardware- und Software-Landschaft oft sehr zeitaufwändig, da Tests für jedes neu entwickelte Gerät neu implementiert werden müssen.

TTCN-3 bietet eine einheitliche, technologieunabhängige Testsprache, wodurch die Testfälle nicht bei Hardware- oder Plattformänderungen angepasst werden müssen. Das Framework ist Modular aufgebaut, wodurch nur noch geringfügige Änderungen in sogenannten Adaptern durchgeführt werden müssen, welche jedoch unabhängig von den Testfällen implementiert werden. Zuletzt muss die Testsuite lediglich für das Zielsystem erstellt werden. Dadurch lassen sich Testfälle einmalig definieren und bei Wechsel von Komponenten müssen nicht die Testfälle, sondern lediglich die Module, die von der Plattform oder Hardware abhängen, angepasst werden. Sind diese Module einmal implementiert, können sie für gleiche Komponenten ohne Änderungen übernommen werden.

Durch Verwenden von TTCN-3 können Fehler vermieden werden, da bestehende Testfälle nicht neu implementiert werden müssen, sondern weiter verwendet werden können. Besonders beim Ändern vorhandener Testfälle ist es lediglich notwendig, die Änderungen einmal in einer gemeinsamen Testsuite vorzunehmen und nicht die Testfälle in allen hardwarespezifischen Projekten einzeln zu ändern.

In den folgenden Kapiteln wird auf die Entwicklung von TTCN-3, seine Struktur, die Darstellungsformen der Core Language und Einsatzbereiche eingegangen und zuletzt Trends aufgezeigt.

2 Historie und Entwicklung

Die Entwicklung von TTCN-3 begann 1984, wobei lediglich eine Testsprache für Konformitätstests für OSI-Protokollimplementierungen vorgesehen war (TTCN). Erst mit der Standardisierung von TTCN-3 entstand daraus eine technologieunabhängige Sprache, die für das Testen jeglicher Art reaktiver, verteilter Systeme geeignet und nicht auf Kommunikationssysteme beschränkt war.

2.1 TTCN

TTCN wurde 1992 als "Tree and Tabular Combined Notation" ursprünglich als Teil des ISO-Standards 9646 Open Systems Interconnection (OSI) – Conformance Testing Methodology and Framework speziell für die Spezifikation von Testfällen für das Konformitätstesten von OSI-Protokollimplementierungen standardisiert [8]. TTCN wurde nur für das Testen von Kommunikationssystemen, vor allem zum Testen von GSM (Global System for Mobile Communications), dem weltweit populärsten Mobilfunkstandard verwendet.

2.2 TTCN-2

TTCN-2 wurde im Jahr 1997 durch die ETSI (European Telecommunication Standards Institute) zertifiziert und von der International Telecommunication Union (ITU-T) angenommen. Die wichtigsten Neuerungen in dieser Version waren:

- Modularisierung
- Parallele Tests
- Teilweise Unterstützung von ASN.1 (Abstract Syntax Notation One) als Schnittstelle

TTCN-2 wurde für die Unterstützung weiterer Kommunikationsprotokolle erweitert. Außer GSM konnten auch die Protokolle ISDN, DECT und INAP mit TTCN-2 getestet werden.

2.3 TTCN-3

Im Jahr 2000 folgte TTCN-3, welches auf Nachfrage führender Hersteller der Telekommunikationsindustrie von der ETSI und der International Telecommunication Union (ITU-T) entwickelt und von der ETSI zertifiziert wurde. Der Name wurde von "Tree and Tabular Combined Notation" in "Testing and Test Control Notation" geändert, da ab dieser Version nicht mehr nur eine tabellarische Darstellung der Sprache ermöglicht wurde, sondern auch eine grafische. Die Abkürzung blieb erhalten. Diese Version brachte gravierende Änderungen mit sich.

Die größte Änderung in dieser Version war, dass diese nicht mehr nur speziell auf das Testen von Telekommunikationssystemen ausgelegt war. TTCN-3 ermöglicht das Testen jeglicher Art von reaktiven, verteilten Systemen mit wohldefinierten Schnittstellen [13]. Hierdurch wurden neue Testarten, wie funktionale Tests, Leistungstests, Skalierungstests, Interoperabilitätstests und inzwischen auch Lasttests, Performance Tests und Real-Time Tests ermöglicht. [12]. Ab dieser Version wurde außerdem nicht nur nachrichten-orientierte, sondern auch prozedurale Kommunikation unterstützt, wodurch TTCN-3 noch universeller einsetzbar wurde.

Weitere Änderungen sind:

- Änderungen an Syntax und Semantik (skriptähnlich)
- Synchrone und asynchrone Kommunikation möglich
- Unterstützung von ASN.1 und IDL (Interface Definition Language) als Schnittstellen
- XML und WSDL zum Testen von Webservices
- Testdefinition über skriptähnliche Sprache oder grafisch möglich
- Definierte Schnittstellen und Module

Die Hauptkomponenten von TTCN-3 sind in einzelnen "Parts" in separaten Dokumenten spezifiziert und versioniert. Außerdem sind Erweiterungen vorhanden, welche auch in separaten Dokumenten spezifiziert wurden.

Nachfolgend sind die Hauptkomponenten von TTCN-3 aufgeführt:

- Part1 Core Language
- Part2 Tabular presentation Format (TFT)
- Part3 Graphical presentation Format (GFT)
- Part4 Operational Semantics
- Part5 Runtime Interface (TRI)
- Part6 Control Interface (TCI)
- Part7 Using ASN.1 with TTCN-3
- Part8 The IDL to TTCN-3 Mapping
- Part9 Using XML schema with TTCN-3
- Part10 Documentation Comment Specification

Zuletzt wurde die Version 4.2.1 des Standards im Juli 2010 aktualisiert. Alle Hauptkomponenten der Spezifikation befinden sich auf diesem Stand, außer Part2 und Part3, welche sich noch auf Version 3.2.1 (Stand 2007) befinden.

3 Struktur von TTCN-3

TTCN-3 ist in verschiedene Module gegliedert, bei denen einige vom zu testenden System (SUT) oder von der verwendeten Plattform, auf der das TTCN-3 Executable ausgeführt wird, abhängen. Diese müssen beim Wechsel der Plattform oder des zu testenden Systems angepasst werden. Die Kommunikation zwischen den einzelnen Modulen ist über definierte Schnittstellen festgelegt.

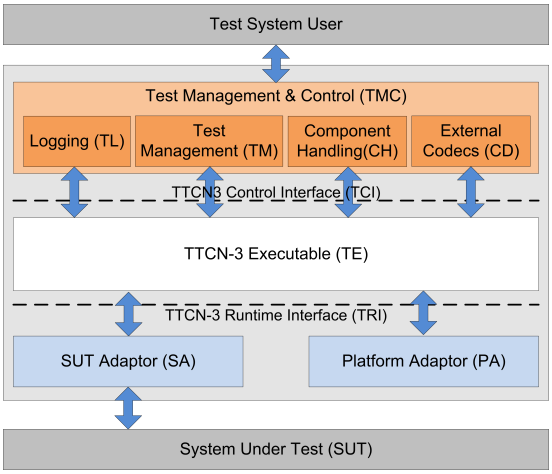

Abb. 1: Modulübersicht TTCN-3[6][13]

Abbildung 1 zeigt einen Überblick über die Struktur von TTCN-3 mit allen Modulen und den dazwischen liegenden Schnittstellen.

3.1 Module

TE (TTCN-3 Executable)
Das TE ist für die Ausführung des TTCN-3-Code verantwortlich. Das TE kann ein ausführbares Programm sein oder über eine Laufzeitumgebung realisiert werden, da dies im Standard nicht festgelegt ist. Dieses Modul ist die zentrale Instanz von TTCN-3, die über spezifizierte Schnittstellen mit allen anderen Modulen verbunden ist.

TM (TTCN-3 Management)
Im TM werden sämtliche Tests, die ausgeführt werden sollen registriert, gestartet und zuletzt deren Ergebnis gespeichert.

TL (TTCN-3 Logging)
Das TL ist für die Anpassung der Logausgaben zuständig. In diesem Modul können alle Ereignisse der Laufzeitumgebung festgelegt werden, für die ein Logeintrag erstellt werden soll. Außerdem ist es in diesem Modul möglich die Formatierung der Ausgabe der Logdaten zu beeinflussen.

CH (Component Handling)
Das CH ist für die Kommunikation der einzelnen Testkomponenten untereinander zuständig. Da diese auch auf verschiedenen Testgeräten verteilt sein können, muss mit dem CH die Kommunikation zwischen den einzelnen Komponenten gesteuert werden.

CD (Coding and Decoding)
Das CD ist verantwortlich für das Decodieren und Encodieren von Daten bei nachrichten- oder prozedurbasierter Kommunikation mit dem SUT. Diese ”externen Codecs” sind optional. Bei Verwendung können diese parallel oder anstatt der im TE vorhandenen Codecs verwendet werden. Der Vorteil der externen Codecs ist die Portabilität zwischen verschiedenen TTCN-3-Systemen und -Tools aufgrund der Verwendung des standardisierten TTCN-3-Control Interface [7].

PA (Plattform Adapter)
Der PA ist für die Anpassung verwendeter Systemfunktionen, wie z.B. Timer, an die verwendete Plattform zuständig. Er muss beim Wechsel der Plattform angepasst werden.

SA (SUT-Adapter)
Das SA ist für die Kommunikation mit dem SUT zuständig. Es implementiert Funktionen des TRI (TTCN-3 Runtime Interfaces) für die nachrichten- und prozedurbasierte Kommunikation mit dem SUT. Dieses Modul muss bei Änderungen am SUT angepasst werden.

3.2 Schnittstellen

TTCN-3 Runtime Interface (TRI)
Das TTCN-3 Runtime Interface bietet Funktionen um ein Test System an die verwendete Plattform und das SUT anzupassen. Hierzu werden verschiedene Funktionen zur Kommunikation mit dem SUT und zum Anpassen plattformspezifischer Komponenten, wie z.B. Timern, zur Verfügung gestellt. [6]

TTCN-3 Control Interface (TCI)
Das TTCN-3 Control Interface bietet Funktionen zur Anpassung des Test Management, Component Handling und Encoding and Decoding an eine bestimmte Test-Plattform an. [7]

3.3 Vorteile der Struktur

Durch die Spezifikation der Module und Schnittstellen wird die Implementierung und der Einsatz von TTCN-3-basierten Tools erleichtert. Dies ermöglicht es den Toolherstellern TTCN-3 flexibler zu implementieren. Für Programmierer, welche die Tools einsetzen, ermöglicht es einen größeren Freiraum durch Implementierung eigener Codecs und Möglichkeiten für spezifische Anpassungen der Adapter.

Beim Wechsel von Plattform oder SUT müssen lediglich die ”Adapter” angepasst werden. Sind Module für bestimmte Plattformen oder SUT vorhanden, können diese bei Verwendung gleicher SUT oder Plattformen wiederverwendet werden. Falls für die Kommunikation mit dem SUT spezielle externe Codecs verwendet werden, muss zusätzlich das Modul Coding and Decoding implementiert werden.

Durch die Schnittstellen-Spezifikation ist es möglich, dem Kunden die Implementierung der Adapter-Module sowie des Moduls für die externen Codecs selbst zu überlassen. Dies ermöglicht SUT- und plattformunabhängige Testsuiten, da die Anpassung auf die Bedürfnisse des Kunden beim Kunden selbst stattfindet. Es besteht jedoch auch die Möglichkeit für bestimmte Codecs, Plattformen und SUT fertige Implementierungen anzubieten.

Ein Toolhersteller, der keine spezifischen Module bereitstellt, muss lediglich die Module TTCN-3 Executable, TTCN-3 Management, TTCN-3 Logging und Component Handling vollständig implementieren. Für die Module Coding and Decoding, Plattform Adapter und SUT-Adapter muss dem Kunden ein einfaches Interface bereitgestellt werden.

Die Modularisierung ermöglicht die Ausführung der Tests für unterschiedliche SUT und Plattformen in einer gemeinsamen Testsuite. Bei Wechsel der Plattform oder des SUT müssen dadurch lediglich die Adapter angepasst werden. Ein Anpassen oder erneutes Implementieren der Testfälle ist dadurch nicht mehr notwendig. [6]

4 Core Language

Die Sprache TTCN-3 ist eine reine Testsprache und eignet sich besonders zum Testen von reaktiven Systemen, wie eingebetteten Systemen, Echtzeitsystemen, sowie Kommunikationsprotokollen. Sie bietet verschiedene definierte Schnittstellen, wie ASN.1, IDL, XML und WSDL für das Einbinden externer Datendefinitionen und verschiedene Darstellungen mit denen ein Benutzer Testfälle erstellen kann (Abbildung 2).

Es gibt drei vordefinierte, standardisierte Darstellungen, die TTCN-3 dem Benutzer zur Erstellung von Tests zur Verfügung stellt.

- Text-Format (Core Notation)
- Tabellarisches Format
- Grafisches Format

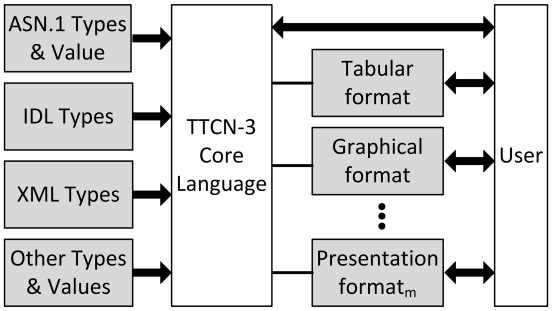

Abb. 2: Übersicht Core Language[4]

Zusätzlich zu den im Standard spezifizierten Formaten, können Hersteller weitere Darstellungsformate bereitstellen. Diese müssen sich lediglich in das Text-Format konvertieren lassen.

Nachfolgend werden die einzelnen Formate erläutert und anhand des ”altstep”-Konstrukts von TTCN-3 beispielhaft die Umsetzung in die verschiedenen Formate aufgezeigt.

4.1 Text-Format (Core Notation)

Das Text-Format, die sogenannten Core Notation, ist die Basissprache von TTCN-3 und wird von den drei Darstellungsformaten am häufigsten eingesetzt. Die Core Notation unterstützt alle Konstrukte der Core Language. Alle Darstellungsformate müssen zunächst in die Core Notation überführt werden, da diese vom Compiler bzw. Interpreter als Eingabesprache verarbeitet wird.

Das Beispiel in Listing 1 zeigt ein altstep-Konstrukt von TTCN-3. Bei Aufruf dieses Konstrukts wird auf das Eintreffen eines der drei erwarteten Ereignisse gewartet und darauf reagiert:

- **Ereignis1**: Eintreffen einer Nachricht vom Typ ”MyMessageOne” von PC1 mit der Bedingung ”x < 5”
  Aktion : Testergebnis auf ”bestanden” setzen, Neuberechnung von x
- **Ereignis2**: Eintreffen einer beliebigen Nachricht von PC2
  Aktion : Wiederholen des altstep
- **Ereignis3**: Ablaufen des Timers T1
  Aktion : Testergebnis auf ”fehlgeschlagen” setzen

Listing 1: altstep Text-Format

```
1  altstep MyAltstep() runs on
       MyComponentType
2  {
3     [x<5] PC1Port.receive(MyMessageOne}
4     {
5        setverdict(pass);
6        x := 7 + 5;
7     }
8     [] PC2Port.receive()
9     {
10       repeat;
11    }
12    [] T1.timeout
13    {
14       setverdict(fail);
15    }
16 }
```

4.2 Tabellarische Darstellung

Die tabellarische Darstellung ist das älteste Darstellungsformat von TTCN-3. Es wurde hauptsächlich aus Kompatibilitätsgründen zur Vorgängerversion, für die einfachere Migration von TTCN-2 zu TTCN-3 spezifiziert und hat keine praktische Relevanz mehr [12][13]. Dieses Darstellungsformat prägte auch den Namen von TTCN ”Tree and Tabular Combined Notation”. Jedoch ist die Verwendung im Gegensatz zur Core Notation sehr aufwändig.

Altstep			
Name	MyAltstep		
Group			
Runs On	MyComponentType		
Purpose			
Comment			
Local Def Name	**Type**	**Initial Value**	**Comment**
Behaviour			

[x<5] PC1Port.receive(MyMessageOne) {
setverdict(pass);
x := 7 + 5;
}
[] PC2Port.receive() {
repeat;
}
[] T1.timeout {
setverdict(fail);
}			

Detailed Comments			

Abb. 3: Tabellarische Darstellung [2]

Abbildung 3 zeigt die tabellarische Darstellung des altstep-Beispiels von Listing 1 (Abschnitt 4.1).

4.3 Grafische Darstellung

Die Grafische Darstellung basiert auf Message Sequence Charts (MSC), ähnlich zu UML-Sequenzdiagrammen.

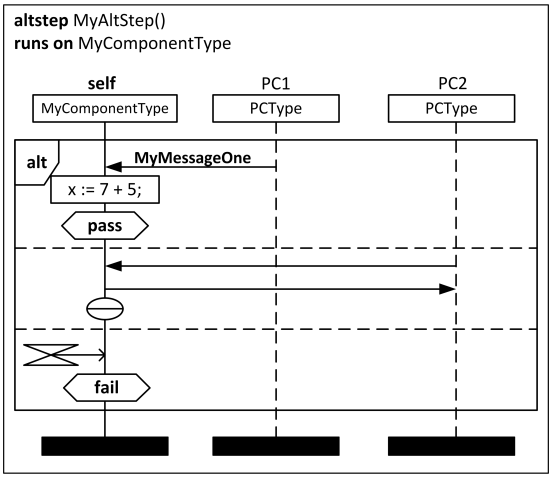

Abb. 4: Grafische Darstellung [3]

Abbildung 4 zeigt die grafische Darstellung des altstep-Beispiels von Listing 1 (Abschnitt 4.1).

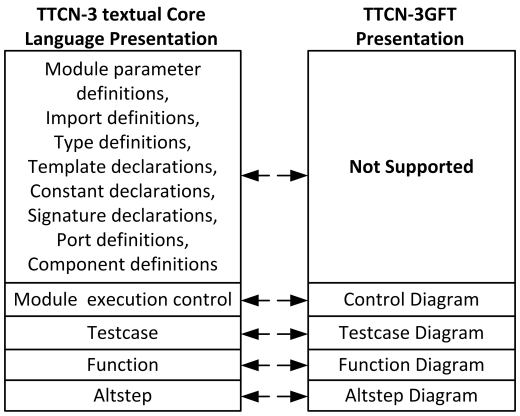

Abb. 5: Von GFT unterstützte Konstrukte [11]

Mit der grafischen Darstellung sind jedoch nicht alle Konstrukte der Core Language darstellbar. Diese müssen in der Core Notation erstellt werden. Abbildung 5 veranschaulicht, welche Konstrukte nicht unterstützt werden. In der grafischen Darstellung lassen sich lediglich Konstrukte visualisieren, die mit dem Testablauf verknüpft sind. Dies sind die Module Execution Control, Testcase, Function und Altstep. Daten-Konstrukte, wie Deklarationen, Definitionen und Parameter, werden somit nicht unterstützt.

Durch zusätzliche Elemente in TTCN-3-Testsuiten kann dies jedoch ermöglicht werden. Hierfür müssen jedoch zusätzliche grafische Elemente zum Erstellen von Datentypen bereitgestellt werden. Dadurch kann ein Benutzer der Testsuite ohne Kenntnisse der Core Notation auch alle notwendigen nicht unterstützten Konstrukte erstellen.

5 Einsatzbereiche

Seit der Veröffentlichung von TTCN-3 haben sich auch dessen mögliche Einsatzbereiche vergrößert. Einige der Einsatzbereiche sind Automotive, Grid Computing, IEEE Protokolle, Medical Systems, Real-Time Testing, HiL-Testing, MBT, Safety Critical Systems, Telekommunikationsprotokolle und Web Services. Im Anschluss wird auf die Themen Automotive, HiL-Testing und MBT eingegangen.

5.1 Automotive

Die Qualitätssicherung von softwareintensiven Systemen in der Automobilindustrie hatte bisher einen geringen Anteil automatischer Tests und eine Vielzahl an oft proprietären Testsystemen und -plattformen.

Im Gegensatz dazu steht das Automotive Open Systems Architecture (AUTOSAR) Konsortium. AUTOSAR strebt die Standardisierung einer Softwarearchitektur und -plattform für die Automobilindustrie an und hat sich dort inzwischen etabliert. Bei einem größeren Pilotprojekt entschied AUTOSAR sich für die funktionalen Tests seiner Basissoftwarekomponenten TTCN-3 zu verwenden. Seit Release 4 2009 werden zudem Echtzeit-Anforderungen sowie Timing unterstützt. [9]

AUTOSAR verwendet die Mappingschnittstelle der TTCN-3 Core Language (Abbildung 6), um eigene Datentypen, Werte und Schnittstellen einzubinden.

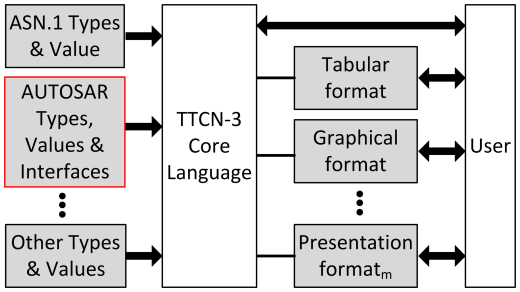

Abb. 6: TTCN-3 AUTOSAR Mapping [10]

Für die Spezifikation der Komponenten werden im Automotive-Bereich verbreitet Message Sequence Charts, teilweise sogar TTCN-3-GFT verwendet. Da TTCN-3 in der grafischen Darstellung Message Sequence Charts anzeigt, kann ein theoretischer Entwurf einer Testsequenz schnell in die Praxis umgesetzt und gegen ein Steuergerät oder eine Simulation getestet werden. Die Erkenntnisse aus der Praxis können dann sofort wieder in die Spezifikationsphase mit einfließen, wodurch die Entwicklung beschleunigt wird.

Einzelheiten zur Verwendung von TTCN-3 bei AUTOSAR können dem Abschnitt 5.2 in den Teilabschnitten ”Testen von Einzelkomponenten” und ”Integrationstests für Komponenten” entnommen werden.

5.2 HiL-Testing

Beim Hardware in the Loop-Testing wird das System, in dem die zu entwickelnde Komponente arbeitet, komplett oder teilweise simuliert. Dadurch lassen sich Tests an realen Systemen stark verringern, da diese zunächst in der simulierten Umgebung ablaufen. Dies hat auch eine Verringerung der Entwicklungszeit und damit der Entwicklungskosten zur Folge.

TTCN-3 ist für Hardware in the Loop-Testing sehr gut geeignet, da durch das komponentenbasierte Konzept lediglich reale Komponenten durch TTCN-3-Testkomponenten ersetzt werden müssen. Für die Testkomponente muss hierbei nur das Kommunikationsverhalten der realen Komponente, die ersetzt werden soll, implementiert werden.

Ein Beispiel für Hardware in the Loop-Testing wird nachfolgend, anhand der AUTOSAR Softwareplattform im Automotive Bereich erläutert. AUTOSAR verwendet TTCN-3-Testkomponenten, um die Gesamtfunktionalität von Einzelkomponenten oder deren Integration in das System zu testen.

Testen von Einzelkomponenten
Um individuelle Komponenten zu Testen, müssen diese vom System vollständig separiert werden. Dies geschieht bei AUTOSAR durch Verwenden eines sogenannten Virtual Functional Bus, welcher selbst durch eine Testkomponente von TTCN-3 ersetzt werden kann (Abbildung 7). Durch diese Maßnahme können alle Funktionen der Komponente über die Testkomponente getestet werden.

Abbildung 7 zeigt einen Controller (C1), dessen Virtual Functional Bus durch eine speziell auf den Controller angepasste Testkomponente ersetzt wurde. Die Testkomponente kann dadurch sämtliche Funktionen des Controllers testen. Außerdem ist für diese Testart kein Testsystem mehr notwendig, da nur der Controller und die Testkomponente benötigt werden.

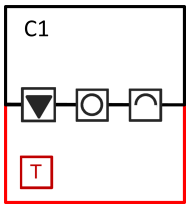

Abb. 7: Testen einer Einzelkomponente [10]

Integrationstests für Komponenten
Ein System kann aus einer Vielzahl anderer Komponenten bestehen. Um individuelle Komponenten, die innerhalb eines Systems integriert werden müssen, zu testen, wird entweder jede Komponente oder nur einzelne Komponenten des Systems durch TTCN-3-Testkomponenten ersetzt. Durch diese Vorgehensweise kann die zu testende Komponente über die TTCN-3-Testkomponenten getestet werden. Jede Testkomponente kann nach Ausführen der Tests eine Aussage darüber Treffen, ob die Integration mit dieser Komponente funktioniert hat oder ein Fehler aufgetreten ist.

Abbildung 8 zeigt ein Testsystem mit einem zu testenden Controller (C1), einem weiteren Controller (C2) und einer Testkomponente (Tester C1), um die Zusammenarbeit mit dem Controller C1 zu testen.

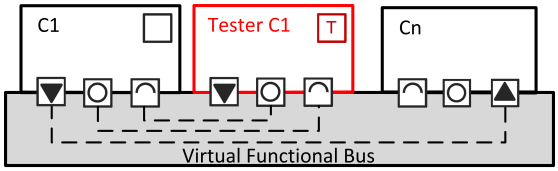

Abb. 8: Integrationstest [10]

5.3 Model Based Testing (MBT)

Testerzeugung aus MSC oder GFT
Wie in Abschnitt 5.1 schon erklärt, werden im Automotive-Bereich verbreitet Message Sequence Charts (MSC), teilweise sogar TTCN-3-GFT zur Spezifikation der Komponenten eingesetzt, welche auch zur Erstellung von Testfällen für TTCN-3 verwendet werden können. Dies ermöglicht direkt das Testen des in der Spezifikation erstellten Modells in den Testfällen. Weiterhin ermöglicht es ein einfaches Vergleichen der Testergebnisse mit der Spezifikation.

U2TP Mapping zu TTCN-3
Für das Testen von Modellen in Form von UML-Diagrammen ist UML 2.0 Testing Profile (U2TP) vorhanden, mit dem direkt über UML-Modelle Testfälle erzeugt werden können. Wie [15] aufzeigt ist es möglich U2TP auf TTCN-3 zu mappen, wodurch das Testen von UML-Modellen über TTCN-3 ermöglich wird. Das Mapping wird anhand des grafischen Formats, sowie der Mappingschnittstelle von TTCN-3 vorgenommen.

U2TP setzt jedoch auf einem höheren Abstraktionslevel wie TTCN-3 an. TTCN-3 bietet gegenüber U2TP den Vorteil, Testfälle im grafischen Format detaillierter darzustellen. Jedoch ist es in TTCN-3 nicht möglich auf die abstraktere Sicht, wie sie U2TP bietet, umzuschalten.

Die Software ”TTmodeler” von TestingTech, ein Plugin für die ”TTworkbench” von TestingTech, ermöglicht es aus UML-2-Diagrammen im XMI-Format TTCN-3-Code zu erzeugen und in der TTworkbench weiter zu verarbeiten.

Motorola verwendet wie TestingTech U2TP um aus UML-2-Diagrammen TTCN-3-Code zu erzeugen und diesen in ihrer eigenen Umgebung weiter zu verwenden [1].

6 Trends

Seit der Standardisierung von TTCN-3 wird die Testsprache immer häufiger in großen Projekten in verschiedenen Industriebereichen als offizielle Testsprache eingesetzt. Einige dieser Bereiche sind Mobile Kommunikation, Breitbandtechnologien, Internet Protokolle und Automotive.

Zu den größten Projekten in diesen Bereichen, bei denen TTCN-3 verwendet wurde zählen:

- WIMAX (802.16)
- UMTS (Universal Mobile Telecommunications System)
- LTE (3GPP Long Term Evolution)
- IPv6 (Internet Protokoll v6)
- SIP (Voice Over IP with the Session Initiation Protocol)
- IMS (IP Multimedia Subsystem)
- MOST (Media Oriented Systems Transport-Bus)
- AUTOSAR

Es soll versucht werden die Anwendungsbereiche von TTCN-3 noch zu vergrößern, um zusätzliche Industriebereiche anzusprechen. Als neue Bereiche sollen Interoperabilitätstests für Betriebssysteme, das Testen großer Webanwendungen und Bank Systeme, intelligente Transportsysteme, Medical Systems, eHealth sowie der Echtzeitbereich für Aviation und Automotive erschlossen werden.

Die größten Themen aktuell sind TTCN-3 für embedded Systems und Model Based Testing mit TTCN-3:

Für den embedded Systems Bereich geht es vor allem um die Entwicklung einer Echtzeiterweitung ”RT TTCN-3 Concepts” und einer Erweiterung zur Unterstützung kontinuierlicher Signale ”Continuous TTCN-3 Concepts”. Diese sollen

Konstrukte für das Testen dieser Systeme bereitstellen. Die Erweiterung "Continuous TTCN-3 Concepts" soll z.B. Konstrukte zum Programmieren von Zustandsautomaten, wie onentry und onexit, zur Verfügung stellen.

Für Model Based Testing mit TTCN-3 sind inzwischen einige kommerzielle Tools oder Plugins, wie "Conformiq OY" von Conformiq Qtronic und das in Abschnitt 5.3 erwähnte Plugin "TTmodeler" von TestingTech vorhanden, die TTCN-3-Code aus UML-Diagrammen generieren.
Außerdem gibt es Überlegungen die Module TE, SA und CD vollständig oder teilweise automatisch aus einem Modell generieren zu lassen. Dies würde die Handhabung von TTCN-3 deutlich vereinfachen.

7 Fazit

TTCN-3 wird inzwischen seit seiner Standardisierung in vielen Industriebereichen verwendet und hat sich in einigen als Standardtestsprache etabliert. Aufgrund der schnellen Weiterentwicklung und der Einsatzfähigkeit in immer mehr Industriebereichen wird sich diese Entwicklung noch ausweiten.

Literatur

[1] BAKER, P. ; JERVIS, C.: Early UML Model Testing using TTCN-3 and the UML Testing Profile. In: *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION, 2007. TAICPART-MUTATION 2007*, 2007, S. 47 –54

[2] ETSI: *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 02: TTCN-3 Tabular presentation Format (TFT)*. 3.2.1, 2007. – URL `http://www.ttcn3.org/StandardSuite.htm`

[3] ETSI: *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 03: TTCN-3 Graphical presentation Format (GFT)*. 3.2.1, 2007. – URL `http://www.ttcn3.org/StandardSuite.htm`

[4] ETSI: *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 01: TTCN-3 Core Language*. 4.2.1, 2010. – URL `http://www.ttcn3.org/StandardSuite.htm`

[5] ETSI: *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 04: TTCN-3 Operational Semantics*. 4.2.1, 2010. – URL `http://www.ttcn3.org/StandardSuite.htm`

[6] ETSI: *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 05: TTCN-3 Runtime Interface (TRI)*. 4.2.1, 2010. – URL `http://www.ttcn3.org/StandardSuite.htm`

[7] ETSI: *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 06: TTCN-3 Control Interface (TCI)*. 4.2.1, 2010. – URL `http://www.ttcn3.org/StandardSuite.htm`

[8] GRABOWSKI, Jens ; SCHMITT, Michael: TTCN-3 - A Language for the Specification and Implementation of Test Cases. (03.2002)

[9] GROSSMANN, J. ; SERBANESCU, D. ; SCHIEFERDECKER, I.: Testing Embedded Real Time Systems with TTCN-3. In: *Software Testing Verification and Validation, 2009. ICST '09. International Conference on*, 2009, S. 81 –90

[10] GROSSMANN, Jürgen ; SCHIEFERDECKER, Ina: TTCN-3 User Conference France: Mapping AUTOSAR Interfaces to TTCN-3, 06.2009

[11] KUMAR, B. ; JAEGER, M. ; JASPERNEITE, J.: A proposal for graphical extension

of TTCN-3 Graphical presentation Format (GFT). In: *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*, Mai 2010, S. 177 –180

[12] VASSILIOU-GIOLES, Theofanis ; PROF. DR.-ING. SCHIEFERDECKER, Ina: Testen mit TTCN-3: Vorteile und Nutzen einer mächtigen und zudem standardisierten Testtechnologie, 2007

[13] WEHRSPANN, Thomas: *Konzeption und Implementierung einer automatisierten Testumgebung*, Technische Universität Clausthal, Diplomarbeit, 2003

[14] WILLCOCK, Colin ; DEISS, Thomas ; TOBIES, Stephan ; SCHULZ, Stephan ; KEIL, Stefan ; ENGLER, Federico: *An Introduction to TTCN-3*. 1. Wiley & Sons, 2005. – URL `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470012242.html`. – ISBN ISBN-10: 0470012242

[15] XI, Liang ; XINXIN, Sun: Research on the Mapping of UML2.0 Testing Profile to TTCN-3. In: *Information and Computing (ICIC), 2010 Third International Conference on* Bd. 4, 2010, S. 280 –283