

# **Erweiterung bewährter Teststrategien zur Erfüllung zukünftiger Testanforderungen in der Automobilbranche**

*Extension of established automotive testing strategies for fulfillment of prospective testing requirements*

Dipl.-Inf. (FH) Susanne Schwarzkopf, C&S group, Wolfenbüttel

## **Zusammenfassung**

Die stetige Weiterentwicklung der Elektronik und Elektrik im Fahrzeug brachte neben neuen Funktionen auch eine Vielzahl von Spezifikationen hervor. Mit dem Entwickeln von Spezifikationen verfolgt die Automobilindustrie das Ziel, eine Interoperabilität von Implementierungen verschiedener Hersteller und mittlerweile darüber hinaus deren Austauschbarkeit zu gewährleisten.

Für die Kommunikation zwischen den diversen Steuergeräten eines Kraftfahrzeugs haben sich mehrere formal spezifizierte Systembusse etabliert. Doch auch für die Softwarearchitektur der Steuergeräte entwickelte sich der Bedarf nach formaler Spezifikation. Spezifikationen beider Arten bedingen den Einsatz von Konformitätstests zur Gewährleistung der Interoperabilität und gegebenenfalls Austauschbarkeit ihrer jeweiligen Implementierungen. Doch obwohl die Ansprüche an das Testen von Hardware anders geartet sind als die an das Testen von Software, sollte man Hardware und Software eines eingebetteten Systems nicht voneinander getrennt betrachten.

Es ist Ziel dieses Dokuments ein Testframework darzustellen und zu erläutern, das das kombinierte Testen der Steuergerätesoftware und –kommunikationsmechanismen erlaubt. Dafür wird nach der Einführung in die Theorie der Konformitätstests ein bewährtes Verfahren für das Testen automotiver Kommunikationsprotokolle vorgestellt. Die innerhalb dieses Verfahrens angewandte Architektur wird erweitert und aus ihr das erwähnte Testframework abgeleitet. Mit Hilfe dieses Frameworks sind die Schnittstellen und die Funktionalität der Software eingebetteter Systeme auf deren Zielplattform testbar. Es wird gezeigt, wie darüber hinaus die externe Kommunikation der Zielplattform und deren Umgebungsbedingungen in die Testausführung einbezogen werden kann.

# 1 Einführung

Die Automobilbranche ist geprägt von schnellem Wandel, Wettbewerb und dem Wunsch vielfältigen Kundenwünschen zu entsprechen oder diese zu wecken. 90 Prozent aller Innovationen basieren überwiegend auf Mikroelektronik und Software. Doch neben neuen Funktionen die durch den vermehrten Einsatz von Elektronik ermöglicht wurden, ergaben sich auch Qualitätsprobleme. Für die Automobilhersteller und Zulieferer wurden Qualität und Zuverlässigkeit der immer komplexer werdenden vernetzten Steuergeräte zu einem beherrschenden Thema. Die aktuelle Herausforderung besteht nun darin, innovative und gleichzeitig zuverlässige Produkte zu entwickeln. Dies kann nur durch verbesserte Organisation und Prozesse erreicht werden. Dabei leistet der Testprozess einen wesentlichen Beitrag zur Absicherung der Funktionalität, Interoperabilität und allgemeinen Qualität.

Das Entwickeln von Standards ist ein Weg, den die Branche schon seit längerem beschreitet. So entstand in den letzten Jahren eine Vielzahl von Spezifikationen. Diese erfordern Konformitätstests als einen unverzichtbaren Bestandteil ihrer Qualitätssicherung. Denn Konformitätstests ermöglichen eine Aussage darüber, ob eine angebliche Implementierung ihrer Spezifikation auch in vollem Umfang entspricht. So gibt es beispielsweise seit Jahren Konformitätstests für Umsetzungen der bekannten standardisierten Kommunikationsprotokolle wie CAN oder LIN. Doch Software rückt immer weiter in den Mittelpunkt der Elektronikentwicklung. So wurden und werden Softwarespezifikationen entwickelt, die entsprechend einen Bedarf an Software-Konformitätstests hervorrufen. Die Ansprüche und somit Methoden, die sich beim Testen von Steuergeräte-Software ergeben sind verschieden gegenüber denen für das Testen von Hardware. Dennoch sollten Software und Hardware eingebetteter Steuerungssysteme gerade im Bereich des Testens nicht isoliert voneinander betrachtet werden.

Das Ziel dieser Arbeit ist es zu beschreiben, wie die (teilweise gegensätzlichen) Ansprüche an das Testen auf Konformität von Steuergeräte-Hardware und -Software in einem Testframework vereinigt werden können. Dieses Testframework, das Inhalt des vierten Kapitels ist, ermöglicht sowohl Hardware-Kommunikationsschnittstellen auf ihre Konformität zur Protokollspezifikation zu testen, als auch die Schnittstellen und Funktionen der darüber liegenden Software-Module auf Konformität hin zu überprüfen.

## 2 Testen auf Konformität

Testen auf Konformität hat das Ziel, die Übereinstimmung (Konformität) einer Implementierung zu ihrer Spezifikation zu offenbaren. Unter Implementierung ist dabei sowohl die Implementierung einer Software, als auch beispielsweise die Umsetzung eines CAN-Netzwerkprotokolltransceivers zu verstehen. Die Konformität einer Implementierung zu ihrer Spezifikation ist eine Voraussetzung dafür, dass die Implementierung in ein Gesamtsystem integriert werden kann, das weitere Umsetzungen des gleichen Standards enthält. Mit einem Konformitätstest verfolgt man somit den Zweck, die Wahrscheinlichkeit zu erhöhen, dass Implementierungen verschiedener Hersteller eines Systems zuverlässig miteinander arbeiten.

Die Notwendigkeit für Konformitätstests ergibt sich aus Folgendem: Spezifikationen beschreiben im Allgemeinen ein Soll-Verhalten. Sie geben hingegen nicht vor, wie dieses Verhalten, also die beschriebene Funktionalität im Detail umzusetzen ist. Daraus folgt, dass es möglich ist, beim Implementieren einer Spezifikation mehrere Lösungswege zu beschreiten. Neben der Tatsache, dass eine Spezifikation keine Implementierungsvorschrift, sondern eine Verhaltensbeschreibung ist, existiert auf dem Gebiet der Automobil-Elektronik/Elektrik ein weiteres Problem: Im Gegensatz zur Telekommunikationsbranche konnte sich in der Automobilindustrie bis heute keine formale Spezifikationsbeschreibungssprache etablieren. Spezifikationen sind infolgedessen meist in Prosa verfasst und enthalten unwillkürlich Interpretationsspielräume.

Aufgrund der vorangehend beschriebenen Umstände ist es notwendig, bei jeder Implementierung nachzuweisen, in wie weit sie die vorgeschriebene und optionale Funktionalität entsprechend ihrer Spezifikation umsetzt. Eine Implementierung gilt dann als konform zur gegebenen Spezifikation, wenn sie ihr zumindest so ähnlich ist, dass sie deren Mindestanforderungen erfüllt.

### 2.1 Konformitätstestspezifikation

Gleichwohl gewisser Interpretationsspielräume enthält jede Spezifikation (mehr oder weniger) eindeutige Anforderungen. Diese dienen als Grundlage für das Testen auf Konformität und stellen die Ausgangspunkte für das Erstellen einer Konformitätstestspezifikation dar. Eine Konformitätstestspezifikation enthält die Spezifikation der Testfälle. Ein Testfall wiederum dient zum Überprüfen einer oder wenn sinnvoll mehrerer Anforderung(en). In einer Konformitätstestspezifikation muss für jede Anforderung mindestens ein Testfall existieren.

Die wichtigste Forderung, die an Konformitätstestspezifikationen gestellt wird ist, dass sie eine 100-prozentige Reproduzierbarkeit eines jeden Testfalls garantiert. Um dies zu ermöglichen muss eine Konformitätstestspezifikationen die folgenden Eigenschaften aufweisen:

- Unabhängigkeit von der spezifischen Implementierung
- Unabhängigkeit von der Testausführung / vom Testhaus
- Unabhängigkeit von spezifischer Testsoftware
- Unabhängigkeit von der Testumgebung

Die letzten zwei Punkte erlauben eine Ausnahme. Eine Konformitätstestspezifikation darf abhängig von Testsoftware oder Testumgebung definiert sein, insofern diese innerhalb der Konformitätstestspezifikation eindeutig beschrieben sind.

Ein Kernaspekt bei der Erstellung von Konformitätstestspezifikationen ist die Erstellung der Testfälle. Diese Thematik wird im Rahmen dieses Dokuments allerdings nicht näher vorgestellt. Es existieren diverse sowohl allgemein bekannte, als auch proprietäre Methoden, die die Testfallableitung unterstützen und darüber hinaus helfen, eine sinnvolle Auswahl an Testfällen zu erzielen. Es soll an dieser Stelle dennoch angemerkt sein, dass eine Testabdeckung von 100 Prozent nur theoretisch möglich ist. Es ist Praktisch nicht umsetzbar (aus Zeit-, Kosten- und oft auch Speicherplatzgründen) alle möglichen Fehler und Fehlerursachen direkt zu überprüfen.

Daher sollte man bei der Testfallableitung Beschränkungen auf möglichst realistische (unter anderem bezogen auf das spätere Einsatzgebiet) und kritische Konstellationen vornehmen.

## **2.2 Gegenwärtige Teststandards und –Strategien**

Neben einschlägig bekannten Testmethoden, die beschreiben, nach welchen Verfahren und Kriterien die Testobjekte überprüft werden sollen, existieren diverse Richtlinien, die angeben, wie beim Testen vorzugehen ist. Dazu zählen beispielsweise die im Folgenden kurz vorgestellten und von der International Standards Organisation (ISO) festgelegten Normen. Innerhalb der ISO-Norm 9646 wurde neben der Testmethodik und den Rahmenbedingungen für Konformitätstests auch deren Beschreibungsart in Form der „Tree and Tabular Combined Notation“ (TTCN) definiert. Aus dieser hat sich mittlerweile die von der ETSI definierte und international standardisierte Testbeschreibungs- und Implementierungssprache „Testing and Test Control Notation Version 3“ (TTCN-3) entwickelt. Auch diese wird im Folgenden näher vorgestellt.

### **2.2.1 Die ISO-Norm 17025**

Die ISO-Norm 17025 (siehe [1] ) beschreibt die „Allgemeinen Anforderungen an die Kompetenz von Prüf- und Kalibrierlaboratorien“. In dieser Norm sind alle Anforderungen genannt, die sichern sollen, dass die Ergebnisse des Laboratoriums glaubwürdig, richtig vergleichbar und für den Anwender nutzbar sind. Testlabore, die ISO 17025 akkreditiert sind, verfügen somit über eine Kompetenzbestätigung, die zeigt, dass sie in der Lage sind bestimmte Konformitätsbewertungsaufgaben durchzuführen. Über diese offizielle Kompetenzbestätigung hinaus ermöglicht das Anwenden der in der ISO 17025 definierten Regeln dem Testlabor richtige, technisch glaubwürdige und nachvollziehbare Testergebnisse zu erzielen.

### **2.2.2 Protokolltests nach ISO 9646**

Die ISO-Norm 9646 (siehe [2]) definiert einen umfangreichen Standard für den Konformitätstest von Protokollimplementierungen nach dem ISO-OSI-Schichtenmodell für offene Systeme (engl.: OSI - Open Systems Interconnection). Dieser besteht aus sieben Teilen und trägt den Titel „Conformance Testing Methodology and Framework“.

Die Charakteristiken dieses Standards sind im Folgenden aufgelistet:

- Beschreibung der Methoden und des Ablaufs von Konformitätstest
- Adaption ist ausschließlich über definierte Schnittstellen erlaubt
  - Ermöglicht Black-Box-Testing
- Definition eines Standards (Vorlage) zur Testfalldefinition
- Basiert auf dem ISO-OSI-Schichtenmodell

Die Ansprüche, die an Konformitätstest gestellt werden, sind vielfältig. So weisen Testobjekte beispielsweise verschiedenartige Strukturen auf und es lassen sich meist nicht alle Einheiten einer Implementierung direkt ansprechen. Dennoch muss ein Test für jede Implementierung anwendbar sein und unabhängig von deren spezifischer Struktur korrekte Ergebnisse liefern. Das Testobjekt ist darüber hinaus meist Bestandteil eines integrierten Schaltkreises und befindet sich in der Regel neben anderen Komponenten in einem Mikrokontroller. Der Zugriff auf das Testobjekt kann nur über die (Kommunikations-)Verbindungen erfolgen, die der Mikrokontroller zur analogen und digitalen Außenwelt zur Verfügung stellt.

Ferner sind Protokollspezifikationen meist sehr komplex (Beispiel: FlexRay-Protokollspezifikation, siehe [3]), lassen aber gleichzeitig den bereits erwähnten Interpretationsspielraum und erlauben verschiedene Implementierungen. Entsprechend vielfältig sind auch die Konformitätstests, die die Wahrscheinlichkeit der Interoperabilität zwischen verschiedenen Implementierungen erhöhen sollen. Ein weiterer zu bedenkender Aspekt ist die Tatsache, dass Konformitätstest im Allgemeinen das Black-Box-Testverfahren

anwenden. Das heißt, für das Überprüfen der Funktionalitäten ist die interne Realisierung nicht bekannt. Da eine Protokollschicht ein geschlossenes System darstellt, ist es nur möglich Informationen über die Schnittstellen zwischen den einzelnen Schichten, den so genannten „Service Access Points“ (SAPs) auszutauschen.

Um diesen mannigfaltigen Ansprüchen an einen Konformitätstest genügen zu können, definiert die ISO 9646 eine international anerkannte Testmethodik. Diese Testmethodik umfasst neben einer (allgemein gebräuchlichen und eingesetzten) Struktur für die Spezifikation von Konformitätstests auch eine etablierte Testarchitektur. Diese ist in Abbildung 1 - Testarchitektur nach ISO 9646 dargestellt.

Die zu testende Implementierung wird als „System Under Test“ (SUT) bezeichnet. Dieses SUT wird vom Testsystem (dunkelblau hinterlegter Bereich) eingerahmt. Es besteht aus „Upper Tester“, „Lower Tester“ und einer „Test Coordination Procedure“ (TCP). Der Upper Tester beobachtet und stimuliert das SUT von dessen oberer Schnittstelle aus; er übernimmt die Funktion der nächsthöheren Schicht. Der Lower Tester beobachtet und stimuliert das SUT von dessen unterer Schnittstelle aus; er übernimmt die Funktion der nächstniedereren Schicht. Die TCP wird durch den Supervisor umgesetzt und kontrolliert und koordiniert Upper und Lower Tester.

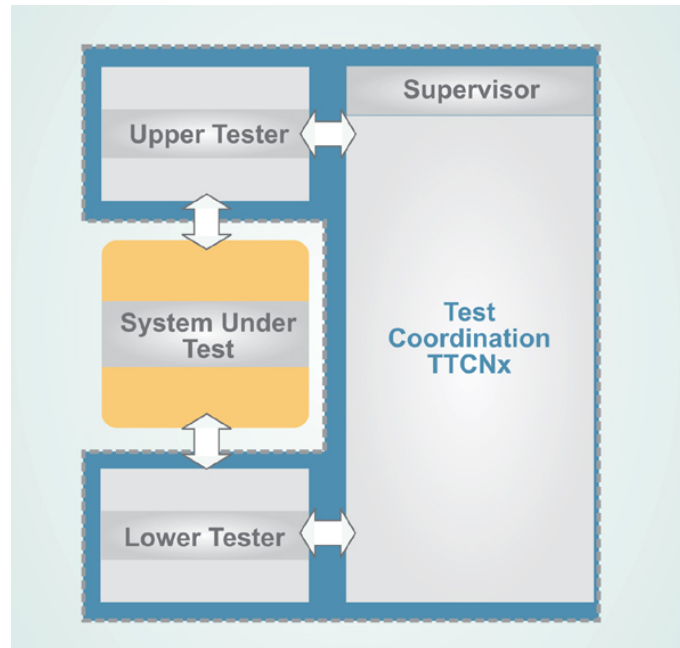


Abbildung 1 - Testarchitektur nach ISO 9646

Die Schnittstelle zwischen Upper Tester und SUT und Lower Tester und SUT an denen die Kommunikation überwacht und gesteuert werden kann, wird „Point of Control and Observation“ (PCO) genannt. Der Upper Tester hat die Aufgabe das Verhalten des SUT zur höheren Schicht hin zu testen, indem er die über den oberen PCO erreichbaren Dienste nutzt. Insofern die nächsthöhere Schicht Bestandteil des Mikrokontrollers ist, muss zumindest ein Teil der Upper Tester Software in den Mikrokontroller integriert werden. Diese Upper Tester Software stimuliert und überwacht somit die Kommunikation am oberen PCO. Der Lower Tester stimuliert und überwacht entsprechend das Verhalten der SUT am unteren PCO.

Alle Testdaten, die an den PCOs erfasst werden und somit das Verhalten des SUT repräsentieren, werden gespeichert. Nach Beendigung eines Testdurchlaufs werden diese mit den Referenzdaten eines Modells verglichen und ausgewertet.

### 2.2.3 Testing & Test Control Notation (TTCN-3)

Neben der durch ISO 9646 definierten Testmethodik gibt es eine Sprache, die die formale Beschreibung und Ausführung von Testfällen ermöglicht. Die „Testing & Test Control Notation Version 3“ (TTCN-3, siehe [4]) ist durch das European Telecommunications Standards Institute (ETSI) seit Oktober 2000 standardisiert. Die Vorgänger-Versionen 1 und 2 waren Inhalt des dritten Teils der ISO 9646.

TTCN-3 versteht sich als neue Testspezifikations- und Implementierungssprache mit dem Anspruch eine „general purpose testing language“ zu sein. Sie wurde speziell für das Black-Box-Testen reaktiver, verteilter Systeme entwickelt. Somit erlaubt sie neben der Beschreibung und dem Ausführen von Konformitätstests beispielsweise auch Integrations-, System-, Last- oder Skalierungstests. Sie wird bereits erfolgreich für das Testen im

Telekommunikationsbereich eingesetzt. So existieren zum Beispiel fertige Testsuites für IPv6 oder VOIP.

Die Testfälle werden vorzugsweise mit Hilfe der Kernsprache, also textuell, beschrieben. Die folgende Abbildung zeigt ein Beispiel eines einfachen in TTCN-3 beschriebenen Testfalls.

```
testcase MyTestcase()
  runs on MyTestComponent
  system SystemComponent
{
  . . .
  MyPort.call (myMethod:{MY_ID}, 3.0 );
  alt {
    [] MyPort.getreply (myMethod:{MY_ID} value "Hello World!") {
      verdict.set (pass);
    }
    [] MyPort.getreply {
      verdict.set (fail);
    }
    [] MyPort.catch (timeout) {
      verdict.set (fail);
    }
  } // end of alt
  . . .
}
```

Abbildung 2 - Beispiel eines Testfalls in der TTCN-3 Kernsprache

Ein Kernaspekt bei TTCN-3 ist das Konzept der abstrakten Testsuite. Das heißt, ein Testfall wird unabhängig vom der Implementierung und Infrastruktur des Testsystems spezifiziert. Um auf dieser abstrakten Ebene die Kommunikation zwischen Testfall und dem SUT zu beschreiben, werden „Stimuli“ des Testfalls und die erwartete Reaktion („Response“) des SUT über Ports spezifiziert. Über diese Ports erfolgt dann im Nachhinein das Mapping zwischen der Abstrakten Testsuite und der Schnittstelle des realen Testsystems. Dieser Mechanismus ist in Abbildung 3 – Port Mapping bei TTCN-3 näher dargestellt.

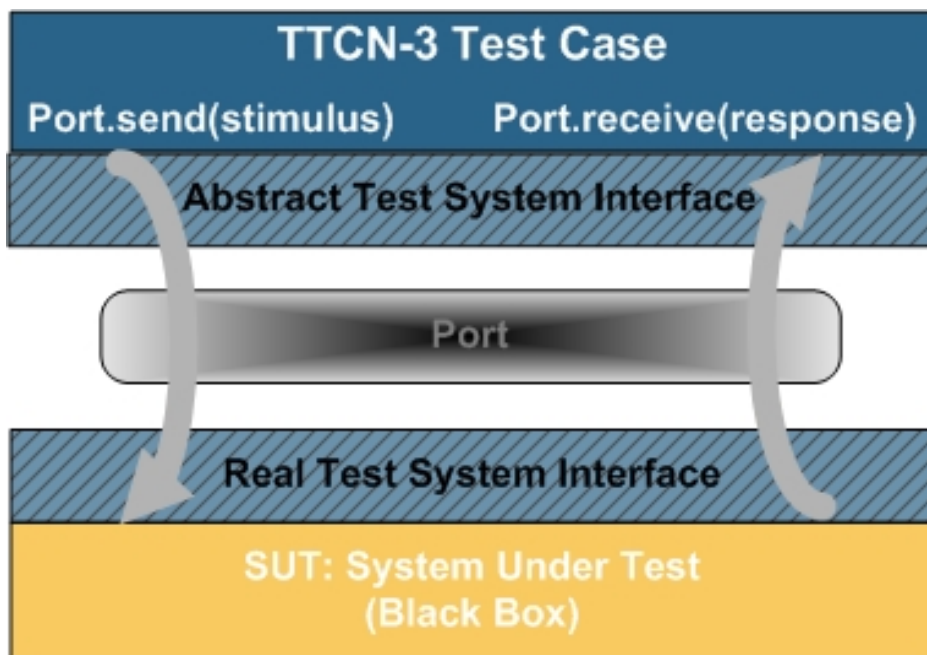


Abbildung 3 – Port Mapping bei TTCN-3

Um eine abstrakt definierte Testsuite auf einer realen Testplattform auszuführen, muss die Testsuite noch adaptiert werden. Dazu stehen zwei spezifizierte Schnittstellen zur Verfügung. Das „TTCN-3 Runtime Interface“ (TRI) und das „TTCN-3 Control Interface“ (TCI).

Über das TRI wird die Adaption des Testsystems definiert. Dies geschieht wiederum über die beiden Subinterfaces:

- „System Adapter“ (SA) welcher die Kommunikation(-schnittstelle) zwischen Testsystem und SUT beschreibt und
- „Platform Adapter“ (PA) um Timer oder externe Funktionen der Plattform zu nutzen.

Mit Hilfe des TCI wird die Schnittstelle zur Adaption des Testsystems realisiert. Das TCI stellt die folgenden Komponenten zur Verfügung:

- Das „Testmanagement“ (TM) zur Interaktion zwischen Nutzer und Testsystem während der Testausführung
- Das „Component Handling“ (CH) zur Verteilung und Kontrolle der Testkomponenten und deren Kommunikation
- Den „CoDec“ (CD) um die Testdaten („Stimuli and Response“) zu kodieren und dekodieren

Die verschiedenen Komponenten eines TTCN-3 Testsystems und die Schnittstellengrenzen (TCI und TRI) sind in Abbildung 4 – Struktur des TTCN-3 Testsystems dargestellt.

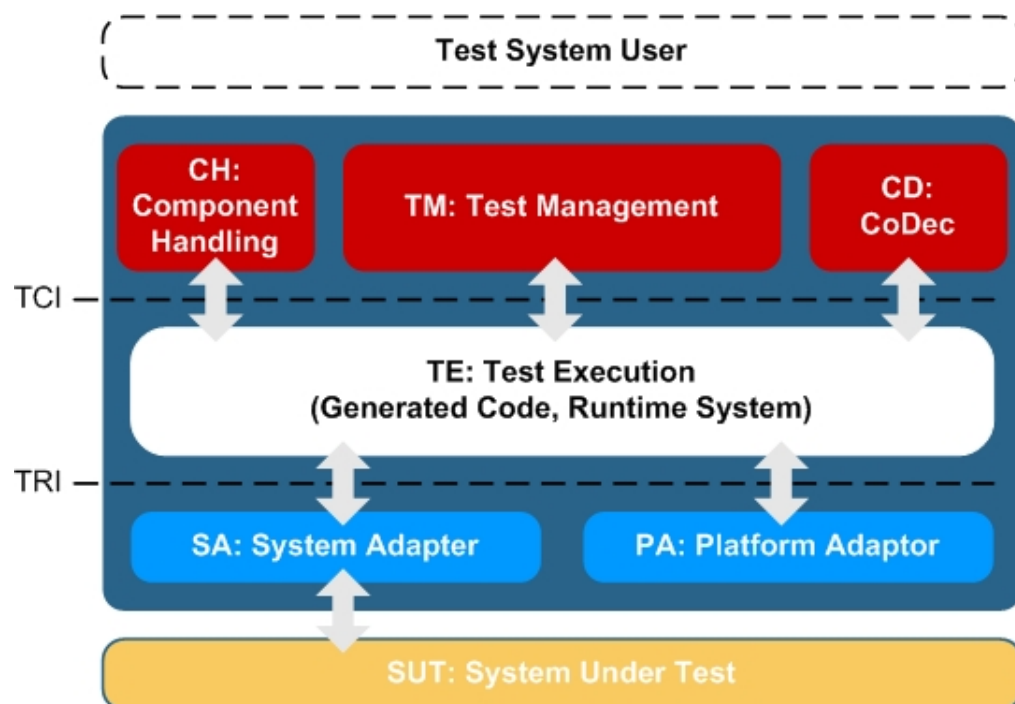


Abbildung 4 – Struktur des TTCN-3 Testsystems

Nach der Adaption an die Testplattform können die kompilierten Testfälle ausgeführt werden. TTCN-3 stellt fünf verschiedene Testresultate (Testurteile) zur Verfügung. Diese lassen sich einzeln für jeden Test und dabei für jede Testkomponente vergeben und sind wie folgt von „none“ bis „error“ geordnet:

**none > pass > inconc > fail > error**

Während der Testausführung gelten für die Testurteile die folgenden zwei Regeln:

- Immer das jeweils schlechteste Urteil ist gültig.
- Das Testurteil „error“ kann nicht explizit, sondern nur automatisch vom Laufzeitsystem gesetzt werden.

Die folgende Tabelle gibt eine Übersicht über die Bedeutungen der Testurteile:

*Tabelle 1 - TTCN-3 Testurteile*

<b>none</b>	Kein Testurteil verfügbar.
<b>pass</b>	Testverlauf wie erwartet. Das Testobjekt hat den Test bestanden.
<b>inconc</b>	Inconclusive, auf deutsch ergebnislos, bedeutet, dass das Testurteil weder einem „pass“ noch „fail“ entspricht. Der Test sollte in diesem Fall eindeutiger spezifiziert werden.
<b>fail</b>	Das Testobjekt hat sich während des Testdurchlaufs nicht wie erwartet verhalten und den Test somit nicht bestanden.
<b>error</b>	Während des Testdurchlaufs wurde vom Laufzeitsystem ein fehlerhaftes Verhalten festgestellt.

Da das Testurteil für jede Testkomponente einzeln ermittelt wird, ist man nach Testdurchführung problemlos in der Lage die mögliche Fehlerursache einzugrenzen, insofern der Test nicht bestanden wurde.

### 3 Der Standard „AUTOSAR“

Aspekte wie Kostendruck und die zunehmend schlechter beherrschbare Komplexität der elektronischen Komponenten in Fahrzeugen führten zu Bestrebungen, einen Architekturstandard für Software-Systeme in der Automobilbranche zu etablieren. Diesem Bestreben hat sich eine Partnerschaft von Automobilherstellern, Zulieferern und Werkzeugherstellern gewidmet und den Standard **AUTOSAR** (AUTomotive Open System ARchitecture, siehe [5]) entwickelt. Das Ergebnis dieser Partnerschaft ist die Spezifikation einer Architektur unabhängiger Software-Module und deren Kombination und Konfiguration. An die definierten Schnittstellen wird die Anforderung gestellt, eine leichte Austauschbarkeit sowohl der Hardware, als auch der oberhalb der Basis-Software-Architektur laufenden Anwendungen zu ermöglichen.

Neben der Spezifikation der einzelnen Basis-Software-Module werden innerhalb von AUTOSAR auch die zugehörigen Konformitätstests spezifiziert, die zum Beispiel die genannte Austauschbarkeit sicherstellen sollen. Dadurch ergeben sich neue Anforderungen an das Testen im Automobilbereich. Es müssen sowohl neu standardisierte Software-Module als auch Hardware-Module, die etablierte Kommunikationsprotokolle umsetzen, getestet werden. Dabei muss beachtet werden, dass auch innerhalb von Software-Modulen, neben „reinen Software-Funktionen“ auch Kommunikationsmechanismen wie sie beispielsweise für das Netzwerkmanagement nötig sind, spezifiziert werden. Das folgende Kapitel beschreibt, wie sich neue Ansprüche, die sich durch AUTOSAR ergeben, mit dem zuvor beschriebenen Vorgehen beim Testen der für jede ECU notwendigen Kommunikationsmechanismen vereinen lassen.

## 4 Erweiterung bewährter Testarchitektur

Das Ziel dieses Kapitels ist, ein Testframework vorzustellen, das durch seine Struktur sowohl Hardware-Kommunikationsschnittstellen auf ihre Konformität zur Protokollspezifikation testet, als auch die Konformität der API und Funktionalität der darüber liegenden Software-Module überprüft. Das Konzept basiert auf dem Konformitäts-Testsystem der C&S group.

### 4.1 Das Testframework – Eine Übersicht

Einen Überblick über dieses Test-Framework gibt die

Abbildung 5 - Struktur des Testframeworks.

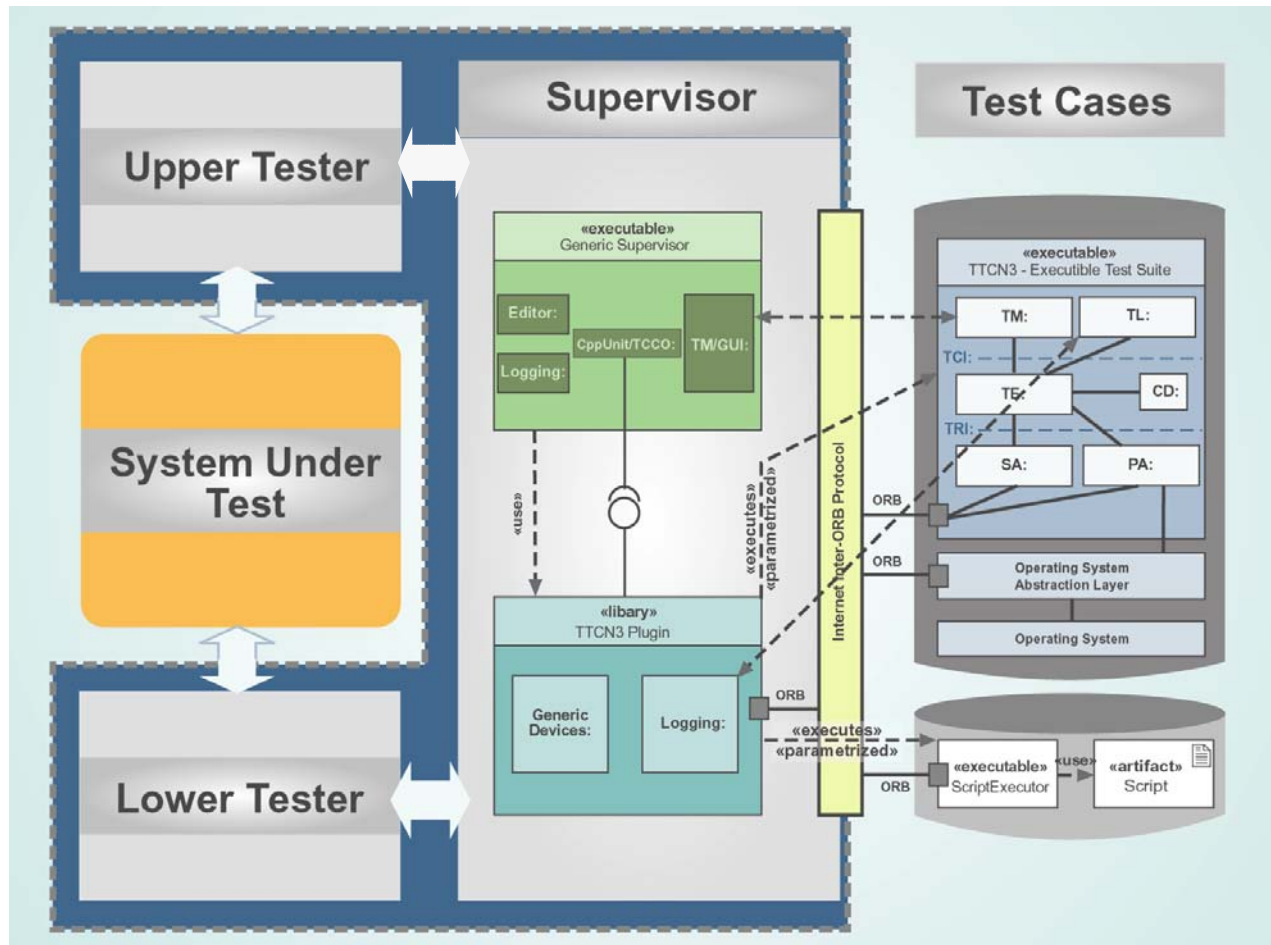


Abbildung 5 - Struktur des Testframeworks

Das hier dargestellte Framework entspricht einer erweiterten Umsetzung der in Abbildung 1 - Testarchitektur nach ISO 9646 dargestellten Architektur aus Lower Tester, Upper Tester und Generic Supervisor. Der Upper Tester wird durch Software umgesetzt, die sich auf dem SUT befindet und dort zur Testlaufzeit ausgeführt wird. Der Lower Tester ist Hardware, die über ein Kommunikationsmedium, entsprechend dem verwendeten Protokoll des SUT, mit diesem verbunden ist. Die Testkoordination wird vom Supervisor übernommen.

Der Supervisor besteht aus zwei Hauptkomponenten: Dem ausführbaren „Generic Supervisor“ (GSV) und einer „TTCN3-Plugin Bibliothek“. Der GSV bietet über eine Grafische Benutzeroberfläche die Möglichkeit des Testmanagements. Der Benutzer kann hierüber steuern, welcher Testfall ausgeführt werden soll. Darüber hinaus ist der GSV für das Testlogging verantwortlich. Über eine Schnittstelle kann der GSV die TTCN3-Plugin Bibliothek

verwenden. Diese ruft die parametrisierbaren Testfälle der ausführbaren TTCN-3 Testsuite auf. Eine TTCN-3 Testsuite enthält alle Testfälle, zum Überprüfen ob Schnittstelle und Funktionalität eines Softwaremoduls entsprechend ihrer Spezifikation implementiert wurden. Die Testsuite befindet sich auf einem Test-PC; in der Abbildung ist diese im Bereich „Test Cases“, außerhalb des dunkelblau hinterlegten Bereichs zu erkennen.

Der Bereich der Test Cases umfasst neben der TTCN-3 Testuite auch den ausführbaren „ScriptExecutor“. Der ScriptExecutor verfügt über Testscripts, die in einem weiteren, von der TTCN-3 Testuite unabhängigen Prozess laufen. Diese Scripts dienen dem Testen der externen Kommunikation des Steuergerätes.

Die Kommunikation zwischen TTCN-3 Testsuite beziehungsweise ScriptExecutor und dem Supervisor erfolgt über das „Internet Inter-ORB Protocol“ (IIOP). IIOP ist das Protokoll zum Fernaufruf innerhalb der „Common Object Request Broker Architecture“ (CORBA) und ermöglicht die Kommunikation zwischen „Object Request Brokers“ (ORBs) verschiedener Hersteller. ORBs sind Vermittler, die eine von Betriebssystem und Programmiersprache unabhängige Kommunikation von Objekten in einem verteilten System erlauben.

## **4.2 Testausführung**

Während der Testausführung sind die weiteren Aufgaben des Supervisors von der Art der auszuführenden Testfälle (TTCN-3 oder Testscript) abhängig. Die Kommunikation zwischen Supervisor und Upper beziehungsweise Lower Tester erfolgt jedoch immer über die in das TTCN-3-Plugin integrierte Komponente „Generic Devices“. Darüber lassen sich zwei Arten von Geräten ansprechen. Zum einen Testhardware, die das Simulieren und Überwachen der externen Kommunikation des Steuergerätes erlaubt. Diese Testhardware besteht meist aus einem Logikanalysator und einem „Pattern Generator“. Zum anderen ermöglicht die Generic Devices Komponente die Kommunikation mit der Upper Tester-Software.

Die Verwendung von Testhardware ist immer dann nötig, wenn die externe Kommunikation des Steuergerätes für den Testfall relevant ist. Die Kommunikation zwischen Supervisor und Lower Tester erfolgt dann mit Hilfe dieser Testhardware. Dieser Kommunikationsweg wird beim Ausführen von Protokolltests immer benötigt.

Beim Ausführen von TTCN-3 Testsuites erfolgt die Kommunikation in Abhängigkeit vom Inhalt des Testfalls. Entweder erfolgt sie nur zwischen der Generic Devices Komponente und dem Upper Tester, oder es erfolgt eine Kommunikation sowohl zwischen Generic Devices und Upper Tester, als auch zwischen Generic Devices und Lower Tester. Letzteres ist beispielsweise der Fall, wenn ein Software-Modul getestet werden soll, das für das Netzwerkmanagement verantwortlich ist.

Im Falle des Ausführens einer TTCN-3 Testsuite ist die Verwendung des Begriffs Upper Tester nicht ganz korrekt. Die entsprechende Tester-Software simuliert nicht nur das Verhalten der darüber liegenden Schicht, sondern sie übernimmt die für die Testausführung notwendige Simulation aller sich um das SUT befindenden Software-Module.

## **4.3 Vorteile durch den Einsatz dieses Frameworks**

Der Einsatz eines solchen, eben vorgestellten Testframeworks erlaubt zum einen eine verbesserte Testausführung: Durch die Verwendung von CORBA können die auszuführenden Tests in einem System beliebig verteilt werden. Da die Umsetzung durch unabhängige Prozesse erfolgt, wird eine parallele Ausführung der Tests ermöglicht. Dieses Konzept ist in den Testsystemen der C&S group erfolgreich eingesetzt.

Für das Testen der Software eingebetteter Systeme ist es darüber hinaus sinnvoll auch die Umgebungsbedingungen des Systems zu berücksichtigen. Zu diesen zählen unter anderem der verwendete Speicher und dessen Ort (integriert oder extern), vorgenommene Optimierungen oder der Takt des Mikrokontrollers. Dadurch, dass die zu testende Software während der Testausführung innerhalb ihrer Plattform läuft, wird es ermöglicht diese und andere Umgebungsbedingungen zu überprüfen.

Zusammengefasst ermöglicht das Testframework das Ausführen von *Combined Conformance Tests* (CCT). Ein CCT testet Software-Funktion und -Schnittstelle eines Moduls innerhalb seiner Zielplattform und gleichzeitig die externe Kommunikation des Steuergerätes oder eines Subsystems. Dadurch wird eine verbesserte, allgemeingültigere Aussage bezüglich der Konformität eines Steuergerätes ermöglicht. Ferner ermöglicht diese Testarchitektur das Testen in Echtzeit und erfüllt die spezifischen Anforderungen, die sich für das Testen unter typischen Automotiv-Bedingungen ergeben. So wird durch die Flexibilität des Testframeworks eine Reduzierung der Testfälle auf die tatsächlichen Anforderungen des Einsatzgebietes eines Steuergerätes ermöglicht.

## A Quellen

- [1] „DIN EN ISO/IEC 17025“, Allgemeine Anforderungen an die Kompetenz von Prüf- und Kalibrierlaboratorien (ISO/IEC 17025:2005); Deutsche und Englische Fassung EN ISO/IEC 17025:2005, August 2005
- [2] “ISO/IEC DIS 9646-1..5”, ISO Norm Konformitäts-Testmethoden und Rahmenwerk, International Organization for Standardization, 1989.
- [3] Internetseiten des FlexRay Konsortiums, <http://www.flexray.com/>
- [4] Internetseite des European Telecommunications Standards Institute (ETSI) zur Testing and Test Control Notation Version 3 (TTCN-3) <http://www.ttcn-3.org/>
- [5] Internetseiten der AUTOSAR Entwicklungs-Partnerschaft <http://www.autosar.org/>
- [6] W. Lawrenz: “Conformance Test of In-Vehicle Communication Protocols – Why?” bzw. “Konformitätstests für Kommunikationsprotokolle in Fahrzeugen - wozu?” , Entwicklerforum Kfz-Elektronik, Ludwigsburg, presentation, May 11, 2005
- [7] W. Lawrenz: “Standard Module Conformance Testing“, Mauritius ICONS'06, April 2006, paper no. 47

## B Kontakt

E-Mail: [S.Schwarzkopf@cs-group.de](mailto:S.Schwarzkopf@cs-group.de)

Anschrift: C&S group  
Fachhochschule Braunschweig/Wolfenbüttel  
Salzdahlumerstr. 46-48  
38302 Wolfenbüttel  
[www.cs-group.de](http://www.cs-group.de)

Telefon: +49/5331 939 6601

Fax: +49/5331 939 6602