'Leveled' Conformance Tests - A Must for Interoperability in Networked Systems

Dr. Wolfhard Lawrenz, Frank Fischer, Karsten Hoffmeister, Maria Scheurer

Modular design and networked solutions are the basic answers to master complexity of today's and future systems. Complexity will even grow dramatically as the desired functionality by the end user is increasing drastically. A basically efficient approach to achieve proper interoperability in distributed systems is to apply standard 'layers' onto which the application itself is built, to specify standard layers as high as possible while minimizing the individual application parts and to check the functionality of these layers by sufficiently efficient conformance tests. In automotive and other applications these 'standard' layers comprise – from the lowest to the highest level – the 'transceiver' layer, the 'network' layer, a 'device driver interface' layer and the 'operating system' layer, which currently is the final interface to the application. In automotive applications the communication layers mostly are based on CAN protocol complemented by CAN software drivers interfacing the operating system OSEK/VDX.This stack of 'standard' levels provides a powerful and neutral interface to the application. But experience over the last years has shown that the implementations of the individual levels differ in their behavior. This is due to several reasons such as the specifications of the individual levels are not precise enough, they are ambiguous, they difficult to implement because of their complexity. Even if there formal and executable specifications for instance based on state machine languages, implementations show defects because the implementers try to optimize and thus they violate unknowingly for instance the constraints related to local variables.

c&s group has worked in the field of conformance tests since more than 5 years and thus has contributed for instance to the development of the ISO CAN Conformance test standard. As a complement to the ISO standard c&s has specified and implemented conformance tests for the CAN register interface and for the related 'higher layer' software drivers. Furthermore c&s is cooperating in a group of auto manufacturers – Daimler Chrysler, BMW, Audi, Volkswagen, PSA – to specify and implement conformance tests for a new generation of fault tolerant CAN transceivers. Tests for the communication and network management part of OSEK/VDX have been specified and implemented. The OSEK tests go beyond the 'normal' test methods which typically check all possible transitions from each state to its neighbor states, because this neglects some typical implementation 'faults', which are due to some optimization side effects.

Looking at the 'tests chain' comprising all the levels being involved to build the bridge from one application on one module to the other modules a high degree of confidence in the conformance of the whole chain is provided. The present paper discusses the constraints given by the distributed system in conjunction with the philosophy how to derive tests on the various levels, how to implement the tests and typical test problems and achieved results.

Introduction

Complex systems in cars grow at a very rapid pace. Enhanced performance such as lower fuel consumption, less pollution, better safety, enhanced comfort require more electronics. Even medium sized cars currently have more than 30 electronic control units – ECUs -, in high end cars there are even more than 100 ECUs. In power train control they serve for engine control, gear box control, brakes control, stability control, etc. In the body control area the ECUs do seat positioning, mirror positioning, power windows control, etc. The multi media applications support radio functionality, TV, telephone, CD player, navigation systems, etc. In current cars there are multiple intelligent air bag controls. In the near future there will be steer by wire, brake by wire and other intelligent functions.

All this complex functionality requires communication between the various ECUs in order to optimize or even enable the overall control loop of the car as a system. Communication typically is performed through multiple networks performing different application oriented protocols.

Obviously predictable and even the safe operation of the complex 'system car' would require good specifications of the components at the various levels as well as sufficiently well specified tests. These requirements would be even more evident for 'standard'-components such as 'standard-communication-links' between the modules.

But very often the specification of the communication components at the various levels is not precise nor complete. In most cases the specification does not exist in a formal language. Therefore formal checks on completeness, in-ambiguity, etc. can not be performed. Due to this there is no way to derive formal verification tests. Tests to check the conformance of such standard components are derived from experience, enhanced by systematic considerations to optimize the tests and to minimize the amount of tests to be executed. Very often the tests in turn become the specification of the component – which may be the wrong way to govern complexity of large systems.

All this is due to the fact that systems grow so rapidly. Because of the competition race there is no time in the pre-development phase for systematic formal approaches. Even worse, in order to have advantages in front of the competition important parts of the specification of standard components are hidden and not published. Interoperability then can only be guaranteed by exhaustive conformance testing, although this may be the wrong way.

Subsequently the state of the art of deriving conformance tests for an insufficiently specified standard communication component is given. The test cases were derived empirically and optimized by systematic treatment. This consequently would lead to a formal more complete specification of the component itself. An iterative optimization approach is given balancing the tests and the specification of the component.

c&s masters the art of deriving conformance tests for an insufficiently specified standard communication component.

Layered Communication Architecure

Proper function of complex systems in cars is essentially depending on the conformance of multiply used standard components. One of the mostly used standard components/functions is communication. Therefore interoperability of the communication modules being supplied by various implementers linking the ECUs into a system is a feature of very high interest. Referring to the ISO-OSI model a communication chain based on e.g. CAN communication protocol [2], [3] could consists of the following modules/layers:

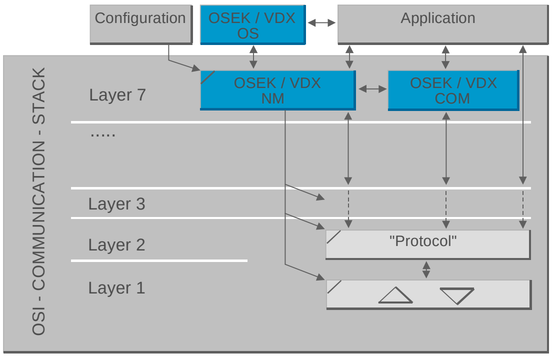


Fig. 1: car system archtitecture

- Application, such as engine control or gear box control, an operating system like OSEK/VDX OS, or an Configuration Management.
- OSI layer 7: OSEK/VDX-Communication and OSEK/VDX-Network-Management
- OSI layer 6 .... 3: may be void
- OSI layer 2: CAN protocol
- OSI layer 1: CAN transceiver and physical wiring

All layers provide their services to the upper layers and use services of the lower layers. The application, a configuration management or an operating system may be the 'final' user of the OSI layer 7. The OSEK-Network Management of layer 7 provide to the 'final' users only local- (node-related) and global- (network-related) management methods (e.g. start-up network or management of different mechanisms for node monitoring), the OSEK-Communication provide the 'real' communication functionality for inter-ECU communication (communication within electronic control units) and also the ECU-internal communication (communication within electronic control units). To provide this services the layer 7 uses the lower layer services which are provided directly per the data link layer (layer 2) or per a higher layer (e.g. layer3). The layer 3 to 6 may be void but there can be another another higher level protocol (e.g. session layer of time triggered CAN). In any case an adaption of the communication layer (layer 2) to the higher layers must exist. The data link layer services are not standardized in case of CAN. This is a problem because CAN implementations of various implementers are too different to exchange them without an adaptation to the higher levels like OSEK-COM. In general layer 2 shall provide services like set up of data which shall be send, indicate received data, set transmit requests and confirm transfer status. The physical layer is the interface between the signal transmission over the physical communication media and the digital 'world' towards the higher OSI-layers.

Layered Conformance Tests

A basically efficient approach to achieve proper interoperability in distributed systems of high complexity is to apply standard 'layers' onto which the application itself is built, to specify standard layers as high as possible while minimizing the individual application parts and to check the functionality of these layers by sufficiently efficient conformance tests.

Apparently the communication link is a crucial factor for the proper function of the whole system. Although there are standardized communication protocols the interoperability of the communication modules may be questioned as the specification of the standards may be partially not strong enough as to ensure that different implementations are sufficiently interoperable. The functionality of a component is controlled by the specification of the component. The specification is mostly given in natural language, which may be ambiguous and not precise and not complete enough. Often some 'formal' specification means are applied partially for instance state diagrams. But for various reasons such as protecting knowledge – intellectual property IP – in front of the competitors even these formal description parts are not sufficiently deeply specified. This willingly takes the risk or even the sure consequence into account that different implementers of the very same specification result into devices which behave differently under certain conditions. This of course is crucial when interoperability of 'standard' components is assumed, being supplied by different component manufacturers.

Baring in mind the constraint of insufficiently specified components and the requirement that these components should be interoperable, one way out of this dilemma is to apply 'sufficiently exhaustive' conformance tests to verify the desired interoperability of components. As no sufficiently detailed and formal description of the component exists a so called black box test technique is applicable. Test stimuli are applied from outside of the component – which is referred to as Implementation under Test IUT – to the accessible inputs, the responses are read from the outputs of the IUT and compared to what is understood from the specification.

In a first step in order to derive a first guess of a sufficiently exhaustive set of test cases an empiric driven selection of a set of test cases, representing practical conditions under which a component later on would work, is a recommendable practical approach. In a second step the individual elements of the above set are organized in such a way that they each would result as the parameterized product of a set of – mostly desired – orthogonal vectors. These vectors may be defined application driven. The parameterized product of the vectors defines the so called System Operational Vector Space – SOVS, comprising all so far known possible operational conditions under which IUT is expected to be functional in the specified way. In a third step a subset of the SOVS is selected empirically, driven from application experience, or even formally. This subset is then expected to be Sufficiently Exhaustive Minimal Set of (conformance) Test Cases SEMSTC on one hand while comprising not too many tests cases on the other hand resulting in a still acceptable test time.

Obviously the approach mentioned above is a practically viable procedure to derive a sufficiently exhaustive set of test cases. But this procedure of course incorporates the risk that the selected subset of tests does not cover sufficiently the behavior of the component. There are several ways to optimize the selection of test cases – see below:

- The selection of the of SEMSTC should be supported by formal methods
- The application of the SEMSTC in conjunction with the practically gained experience that in real application cases will be detected that tested components will show deficiencies in their desired interoperability will lead to an iterative optimization process influencing the re-specification of the test cases and the component. The newly detected problem is analyzed leading to:
- A redefinition of the SOVS, their parameters and SEMSTC in such a way that the new case systematically is a member of the sets mentioned above.
- A redefinition of the specification of the component, leading to a more precise, more detailed, more formal result, without necessarily being too stringent to the implementers. Based on that 'more' formal methods could be applied to derive tests which would then lead to a better SEMSTC. The black box test method would then turn more into a gray box test technique.

OSEK/VDX NETWORK MANAGEMENT TESTS - In the scope of OSEK/VDX the Network Management system (NM) provides standardised features which ensure the functionality of inter-networking by standardised interfaces. The essential task of NM is to ensure the safety and the reliability of a communication network for ECUs. According to the car system architecture (Fig. 1) NM comprises the following standardised interfaces:

- Configuration/OS/Application ↔ OSEK/VDX NM
- OSEK/VDX NM ↔ OSEK/VDX COM
- OSEK/VDX NM ↔ communication driver/medium

c&s has developed a test system for testing the conformance of implementations of the OSEK/VDX NM concept to the OSEK/VDX NM specification. The development comprised the following steps:

- derivation of the test cases
- planing and realization of the test environment
- implementation of tester
- automation of test execution and verification

Derivation of Test Cases - The specification of OSEK/VDX NM is given as a state machine. Fig. 2 shows an example.

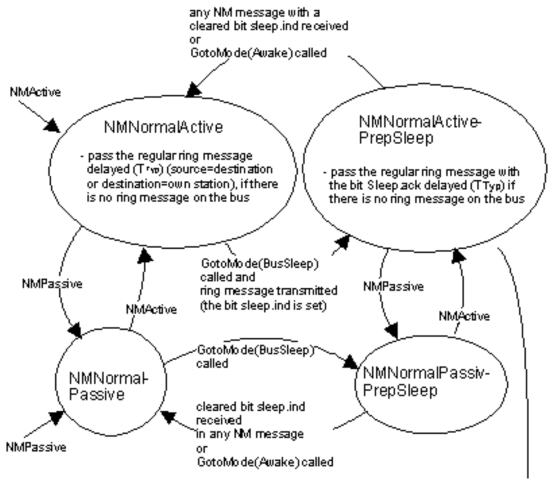

any NM message with a
cleared bit sleep.ind received
or
GotoMode(Awake) called
NMActive
NMNormalActive
- pass the regular ring message delayed (T Typ) (source=destination or destination=own station), if there is no ring message on the bus
NMNormalActive-PrepSleep
- pass the regular ring message with the bit Sleep.ack delayed (T Typ) if there is no ring message on the bus
NMPassive
NMActive
GotoMode(BusSleep) called and
ring message transmitted
(the bit sleep.ind is set)
NMPassive
NMActive
GotoMode(BusSleep) called
NMNormal-Passive
NMNormalPassiv-PrepSleep
cleared bit sleep.ind received
in any NM message
or
GotoMode(Awake) called
NMPassive

Fig. 2: part of OSEK/NM specification

A first approach to deriving the test cases is the coverage of each transition from one state to another with at least one test case. Further there is a need to test each condition which leads to such a transition. The test cases derived by this way are the minimum quantum of test cases which must be executed. This quantum of test cases is also commended by a group which deals with methods and tools for the validation of OSEK/VDX - based distributed architectures (Modistarc).

Fig. 3 shows an example for one test case, which covers the transition from 'normal state' to 'BusSleep state' under a certain condition.

3.2.1.2 Transition to NMBusSleep: NMNormal / NMActive / Initiator Node of Residual System

Initial State	SystemConfiguration	
	- Number of Nodes in Residual System	: 1
	- Composition	: residual system (c&s node) + IUT
	Net Management Settings	
	- Operating Mode	: 1
	- Local Management Settings	: networkstatus.bussleep = 0
	States of Net Management	
	- Main State	: NMNormal
	- Parallel State	: NMActive
	Residual System + IUT	: logical ring existing networkstatus.bussleep(residual system) = 0
Test Steps	- residual system GotoMode(BusSleep) - IUT GotoMode(BusSleep)	
Response	- IUT must send next ring message with bit sleep.ind = 1 - verification NMBusSleep NMActive after $T_{WaitBusSleep}$ after receiving ring message with bit sleep.ack = 1 - NM must call ApplBusSleep - status byte according to specification - NM must call ApplRingTimeElapsed according to specification	

Fig. 3: test case 3.2.1.2 Transition to NMBusSleep: NMNormal / NMActive / Initiator Node of Residual System

c&s however has perceived - and experience has confirmed - that this 'formal' methodology for deriving test cases is not sufficient: as implementers typically try to 'optimize' by for instance saving local variables for more than one state environment, additional tests must be defined checking more than the direct neighbors of the states. Therefore when deriving the test cases sequences of successive state transitions must be taken into account. Another important point which must be considered when deriving the test cases is the 'real' environment of an implementation. How does the NM under test cooperate with other NM implementations? Which effects has a high bus load to the functionality of the NM implementation under test? Which effects have bus failures e.g. open wire or short circuit of a bus line to the implementation? The test specification developed by c&s comprises test cases which take account into all these point of views:

- each transition from one state to another is covered by at least one test case
- each condition which leads to such a transition is at least covered with one test case
- test cases considering at least the pre-state n-1 when testing the transition from state n to state n+1

- test cases considering the 'environment' of an implementation as bus failures, number of nodes in a system, wrong data, etc.

<u>Tester Architecture</u> - The testing architecture is a black box testing architecture: the functionality of the implementation under test (IUT) is observed and controlled at external points of the IUT, the details of the respective NM implementation are not visible. The tester architecture is shown in Fig. 4.

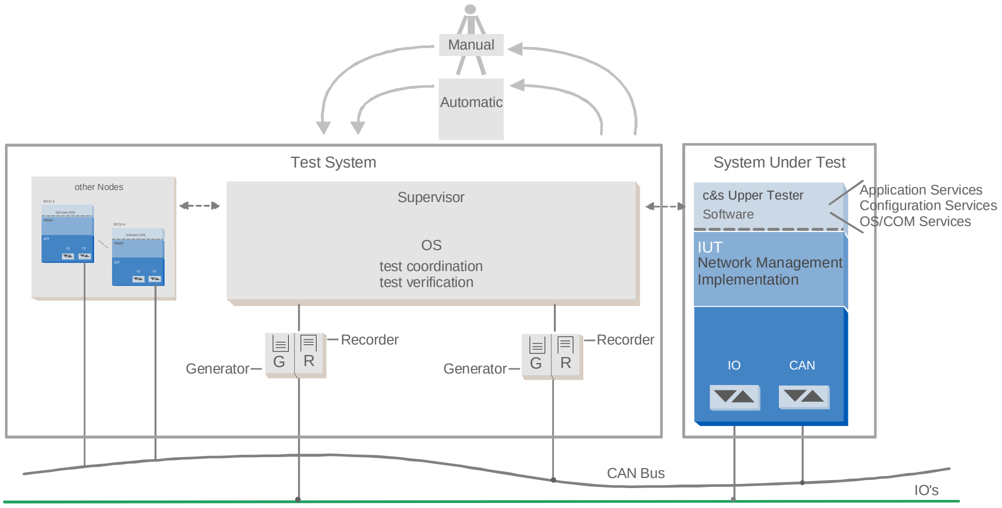


Fig. 4: tester architecture for OSEK/VDX network management tests

The node with the NM implementation under test consists of a communication chip (e.g. CAN) with µC, the implementation under test and a specific Upper Tester software developed by c&s. The realization of the supervisor comprises one node with a enhanced emulation of NM, a tester software to coordinate the tests, a program for automatically verification of the tester and generation of the test report, an user interface for starting one/several/all tests, observing test execution and doing several settings. The residual system comprises a dedicated number of nodes, each containing a communication chip (e.g. CAN) with µC, an NM implementation and a specific software developed by c&s. The implementation of the tester was realized with future oriented tools such as Matlab/Simulink/Stateflow in conjunction with realtime hardware. In order to execute the tests c&s adapts its test system to the actual hardware of the processor on which OSEK NM is implemented.

<u>Conclusion OSEK/VDX NM Tests</u> - Due to the enhanced derivation of test cases on the firm basis of experience the test system developed by c&s detects more faults than others. The test cases derivated by c&s not only check the conformance of an implementation to its specification. Therefore potential faults and loopholes in a specification are detected when deriving and executing the test cases. Although the specification of the OSEK/VDX NM is given in a rather precise and nearly 'formal' form there may be a need to complete the specification. This interaction between 'specifying an implementation' and 'testing an implementation' is exactly the way that leads to a precise and tighter specification, which is the condition precedent to interoperability.

CAN PROTOCOL TESTS - The CAN protocol is a CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) protocol that is targeted for automotive real-time applications. Many European cars use the CAN protocol for in-vehicle data communication.

As shown in Fig. 1 the protocol module, in this case a CAN protocol implementation, is part of layer 2 of the model; according to this model the interfaces of the protocol module are the following ones:

upper services to:

- OSEK / VDX NM
- OSEK / VDX COM
- Application

lower services of:

- physical layer (PMA/MDI - Physical Medium Attachment / Medium Dependent Interface)

The c&s group is worldwide no. 1 in testing of new CAN chip implementations. More than 15 man-years have been put so far into the development of the test technique. Over 100 tests have been done on emulated CAN versions (FPGA, Quickturn boxes, etc.), 1st and 2nd silicon and on software drivers for CAN. Almost any of the major semiconductor manufacturers' chips have been tested by c&s group. There are 5 CAN test systems at c&s site. Tests are performed on various levels:

- ISO standard test
- c&s extended tests on:
  - ISO tests
  - processor interface functionality
  - processor interface performance
  - robustness = 24 hours random tests
- customer coaching to discuss and locate problems
- authentification, if required by customer

Derivation of Test Cases - The International Standard ISO 16845 specifies the methodology and the abstract test suite necessary to check the conformance of any CAN implementation to the harmonised CAN specifications in ISO 11898 and CAN Specification - Bosch - Version 2.0.

Standard conformance tests are performed on the assumption of a standardized simplified receive and transmit register interface between the CAN device under test and the Upper Tester. This simplification never corresponds to a real CAN implementation. But obviously the real CAN register implementation with their corresponding control bits have a great influence on the proper function of a CAN device. That is why the c&s group always is requested to perform additional c&s group defined register tests.

As a result a set of additional 'de facto' standard register test cases was developed. Each of these extended register test cases is dedicated to perform specific checks on special characteristics of CAN registers. For solutions comparable to the Basic CAN architecture, there are different implementations to be checked such as transmit buffer registers with/without internal prioritization, different write capabilities onto the transmit register queue which influence heavily the overall arbitration process and thus the latency times of message transfers, etc. Concerning CAN solutions comparable to the Full CAN architecture, specifically adapted checks must be executed to test the individual solution of structure and number of transmit and receive registers, the kind of masking capabilities, the access techniques to the registers, etc. There are various solutions for the status and control registers which must be checked such as receive/transmit error counters, time stamps, transmission success checks, interrupts, non unified position and definition of status/control bits, etc. Following a short extract of a test list for the Processor Interface CAN Conformance Tests.

- RECEIVE INTO MAILBOX 0
- ABORT MAILBOX X
- TRANSMIT MAILBOX PRIORITY AND CAN BUS ARBITRATION
- ACCEPTANCE FILTER STANDARD FRAME, 1ST PATTERN
- BUS RECEIVE OVERLOAD WARNING
- ENTERING SLEEP MODE DURING TRANSMISSION
- CHECK ALL SPECIAL FEATURES OF YOUR CAN IMPLEMENTATION WHICH WERE NOT YET TESTED.

Tester Architecture - The execution of the tests is based on the 'OSI coordinated test method' as specified in the standard ISO

9646. The implementation under test (IUT) typically is stimulated by the Test System, respective the Lower Tester through the underlying Service Provider. The stimulation consists of the test steps of the test cases. The IUT then undergoes the actual checking procedure: It processes the stimulus in conjunction with a so called Upper Tester, representing the potentially lacking higher layers complementing the IUT to a complete system module. The Implementation under test responds to the Tester with the result of the processed stimulus. The tester typically then compares the received information with the expected result, which is defined by the standard application specification of the Implementation under Test. If okay, the test is passed, the IUT is conformant to it's specification with respect to that test case. The tester proceeds to the next test case. If not okay, the test has failed. The IUT is not conformant with respect to that test case., corresponding actions need to be taken.

The test system structure as depicted in Fig. 5 is a direct translation of the OSI Coordinated Test Method. The Implementation under Test – the 'new' CAN device – is connected to the Tester through the so called stress box, representing the Underlying Service Provider. As such the stress box connects the signals between the stimulation and analyzing units and the RX and TX pins of the CAN device. The Tester – respectively the Lower Tester – consists of a standard PC with a Windows™ operating system with off the shelf standard data generators and logic analyzers attached to it. The analyzer and data generator are responsible for the execution of the real time critical parts of the tests. The non real time critical parts of the conformance tester is implemented in software. On the other side there is the Implementation under Test – the 'new' CAN device – with it's Upper Tester represented by a micro controller and the related upper tester specific software.

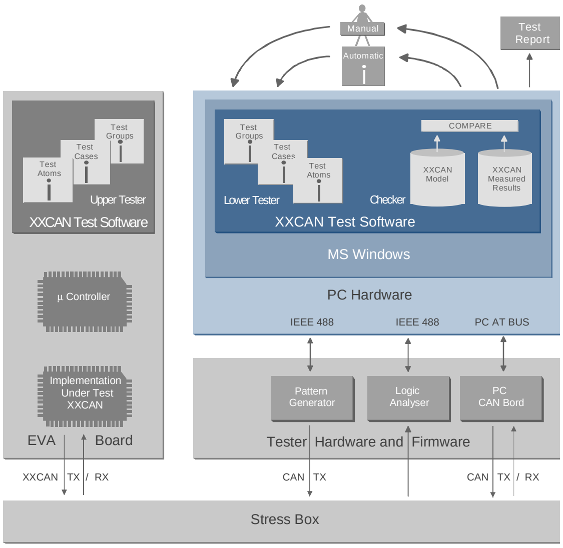


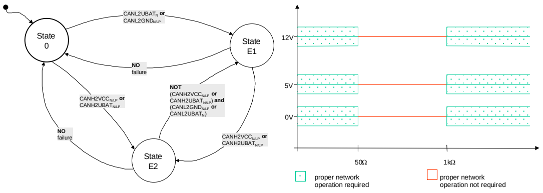
Fig. 5: tester architecture for CAN protocol Tests

<u>Conclusion CAN Protocol Tests </u> - In general, the future will show us more and more critical constellations which should be observed by a test system. Hence the amount of test cases will be incremented continuously. This new test cases arise from the inexact protocol specification. The new test cases should be assumed to the test specification, and finally the inexact protocol specification must be corrected to prevent further implementation errors.

FAULT TOLERANT CAN TRANSCEIVER TESTS - The transceiver is the interface between the analog signal transmission over the physical communication media and the digital 'world' towards the higher OSI-layers. The latter could comprise – as it is the case for CAN – lines for digital signals such as Receive_Data, Transmit_Data, Error_Signal, etc. Towards the physical media there may be analog lines such as CAN_High, CAN_Low, etc. The redundancy of these lines is used for differential mode signal transmission which provides a good means for high common mode noise injection immunity. In case of a failure – short circuit or broken wire – the redundancy is used together with the failure detection mechanism of the fault tolerant transceiver, to switch off the defect line, communicate with a reduced

signal to noise ratio and signal the error to higher layers. Finally there are power supply lines for supplying the component with energy. There are means provided to switch the component from normal mode into low power or power off mode. For more details refer to [4].
c&s is cooperating in a group of automobil manufacturers - DaimlerChrysler, BMW, Audi, Volkswagen, PSA - to specify and implement the world wide first and only conformance tests for a new generation of fault tolerant CAN transceivers.

Derivation of test cases - The specification of the fault tolerant CAN transceiver is given in natural language. Some 'formal' specification means are applied partially for instance state diagrams and tables to describe the behavior of the physical media failure detection, see Fig. 6.



State 0
State E1
State E2
CANL2UBAT$_N$ or CANL2GND$_{N/LP}$
NO failure
CANH2VCC$_{N/LP}$ or CANH2UBAT$_{N/LP}$
NOT (CANH2VCC$_{N/LP}$ or CANH2UBAT$_{N/LP}$) and (CANL2GND$_{N/LP}$ or CANL2UBAT$_N$)
NO failure
CANH2VCC$_{N/LP}$ or CANH2UBAT$_{N/LP}$
12V
5V
0V
50Ω
1kΩ
proper network operation required
proper network operation not required

Fig. 6: Transceiver Failure detection state diagram with table specifying the operational areas for short circuits

Because most parts of the specification are given in natural language there may be many faults and ambiguities in the specification and consequently in the implementation. Therefore exhaustive conformance testing is a must. To ensure that a high share of faults in the specification and the implementation is detected the application of a systematically method is essential when deriving the test cases. The application of the 'SOVS-method' developed by c&s led to a SOVS comprising all so far known possible operational conditions under which the transceiver is expected to be functional in the specified way. On the firm base of experience of c&s the SOVS for testing a transceiver is derived as shown in Fig. 7. The experience has been gained from practical applications at the automotive companies sites' and the experience in testing and organizing tests within the research work at the c&s group. For further details refer to [5], [6].

The tables given in Fig. 7 already show how large the number of test cases directly derived from SOVS would be. Undoubtedly it were impossible to execute these test in a practically acceptable amount of time.

Transceiver-SOVS =	**{ System Configuration } x { Communication } x { Power Supply } x { GND Shift } x { Op. Modes } x { Failure }**	
System Configuration	Baud rate	... 5 k Bd ...... 125 kBd
	Termination	Calculated total termination = ... 100Ω ......
	Topology	Bus, ring, star, mesh, ...
	Composition	Homogeneous, heterogeneous (ratio), ...
	Number of nodes	1, 2, ...40, ...
	Environmental conditions	Temperature (...20°C...), moisture, shock, ...
	......	
Communication	Nodes' interaction	- Logical ring: - node x receives token - node x transmits token to node x+1 - after 1 cycle all nodes transmit 1 message leading to an arbitration conflict ... Arbitrary communication ....
	Identifier	Any, special, ...
	Data	Any, nodes reference, ...
	......	
Power Supply	....	
Ground Shift	....	
Operational Modes	....	
Failure	Single bus failure	- no failure - short circuit: - CL_Vx(up)@Rx with: - Vx = [... -3 V ..... + 18 V ....] - Rx = [.... 0Ω .. 50.000Ω .....] - CH_Vx(up)@Rx - CL_OW@Rx(up) .... - open circuit: - CL_OW@Rx(down) ....
	1,5 bus failures	- apply CL_BAT + CL_CH then remove CL_BAT - apply CL_GND + CL_CH then remove CL_GND ...
	2 bus failures	- apply CL_BAT + CL_CH - apply CL_GND + CL_CH ...
	n bus failures	- ...
	Location of failure	At node 1, 2, ...
	....	

Combinatorial Tests based on the Vectors	**1:6**						**2:6**						**3:6**			**..**	**6:6**
System configuration	☒						☒			☒			☒				☒
Communication		☒						☒			☒			☒			☒
Power Supply			☒						...			..			..		☒
Ground Shift				☒												..	☒
Operational Modes					☒					☒	☒		☒	☒			☒
Failure						☒	☒	☒					☒	☒			☒

Fig. 7: SOVS and variation of all vectors for transceiver tests

Therefore a second step must be applied to reduce the number of tests by selecting 'carefully' a subset of tests to come to a significantly reduced Sufficiently Exhaustive Minimal Set of (conformance) Test Cases SEMSTC. From practically gained experience by the above mentioned car manufacturers and c&s group the following settings seem to be reasonable to reduce the number of test cases to the SEMSTC shown in Fig. 8.

Transceiver-SEMSTC =	*Reduction on [* { System Configuration } x { Communication } x { Power Supply } x { GND Shift } x { Op. Modes } x { Failure }*]*	
System Configuration - reduced - *(constant standard system)*	Baud rate	100 kBd
	Termination	Calculated total termination = 100Ω
	Topology	Bus
	Composition	Homogeneous
	Number of nodes	40
	Environmental conditions	Temperature, moisture, shock, ... = ambient
	......	
Communication *- reduced –* *(constant standard com)*	Nodes' interaction	- Logical ring: - node x receives token - node x transmits token to node x+1 - after 1 cycle all nodes transmit 1 message leading to an arbitration conflict
	Identifier	Respective name of node
	Data	Name of logical successor; at arbitration: no data
	......	
Power Supply *- reduced -*	....	
Ground Shift *- reduced -*	....	
Operational Modes - reduced -	....	
Failure - reduced -	Single bus failure	- no failure - short circuit: - CL_Vx(up)@Rx with: - Vx = [... -3 V ..... + 18 V ....] - Rx = [.... 0Ω .. 50.000Ω .....] - CH_Vx(up)@Rx - CL_OW@Rx(up) .... - open circuit: - CL_OW@Rx(down) ......
	1,5 bus failures	- apply CL_BAT + CL_CH then remove CL_BAT - apply CL_GND + CL_CH then remove CL_GND ...
	Location of failure	Between node 39 and 40
	....	

Combinatorial Tests based on the Vectors *- reduced -*	**1:4**	**2:4**	**3:4**	**4:4**
System configuration	Constant	Constant		
Communication	Constant	Constant		
Power Supply	☒	☒		
Ground Shift	☒	☒		
Operational Modes	☒	☒		
Failure	☒	☒ ☒ ☒		

Fig. 8: SEMSTC and restricted variation of vectors for transceiver tests

<u>Tester Architecture -</u> Correspondingly to the derivation of test cases the resulting tester architecture depicted in Fig. 9 and a detailed definition of for example a Short Circuit test case – Fig. 9 – would result.

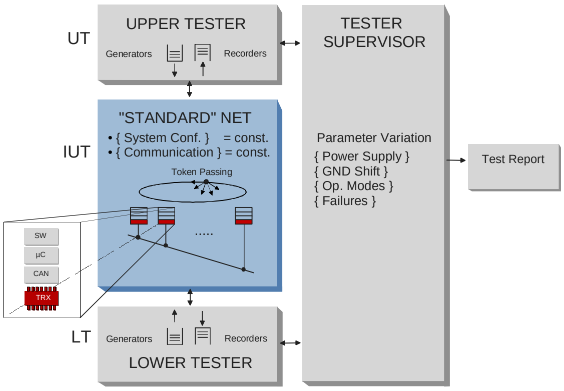


Fig. 9: tester architecture for fault tolerant CAN transceiver tests

<table>
<tr><td>Constants</td><td><b>Power Supply = 12V / GND Shift = 0V / Op. Mode = Normal</b></td></tr>
<tr><td></td><td>Test procedure : Short Circuit Failure (CL_Vx, CH_Vx)</td></tr>
<tr><td></td><td>Operating area : Short Circuit Operating Area</td></tr>
<tr><td>Initial State</td><td>System Configuration : constant as specified in <i>Standard Net</i>
Communication : constant as specified in <i>Standard Net</i>
Op. Mode : Normal Mode
Power Supply : 12V
GND Shift : 0V
<u>resistor range for error generator R/U:</u>
rx_start : 0Ω
rx_stop : 50KΩ
rx_next : depending on steps
<table>
<tr><td colspan="2">range</td><td>step</td></tr>
<tr><td>0Ω -</td><td>10Ω</td><td>1Ω</td></tr>
<tr><td>10Ω -</td><td>50Ω</td><td>5Ω</td></tr>
<tr><td>50Ω -</td><td>250Ω</td><td>10Ω</td></tr>
<tr><td>250Ω -</td><td>1.000Ω</td><td>50Ω</td></tr>
<tr><td>1.000Ω -</td><td>10.000Ω</td><td>1.000Ω</td></tr>
<tr><td>10.000Ω -</td><td>50.000Ω</td><td>10.000Ω</td></tr>
</table>
<u>voltage range for error generator R/U:</u>
vx_start : 16V
vx_stop : -3V
vx_next : depending on steps
<table>
<tr><td colspan="2">range</td><td>step</td></tr>
<tr><td>16V -</td><td>-3V</td><td>0.1V</td></tr>
</table>
</td></tr>
<tr><td>Test Steps</td><td>execution of communication as specified in <i>Standard Net</i></td></tr>
<tr><td>Response</td><td>test results must match the operating areas as defined for short circuit failures</td></tr>
</table>

Fig. 10: test case 4.3.6.7: CH_Vx(down)@Rx

<u>Conclusion Fault Tolerant CAN Transceiver Tests</u> - Practical experience will show whether the above choice of SEMSTC is appropriate to check the insufficiently specified CAN Transceivers for interoperability. Probably real applications will lead to some unexpected behavior which will need further analysis and which will then lead to the described formal expansion of the vector space and as well as more refinement and formal description of the component specification.

Summary

A first approach to tighter and more precise specifications is the description of protocol features in a 'formal language'. If a specification is given in a completely formal form formal checks on completeness, in-ambiguity, etc. would be applicable, for further information about such formal methods refer to [15]. The application of formal methods for deriving test cases on a formal specification does not redundantize the application of empiric methods as they have been discussed in the present paper. Empiric and formal testing approaches are not antagonist. They are further more complementary. Therefore a reliable methodology for design and test should integrate both of them. On one hand, because of the know-how of test experts cannot be qualitatively compared with the capabilities of formal methods which are 'artificially' and systematically applied to a generally incomplete specification. So an interesting approach would be to improve the formal methods with data provided by experts. On the other hand, the empirically defined test sequences, in which the community is confident in, can be analyzed in a process of applying formal methods.

It is obvious that a confident test could not be performed without a complete and non-ambiguous specification even if it is done at a high level of abstraction. This specification is useful to obtain relevant and confident test cases. Furthermore, if the specification is detailed enough it is possible to perform a simulation process in order to evaluate the coverage of executive path of the specification. An high path coverage allows the confidence into test process to be increased.

Based on that some next steps concentrate on the integration process based on existing tools. Such a integration process will put together the test philosophies as well as the currently not harmonized and not cooperating tools on network simulation/emulation and process simulation.

References

[1] W. Lawrenz: Networked Systems High Level Design & Test Philosophy and Tools; SAE Conference and Show Detroit, paper 950296, 27.03. - 03.03.95

[2] W. Lawrenz, editor: CAN - Controller Area Network from Theories to Application; Springer Verlag, 1997, ISBN 0-387-94939-9

[3] ISO 11898 Road Vehicle – Controller Area Network (CAN), part 1; International Standard ISO 11898, 1999

[4] Specification of a Fault Tolerant CAN Transceiver; GIFT Working Group at Fachhochschule Wolfenbuettel, c&s group, Germany, 2000

[5] Test Specification for a Fault Tolerant CAN Transceiver; ICT Working Group at Fachhochschule Wolfenbuettel, c&s group, Germany, 2000

[6] M. Scheurer: Entwicklung einer Methode zur systematischen Ableitung von Testfällen für Konformitätsprüfungen in der Informationstechnologie ; Diploma Thesis at Fachhochschule Wolfenbuettel, c&s group, Germany, July 2000

[7] Abramovici, M.A. Breuer, A.D. Friedman: Digital testing and testable design; Computer Science Press, 1990.

[8] A.L.Courbis, J.F. Santucci, N.Giambiasi : Automatic Behavioral Test Pattern Generation for Digital Circuits; 1st IEEE Asian Test Symposium, Hiroshima, Japan, November 1992, pp. 112-117

[9] S.J. Chandra, J.H. Patel: Experimental evaluation of testability measures for test generation; IEEE Trans. On CAD, Vol. 8, N°1, pp.93-97, January 1989.

[10] Z. Manna, A. Pnueli: How to cook a temporal proof system for your pet language; Report n°STAN-CS-82-954, Department of Computer Sciences, Stanford University , USA, 1982

[11] K.T. Cheng, A.S. Krishnakumar: Automatic functional test generation using the Extended Finite State Machine Model; 30th ACM/IEEE Design Automation Conference, USA, 1993

[12] M. Larnac, V. Chapurlat, J. Magnier, B. Chenot: Formal Representation and Proof of the Interpreted Sequential Machine Model; EUROCAST'97, LNCS 1333, Springer, Las Palmas de Gran Canaria, Spain, 1997

[13] J. Magnier: Représentation Symbolique et Vérification Formelle de machines séquentielles; State Thesis University of Montpellier II, France, 1990

[14] K. Hoffmeister: Applikationsbedingte Kommunikationsanforderungen im verteilten Kraftfahrzeug-Echtzeitsystem und deren Testbarkeit; Diploma Thesis at Fachhochschule Wolfenbuettel, c&s group, Germany, July 2000

[15] W. Lawrenz, Anne-Lise Courbis, Janine Magnier: Car System Conformance Testing, International Conference Systems Engineering and Information&Communication Technology, Nimes TIC 2000, September 2000

c&s - communication and systems group
University of Applied Science
Salzdahlumer Straße 46/48
D-38302 Wolfenbüttel
Germany

Tel. +49 5331 939-642
Fax. +49 5331 939-641

W.Lawrenz@fh-wolfenbuettel.de
www.cs-group.de