Diagnosis of V2X communication via evaluation modules and textual rule sets
Presented by Tim Ruß
Motivation

Paris, 16-18 October 2018
ifak: Institute for Automation and Communication

- Applied research
- Test laboratories
Examples for our communication domains

Factory Automation

Vehicle-to-X (also: Car2X)
Manual process: Checking Wireshark records

- Search for connection / startup sequence
- Search for errors (alarms, connection releases...)

→ Demand to automate such processes
Agenda

- Scope and methods
- Approach
- Use case
- Summary
Scope and methods
Scope and terminology

- Network communication of distributed (computer) systems:
  - OSI reference model
  - Protocol Data Units of higher levels: “messages”
- “Checking” those messages:
  - Verification: If a specification is correct according to the design. Prior to the implementation.
  - Validation: Check if customer expectations fulfilled
  - Conformance check: Check if an implementation matches the underlying specification
Testing methods for software (e.g. source code)

- **Architecture:** Test system around System Under Test (SUT)
  - Stimulation via input parameters
  - Check of states or return values
  - E.g. Unit Tests
- **Coverage criteria**
  - E.g. check program execution paths
Testing network protocols

- Fuzzing: Generate random input/network data
- Conformance check
  - Replace other protocol layers with test system (upper tester, lower tester)
  - Run subsequent test cases...
    - Difficult for already running systems without test interfaces
    - Demand for diagnosis after commissioning
Protocol specifications given in different formats

Example: Generation frequency of V2X status messages (CAMs) as plain text:

6.1.3 CAM generation frequency management for vehicle ITS-Ss

The CAM generation frequency is managed by the CA basic service; it defines the time interval between two consecutive CAM generations. Considering the requirements as specified in ETSI TS 101 539-1 [i.8], ETSI TS 101 539-2 [i.9] or ETSI TS 101 539-3 [i.10] the upper and lower limits of the transmission interval are set as follows:

- The CAM generation interval shall not be inferior to \( T_{\text{GenCamMin}} = 100 \text{ ms} \). This corresponds to the CAM generation rate of 10 Hz.
- The CAM generation interval shall not be superior to \( T_{\text{GenCamMax}} = 1000 \text{ ms} \). This corresponds to the CAM generation rate of 1 Hz.
Formal graphical descriptions for network protocols (1)

Time Flow Charts

<table>
<thead>
<tr>
<th>Service User</th>
<th>Service Provider</th>
<th>Service User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect Request</td>
<td></td>
<td>Connect Indication</td>
</tr>
<tr>
<td>Connect Confirm</td>
<td></td>
<td>Connect Response</td>
</tr>
</tbody>
</table>

Sequence Charts

Controller
- Connect Request
- Connect Response
- Write Request
- Write Response OK
- Control Request PE
- Control Response PE OK
- Control Request AR
- Control Response AR OK

Device
- loop (min=1, max=*)
- Connect Response OK
Formal graphical descriptions for network protocols (2)

Finite State Machines

Petri nets

Aim: Check requirements and errors at the same time (like parallel test cases, but not all have to be executed) \(\rightarrow\) diagnosis
Approach
Approach: Using a Petri net to compare network messages against requirements.
Case 1: Check a sequence of messages

Example: “A connect request is followed by a connect response.”
Case 2: Check parameter values within messages

Example: “The ErrorCode of the response must be 0.”

```
Response → [ErrorCode == 0] (success)
```

```
Response → [ErrorCode != 0] (error)
```
Whole toolchain & workflow

- User/Tester
- Textual "rules"
  - Currently own DSL
- Parser
  - ANTLR
- Petri net generator
  - C++, (Python)
- Firing transitions
- Verdicts
- System Under Test
- Messages
- Decoder
  - C++, PCAP

ANTLR, C++
Creation of structures via subnets

- Structures for control flow, e.g. from UML Sequence Diagrams
- Considered most important:
  - Alternatives ("alt")
  - Parallels ("par")
  - Loops ("loop")
  - Errors ("not")
- Start and end places as interfaces between structures
Description of net creation with Python

<table>
<thead>
<tr>
<th>Text</th>
<th>Mathematical</th>
<th>Python</th>
</tr>
</thead>
</table>
| „A net $N$ consists of places $P$, transitions $T$ and flow relations $F$.“ | $N = (P; T; F)$ | class Petrinet:
  def __init__(self):
    self.places = []
    self.transitions = []
    self.arcs = [] |
```python
def createPar(self, interfaces, operandCount):
    startTrans = self.petrinet.addTransition()
    self.petrinet.addArc(interfaces.getStartPlace(), startTrans)
    endTrans = self.petrinet.addTransition()
    self.petrinet.addArc(endTrans, interfaces.getEndPlace())
    operands = []

    for i in range(0, operandCount):
        operandStartPlace = self.petrinet.addPlace()
        self.petrinet.addArc(startTrans, operandStartPlace)
        operandEndPlace = self.petrinet.addPlace()
        self.petrinet.addArc(operandEndPlace, endTrans)
        operandInterfaces = NetInterfaces(operandStartPlace, operandEndPlace)
        operands.append(operandInterfaces)

    return operands
```

possible interfaces to other structures
Use Case

Paris, 16-18 October 2018
Use Case: Vehicle-to-X communication (V2X)

- Vehicles send out status messages periodically (Cooperative Awareness Messages, CAM)
- Monitoring modules shall capture and check messages according to user defined rules
User-selected requirements for CAM payload

- User defined: StationType == 11 (Tram)
- From CAM protocol specification: $1 \text{ Hz} \leq f \leq 10 \text{ Hz}$
Check the vehicle role (1)

“Every CAM shall have a station type of 1.”
Check the vehicle role (2)

“Every CAM shall have a station type of 1.”
Check the message interval (1)

“The transmit frequency shall always be between 1 Hz and 10 Hz.”

\[ \Delta t = t_{\text{CAM}(n)} - t_{\text{CAM}(n-1)} \]
Check the message interval (2)

“The transmit frequency shall always be between 1 Hz and 10 Hz.”
Check parameter and frequency

Diagram:

- Global start
  - Parameter loop start
    - ...
      - Loop end
  - Frequency loop start
    - ...
      - Loop end
  - Global end (all processes finished)
Paris, 16-18 October 2018

Summary
Summary

- Check several requirements and detect errors at the same time → Diagnosis
- User-selected requirements as “rules”
- Petri subnets with interface places
- Connect subnets
- React to decoded messages (move tokens), create verdicts
Thank you for your kind attention
Tim Ruß
tim.russ@ifak.eu
import <denm_cancelation.components> as stations

Rule {
    name: "DENM Cancelation";
    description: "Abkündigung einer DENM prüfen"
    Declaration {
        Signal {
            name: "V2X_DENM";
            use DENM.causeCode as cause;
        }
    }
    sequence {
        Loop [3] {
            Message CyclistWarning(from == rsu and to == car1);
            Message CyclistWarning(from == rsu and to == Any);
        }
        Message CyclistWarning(from == rsu and to == Any and canceled == true);
    }
}