

## Developing, Simulating and Testing for Ethernet System

Database, Security and XiL Testing

## Agenda

1. Data source vCDL Beyond arxml for Ethernet and SOA

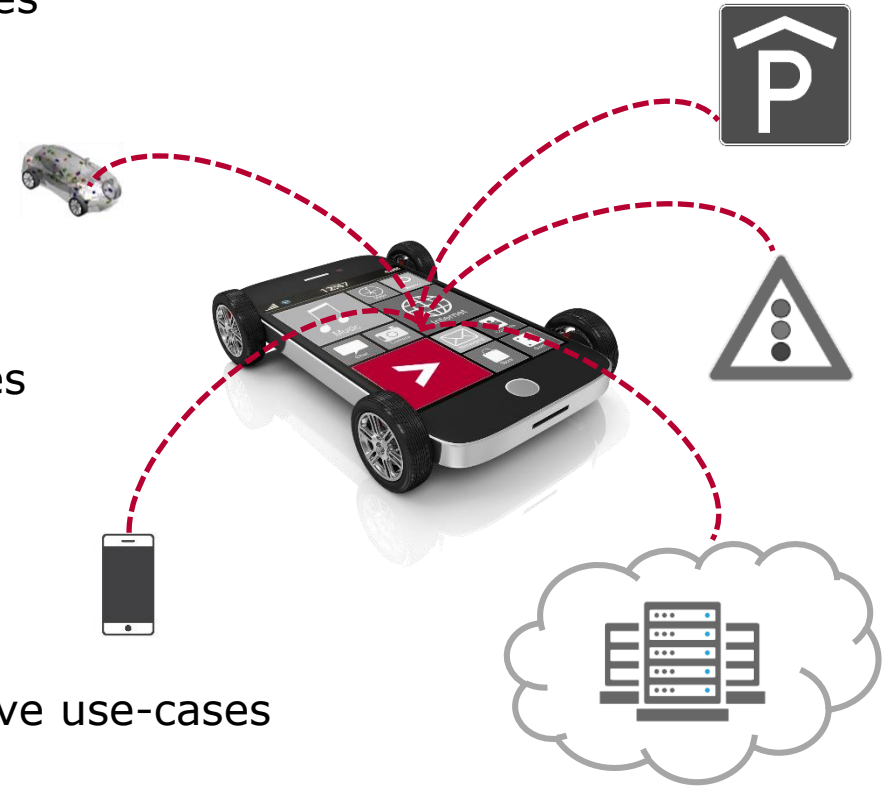
2. Network Interface for Automotive Ethernet

3. Communication Testing, Security and Analysis

4. More SOA – More Software – More Testing

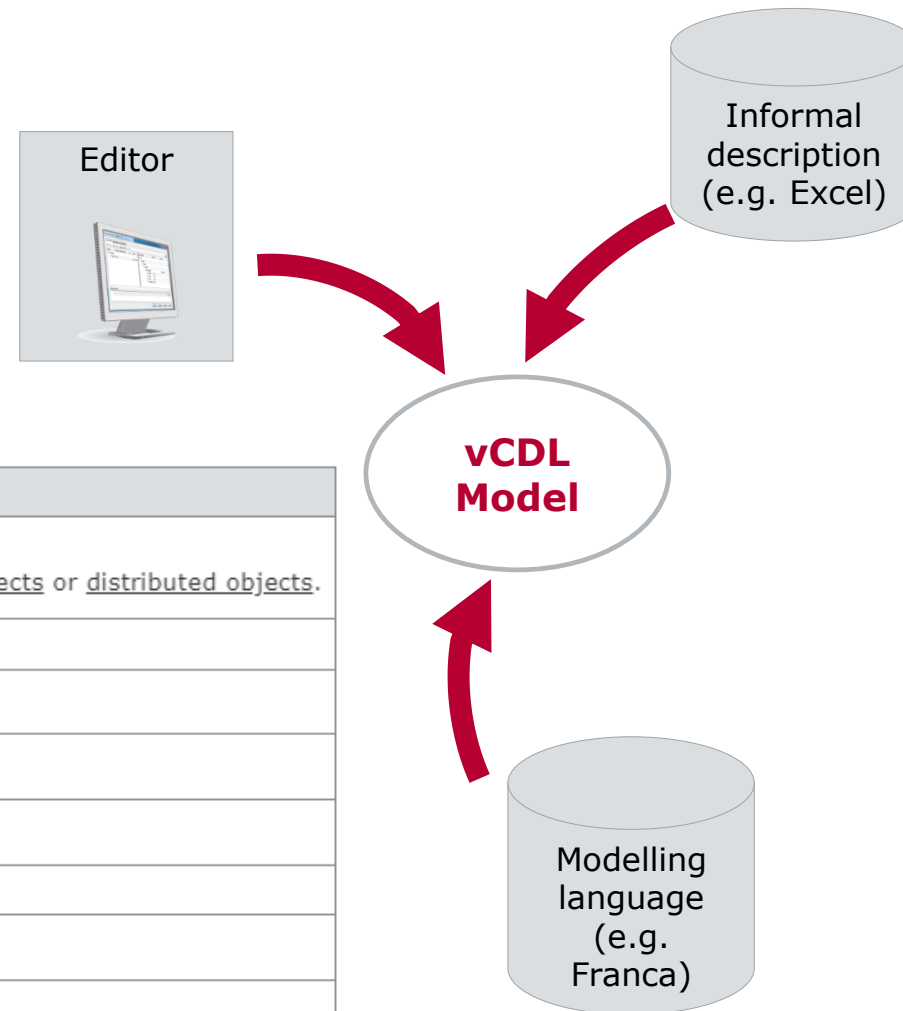
## vCDL – A Domain-Specific Language for Service-Oriented Communication

- ▶ Autonomous driving is challenging for current E/E architectures
  - ▶ Interconnection of onboard and offboard systems
  - ▶ Huge number of protocols involved
    - > CAN, LIN, FlexRay, Ethernet, Car2X
  - ▶ Continuous software updates require a flexible architecture
- ▶ Service-oriented Architectures (SOA) address these challenges
  - ▶ Definition of services independent of underlying protocols
- ▶ vCDL at a glance
  - ▶ vCDL = **V**ector **C**ommunication **D**escription **L**anguage
  - ▶ Unified modelling of SOA for automotive and non-automotive use-cases
  - ▶ Very lightweight compared to AUTOSAR
  - ▶ Support of different communication protocols
  - ▶ Convenient editor with syntax highlighting and code completion available



## vCDL Use Cases

- ▶ Simulation & test without AUTOSAR descriptions, e.g.
  - ▶ Rapid prototyping, no description available
  - ▶ Development of tests with an informal Excel description
- ▶ Extension of AUTOSAR based descriptions, e.g.
  - ▶ Adding further services



Previous Element / Previous Workflow	New Element / New Workflow
<b>Database Editor</b> <ul style="list-style-type: none"> <li>▶ Creating/changing/deleting PDUs, etc.</li> </ul>	<b>Model Editor</b> <ul style="list-style-type: none"> <li>▶ Enables creating, deleting and changing <u>communication objects</u> or <u>distributed objects</u>.</li> </ul>
<b>Network Node:</b> <ul style="list-style-type: none"> <li>▶ Settings (e.g. TCP/IP configuration)</li> <li>▶ Adding CAPL code, etc.</li> <li>▶ Interaction layer configuration</li> </ul>	<ul style="list-style-type: none"> <li>▶ <b>Model Editor</b> (Bindings)</li> <li>▶ <u>Application models</u> in Communication Setup</li> <li>▶ <b>Model Editor</b> (PDU Timings, Bindings)</li> </ul>
<b>Network:</b> <ul style="list-style-type: none"> <li>▶ Assign application channel</li> <li>▶ Add database/ Import wizard</li> </ul>	<ul style="list-style-type: none"> <li>▶ <b>Model Editor</b> (Bindings)</li> <li>▶ <u>Data sources</u> in Communication Setup</li> </ul>

## Core vCDL Language Design

Core Language (syntactically similar to C++):

- ▶ Human readable, ease of use, mergeable
- ▶ As few keywords as possible (but not less)
- ▶ Reasonable defaults for optional settings
- ▶ Support versioning and backward compatibility
- ▶ Consistent with programming languages
- ▶ Split models across multiple files
- ▶ Complex data types:
  - ▶ array, list, struct, union, map, enum, bit field
- ▶ Primitive data types:
  - ▶ bool, [u]int[1-64], float, double, string, time
- ▶ Physical and textual encodings for numeric data types



## Distributed Objects

- ▶ Logical objects, **representing communicating entities** in a distributed system
  - ▶ For example: ECUs, applications, sensors, ...
  - ▶ Similar to objects in common programming languages: C++, Java

### Characteristics

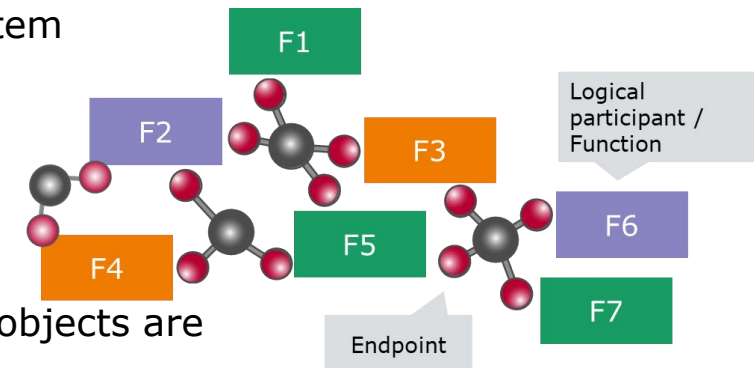
- ▶ Independent from technologies and paradigms until properties of distributed objects are defined
- ▶ Properties of distributed objects depends on the use case (e.g., ADAS, IoT, SIL, etc.)
- ▶ Can be used to describe provider and consumer side independently

### Application Layer Support

- ▶ Definition of consumable data and data types (scalar and complex data types are supported)
- ▶ Definition of how data is consumed (as value, field, event, method)

### Binding Support

- ▶ Attributes for used communication technology (e.g., MQTT, SOME/IP, HTTP, etc.)



```
namespace NField
{
    struct Name
    {
        string firstname;
        string lastname;
    }

    [Binding="Abstract"]
    interface ProvidedField
    {
        [CommunicationPattern="PublishSubscribe"]
        provided field Name GetSetName;
        provided field string GetSetName;
        provided field double GetOnly { get; }
        provided field int32 NotifyOnly { notify; }
        provided field double SetOnly { set; }
        internal field string InternalField;
    }

    object FieldProvider : ProvidedField;

    object FieldConsumer : reverse<ProvidedField>;
}

```

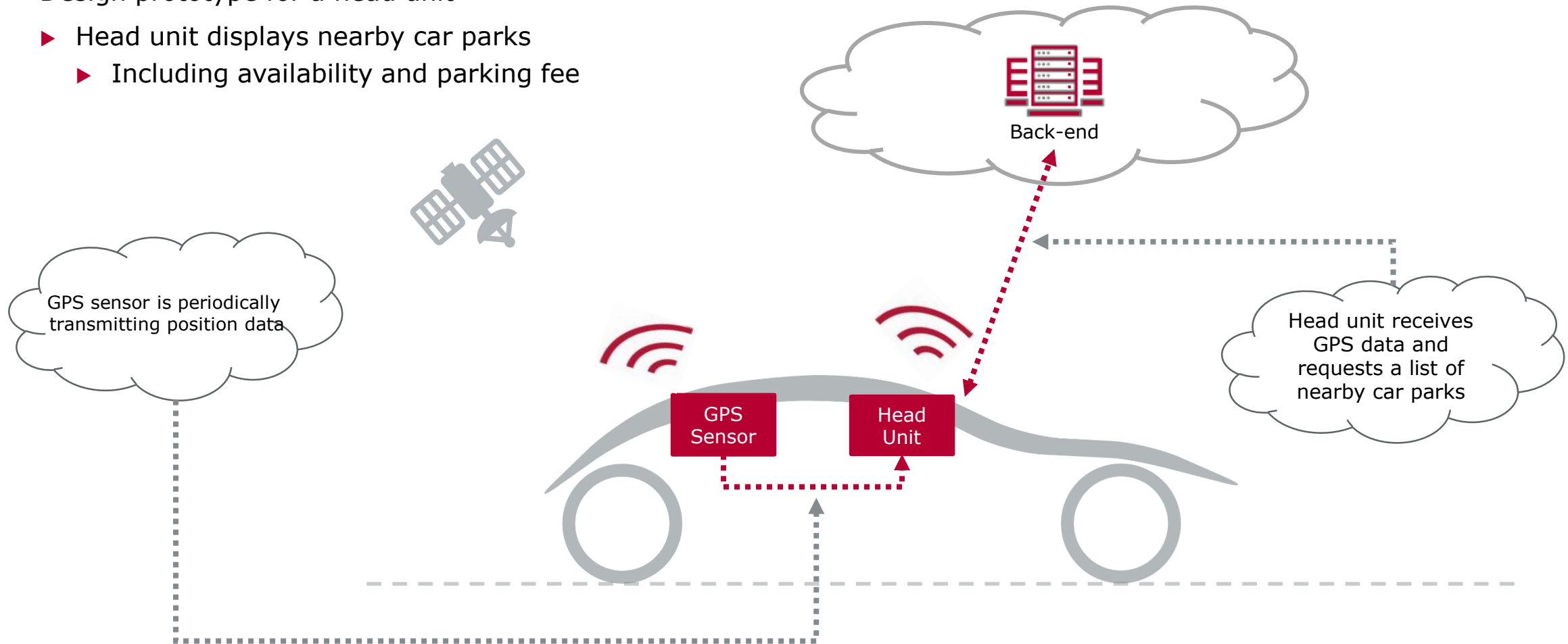
## vCDL Building Blocks

- ▶ Datatypes, e.g.
  - ▶ Basic / predefined datatypes: int, double, bool, enum, string
  - ▶ Custom defined datatypes: struct, union, list, array
  
- ▶ Interfaces
  - ▶ Definition / grouping of communication interfaces
    - > E.g. GPS sensor, Headunit, Camera
  - ▶ Communication type and direction
    - > Method: RPC, Client/Server
    - > Data: Sender/Receiver, Event-based
  
- ▶ Distributed objects
  - ▶ Instantiation of interfaces
  - ▶ Same interface may be instantiated multiple times (e.g. Front camera, rear camera)

## Application Scenario example

Design prototype for a head unit

- ▶ Head unit displays nearby car parks
  - ▶ Including availability and parking fee



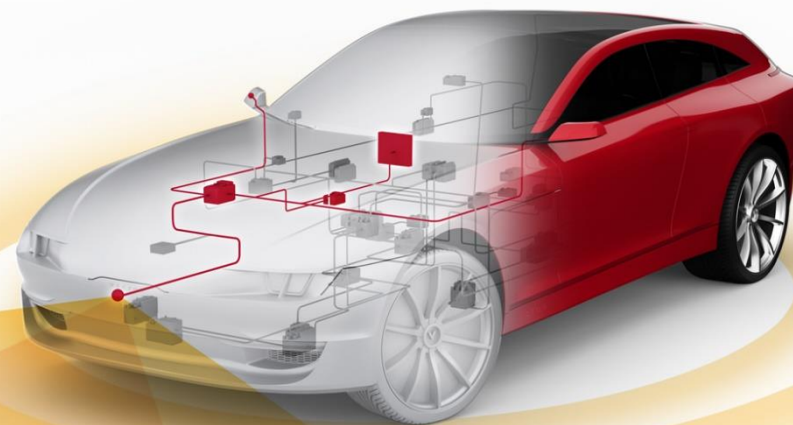


## vCDL Datatypes

```
namespace DataTypes
{
    struct Coordinates_struct
    {
        double lattitude;
        double longitude;
    }

    enum Parking_Type
    {
        CAR_PARK = 1, UNDERGROUND = 2, PARKING_SPACE = 3
    }

    struct Parking_Info_struct
    {
        Parking_Type type;
        uint32 capacity;
        double availability;
        double fee;
        Coordinates_struct position;
        string name;
    }
}
```



## vCDL Interfaces

```
interface IGPS_Sensor
```

```
{  
    provided data Coordinates_struct Coordinates;  
}
```

```
interface IHead_Unit
```

```
{  
    consumed data Coordinates_struct Coordinates;  
    consumed method array<Parking_Info_struct, 5> Get_Parking_Info(in Coordinates_struct coordinates);  
}
```

```
interface IBackend
```

```
{  
    provided method array<Parking_Info_struct, 5> Get_Parking_Info(in Coordinates_struct coordinates);  
}
```

## vCDL Distributed Objects

[*Binding* = "Abstract"]

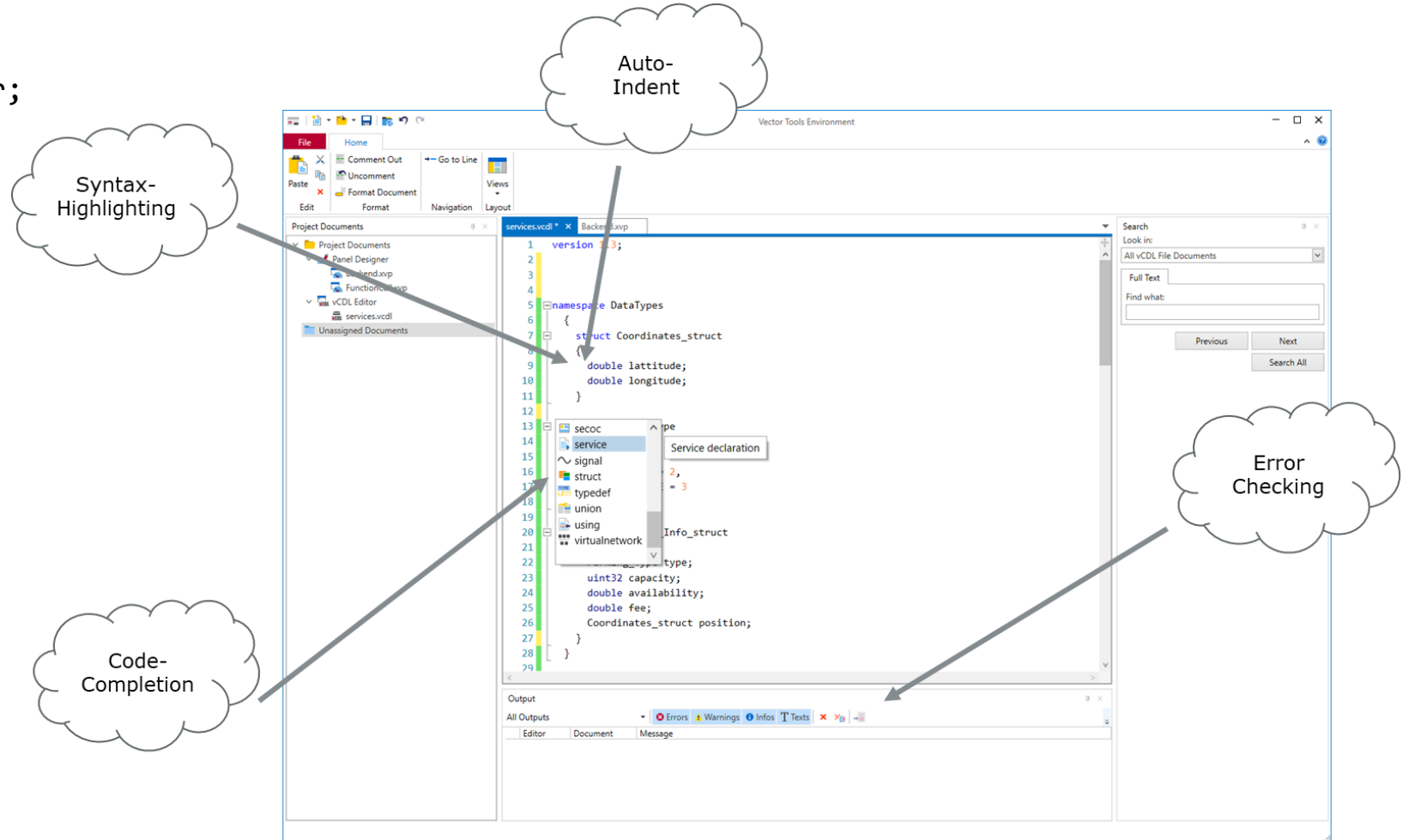
```
object GPS_Sensor : IGPS_Sensor;
```

[*Binding* = "Abstract"]

```
object Head_Unit : IHead_Unit;
```

[*Binding* = "Abstract"]

```
object Backend : IBackend;
```



# Visualization of Services

The screenshot displays the 'System and Communication Setup' interface. On the left, a tree view shows the 'System Definition' structure, including 'Endpoints' and 'Services'. A central table lists service configurations:

Name	State	Role	Binding	System Setup
GPS_Service[GPS_S...	Simulated	Provider	SOME/IP	-
GPS_Service[Head_U...	Simulated	Consumer	SOME/IP	-
Park_Info_Service[Ba...	Simulated	Provider	Abstract	-
Park_Info_Service[He...	Simulated	Consumer	Abstract	-

On the right, the 'Endpoint Properties' panel shows details for the selected 'Head\_Unit' endpoint, including its CO (GPS\_Service), Name (Head\_Unit), Role (Consumer), State (Simulated), Binding (SOME/IP), and a checked 'Generate C# API' option. Below this, a communication diagram shows two endpoints: 'Head\_Unit Simulated' and 'GPS\_Sensor Simulated' connected by a line. The bottom right panel shows a table for 'Communication Object 'GPS\_Service'' with one entry:

Type	Name	Details
Event	Coordinates	

Two callout clouds are present: 'System overview and configuration' with an arrow pointing to the left sidebar, and 'Visualization of service communication and roles' with an arrow pointing to the communication diagram.

## CANoe - Service Orientation

- ▶ Built-in “service-oriented communication” instead of “network specific elements”

The screenshot shows the CANoe Trace window with a table of communication events. The table has columns for Time, Name, Object Type, Object Detail, Dir, Src, Dest, and Data. The data is organized into a tree view on the left, showing nested objects like coordinates and ReturnValue.

Time	Name	Object Type	Object Detail	Dir	Src	Dest	Data
0.000000	GPS_Service	Service	Service State		GPS_Sensor		Available
0.000013	GPS_Service.Coordinates	Event	Subscription State		[Head_Unit]	GPS_Sensor	Subscribed
0.000000	Park_Info_Service	Service	Service State		Backend_Cloud		Available
0.000007	GPS_Service	Service	Connection State		[Head_Unit]	GPS_Sensor	Connected
0.002000	Park_Info_Service	Service	Connection State		[Head_Unit]	Backend_Cloud	Connected
0.000000							
0.000000							
0.000003	GPS_Service	Service	Connection State		Head_Unit	[GPS_Sensor]	Available
0.000011	GPS_Service.Coordinates	Event	Subscription State		Head_Unit	[GPS_Sensor]	Subscribed
0.003000	Park_Info_Service	Service	Connection State		Head_Unit	[Backend_Cloud]	Available
14.854924	GPS_Service.Coordinates	Event	Provider Value	Tx	GPS_Sensor		
	~ latitude						
	~ longitude						
	~ z						
14.855001	GPS_Service.Coordinates	Event	Value	Rx	GPS_Sensor	Head_Unit	
14.855001	Park_Info_Service.Get_Par...	Method	Call	Tx	Head_Unit	Backend_Cloud	
14.856001	Park_Info_Service.Get_Par...	Method	Call	Rx	Head_Unit	Backend_Cloud	
	coordinates						
	~ latitude						
	~ longitude						
	~ z						
14.856001	Park_Info_Service.Get_Par...	Method	Return	Tx	Backend_Cloud	Head_Unit	
14.857001	Park_Info_Service.Get_Par...	Method	Return	Rx	Backend_Cloud	Head_Unit	
	ReturnValue						
	[0]						
	~ type						
	~ capacity						
	~ availability						
	~ fee						
	position						
	~ latitude						
	~ longitude						
	~ z						
	[1]						
	[2]						
	[3]						
	[4]						

Annotations:

- Observation of events:** Points to the top of the trace table.
- Observation of method calls and returns:** Points to the 'Park\_Info\_Service.Get\_Par...' method call and return entries.
- Observation of service states:** Points to the 'Service State' and 'Connection State' entries in the trace table.

# vCDL - SOME/IP, MQTT, DDS

```
[version=1.0, serviceId=11]
service Calculator
{
    void Add(int32 operand1, int32 operand2, out
float result)
        ;
    void Subtract(int32 operand1, int32 operand2,
out float result)
        ;
    void Multiply(int32 operand1, int32 operand2, out
float result)
        ;
    void Divide(int32 operand1, int32 operand2, out
float result)
        ;

    [udpEndpoint="192.168.1.10:40000",
sdMulticastEndpoint="239.0.0.1:30490"]
    consumer Terminal;

    [simulated = false, instanceId = 1];
    provider VC121;
}
```

```
version 1.1;
namespace Home
{
    namespace Data
    {
        interface IProvidedData
        {
            [CommunicationPattern="PublishSubscribe"]
            [AutoConnect=true]
            [Topic:"Bathroom/VentilatorSpeed"]
            provided data int32 Speed;
        }
        interface IConsumedData
        {
            [CommunicationPattern="PublishSubscribe"]
            [AutoSubscribe=true]
            [AutoConnect=true]
            [Topic:"Bathroom/VentilatorSpeed"]
            consumed data int32 Speed;
        }

        [Binding="MQTT"]
        object ProvidedDataInt : IProvidedDataInt
        [Binding="MQTT"]
        object ConsumedDataInt : IConsumedDataInt;
    }
}
```

```
struct positionType {
    float longitude;
    float latitude;
}

struct statusType {
    float speed;
    bool break;
}

[Binding="DDS",
DDS::Reader::Reliability=RELIABLE,
DDS::Reader::History=KEEP_ALL]
interface vehicle
{
    // Topic subscribed by CANoe
    [DDS::Topic::Name="/gps/position"]
    consumed data positionType position;

    // Topic published by CANoe
    [DDS::Topic::Name="/vehicle/status"]
    provided data statusType status;
}
```

Time	ID	Name	CO Type	CO Detail	Side	Src	Dest
0.000000	CO:	Calculator	Service	Service State	providerSide	Calculator_Provider	
7.000000	CO:	Calculator.State	Event	Provider Value	providerSide	Calculator_Provider	
4.160468	CO:	Calculator.Add	Method	Call	consumerSide	Calculator_Consumer	Calculator_Provider
4.160568	CO:	Calculator.Add	Method	Return	consumerSide	Calculator_Provider	Calculator_Consumer

▼ Publish message

Topic: **JSONTest**

Data: `{"aInt" : 142, "aDbI" : -11.23, "aStr" : "Hello world!", "aFlag":false}`

109.102619 Panel.payload

- aInt: 142 142 -
- aDbI: -11.2300 -11.2300 -
- aStr: Hello world!
- 0: Hello world!
- aFlag: 0 0 -

Time	Name	Dir	Binding Info
0.012370	DDSMonitor.subscriptionMatchedEvent	Rx	DDS
0.012370	DDSMonitor.publicationMatchedEvent	Rx	DDS
44.898838	Vehicle.status	Tx	DDS
10.0000	Vehicle.position	Rx	DDS

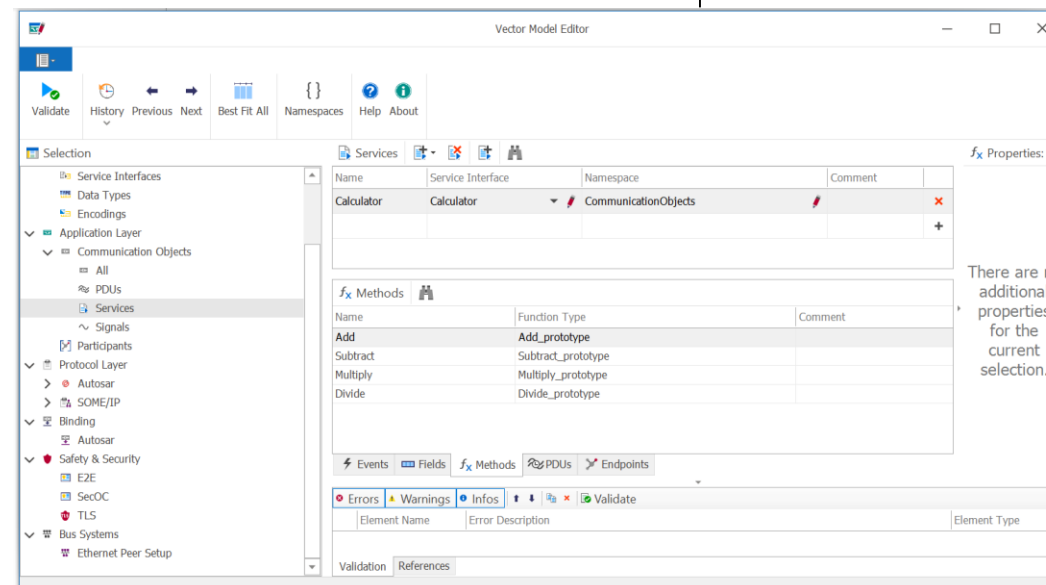
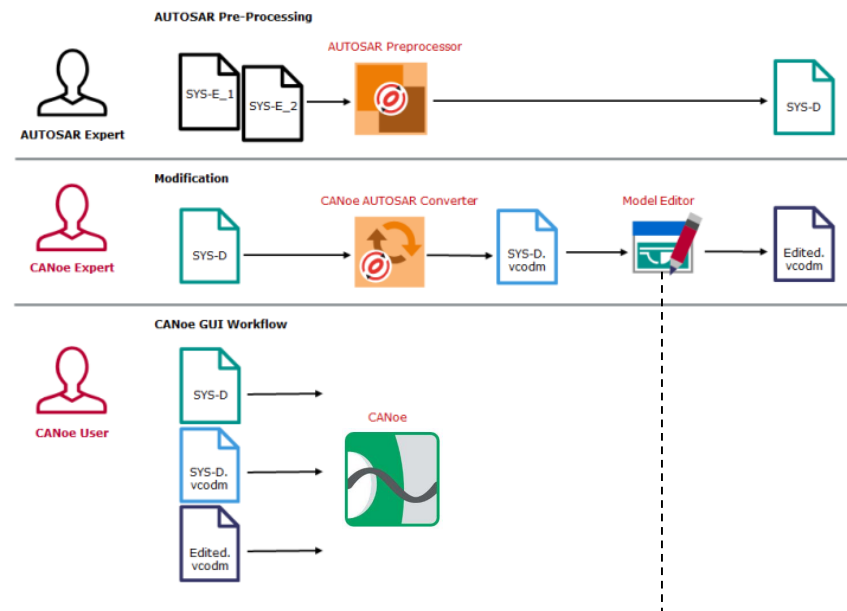
## Beyond vCDL – \*.arxml

### ▶ Current Situation

- ▶ In many cases OEMs deliver AUTOSAR extracts only which must be merged to a system description
- ▶ Merging may take hours due to the size of the files
- ▶ Each CANoe user would need to perform these merges
- ▶ Modifications of AUTOSAR databases is very complicated

### ▶ Solution

- ▶ Instead of groups in the Communication Setup, the new **AUTOSAR PreProcessor** Tool is provided and may be used by an AUTOSAR expert providing the result to a team of CANoe users
- ▶ The new **AUTOSAR Converter** can be used to convert system descriptions in into the CANoe format vcodm which can be modified using the Model Editor



## Agenda

1.

Data source vCDL Beyond arxml for Ethernet and SOA

2.

**Network Interface for Automotive Ethernet**

3.

Communication Testing, Security and Analysis

4.

More SOA – More Software – More Testing



# Vector Ethernet Hardware Overview

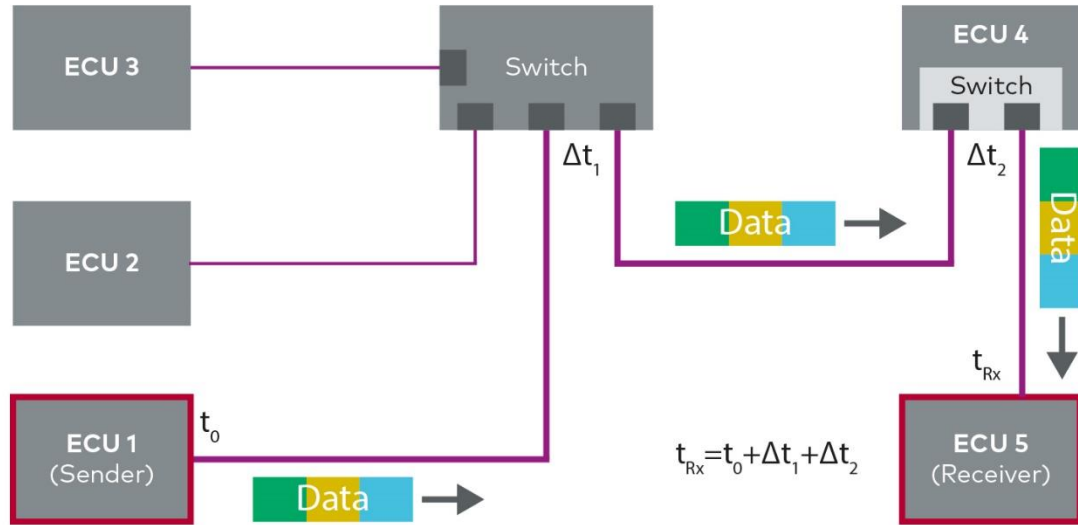
USB to Ethernet Adapter

Ethernet Network Interfaces

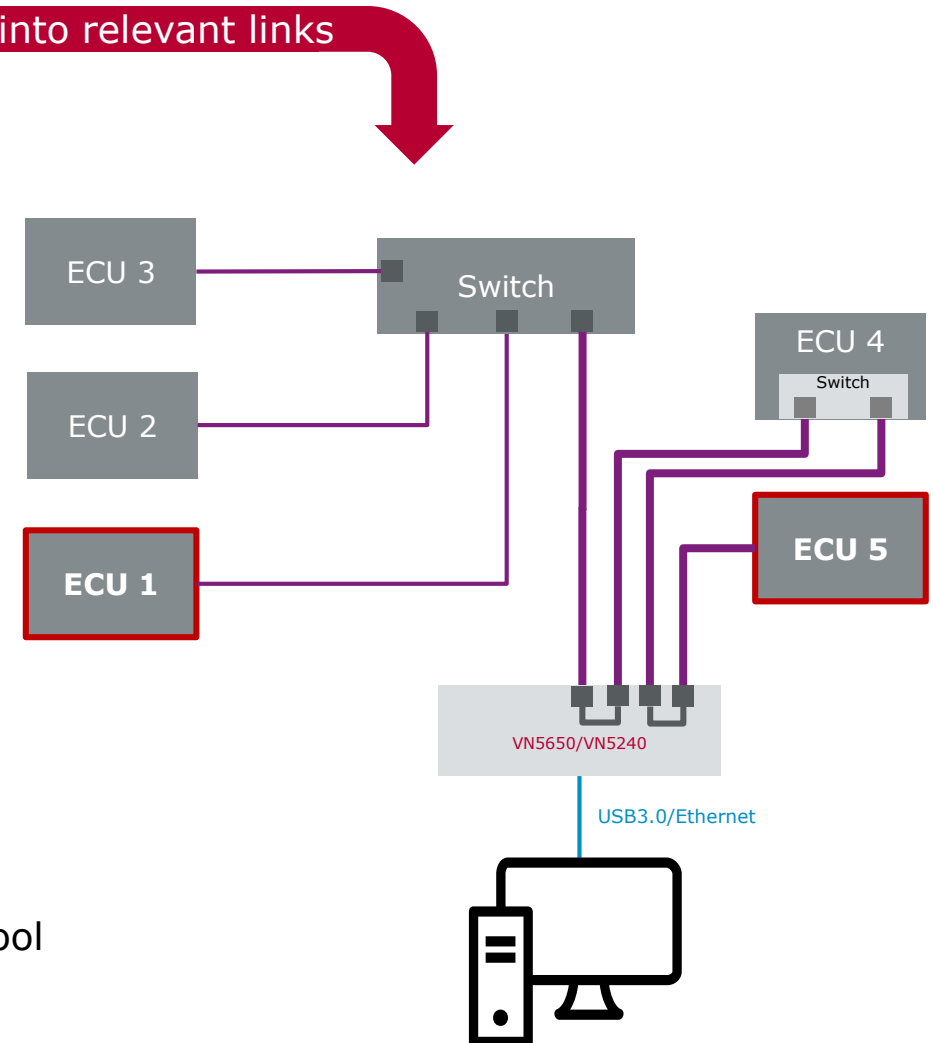


Features and supported use cases

## Network Analysis



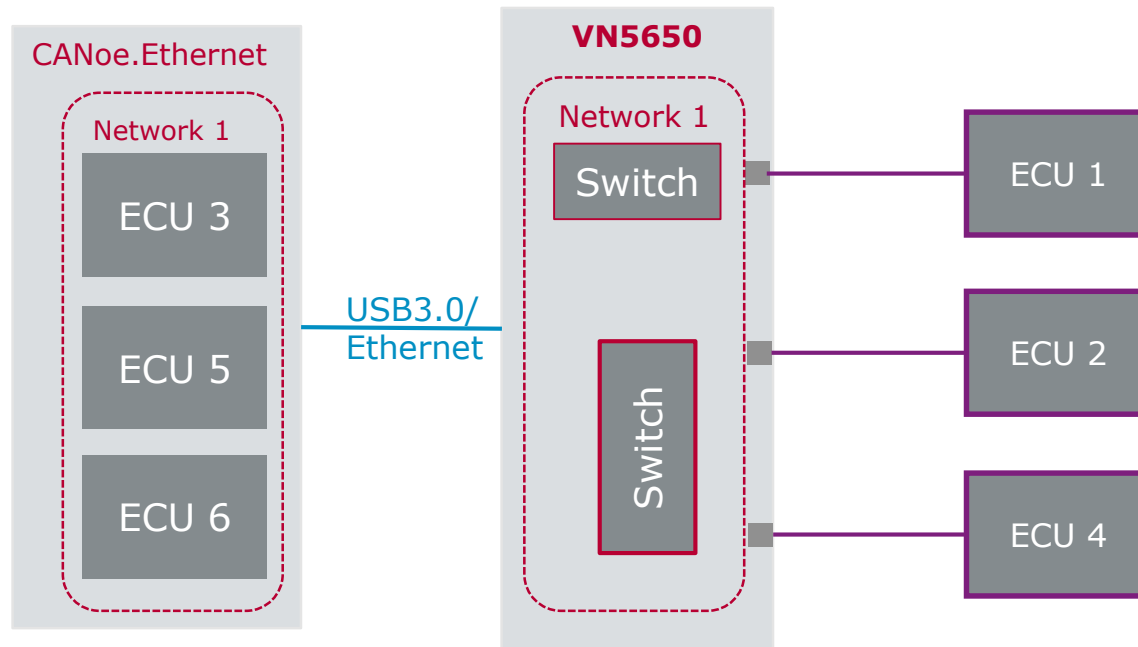
TAP into relevant links



► **Verification of entire data paths:**

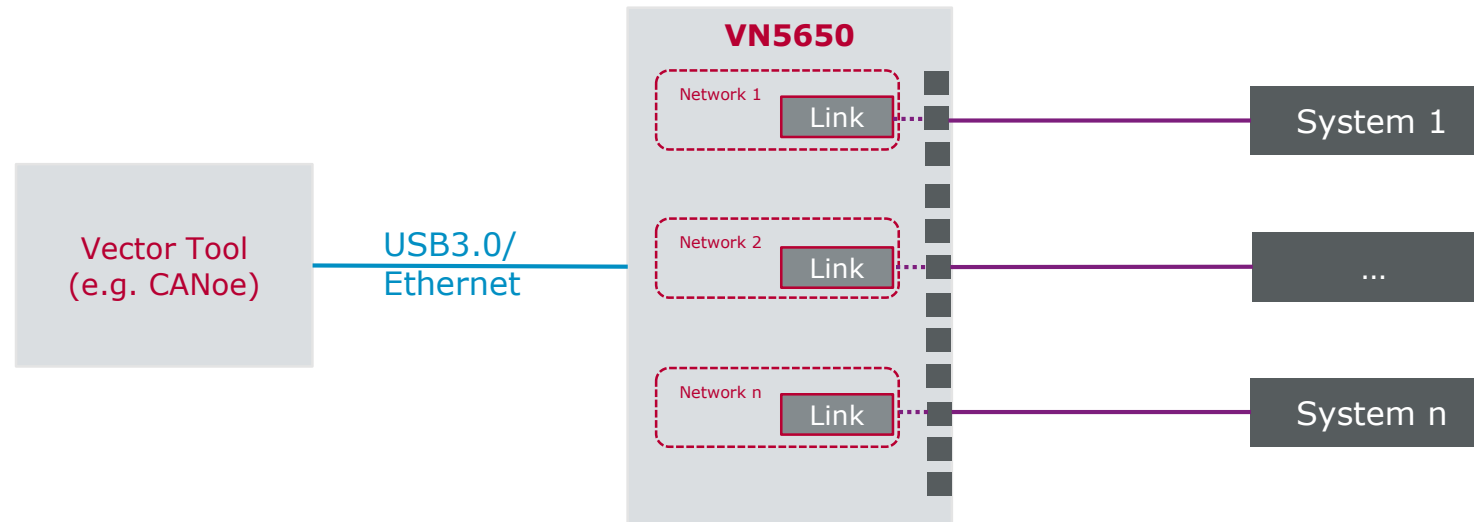
- Transparent data validation (e.g. Frame Errors)
- End-to-End transmission times
- Pass-through times of switches
- Information about dropped frames
- Possibility to affect communication with frames sent by a tool

## Simulation



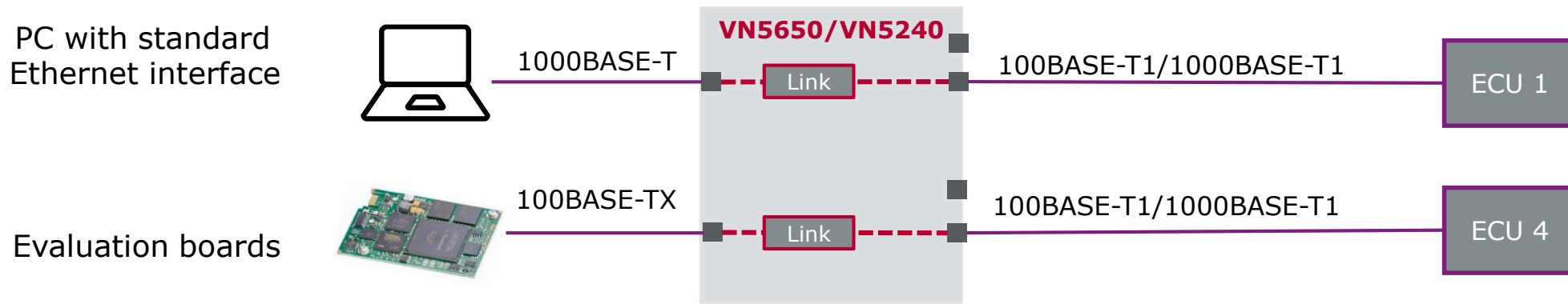
- ▶ **Simulation within an existing network**
  - ▶ Simple network access over integrated switch
    - ▶ variable ECU wiring possible, without simulation impact

## Direct Access



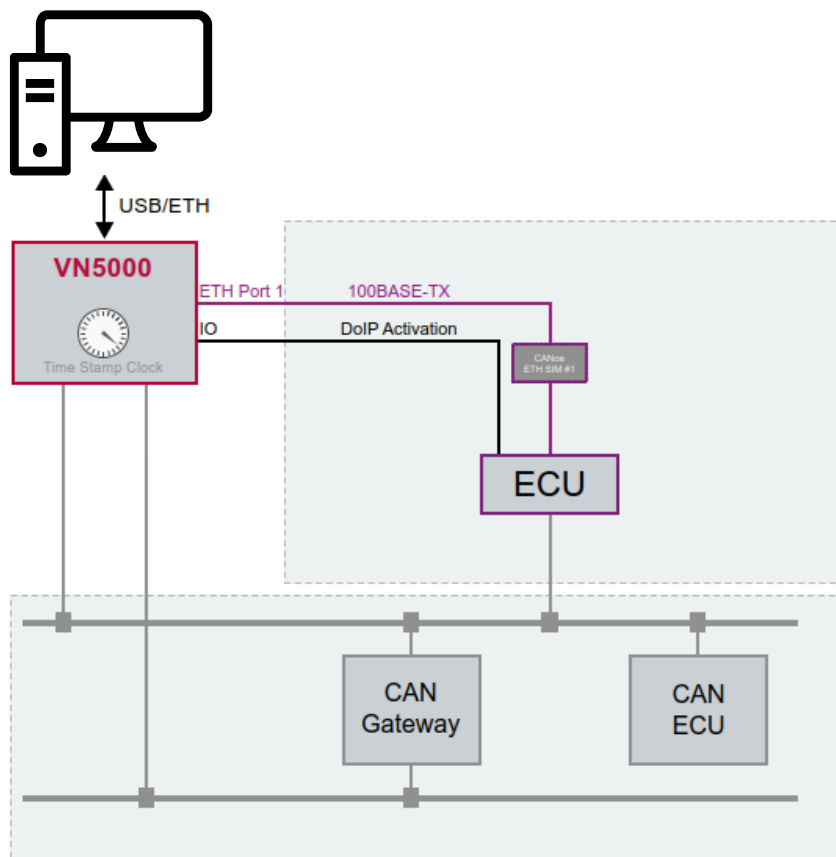
- ▶ **Individual access to each link e.g. for**
  - ▶ Flash reprogramming of ECUs (Electronic Control Units)
  - ▶ Vehicle diagnostics
  - ▶ Test benches (test of multiple identical systems)
  - ▶ Test of ECUs with multiple ports

## Media Conversion

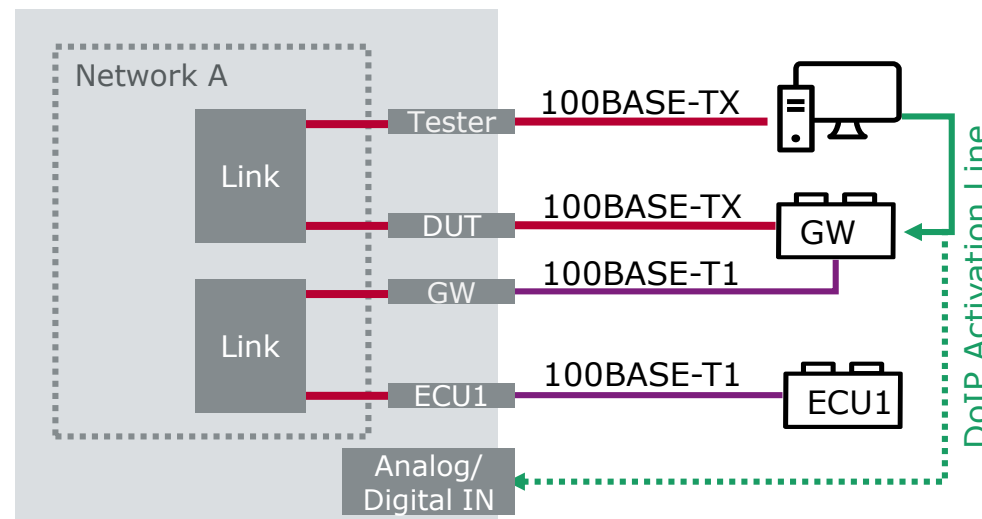


- ▶ Media conversion between different physical layers
- ▶ VN5240:
  - ▶ Up to 3 converters between IEEE 100BASE-T1/1000BASE-T1 and IEEE 100BASE-TX/1000BASE-T
- ▶ VN5650:
  - ▶ Up to 4 converters between IEEE 100BASE-T1/1000BASE-T1 and IEEE 100BASE-TX/1000BASE-T

## Diagnostics Over IP



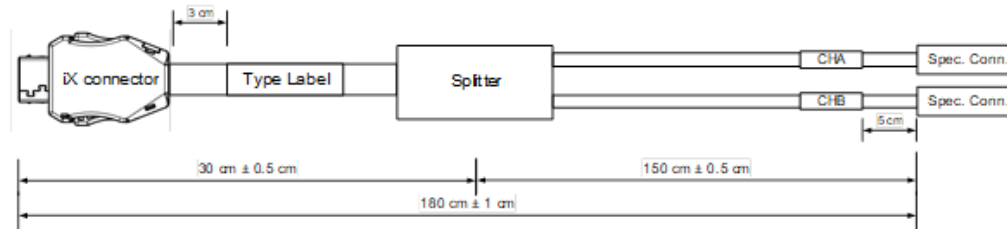
- ▶ Access on diagnostic links via the 100BASE-TX ports as a tester
- ▶ Set/Read DoIP Activation Line by using the onboard analog/digital signal interface
- ▶ TAP in-between the tester and DUT communication
  - ▶ Capture diagnostic requests/responses
  - ▶ Capture forwarded messages on in-vehicle side
  - ▶ Capture DoIP Activation signaling



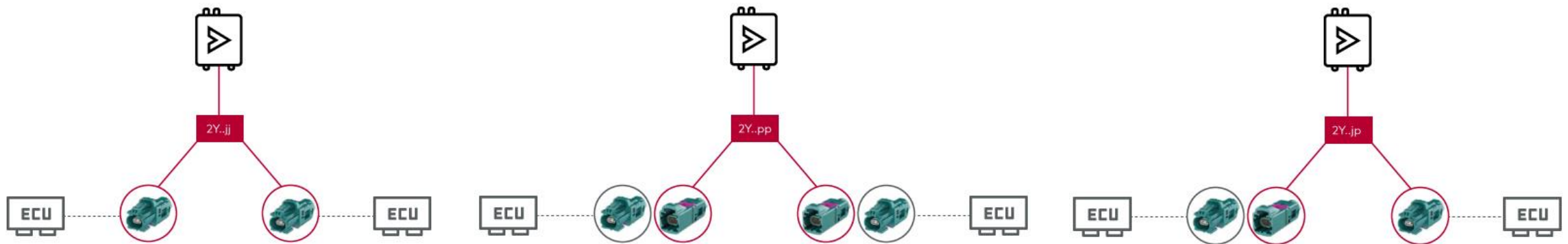
## AEcable Family (Cable Sets for 100BASE-T1/1000BASE-T1)

- ▶ Different cable variants are available, to adapt ix industrial to different other plug systems:

- ▶ H-MTD
- ▶ HSD
- ▶ MATEnet
- ▶ DSUB9
- ▶ RJ45





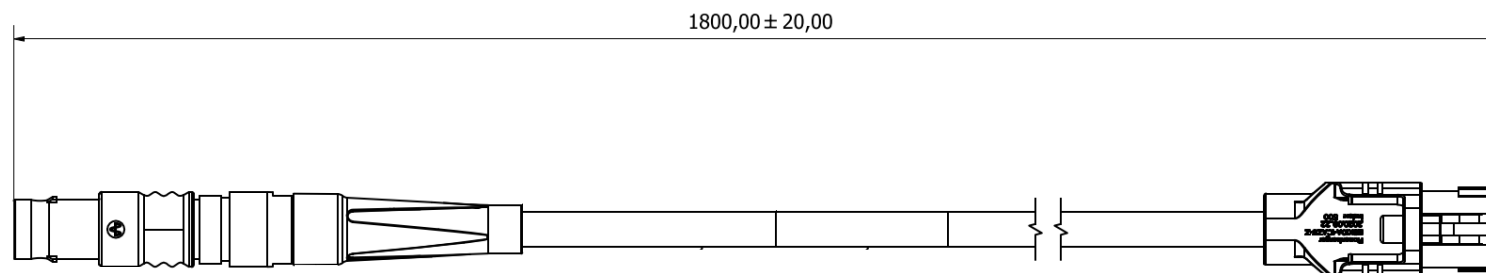
- ▶ The cables are available with plug/header (male contacts) or jack/frame (female contacts)



- ▶ Further details can be found in the Vector Network Interface accessories manual

## Cables for MultiGBASE-T1 (2.5 Gbps, 5 Gbps and 10 Gbps)

Part No	Article name	Connector Type	Photo
05218	AEcable MultiGig EVA	CHA with open end For EVALuation purposes	
05216	AEcable MultiGig H-MTD Zj	CHA: H-MTD, coding Z, type jack	
05117	AEcable MultiGig H-MTD Zp	CHA: H-MTD, coding Z, type plug	

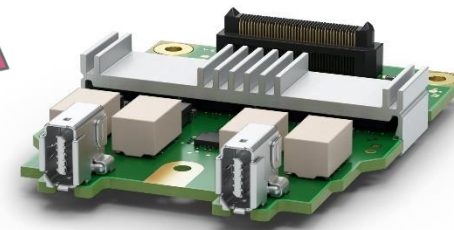
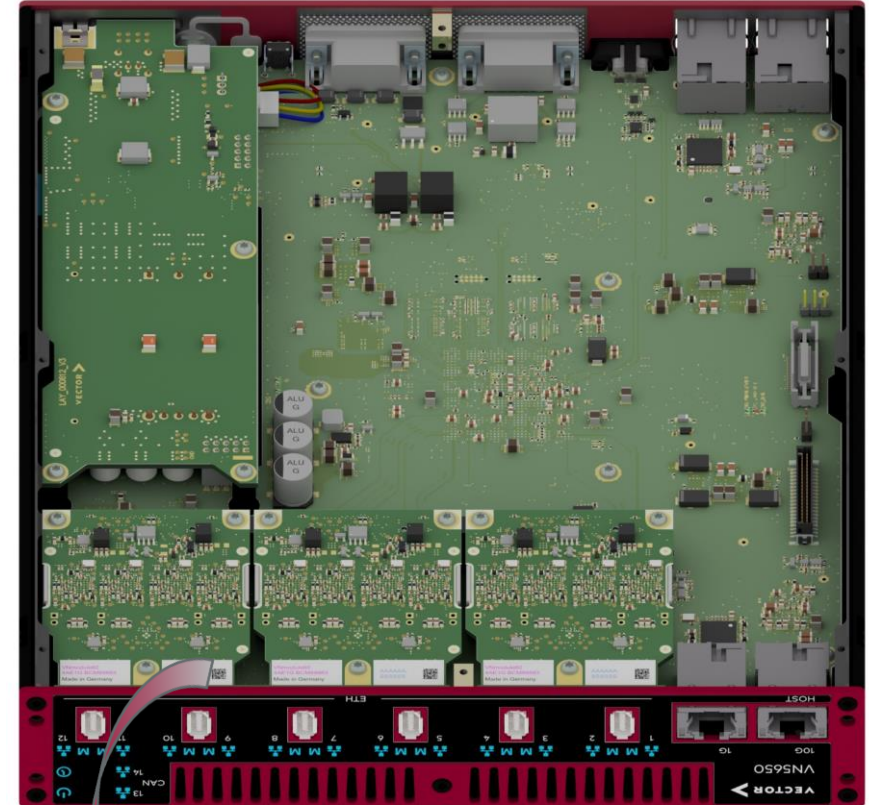


Connector for Vector Interfaces: YCP-BPR09ACX-S1MSCDX-051X YAMAICHI ELECTRONICS



## VNmodules – Flexible Physical Layer Modules

- ▶ VN5650 is flexible and modular hardware interfaces
  - ▶ Interchangeable PHY modules
    - > Different modules for current and upcoming Ethernet physical layers: 10BASE-T1S, 100BASE-T1, 1000BASE-T1, MultiGig Automotive Ethernet, MACsec



	Standard	Connector	PHY	Features
<b>Already Available</b>				
VNmodule60 4AE1G BCM89883	IEEE 100/1000BASE-T1 (4 ports)	ix Industrial	Broadcom BCM89883	OPEN Alliance TC10 @ 100BASE-T1
VNmodule60 4AE1G 88Q2112	IEEE 100/1000BASE-T1 (4 ports)	ix Industrial	Marvell 88Q2112-A2	1000BASE-T1 Legacy Mode
VNmodule60 1AE10M LAN8670	IEEE 10BASE-T1S (1 port) IEEE 100/1000BASE-T1 (2 ports)	ix Industrial	Microchip LAN8670 (10BASE-T1S); Broadcom BCM89883 (100/1000BASE-T1)	OPEN Alliance TC10 @ 100BASE-T1
VNmodule60 4AE1G RTL9010AA	IEEE 100/1000BASE-T1 (4 ports)	ix industrial	Realtek RTL9010AA	OPEN Alliance TC10 @ 100BASE-T1
VNmodule60 2AE10G BCM89890	IEEE 2.5/5/10GBASE-T1 (2 Ports)	Yamaichi Y-Circ	Broadcom BCM89890M Rev. B0	OPEN Alliance TC10* MACsec*
VNmodule60 4AE1G 88Q2221M	IEEE 100/1000BASE-T1 (4 ports)	ix Industrial	Marvell 88Q2221M Rev. B1	OPEN Alliance TC10 @ 100BASE-T1 OPEN Alliance TC10 @ 1000BASE-T1 MACsec IEEE 802.1AE

## Agenda

1. Data source vCDL Beyond arxml for Ethernet and SOA

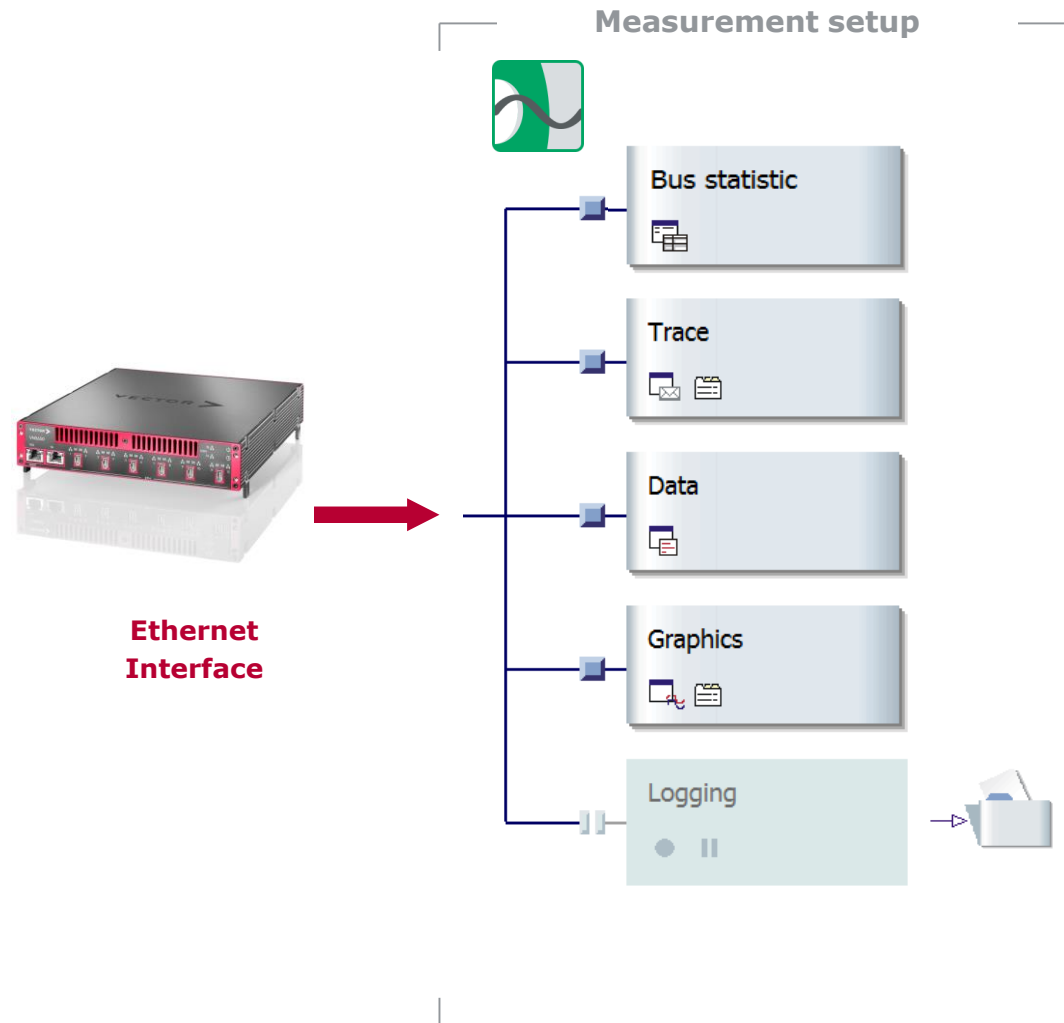
2. Network Interface for Automotive Ethernet

3. Communication Testing, Security and Analysis

4. More SOA – More Software – More Testing

# Communication Testing, Security and Analysis

## Measurement and Analyzing



SOABasicAsrAdaptive.cfg \* [Simulated Bus] - Vector CANoe /pro

**Communication Setup**

- Application Layer Objects > Participants > Consumer
- Consumer: On/Simulated (CANoe)
- Generate C# API: checked

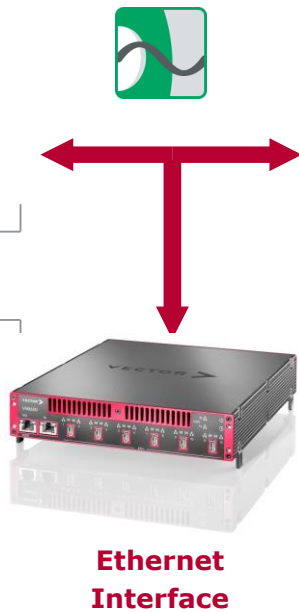
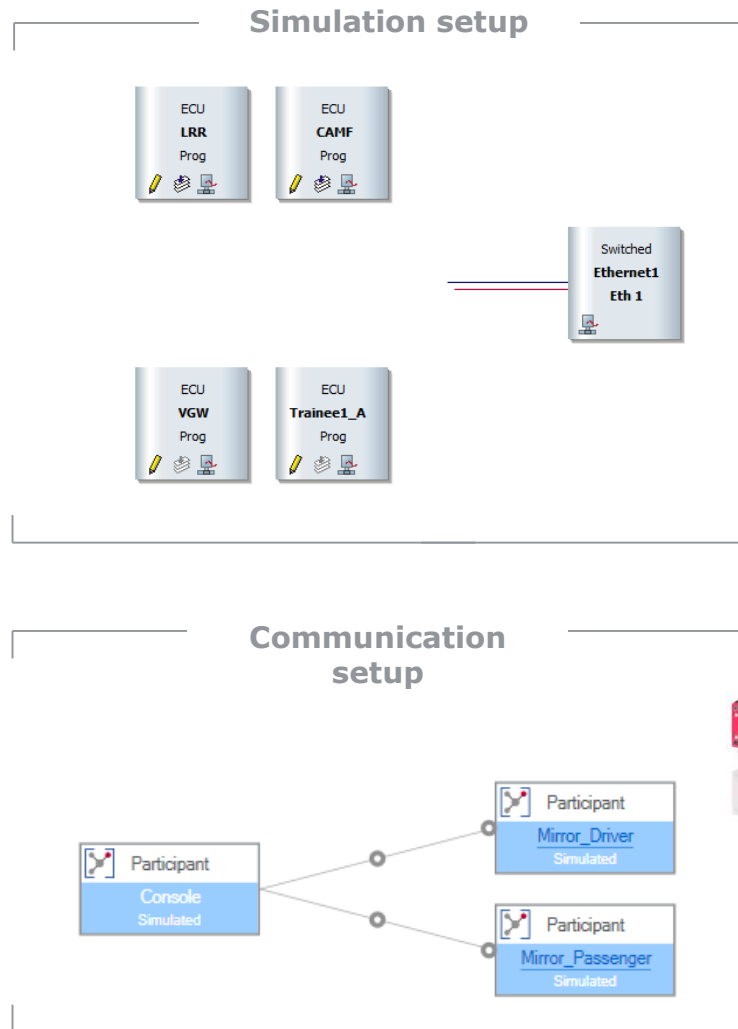
**Protocol Monitor**

Diagram View: All Networks

**Trace**

Time	Name	Object Type	Object Detail	Dir	Src	Dest	Data
0.000000	Calculator	Service	Service State		Provider	Provider	Available
0.000010	Calculator.State	Event	Subscription State		[Consumer]	Provider	Subscribed
20.000000	Calculator.State	Event	Value	Tx	Provider	Consumer	00 00 00 00 00 00 00 ...
0.000000	Calculator	Service	Connection State		[Consumer]	Provider	Connected
0.000001				Rx	02:00:00:00:01:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 24 ...
0.000001	Special::ServiceDisco...			Rx	02:00:00:00:01:FF	01:00:58:00:00:01	C0 00 00 00 00 00 00 10 ...
18.450001	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 30 ...
18.450001	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	01:00:58:00:00:01	C0 00 00 00 00 00 00 10 ...
0.000003				Tx	02:00:00:00:01:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 24 ...
0.000003	Special::ServiceDisco...			Tx	02:00:00:00:01:FF	01:00:58:00:00:01	C0 00 00 00 00 00 00 10 ...
18.450002	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 30 ...
18.450002	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	01:00:58:00:00:01	C0 00 00 00 00 00 00 10 ...
0.000004	Calculator	Service	Connection State		Consumer	Provider	Available
0.000012	Calculator.State	Event	Subscription State		[Consumer]	Provider	Subscribed
20.000001				Rx	02:00:00:00:00:FF	02:00:00:00:01:FF	06 82 9F 41 00 00 00 18 ...
18.450005	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	02:00:00:00:01:FF	C0 00 00 00 00 00 00 10 ...
20.000002				Tx	02:00:00:00:00:FF	02:00:00:00:01:FF	06 82 9F 41 00 00 00 18 ...
18.450006	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	02:00:00:00:01:FF	C0 00 00 00 00 00 00 10 ...
18.450003				Rx	02:00:00:00:01:FF	02:00:00:00:00:FF	FF FF 81 00 00 00 00 30 ...
18.450003	Special::ServiceDisco...			Rx	02:00:00:00:01:FF	02:00:00:00:00:FF	C0 00 00 00 00 00 00 10 ...
18.450004				Tx	02:00:00:00:01:FF	02:00:00:00:00:FF	FF FF 81 00 00 00 00 30 ...

# Simulation in CANoe



SOABasicAsrAdaptive.cfg \* [Simulated Bus] - Vector CANoe /pro

**Communication Setup**

- Application: Calculator[Consum... On/Simulated (CANoe)
- Participant: Consumer
- Participant: Provider

**Protocol Monitor**

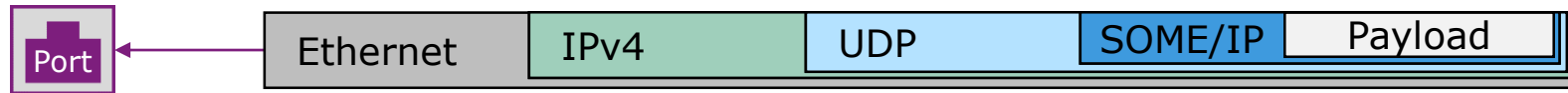
Diagram View: All Networks

**Trace**

Time	Name	Object Type	Object Detail	Dir	Src	Dest	Data
0.000000	Calculator	Service	Service State		Provider	Provider	Available
0.000010	Calculator.State	Event	Subscription State		[Consumer]	Provider	Subscribed
20.000000	Calculator.State	Event	Value	Tx	Provider	Consumer	00 00 00 00 00 00 00 ...
0.000000	Calculator	Service	Connection State		[Consumer]	Provider	Connected
0.000000	Calculator	Service	Connection State		[Consumer]	Provider	Connected
0.000001	Special::ServiceDisco...			Rx	02:00:00:00:01:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 24 ...
0.000001	Special::ServiceDisco...			Rx	02:00:00:00:01:FF	01:00:58:00:00:01	C0 00 00 00 00 00 10 ...
18.450001	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 30 ...
18.450001	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	01:00:58:00:00:01	C0 00 00 00 00 00 10 ...
0.000003	Special::ServiceDisco...			Tx	02:00:00:00:01:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 24 ...
0.000003	Special::ServiceDisco...			Tx	02:00:00:00:01:FF	01:00:58:00:00:01	C0 00 00 00 00 00 10 ...
18.450002	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	01:00:58:00:00:01	FF FF 81 00 00 00 00 30 ...
18.450002	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	01:00:58:00:00:01	C0 00 00 00 00 00 10 ...
0.000004	Calculator	Service	Connection State		Consumer	Provider	Available
0.000012	Calculator.State	Event	Subscription State		[Consumer]	Provider	Subscribed
20.000001	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	02:00:00:00:01:FF	06 82 9F 41 00 00 00 18 ...
18.450005	Special::ServiceDisco...			Rx	02:00:00:00:00:FF	02:00:00:00:01:FF	C0 00 00 00 00 00 10 ...
20.000002	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	02:00:00:00:01:FF	06 82 9F 41 00 00 00 18 ...
18.450006	Special::ServiceDisco...			Tx	02:00:00:00:00:FF	02:00:00:00:01:FF	C0 00 00 00 00 00 10 ...
18.450003	Special::ServiceDisco...			Rx	02:00:00:00:01:FF	02:00:00:00:00:FF	FF FF 81 00 00 00 00 FF ...
18.450003	Special::ServiceDisco...			Rx	02:00:00:00:01:FF	02:00:00:00:00:FF	C0 00 00 00 00 00 10 ...
18.450004	Special::ServiceDisco...			Tx	02:00:00:00:01:FF	02:00:00:00:00:FF	FF FF 81 00 00 00 00 30 ...

# Protocol Monitor And Trace Window

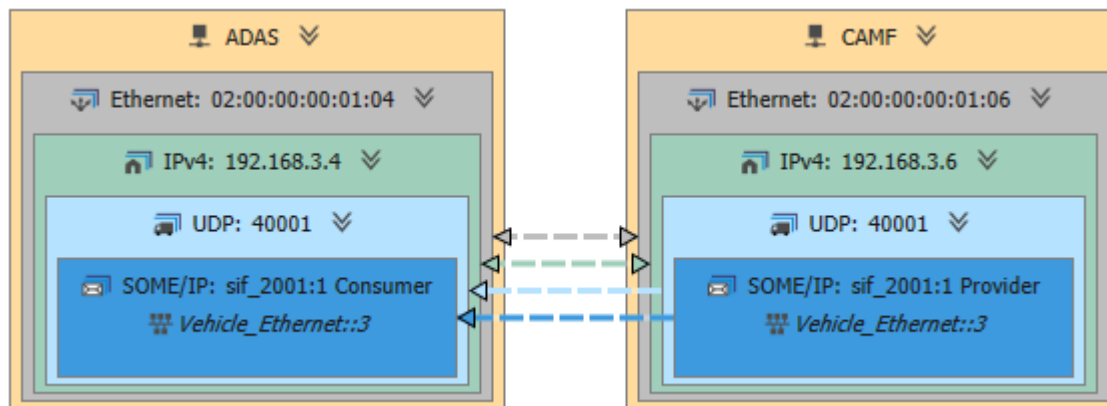
Ethernet Frame:



Trace

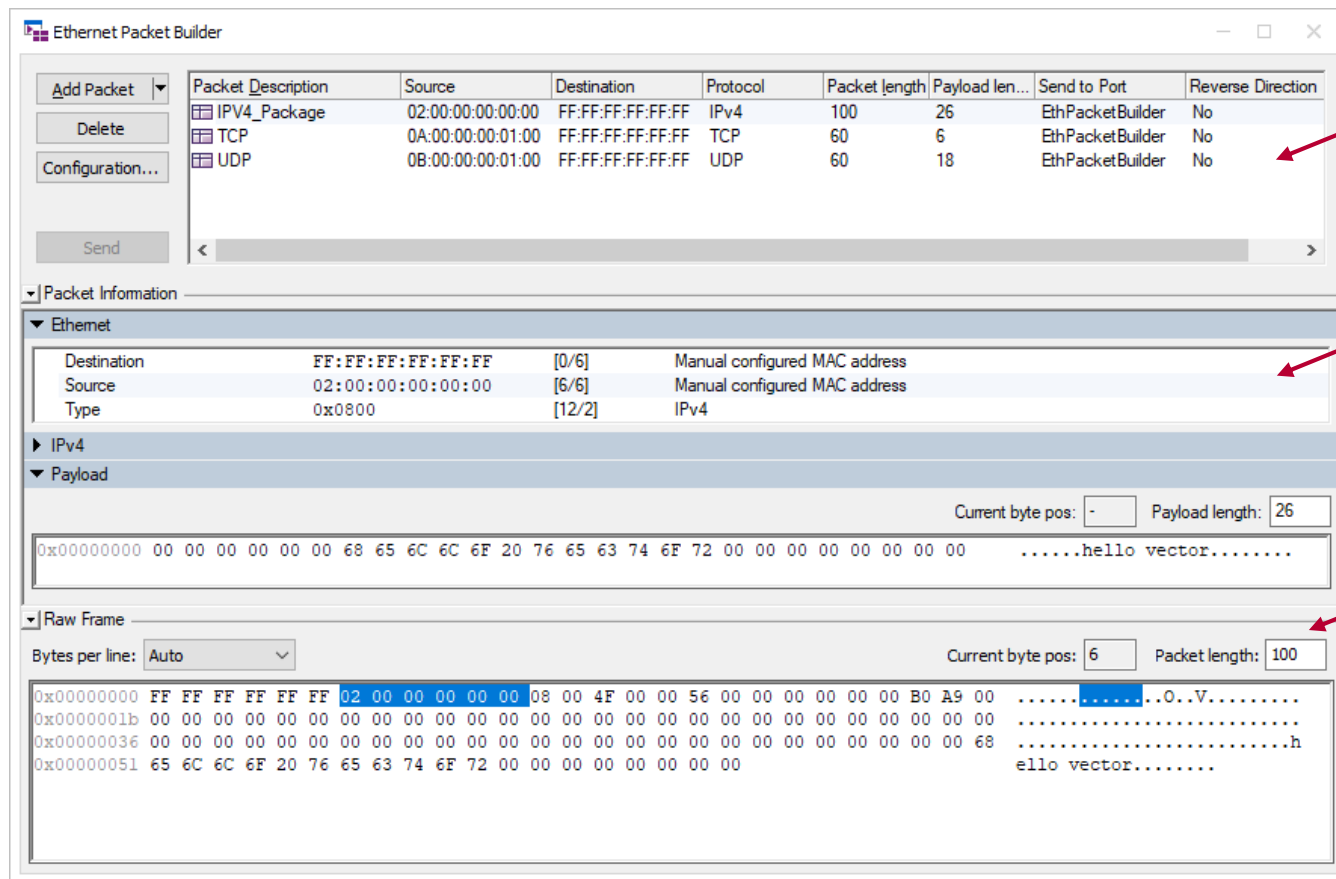
Time	Chn	VLAN...	Sim	Dir	Protocol	Sender Node	Receive Node	Service	Ser...	Method	Message Type	Protocol
0:00:00:00:49.479981	Eth 1	5	s	Tx	someip	CD_Changer	HeadUnit	Special		ServiceDiscovery	NOTIFICATION	SOME/IP
0:00:00:00:49.500010	Eth 1	5	s	Tx	someip	CD_Changer	HeadUnit	CDPlayer	1	PeakLevel	NOTIFICATION	
0:00:00:00:49.800010	Eth 1	5	s	Tx	someip	CD_Changer	HeadUnit	CDPlayer	1	PeakLevel	NOTIFICATION	
0:00:00:00:50.000011	Eth 1	5	s	Tx	someip	CD_Changer	HeadUnit	CDPlayer	1	TimePosition	NOTIFICATION	
0:00:00:00:50.100010	Eth 1	5	s	Tx	someip	CD_Changer	HeadUnit	CDPlayer	1	PeakLevel	NOTIFICATION	

Protocol Monitor



# Ethernet Packet Builder

- ▶ Ethernet Packet Builder: The window is divided into several parts



## ▶ Package List

Packet description, packet length and payload length can be modified.

## ▶ Packet Information

Protocol header fields as well as the payload of the packet can be modified.

Constant MAC Ids as well as MAC Ids of real adapters can be selected.

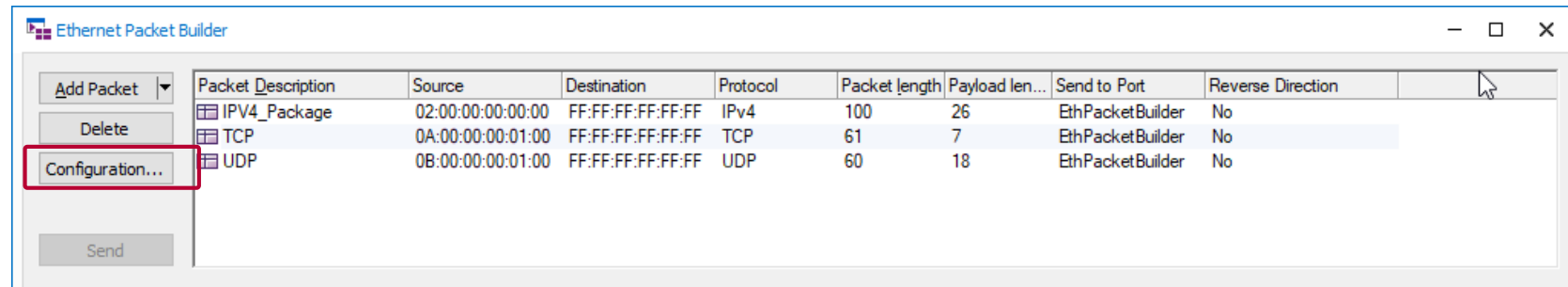
## ▶ Raw Frame

Hexadecimal presentation field:  
Raw frames can be imported

ASCII presentation field:  
Data can be copied from the clipboard or can be edited

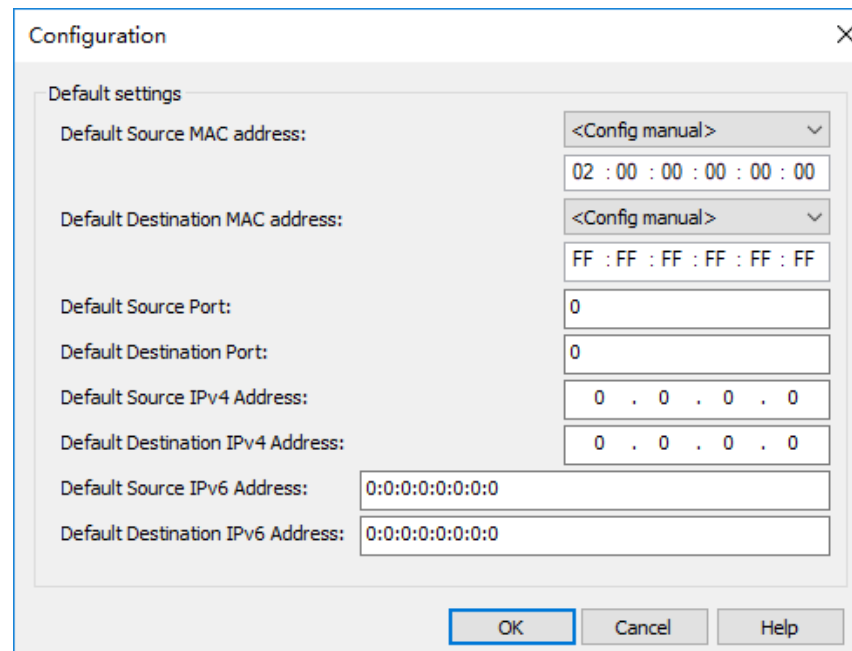
Bytes per line and Packet length can be adjusted.

# Ethernet Packet Builder



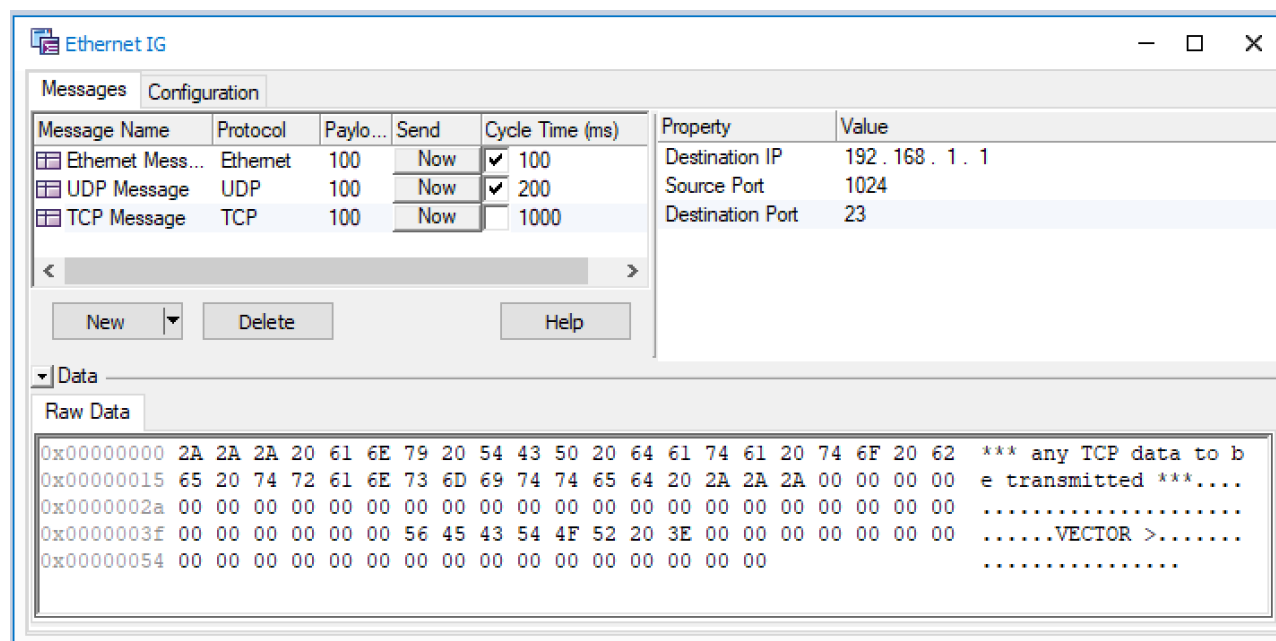
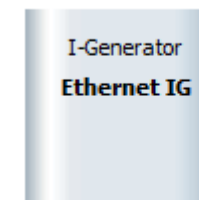
► **Settings:**

Default values for source and destination MAC ID, IP addresses and ports of added packets



## Ethernet Interactive Generator Block

- ▶ Periodic or spontaneous transmission of Ethernet packets
- ▶ IPv4 Socket-based transmission of UDP and TCP data
- ▶ Autonomous establishment of TCP connection
- ▶ Payload can be modified at any time.



\*Only CANoe; feature is in maintenance mode



## Ethernet Packet

- Send one ethernet packet (without IP information)

- Output (Ethernet)

```
void output(ethernetPacket packet)
```

```
on key 'a'
{
  ethernetPacket txPacket;
  int i;

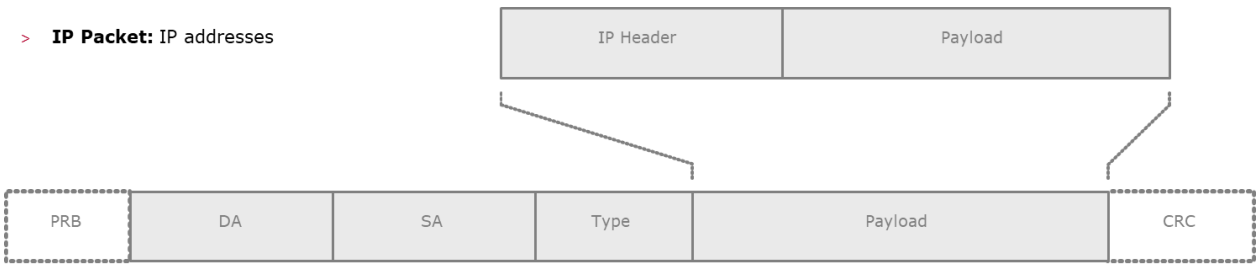
  txPacket.msgChannel =1;
  txPacket.source = ethGetMacAddressAsNumber("20:00:00:00:00:01");
  txPacket.destination = ethGetMacAddressAsNumber("FF:FF:FF:FF:FF:FF");
  txPacket.type = 0XF123;
  txPacket.Length = 100;
  for(i=0;i<txPacket.length;i++)
  {
    txPacket.byte(i) = i;
  }

  output(txPacket);
}
```

General	
Type:	Ethernet Packet
Channel:	Ethernet1 (Eth 1)
Ports:	ECU2
Frame Checksum:	0
Packet Length:	114 bytes 72
Direction:	Rx
+ Data	
Ethernet	
Destination:	broadcast FF:FF:FF:FF:FF:FF
Source:	20:00:00:00:00:01
Type:	61731 F123
Payload	
Length	100 bytes
000h	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12
020h	20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32
040h	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52
060h	60 61 62 63

# Send IP Segments with CAPL

## ► Example



```

on key 'b'
{
    ethernetPacket IpPacket;

    IpPacket.source.ParseAddress("20:00:00:00:00:01");
    IpPacket.destination.ParseAddress("20:00:00:00:00:02");

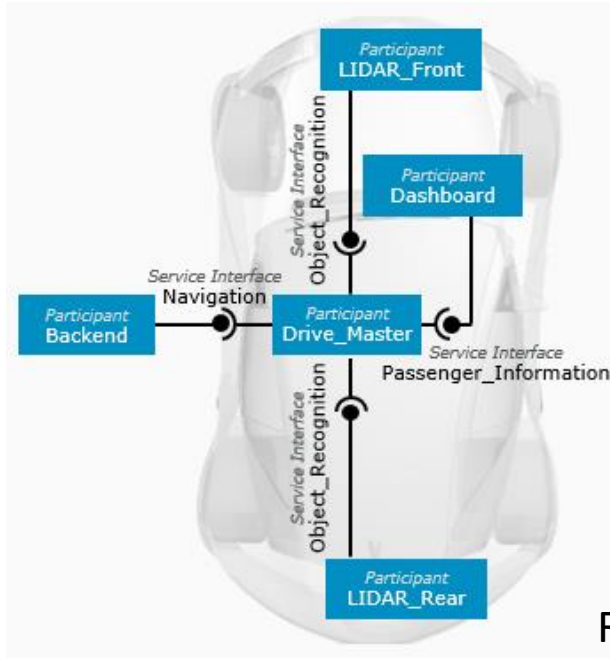
    IpPacket.ipv4.init(); //Initialize the protocol
    IpPacket.ipv4.source.ParseAddress("192.168.100.1");
    IpPacket.ipv4.destination.ParseAddress("192.168.100.2");
    IpPacket.ipv4.ResizeData(100); //Resizes the payload of a protocol
    IpPacket.ipv4.byte(0) = 0xff;

    IpPacket.CompletePacket();
    //Calculates the checksum and length field for all protocols
    output(IpPacket);
}
    
```

```

Ethernet
  Destination: 20:00:00:00:00:02
  Source: 20:00:00:00:00:01
  Type: IPv4 0800
  Payload
    IPv4
      Version: 4 4
      Header Length: 20 bytes 5
      Differentiated Service Field DSCP=0 ECN=Not-ECT 00
      Total Length: 120 bytes 0078
      Identification: 0 0000
      Control Flags 0
      Fragment Offset: 0 bytes 0000
      Time to Live: 64 40
      Protocol: HOPOPT 00
      Checksum: 12594 3132
      Source: private 192.168.100.1
      Destination: private 192.168.100.2
      Payload
        Length 100 bytes
        000h FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        020h 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        040h 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        060h 00 00 00 00
    
```

# Application, Conformance and Robustness Testing with SOMEIP




Application

Conformance

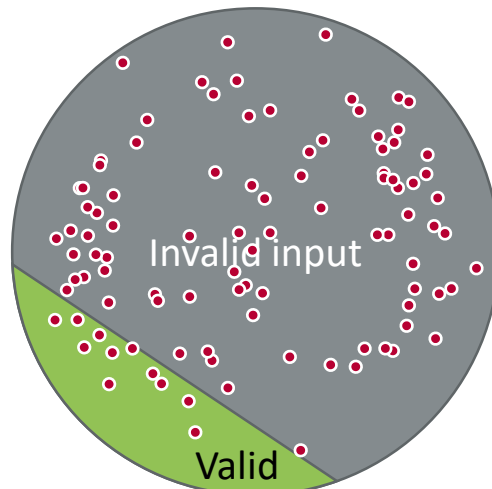


Robustness

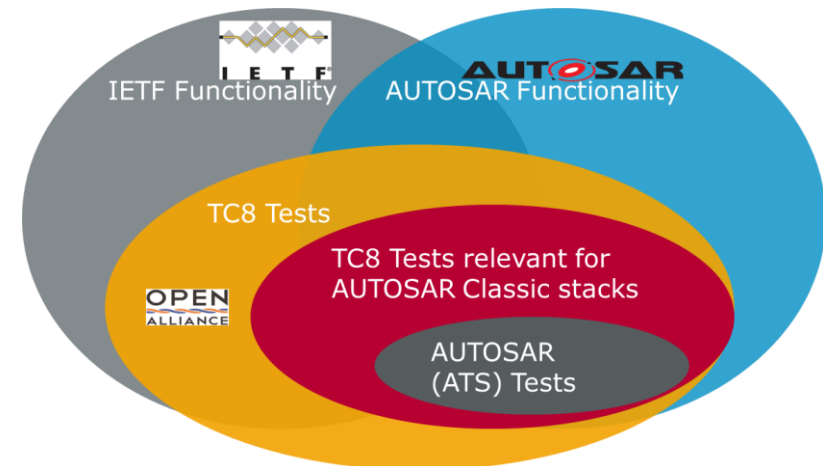
OPEN Alliance Automotive Ethernet ECU Test Specification Layer 3-7  
TC8 ECU Test



The logo for OPEN ALLIANCE features the word 'OPEN' in large black letters above a stylized wave in blue and orange, with the word 'ALLIANCE' below it.

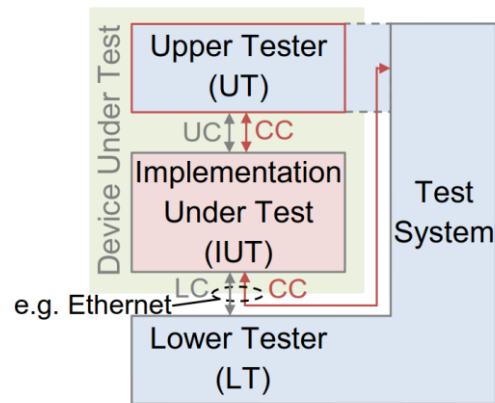


Fuzz Testing



## SOME/IP Server and ETS testing in TC8

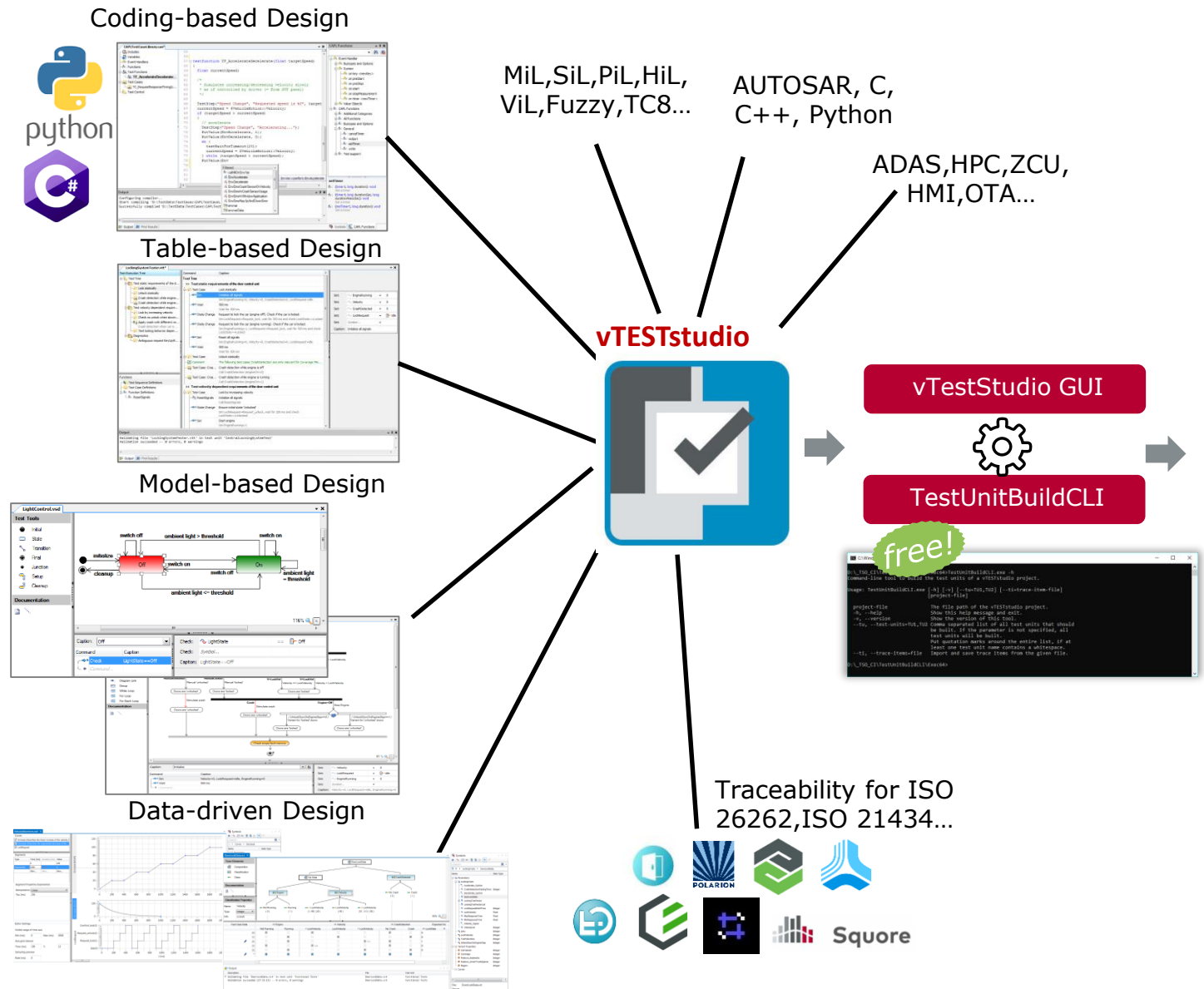
- ▶ CANoe Option Ethernet supports TC8 test specification
- ▶ The configuration does **not** require extra licensing
- ▶ A simulation of the DUT (Golden Device) is included
- ▶ Source code with vTESTstudio for free



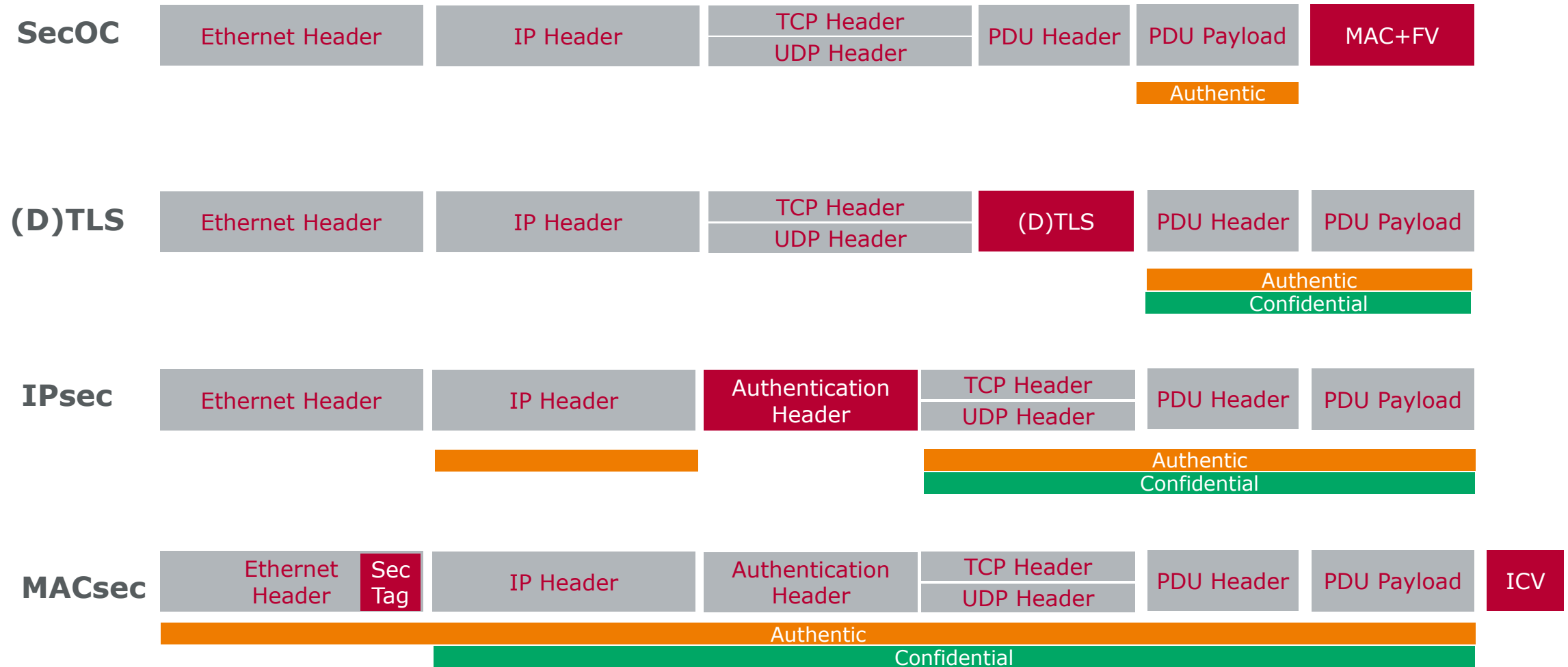
Scheme of the test environment using the control channel through the IUT itself

Title	Verdict	Runtime
✓ SOME/IP Testcases	✓	21.656s
> ✓ SOME/IP Server Tests	✓	1.595s
✓ SOME/IP ETS Tests	✓	20.058s
✓ SOME/IP_ETS_001	✓	0.007s
✓ SOME/IP_ETS_002	✓	0.004s
✓ SOME/IP_ETS_003	✓	0.004s
✓ SOME/IP_ETS_004	✓	0.004s
✓ SOME/IP_ETS_005	✓	0.004s
✓ SOME/IP_ETS_007	✓	0.004s
✓ SOME/IP_ETS_008	✓	0.005s
✓ SOME/IP_ETS_009	✓	0.005s
✓ SOME/IP_ETS_019	✓	0.005s
✓ SOME/IP_ETS_021	✓	0.007s
✓ SOME/IP_ETS_022	✓	0.005s
✓ SOME/IP_ETS_027	✓	0.009s
✓ SOME/IP_ETS_028	✓	0.005s
✓ SOME/IP_ETS_029	✓	0.007s
✓ SOME/IP_ETS_030	✓	0.005s
✓ SOME/IP_ETS_031	✓	0.004s
✓ SOME/IP_ETS_032	✓	0.004s
✓ SOME/IP_ETS_033	✗	0.007s
✓ SOME/IP_ETS_035	✓	0.008s
✓ SOME/IP_ETS_038	✓	0.005s
✓ SOME/IP_ETS_039	✓	0.005s
✓ SOME/IP_ETS_040	✓	0.006s
✓ SOME/IP_ETS_041	✗	0.006s
✓ SOME/IP_ETS_042	✗	0.005s
✓ SOME/IP_ETS_043	✗	0.005s
✓ SOME/IP_ETS_044	✓	0.005s
✓ SOME/IP_ETS_045	✗	0.006s

# What is vTESTstudio?



# Security and Protection

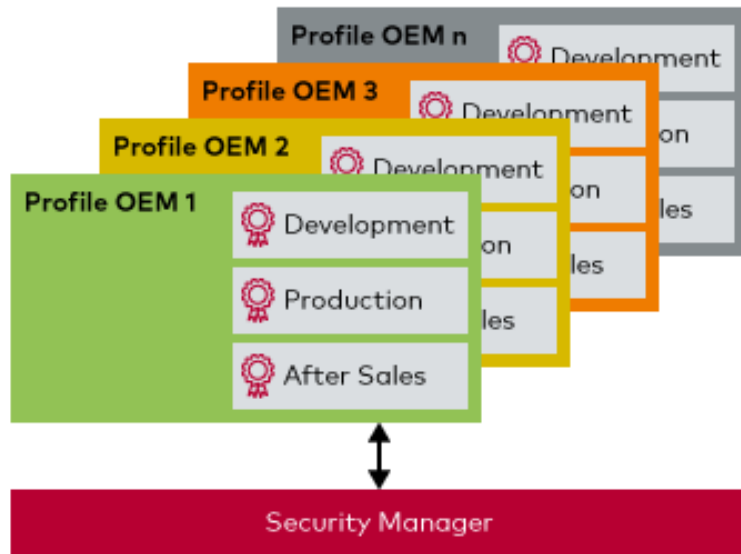


## Smart Charging with TLS in ISO 15118 via PLC



	ISO 15118-2	ISO 15118-20
TLS is mandatory for the following use-cases:	<ul style="list-style-type: none"> <li>▶ Plug &amp; Charge</li> <li>▶ Value-Added-Service</li> </ul>	<ul style="list-style-type: none"> <li>▶ TLS is always mandatory (EIM &amp; PnC)</li> </ul>
Supported Cipher Suites (TLS 1.2)	<ul style="list-style-type: none"> <li>▶ TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256</li> <li>▶ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</li> </ul>	<ul style="list-style-type: none"> <li>▶ TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256</li> <li>▶ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</li> </ul>

# Tools without Security is **challenges!**



## Communication analysis

- ✓ Reading data
- ✗ Verification of authentication information
- ✗ Check data integrity
- ✗ Detection of old / replayed data

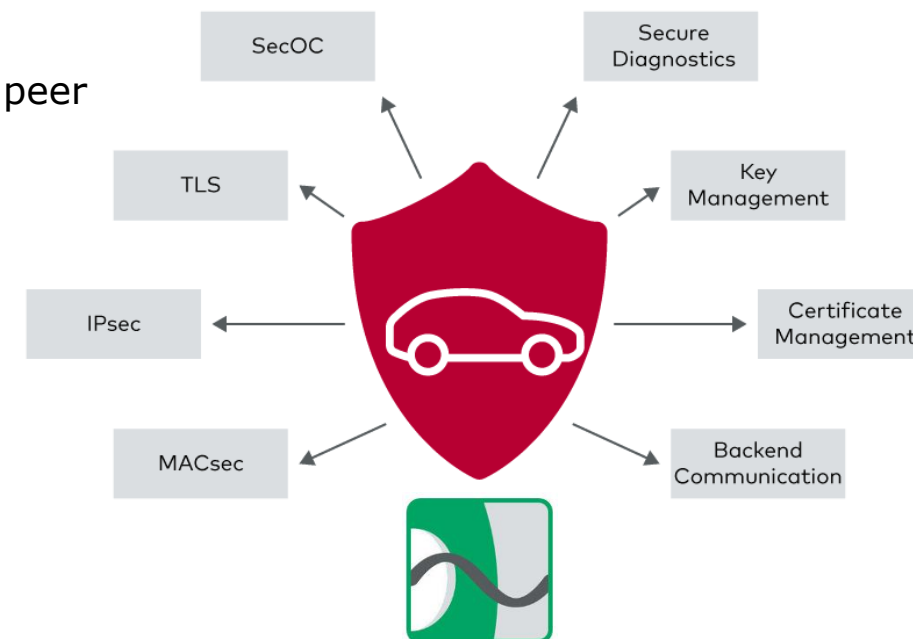
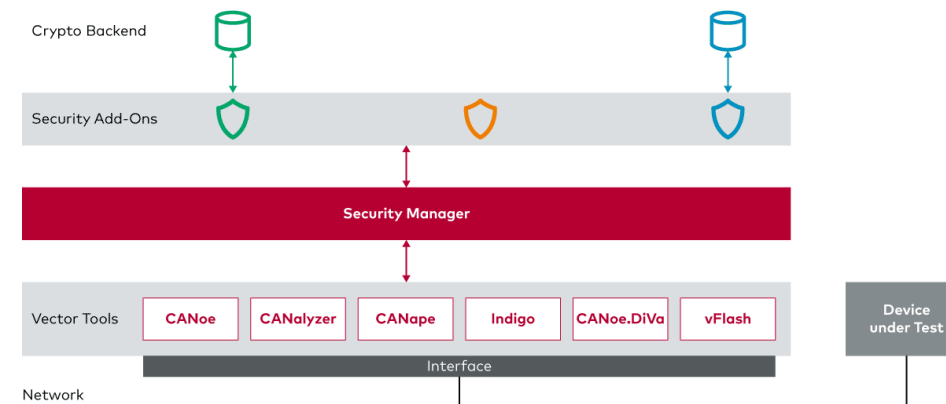
## Remaining bus simulation / Stimulation

- ✗ Authenticator calculation
- ✗ Freshness management



## Testing of Security-Protected ECUs and Networks with the Security Manager

- ▶ Communication: SecOC
- ▶ Diagnostics: Authentication
- ▶ Diagnostics: Variant Coding
- ▶ TLS: Simulation of Client and Server
- ▶ TLS: TLS Observer using Master Secret
- ▶ TLS: DoIP over TLS
- ▶ IPsec: IKEv2 support for certificate based peer authentication, dead peer detection, IKE fragmentation and IKE rekeying
- ▶ IPsec: Import of StrongSwan IPsec configurations
- ▶ IPsec: Full control of the Security Policy Database
- ▶ MACsec: Standard 802.1 AE 2018
- ▶ MACsec: Key Agreement (MKA)-Protokoll 802.1.x-2020
- ▶ V2X certificate communication



## Diagnostics: DoIP via TLS

- ▶ Encrypted DoIP communication via TLS
  - ▶ Support for built-in diagnostic channel (tester and simulation)
  - ▶ Support of cypher suites defined in ISO 13400-2:2019, e.g. Null-Cipher for debugging purposes
  - ▶ Interpretation of diagnostics communication even for fully encrypted communication
  - ▶ Support of DoIP protocol version 3
  - ▶ Configuration of security profile in Security Manager

Stack	Used by Node	Use Security	Security Profile
Operating System Stack	-	<input type="checkbox"/>	No Security Profile is assigned to this stack.
Shared CANoe Stack	-	<input checked="" type="checkbox"/>	DoIP over TLS Demo (File based PKI): Public Key Infrastructure for the CANoe T...
Individual Stack (ECU 1)	ECU 1	<input checked="" type="checkbox"/>	DoIP over TLS Demo (File based PKI): Public Key Infrastructure for the CANoe T...

## Agenda

1. Data source vCDL Beyond arxml for Ethernet and SOA

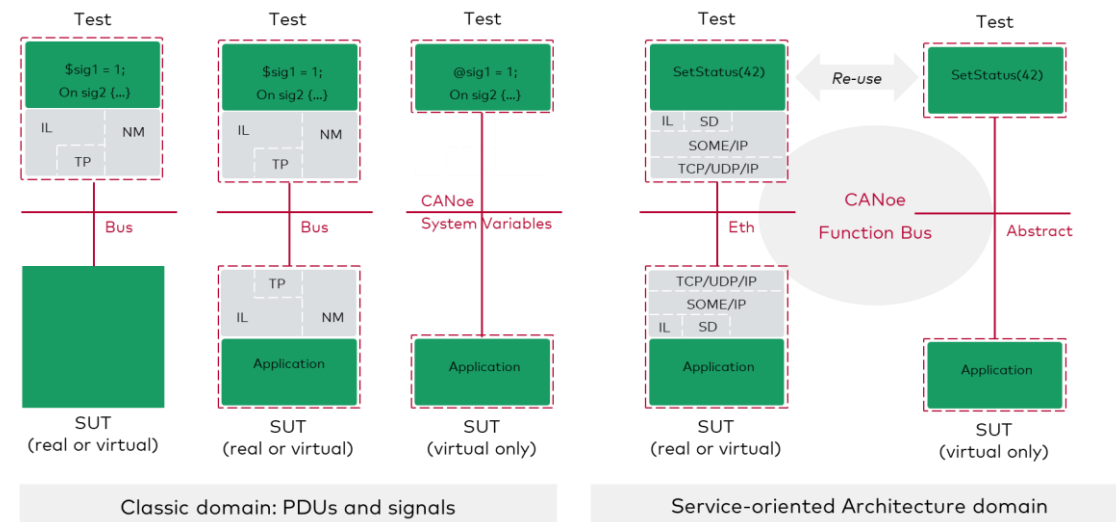
2. Network Interface for Automotive Ethernet

3. Communication Testing, Security and Analysis

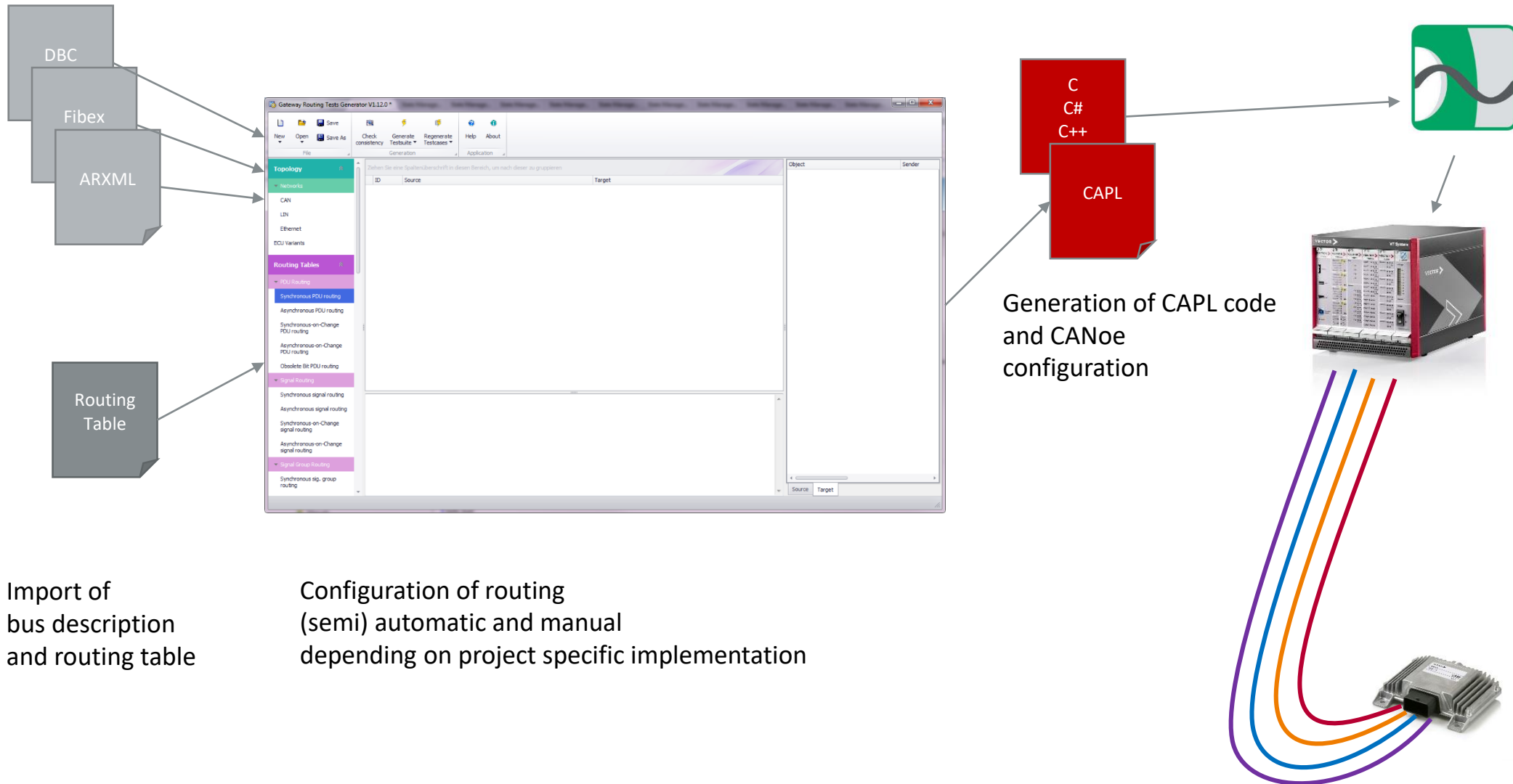
4. More SOA – More Software – More Testing

## More than TC8, not only Ethernet

- ▶ Cars are not just another IT software
  - ▶ Testing of all production variants highly desired
  - ▶ Tests must be performed on various integration levels
    - > Software component level
    - > ECU level
    - > Subsystem level
    - > Entire vehicle network level
    - > Test drive
  
- ▶ Most important concepts
  - ▶ Simultaneous operation of all networks
  - ▶ Same time base for all networks and application layer objects
    - > Allows testing of gateway applications
  - ▶ Scalability (distributed operation on multiple PCs) for HPC and ZCU



# Gateway tester



# HiL



**Stimulation Modules**  
analog VT2004A  
digital VT2516A

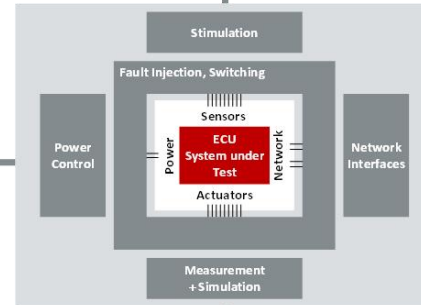


**Rotation Sensor Module** VT7820  
**Smart Charging Module**  
VT7970 / VT7971

**Network Interface Modules**  
VT6104B / VT6204B  
VT2710 / VT6306B



**Power Modules**  
VT7001A (40V)  
VT7101 (60V)



**Real-Time Modules**  
Atom VT6020  
Core i7 VT6060

**General Purpose Modules**

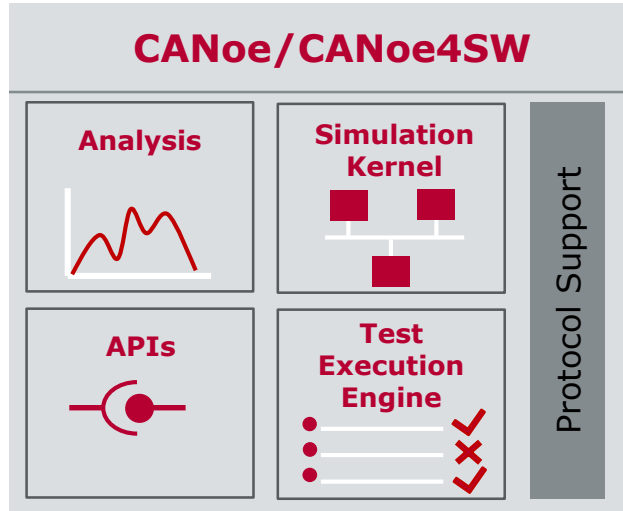
- current VT2808
- analog VT2816A
- digital VT2848
- Relais VT2820
- Matrix VT2832
- Multi VT5838



**Load + Measurement Modules**  
VT1004A (40V)  
VT1104 (60V)



# Connection Scenarios for SUT for connectivity



Scenario:  
▶ CANoe can directly connect to the Internet/SUT with IP based protocols

Examples:

- ▶ HTTP via OS TCP/IP stack
- ▶ MQTT via broker which is reachable in LAN/WAN

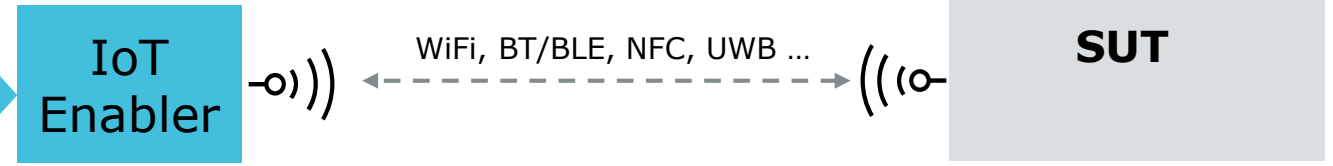
1

Scenario:  
▶ SUT cannot connect to the Internet, i.e. non IP based protocols

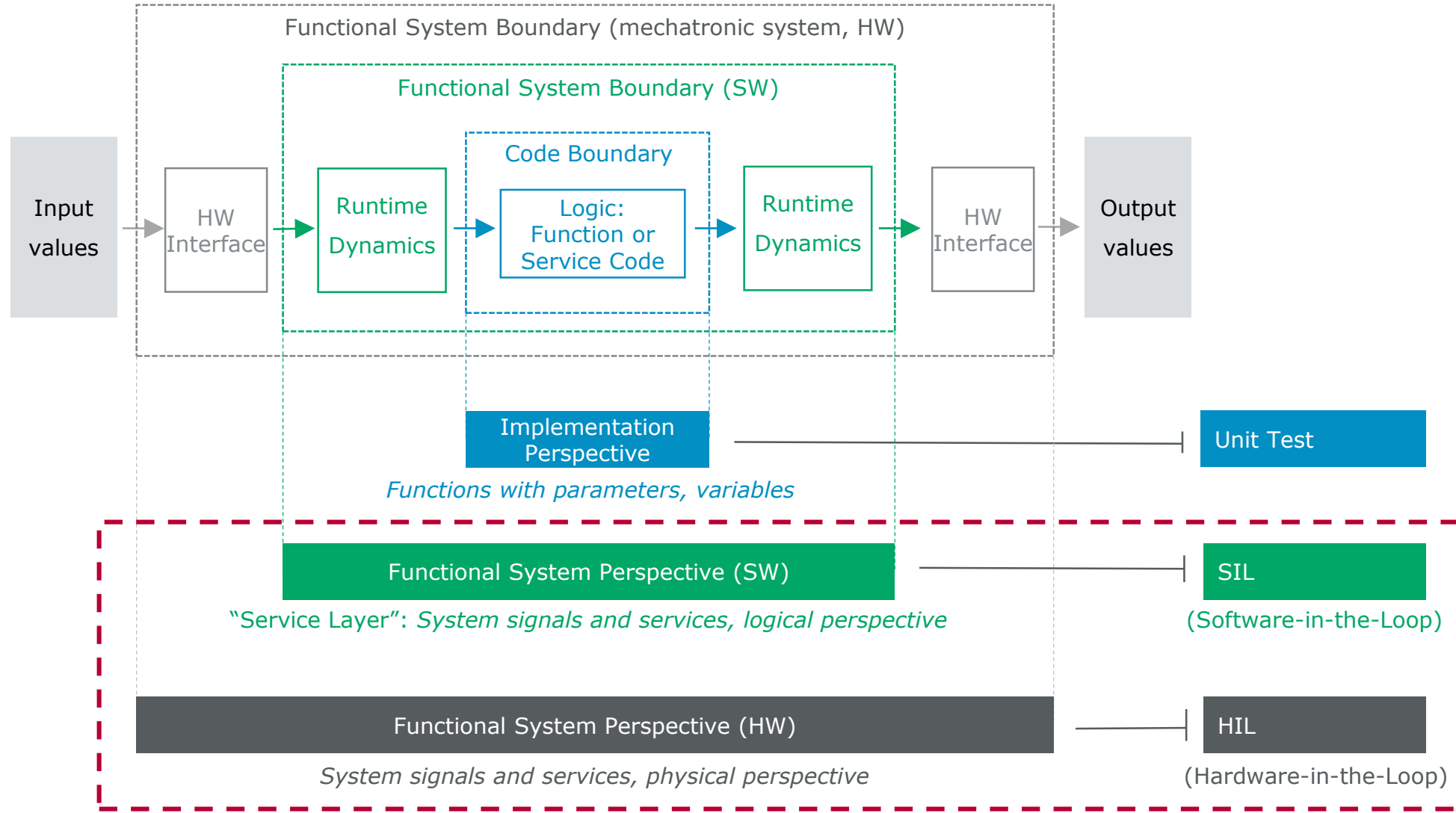
Examples:

- ▶ BT/BTLE
- ▶ UWB

2



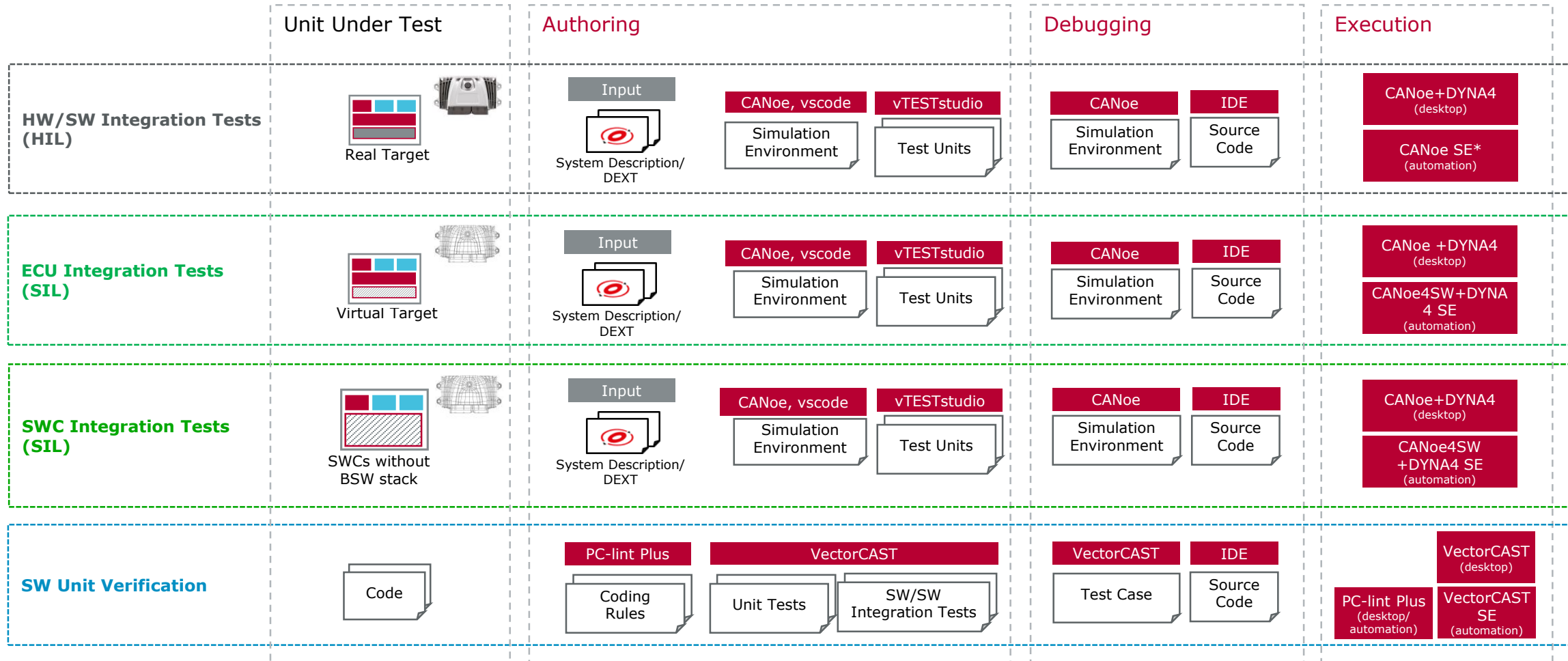
# Layers of Test Interfaces



<focus of today>



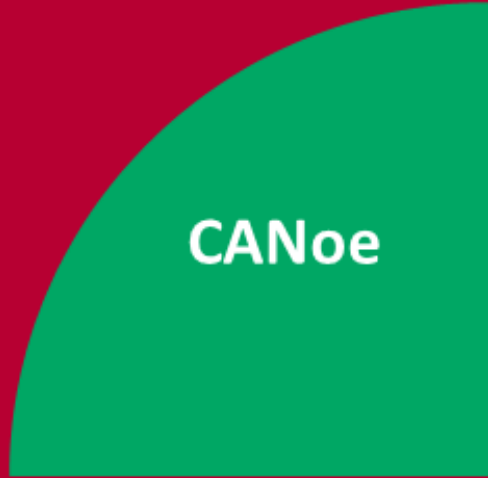
# Test Methods – Summary



\* CANoe SE is not yet available. In the meantime, CANoe pro TBE can be used for automation.



... on PC



**CANoe**



**CANoe  
Standalone**



... on network  
interface



... on PC



**CANoe4SW**



**CANoe4SW  
Server Edition**



... on server

[Kevin.fan@vector.com](mailto:Kevin.fan@vector.com)