Developing, Simulating and Testing for Ethernet System

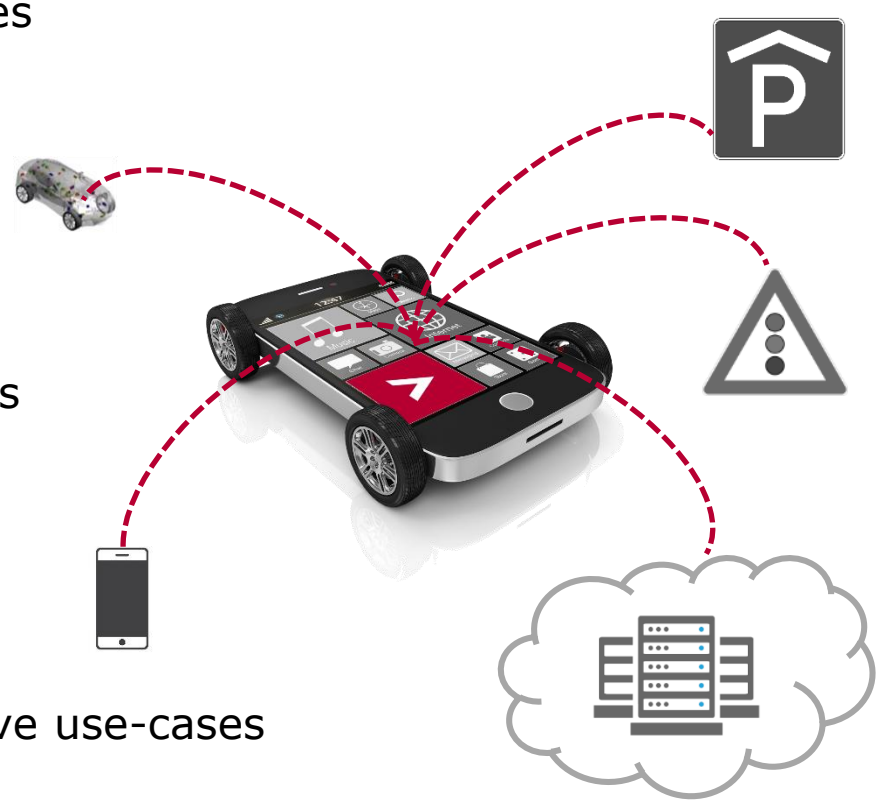
Database, Security and XiL Testing

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

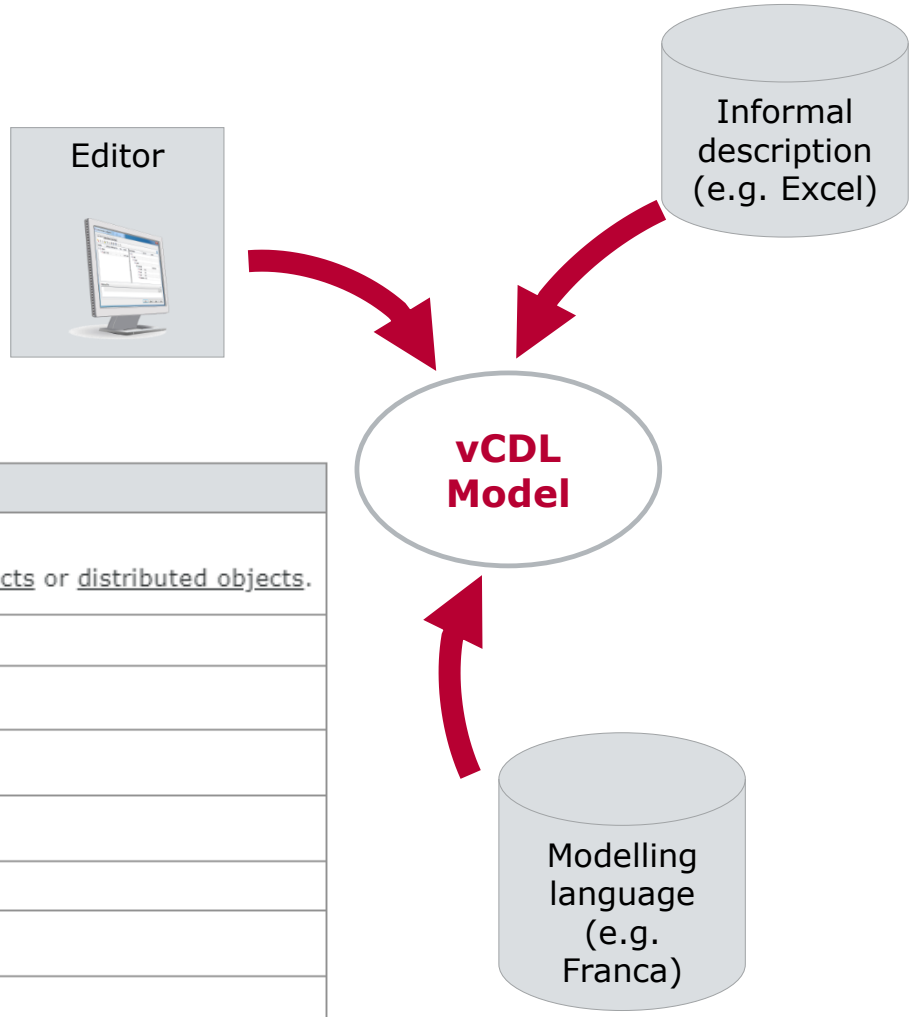


- Service-oriented Architectures (SOA) address these challenges
  - Definition of services independent of underlying protocols

- 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
Database Editor ▸ Creating/changing/deleting PDUs, etc.	**Model Editor** ▸ Enables creating, deleting and changing communication objects or distributed objects.
Network Node:	
▸ Settings (e.g. TCP/IP configuration)	▸ **Model Editor** (Bindings)
▸ Adding CAPL code, etc.	▸ Application models in Communication Setup
▸ Interaction layer configuration	▸ **Model Editor** (PDU Timings, Bindings)
Network:	
▸ Assign application channel	▸ **Model Editor** (Bindings)
▸ Add database/ Import wizard	▸ Data sources in Communication Setup

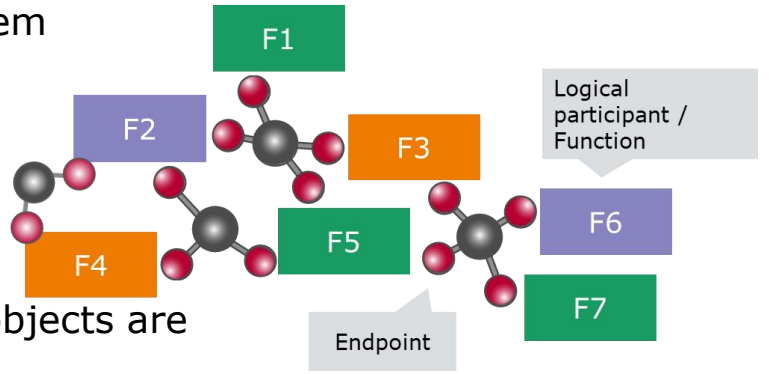
Core vCDL Language Design

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, …
  - Similiar to objects in common prorgamming 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  GetSetNotify;
    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

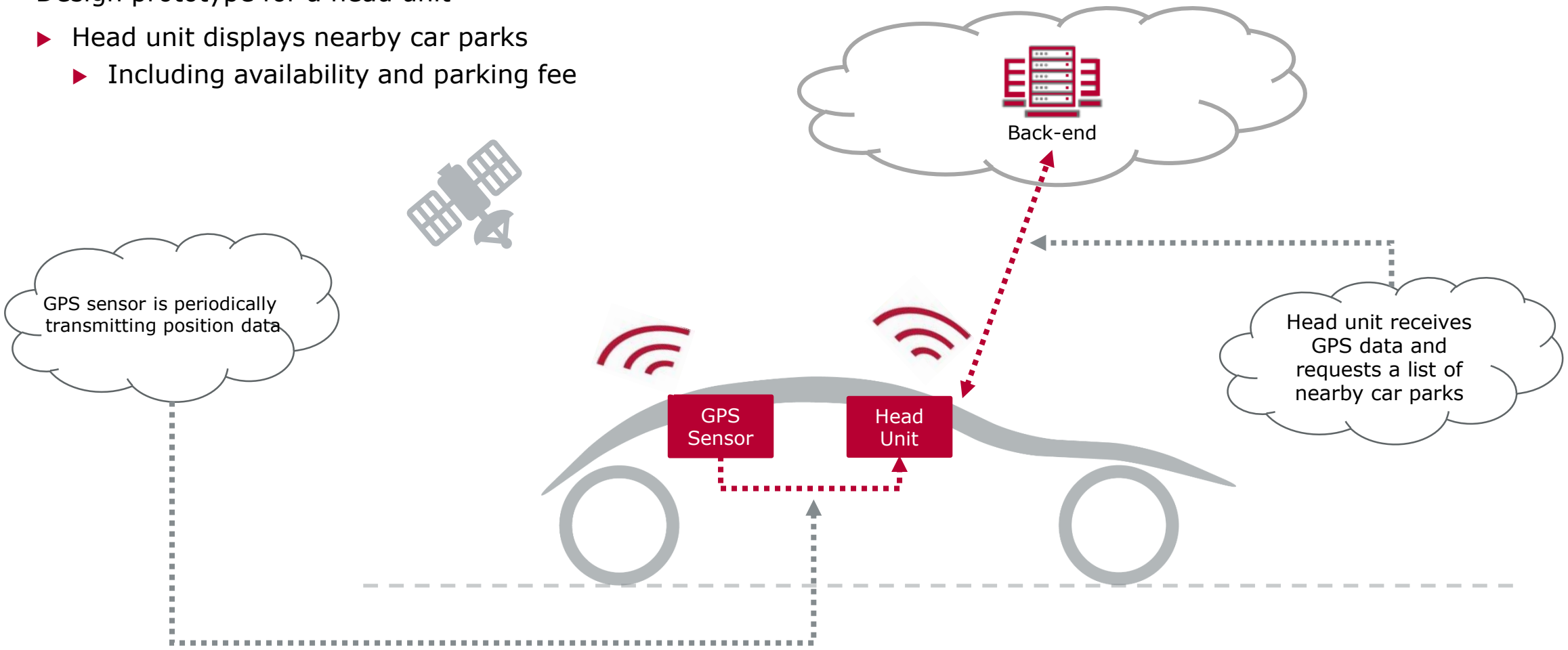
- Interfaces
  - Definition / grouping of communication interfaces
    - E.g. GPS sensor, Headunit, Camera
  - Communication type and direction
    - Method: RPC, Client/Server
    - Data: Sender/Receiver, Event-based

- Distributed objects
  - Instantiation of interfaces
  - Same interface may be instantiated multiple times (e.g. Front camera, rear camera)

Application Scenario example

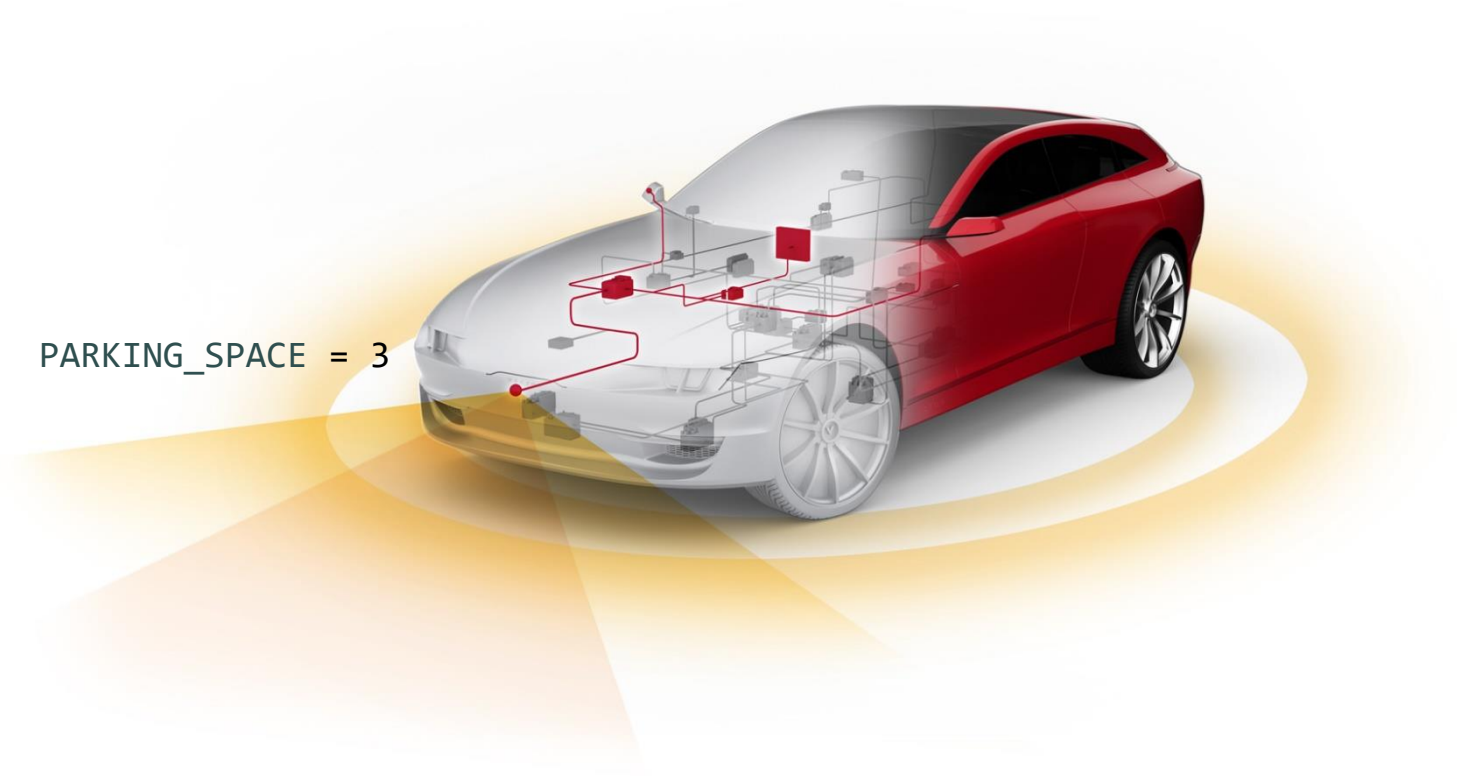
Design prototype for a head unit

- 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;
    }
}
```



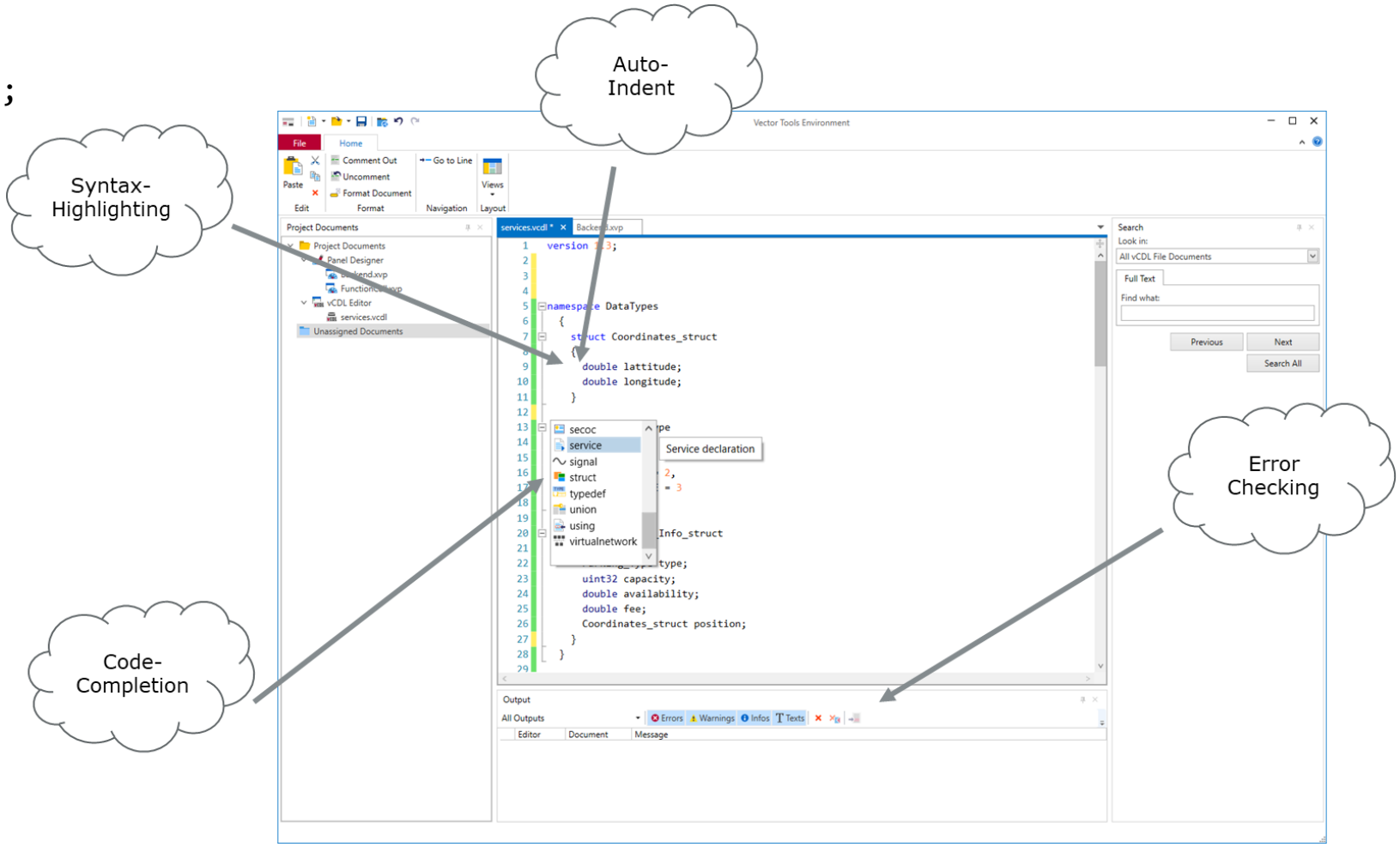
VECTOR

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 Disrtributed Objects

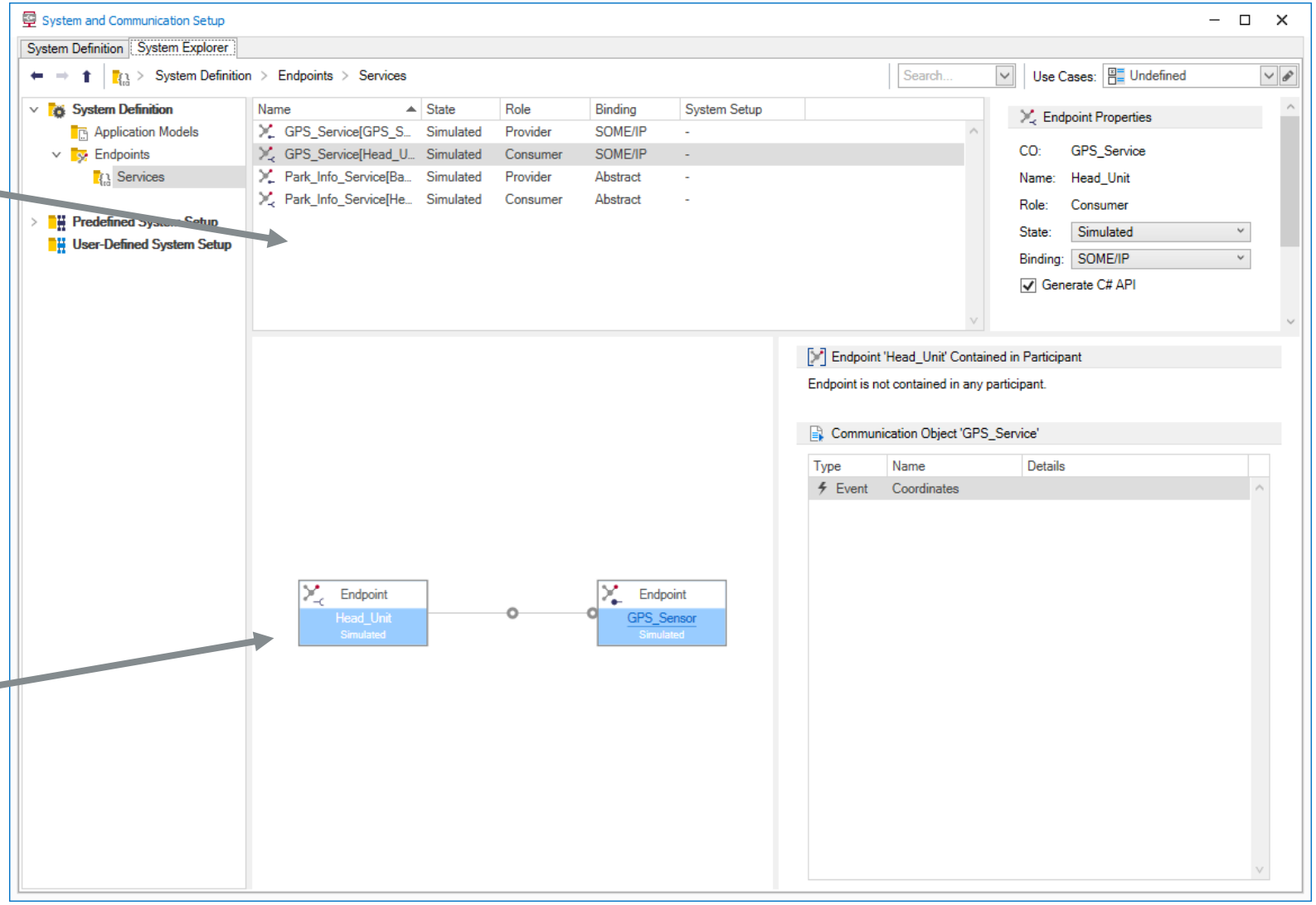
```
[Binding = "Abstract"]
object GPS_Sensor : IGPS_Sensor;

[Binding = "Abstract"]
object Head_Unit : IHead_Unit;

[Binding = "Abstract"]
object Backend : IBackend;
```



Visualization of Services

System overview and configuration

Visualization of service communication and roles



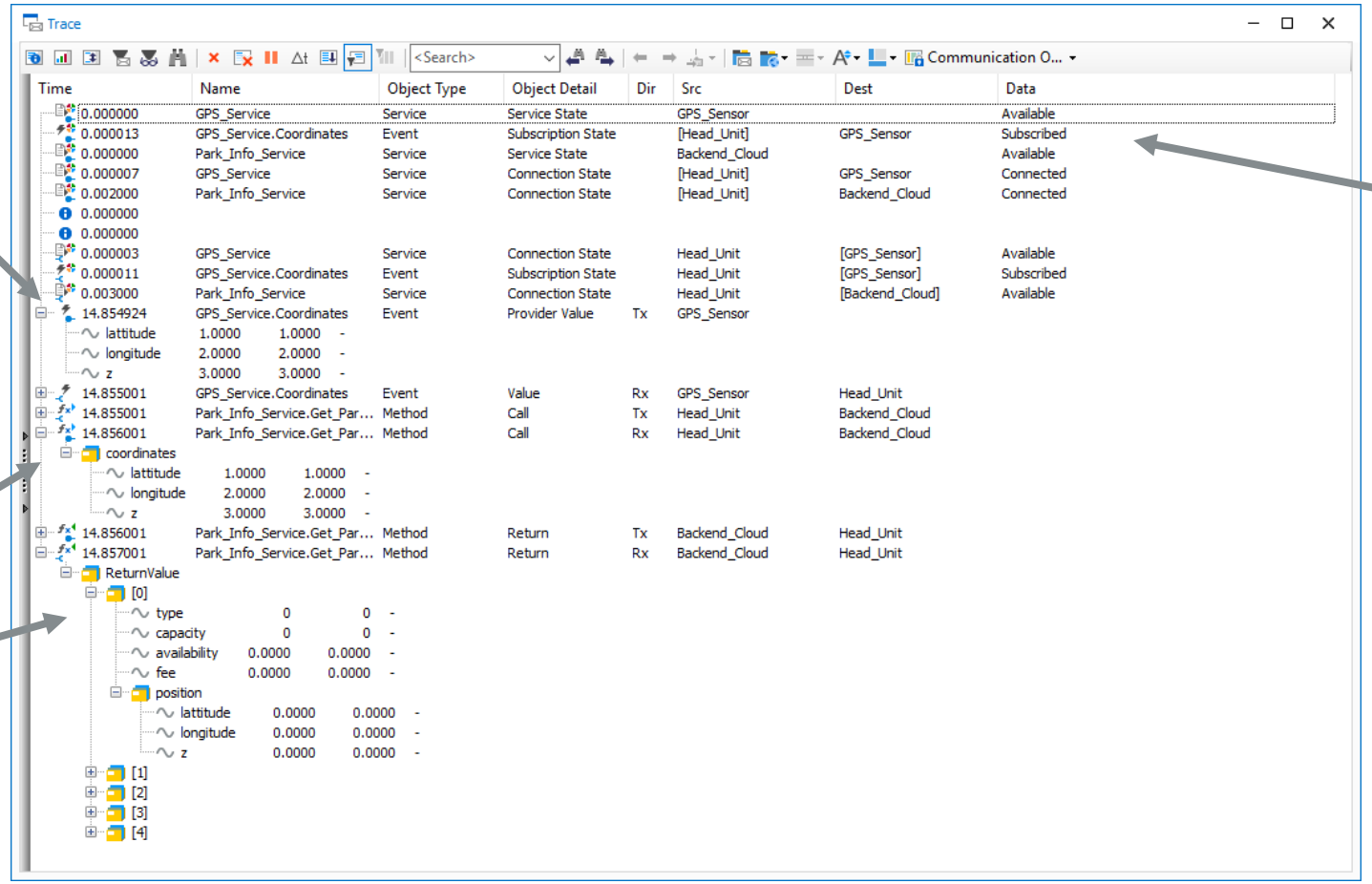
CANoe - Service Orientation

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

Observation of events

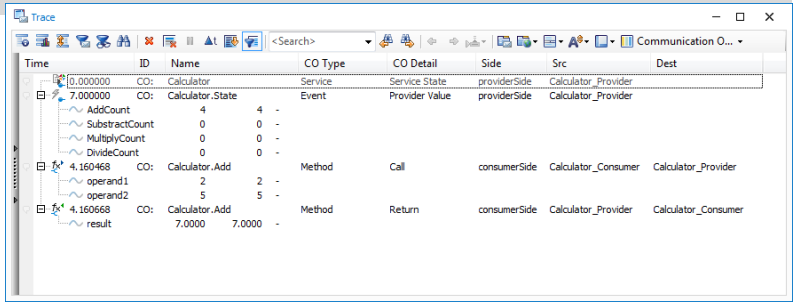
Observation of method calls and returns

Observation of service states

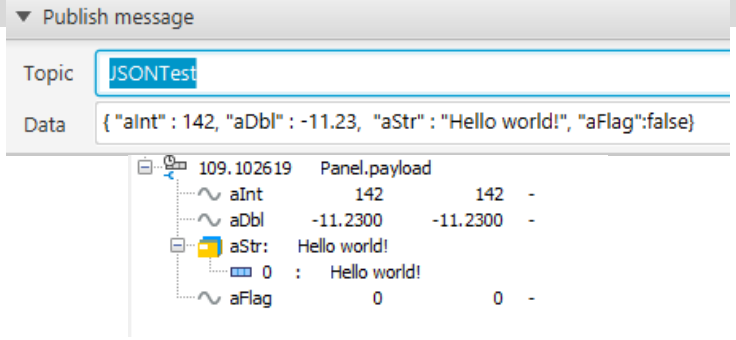


vCDL - SOME/IP,MQTT,DDS

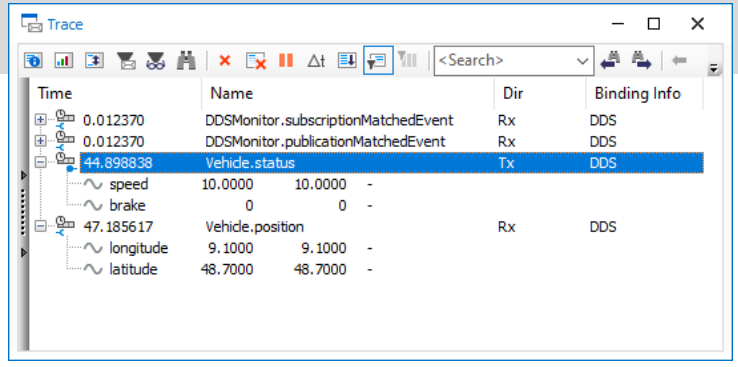
```
[version=1.0, serviceId=11]

service Calculator

{

    void Add(int32 operand1, int32 operand2, out
float result)                    ;

    void Substract(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;
 }
}
```

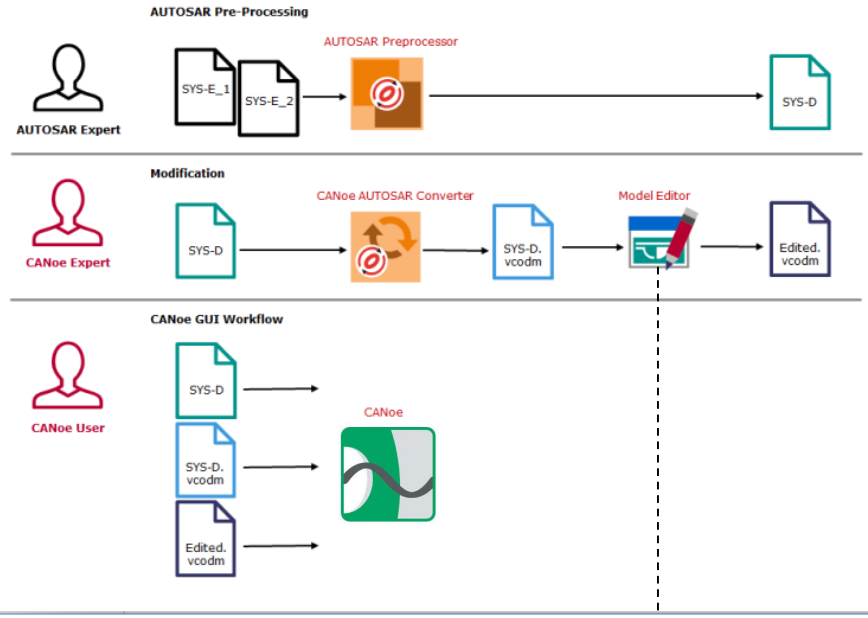


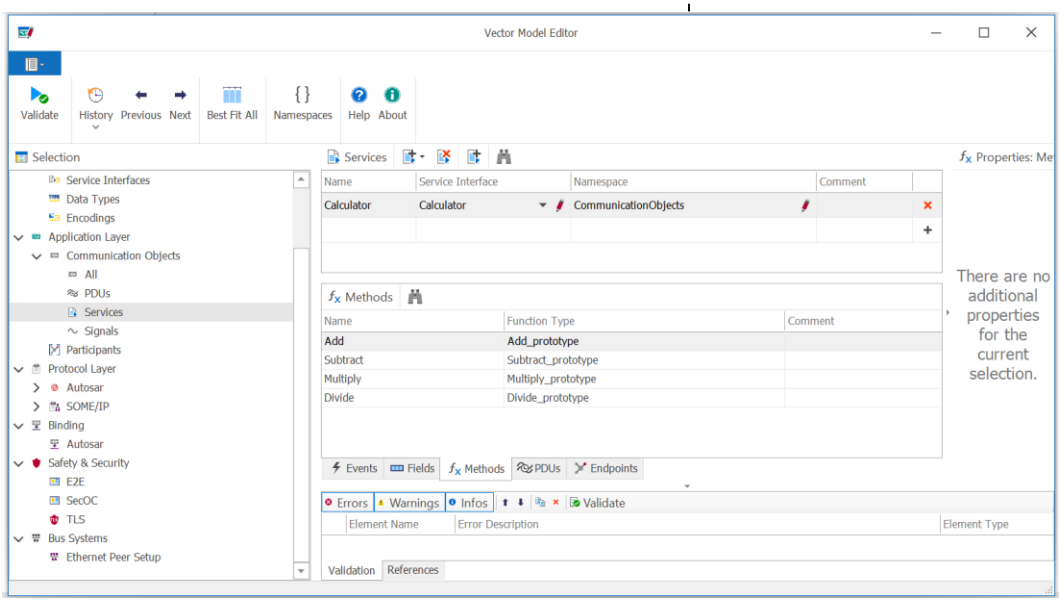

```
struct positionType {
 float longitude;
 float latitude;
}
struct statusType {
 float speed;
 bool break;
}

[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;
  }
```



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

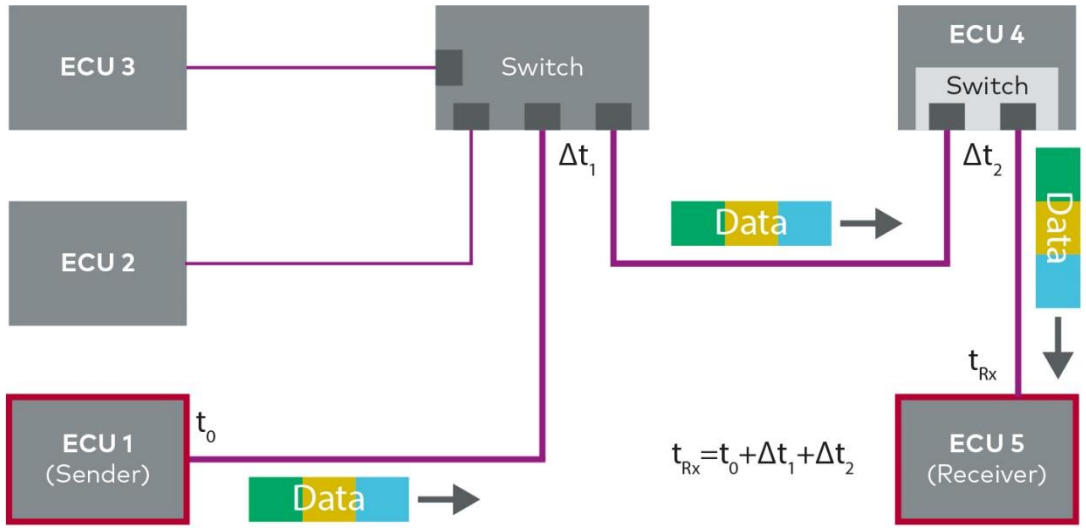
Vector Ethernet Hardware Overview

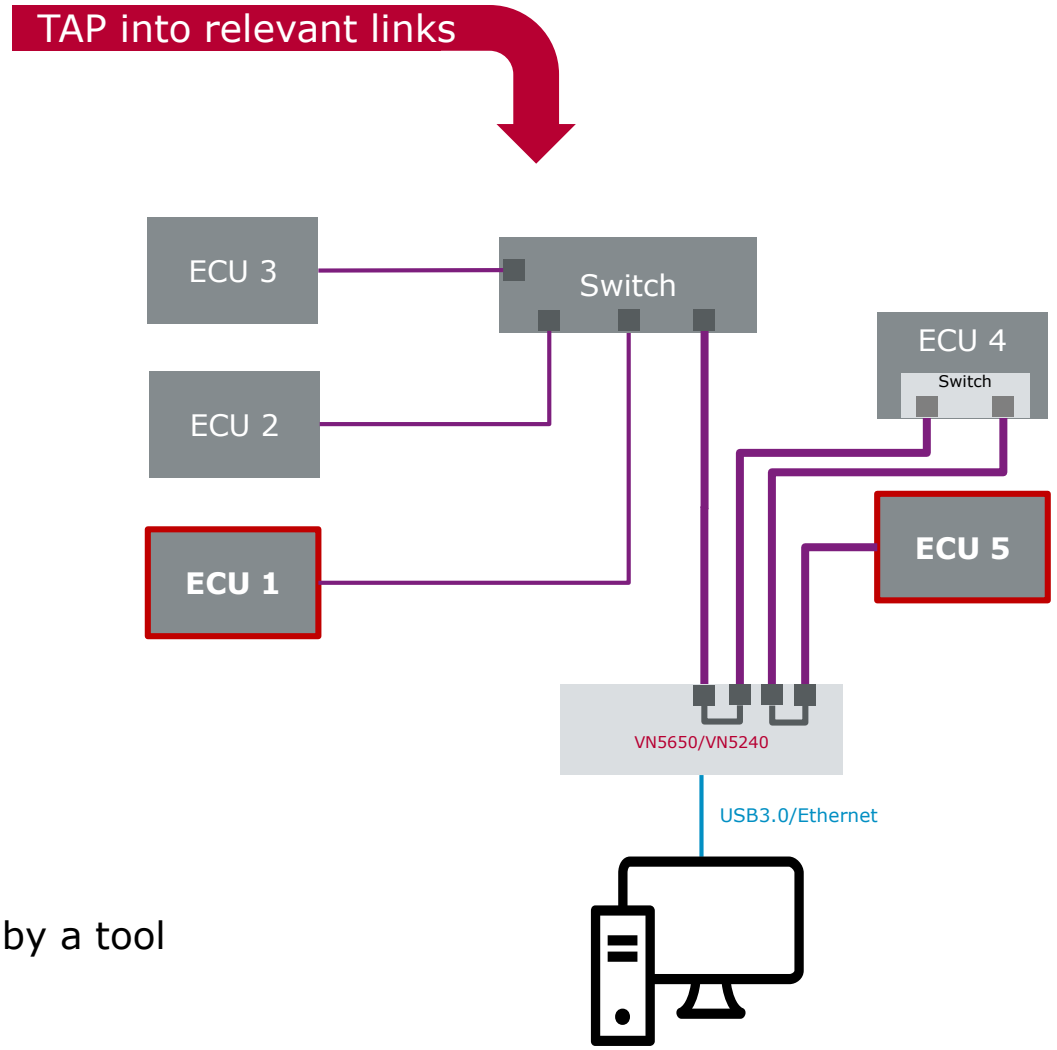
USB to Ethernet Adapter

Ethernet Network Interfaces

Features and supported use cases

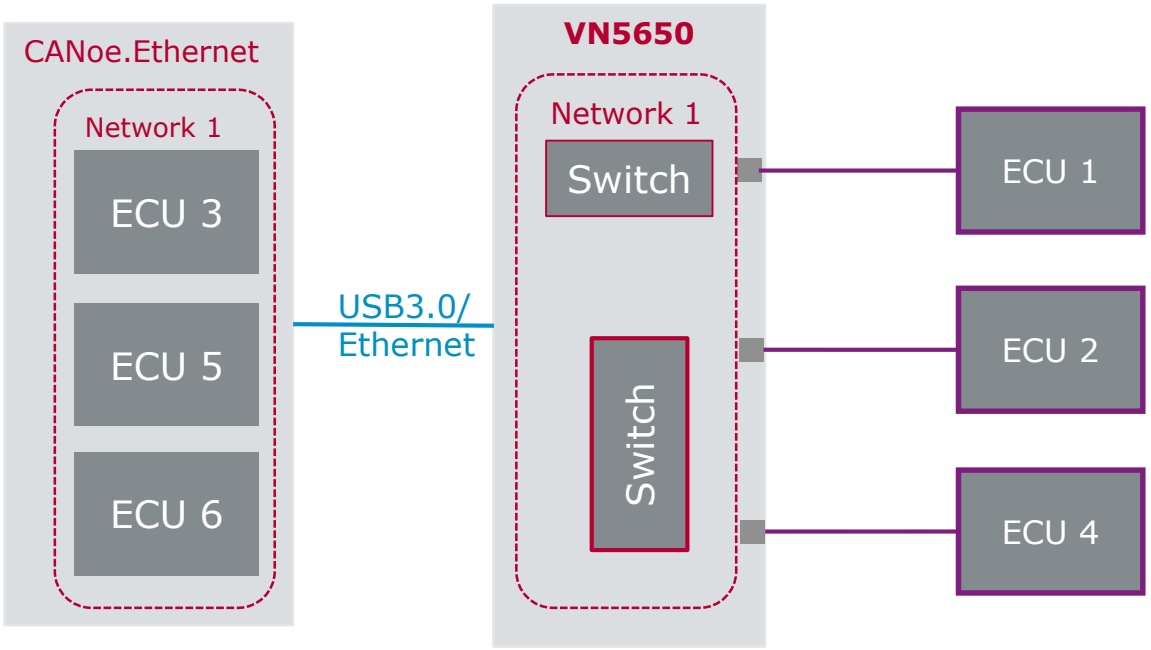
Network Analysis





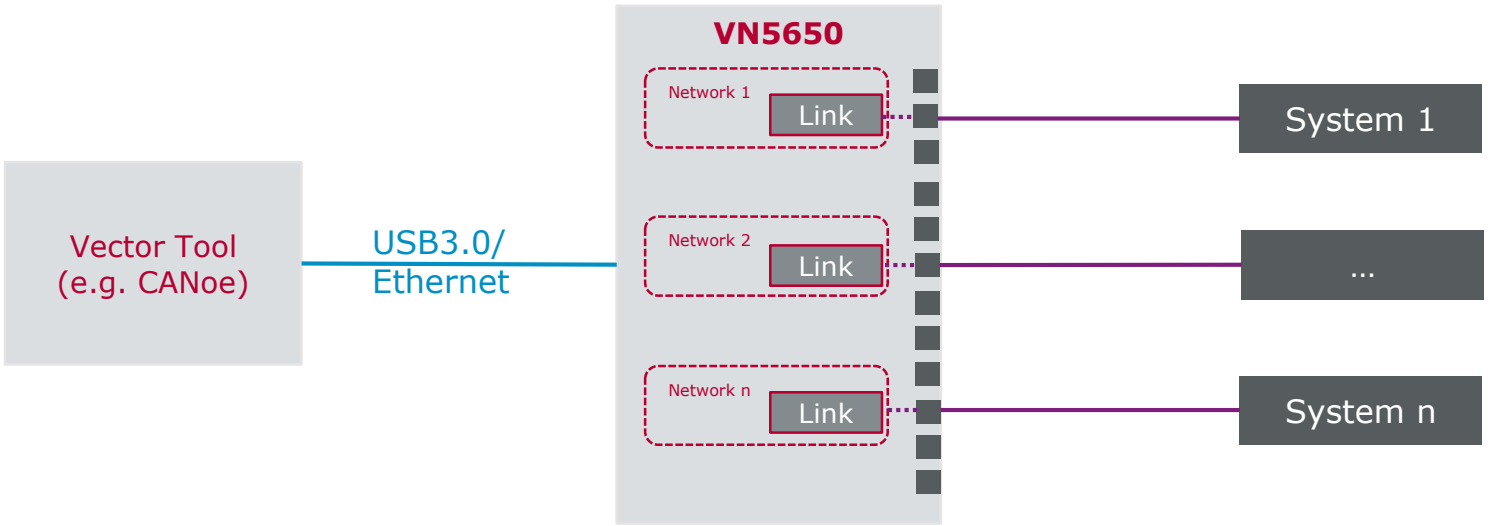
- **Verification of entire data paths:**
  - Transparent data validation (e.g. Frame Errors)
  - End-to-End transmission times
  - Pass-through times of switches
  - Information about dropped frames
  - Possibility to affect communication with frames sent by a tool

Simulation



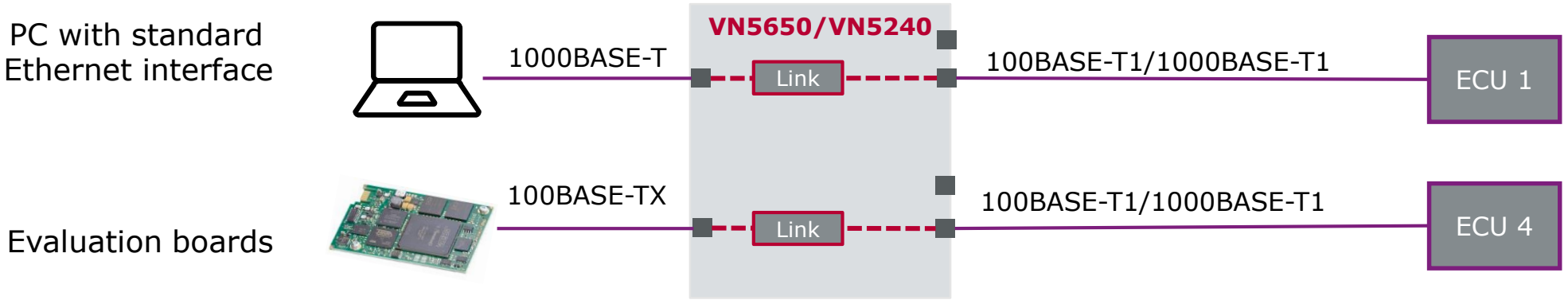
- **Simulation within an existing network**
  - Simple network access over integrated switch
    - variable ECU wiring possible, without simulation impact

Direct Access



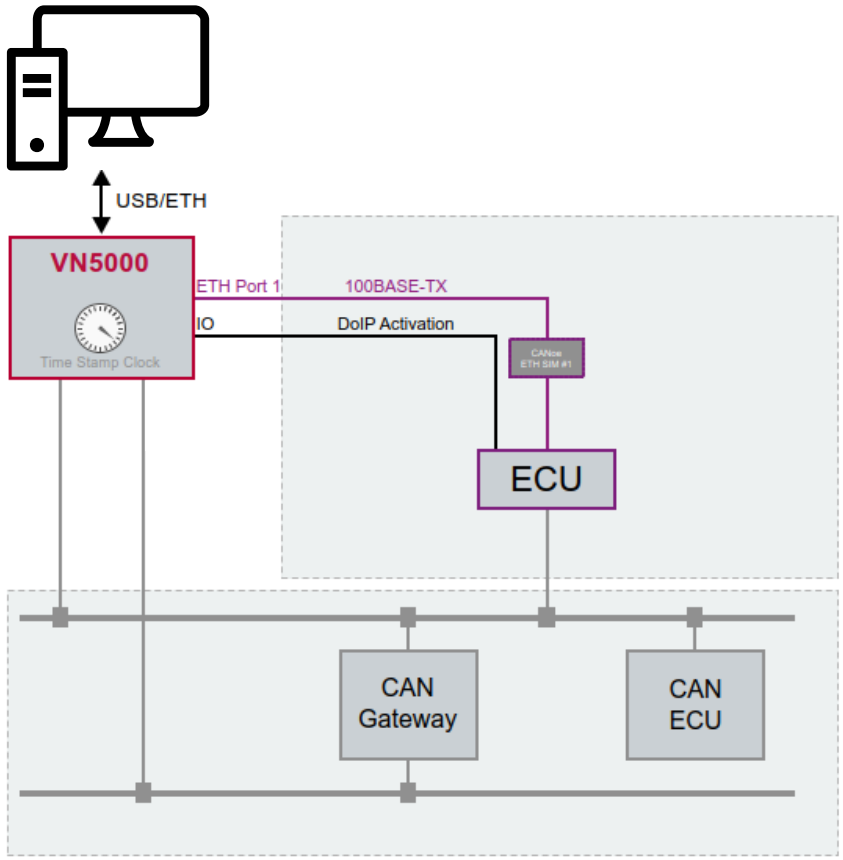
- **Individual access to each link e.g. for**
  - Flash reprogramming of ECUs (Electronic Control Units)
  - Vehicle diagnostics
  - Test benches (test of multiple identical systems)
  - Test of ECUs with multiple ports

Media Conversion

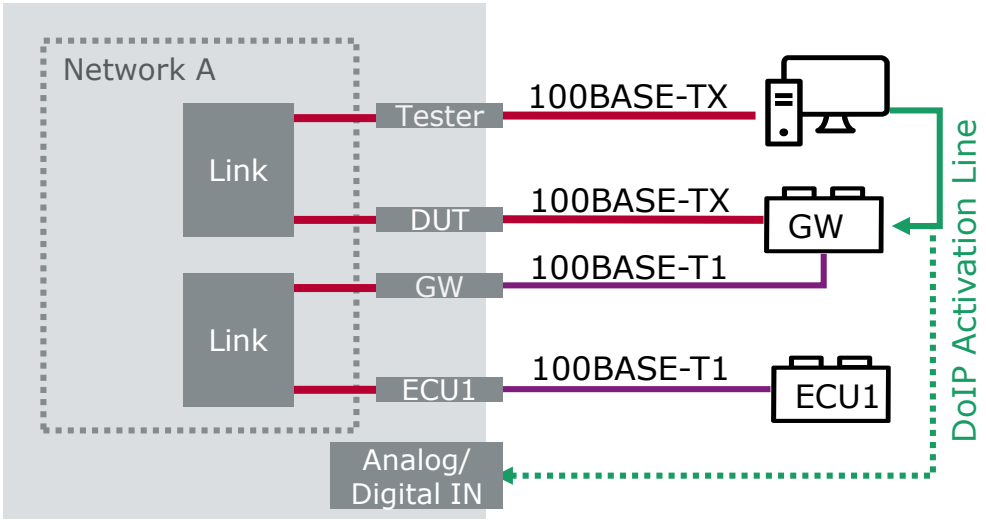


- Media conversion between different physical layers
- VN5240:
  - Up to 3 converters between IEEE 100BASE-T1/1000BASE-T1 and IEEE 100BASE-TX/1000BASE-T
- VN5650:
  - Up to 4 converters between IEEE 100BASE-T1/1000BASE-T1 and IEEE 100BASE-TX/1000BASE-T

Diagnostics Over IP

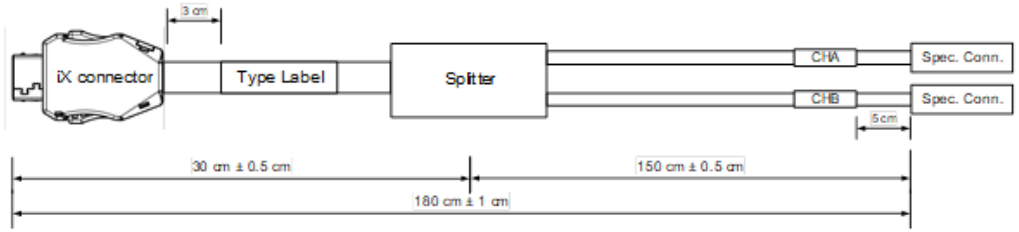


- Access on diagnostic links via the 100BASE-TX ports as a tester
- Set/Read DoIP Activation Line by using the onboard analog/digital signal interface
- TAP in-between the tester and DUT communication
  - Capture diagnostic requests/responses
  - Capture forwarded messages on in-vehicle side
  - Capture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

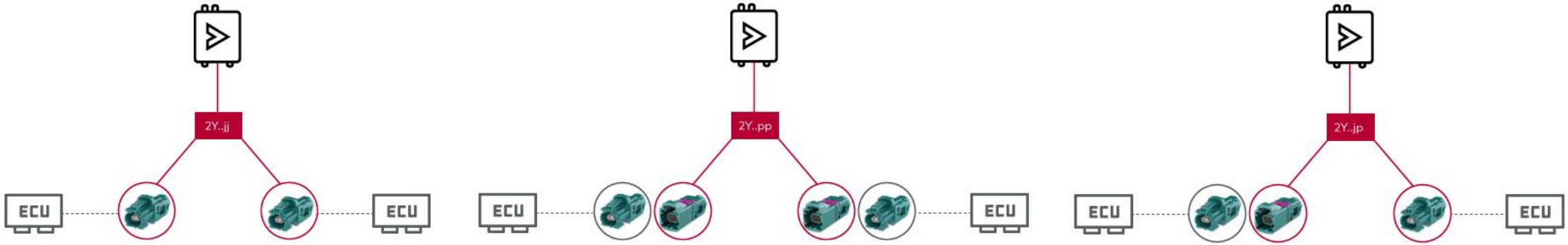


AEcable Family (Cable Sets for 100BASE-T1/1000BASE-T1)

- Different cable variants are available, to adapt ix industrial to different other plug systems:
  - H-MTD
  - HSD
  - MATEnet
  - DSUB9
  - RJ45



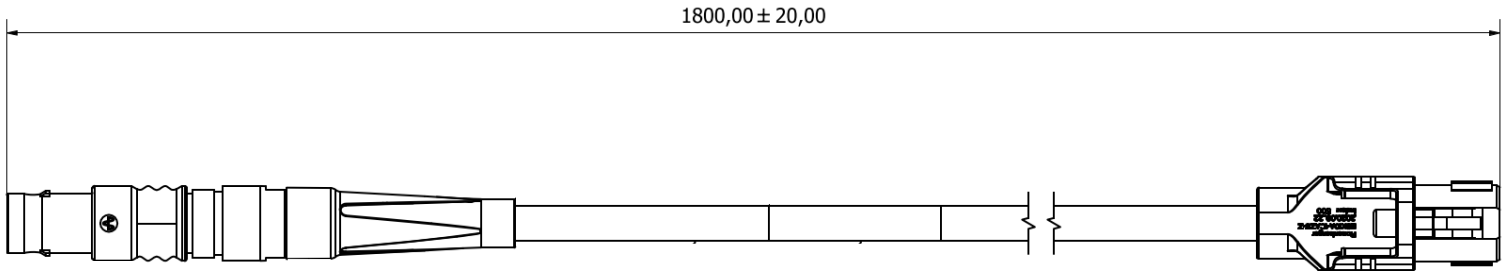
- The cables are available with plug/header (male contacts) or jack/frame (female contacts)



- 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	

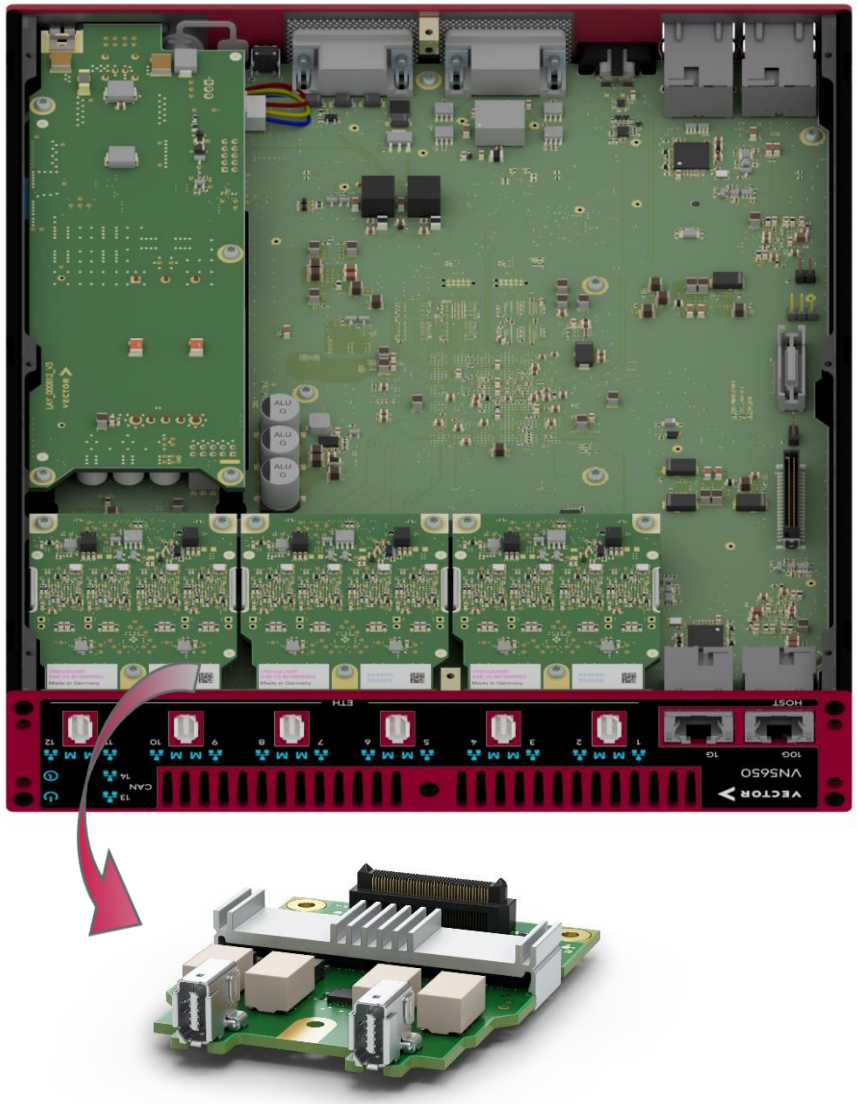


Connector for Vector Interfaces: YCP-BPR09ACX-S1MSCDX-051X YAMAICHI ELECTRONICS

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, 10000BASE-T1, MultiGig Automotive Ethernet, MACsec

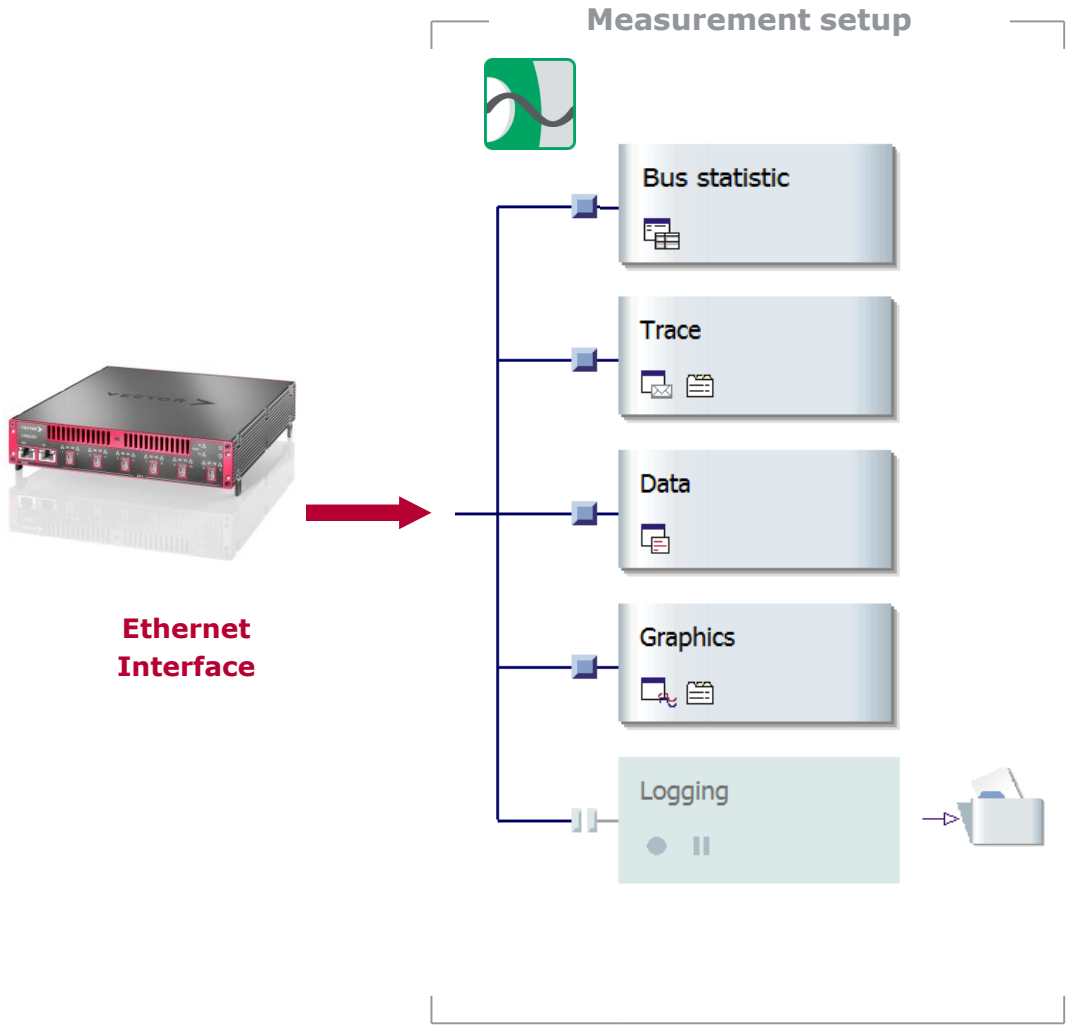
	Standard	Connector	PHY	Features
Already Available				
VNmodule60 4AE1G BCM89883	IEEE 100/1000BASE-T1 (4 ports)	ix Indrustrial	Broadcom BCM89883	OPEN Alliance TC10 @ 100BASE-T1
VNmodule60 4AE1G 88Q2112	IEEE 100/1000BASE-T1 (4 ports)	ix Indrustrial	Marvell 88Q2112-A2	1000BASE-T1 Legacy Mode
VNmodule60 1AE10M LAN8670	IEEE 10BASE-T1S (1 port) IEEE 100/1000BASE-T1 (2 ports)	ix Indrustrial	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 Indrustrial	Marvell 88Q2221M Rev. B1	OPEN Alliance TC10 @ 100BASE-T1 OPEN Alliance TC10 @ 1000BASE-T1 MACsec IEEE 802.1AE

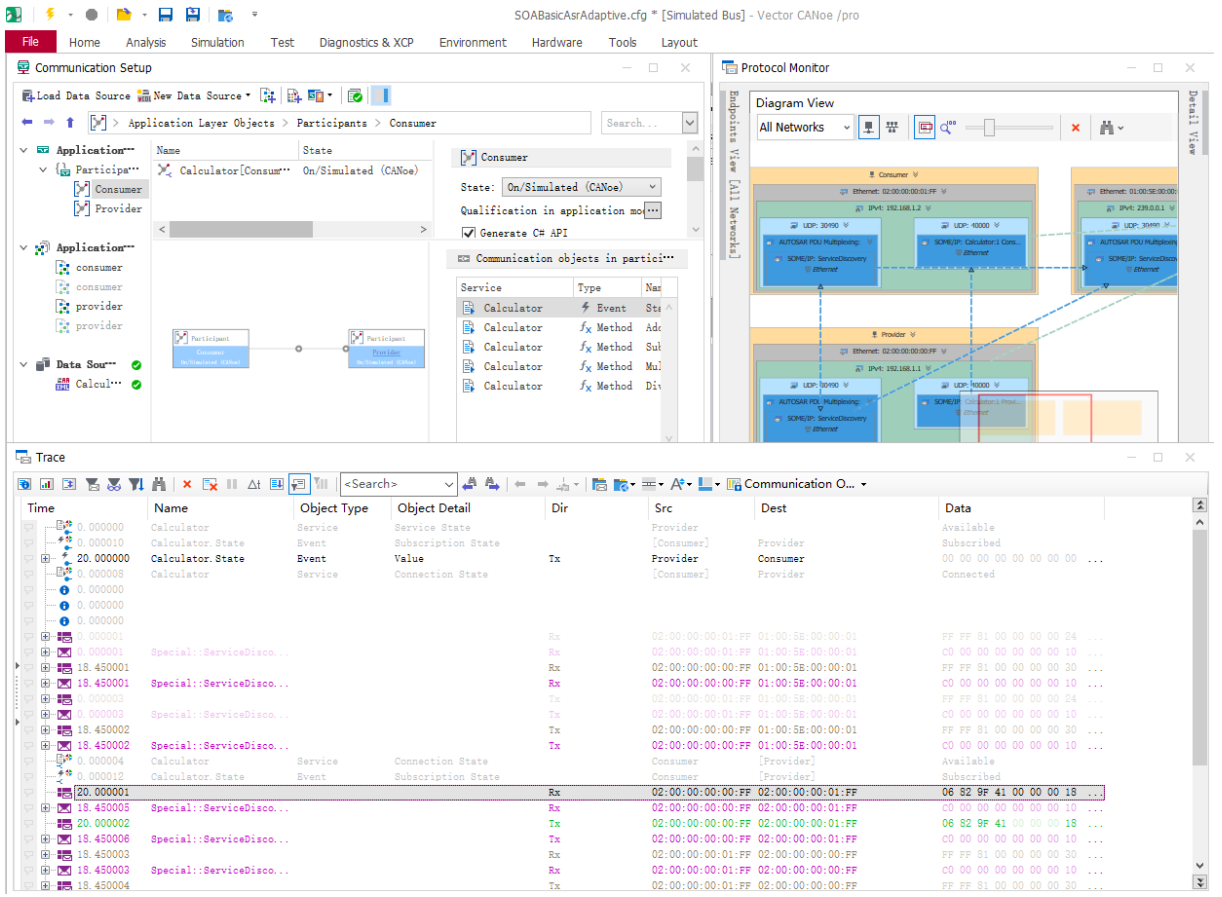


Agenda

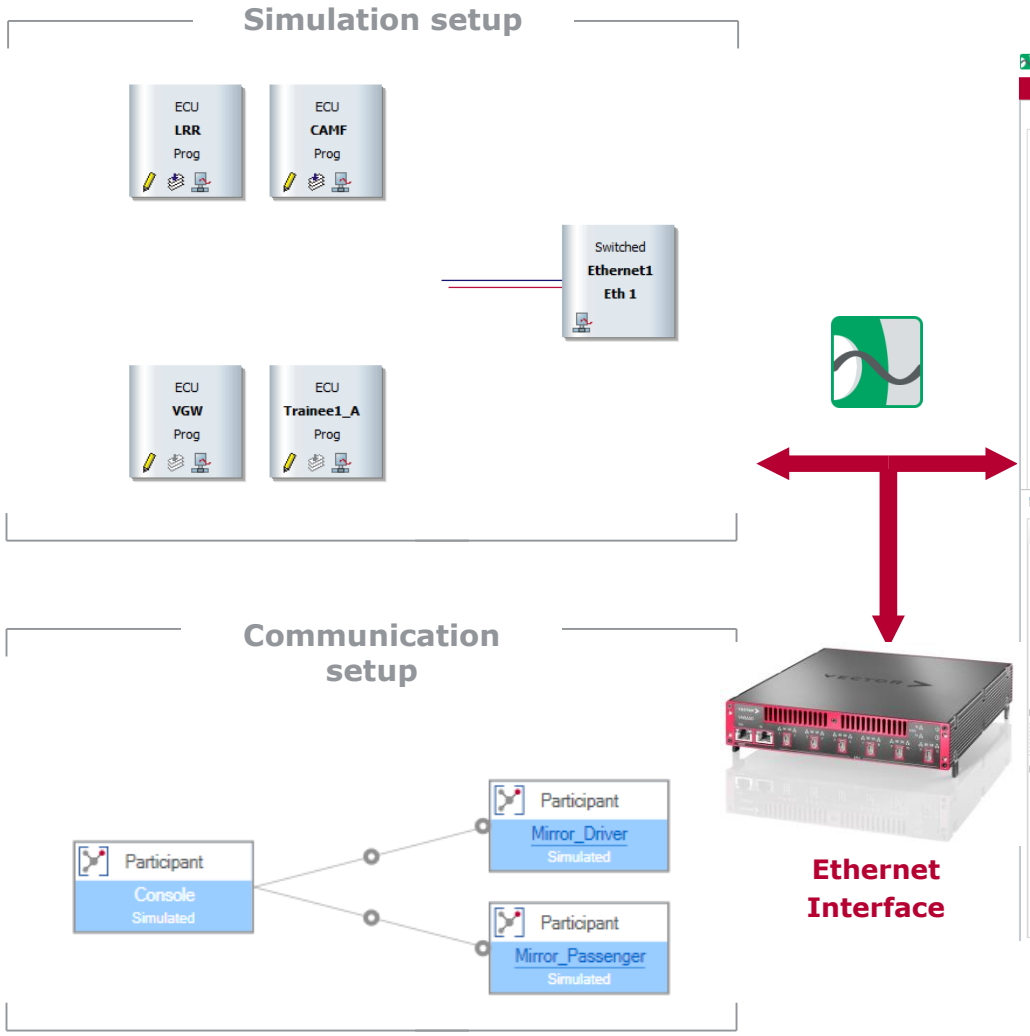
1.	Data source vCDL Beyond arxml for Ethernet and SOA
2.	Network Interface for Automotive Ethernet
3.	Communication Testing, Security and Analysis
4.	More SOA – More Software – More Testing

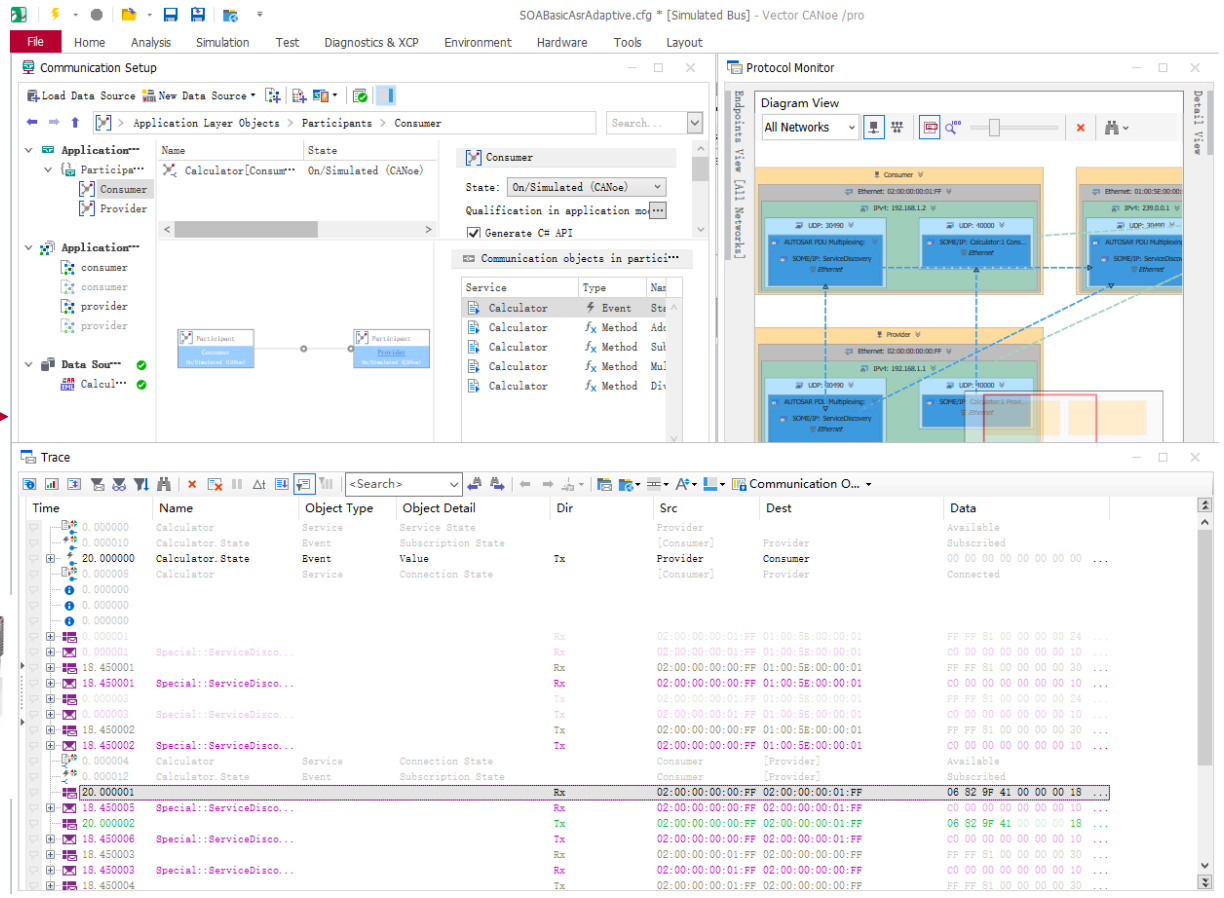
Measurement and Analyzing





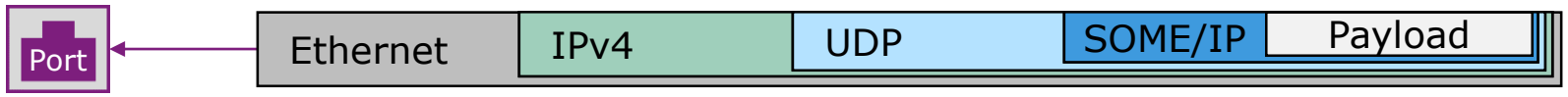
Simulation in CANoe





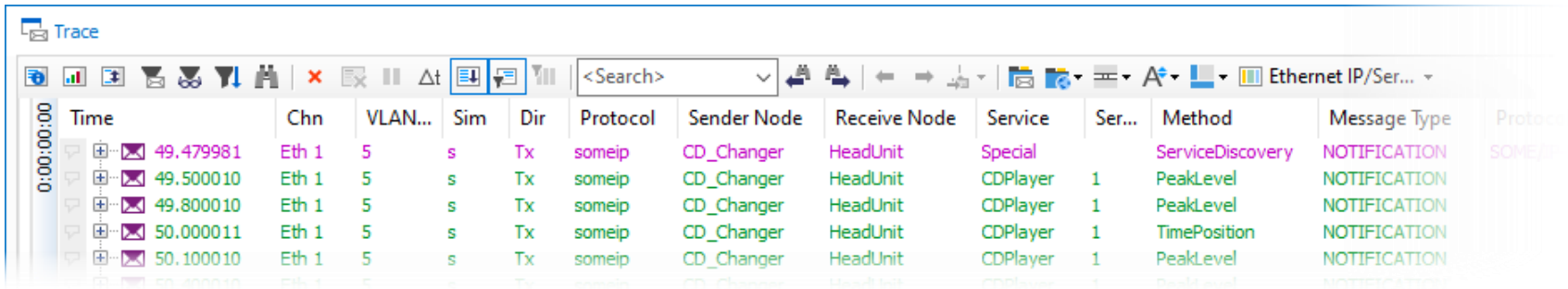
Protocol Monitor And Trace Window

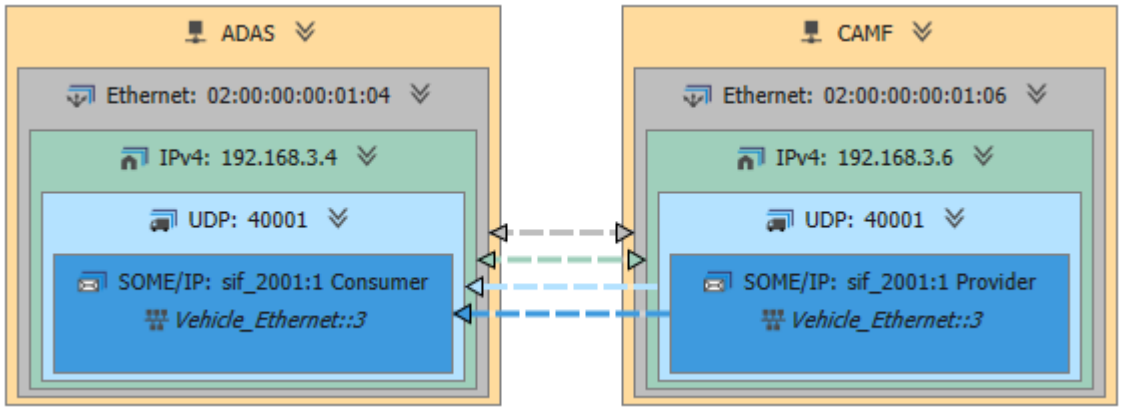
Ethernet Frame:



Trace

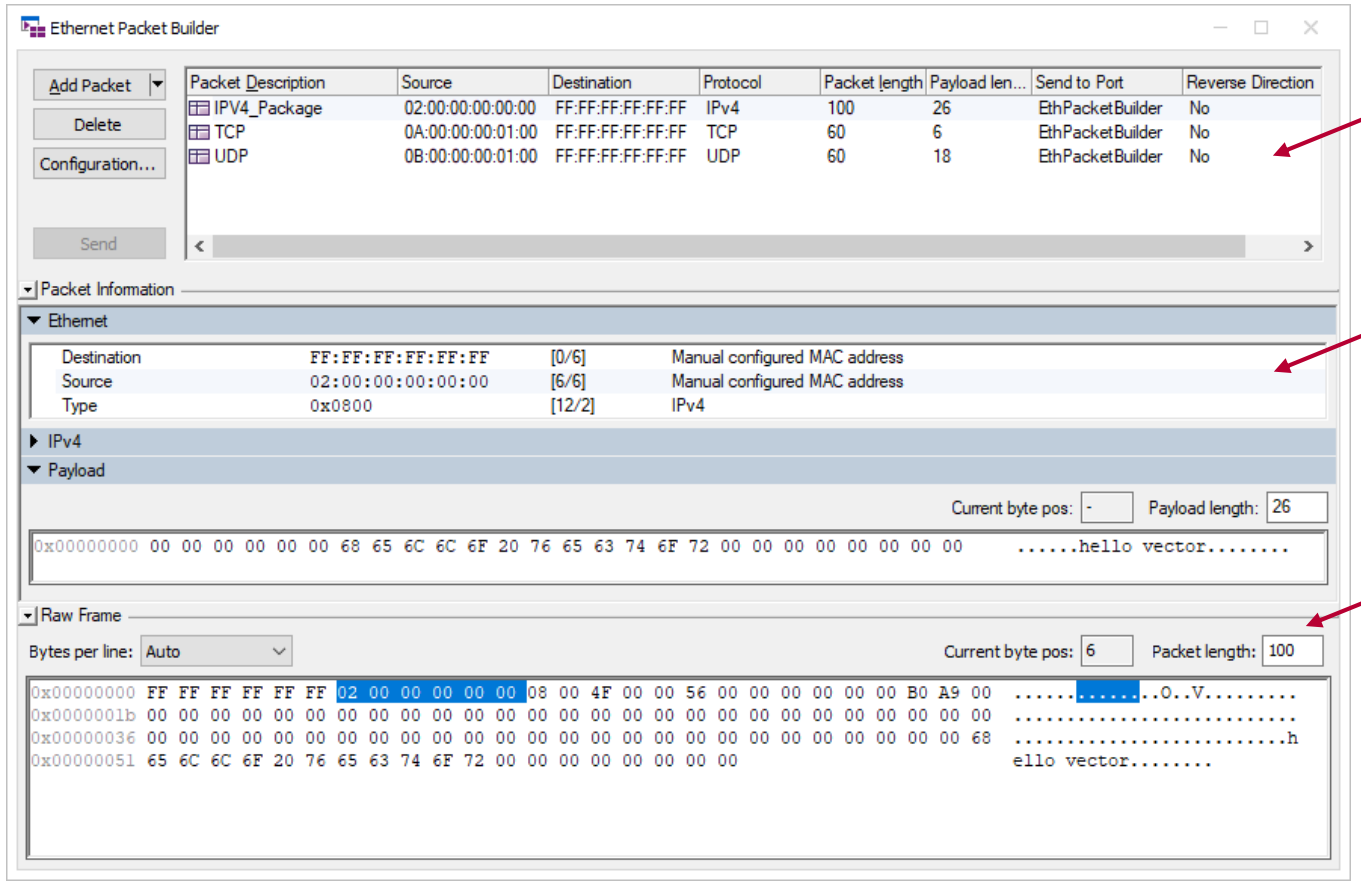
Protocol Monitor





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.

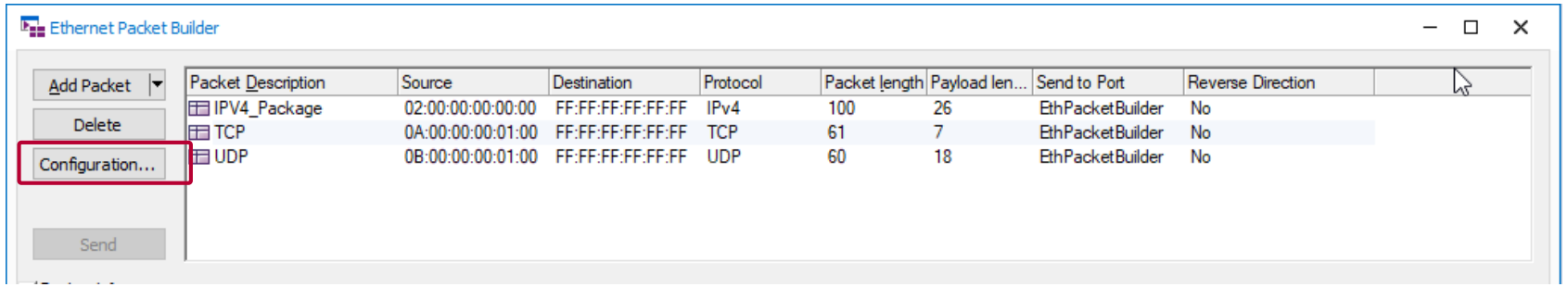
- **Raw Frame**

Hexadecimal presentation field:
Raw frames can be imported

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

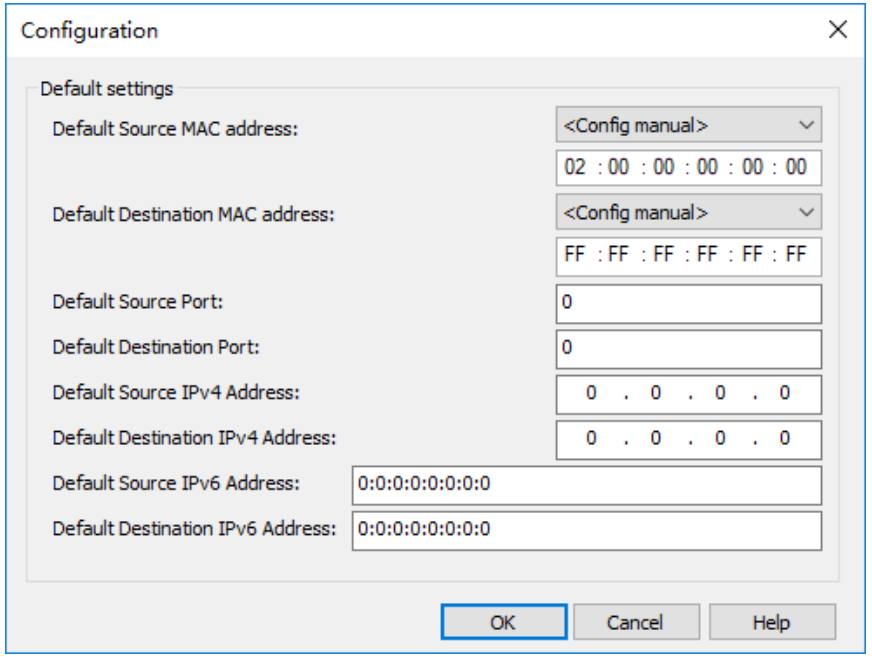
Bytes per line and Packet length can be adjusted.

Ethernet Packet Builder



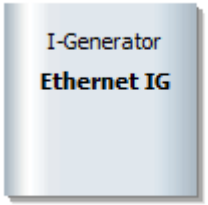
- **Settings:**

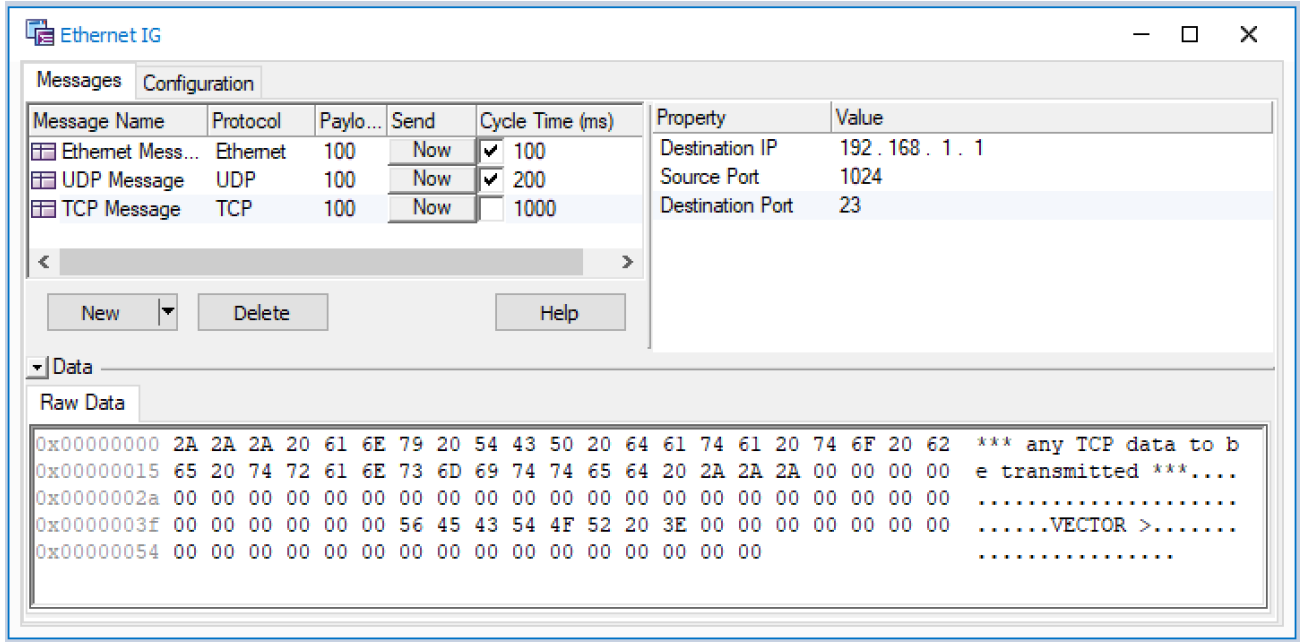
Default values for source and destination MAC ID, IP addresses and ports of added packets



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.





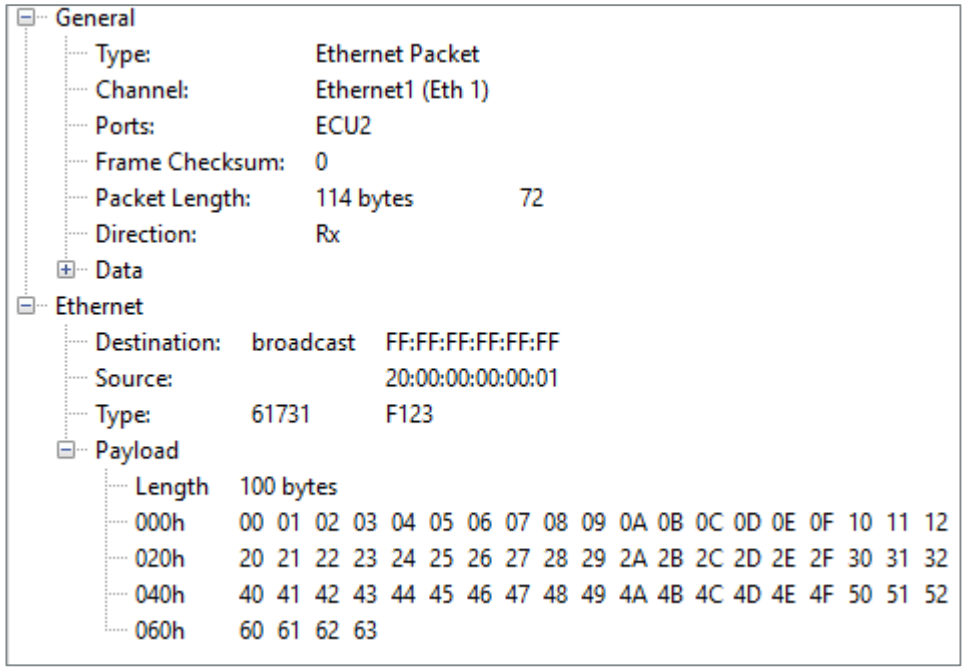
*Only CANoe; feature is in maintenance mode

Ethernet Packet

- Send one ethernet packet (without IP information)
  - Output (Ethernet)

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

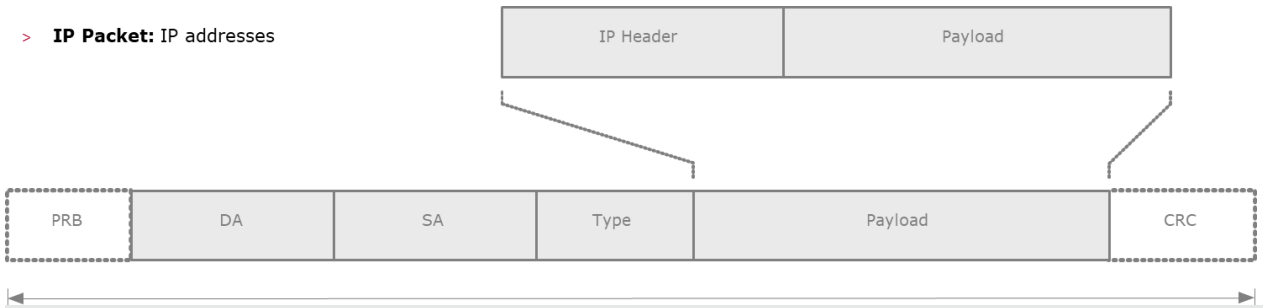
```
on key 'a'
{
  ethernetPacket txPacket;
  int i;

  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);
}
```



VECTOR

Send IP Segments with CAPL

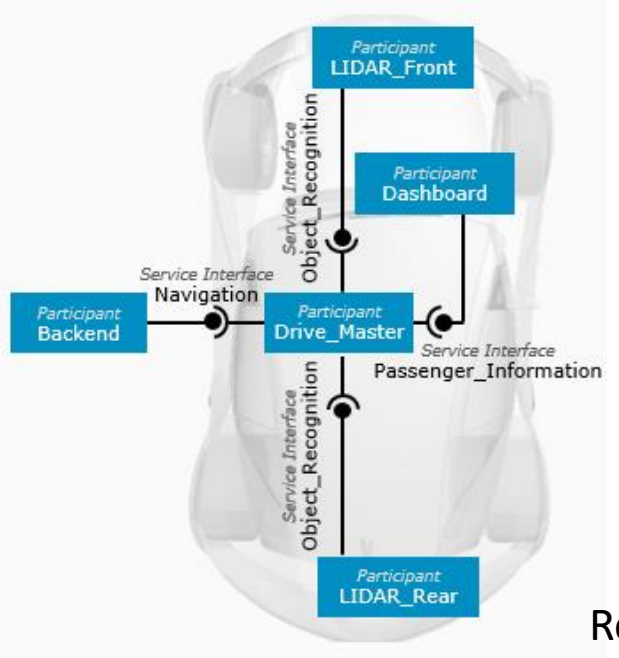
Example




```
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 00
      020h      00 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 00
      060h      00 00 00 00
```

Application, Conformance and Robustness Testing with SOMEIP



Application

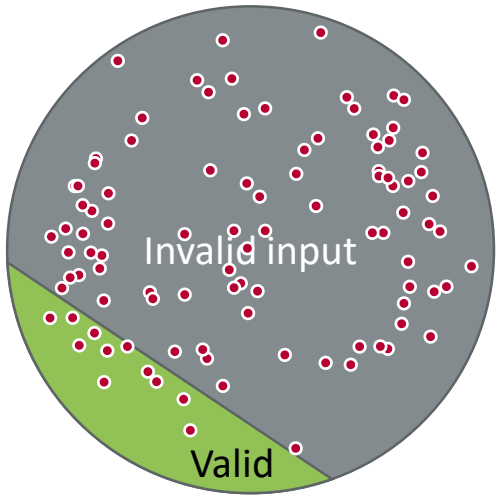
Conformance

Robustness

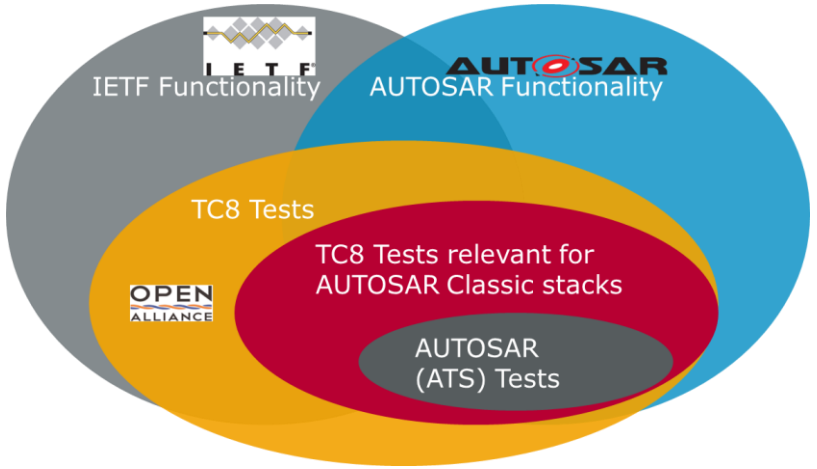
OPEN Alliance Automotive Ethernet ECU Test Specification Layer 3-7

TC8 ECU Test

OPEN ALLIANCE

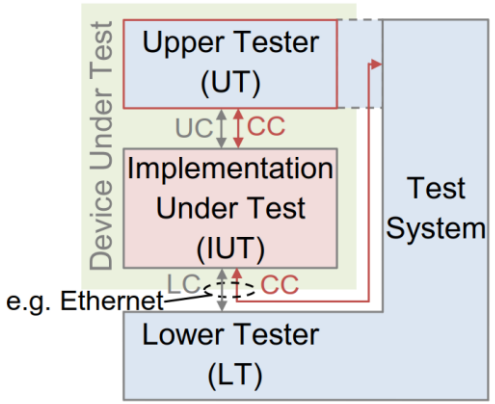


Fuzz Testing

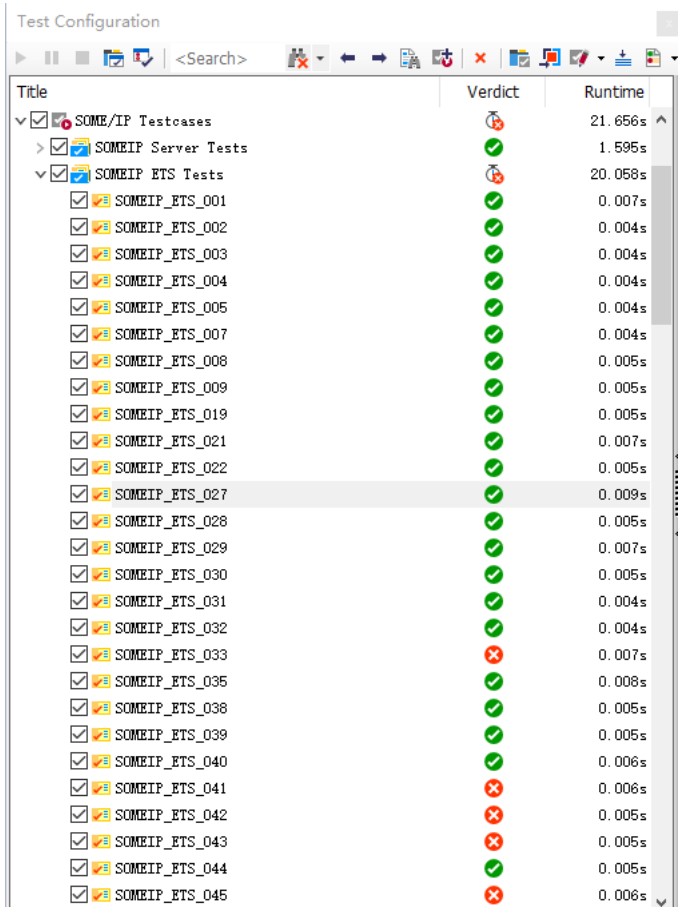


SOME/IP Server and ETS testing in TC8

- 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 though the IUT itself



What is vTESTstudio?

Coding-based Design

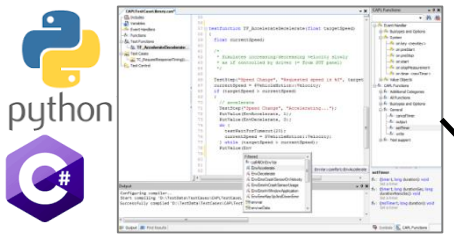
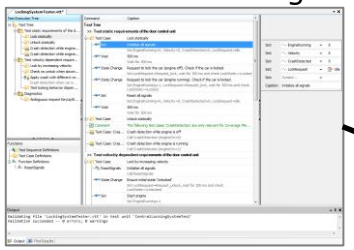
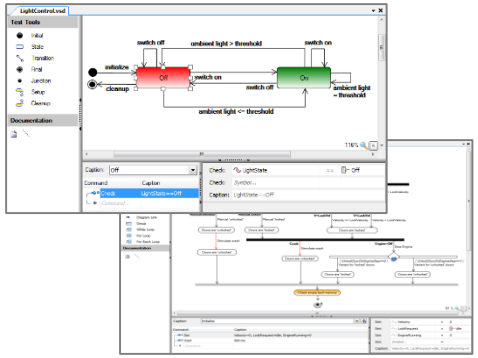


Table-based Design



Model-based Design



Data-driven Design

MiL,SiL,PiL,HiL,
ViL,Fuzzy,TC8…

AUTOSAR, C,
C++, Python

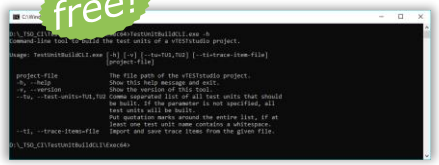
ADAS,HPC,ZCU,
HMI,OTA…

vTESTstudio

vTestStudio GUI

TestUnitBuildCLI

free!



Traceability for ISO
26262,ISO 21434…

POLARION

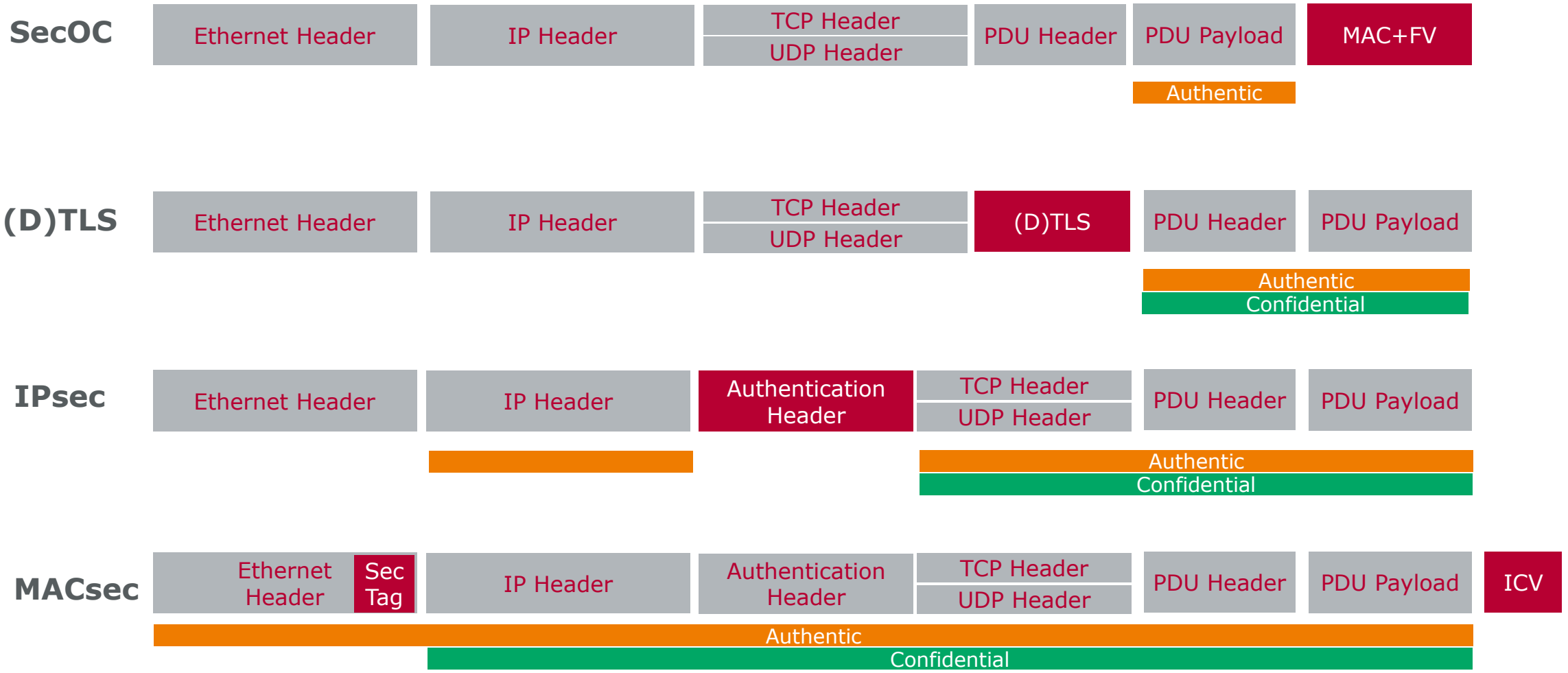
Squore

Vector Tools

3rd party HW/SW

VECTOR

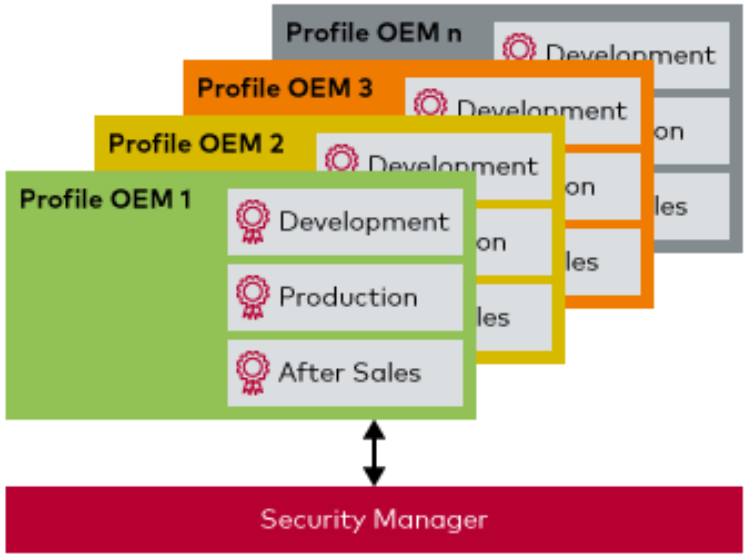
Security and Protection

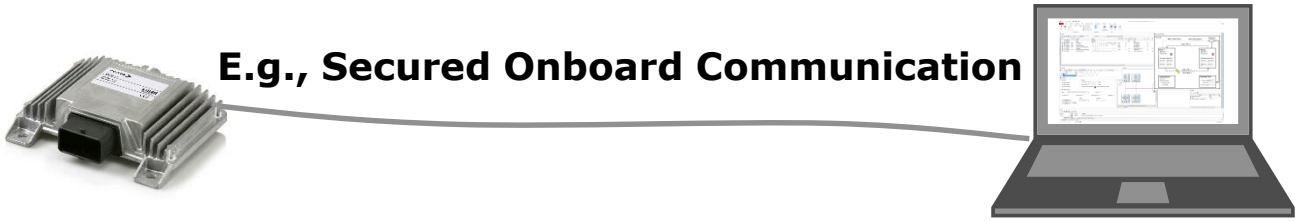


Smart Charging with TLS in ISO 15118 via PLC

	ISO 15118-2	ISO 15118-20
TLS is mandatory for the following use-cases:	▸ Plug & Charge ▸ Value-Added-Service	▸ TLS is always mandatory (EIM & PnC)
Supported Cipher Suites (TLS 1.2)	▸ TLS_ECDH_ECDSA_WITH _AES_128_CBC_SHA256 ▸ TLS_ECDHE_ECDSA_WITH _AES_128_CBC_SHA256	▸ TLS_ECDH_ECDSA_WITH _AES_128_GCM_SHA256 ▸ TLS_ECDHE_ECDSA_WITH _AES_128_GCM_SHA256

Tools without Security is **challenges!**





Communication analysis

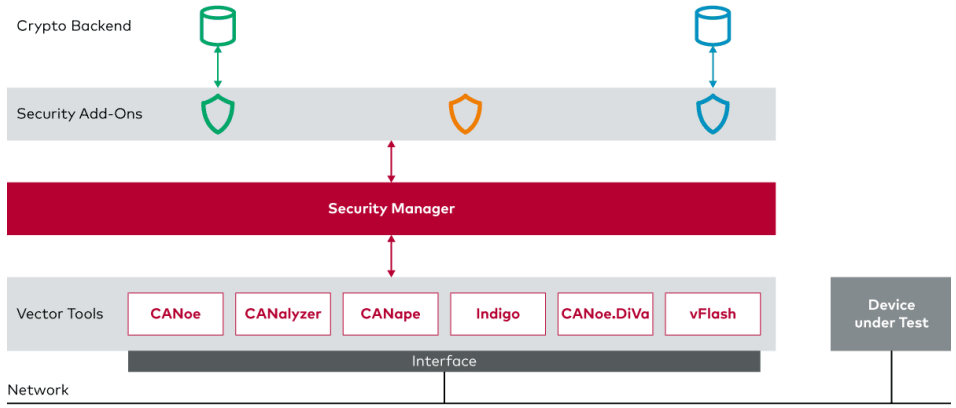
- Reading data
- Verification of authentication information
- Check data integrity
- Detection of old / replayed data

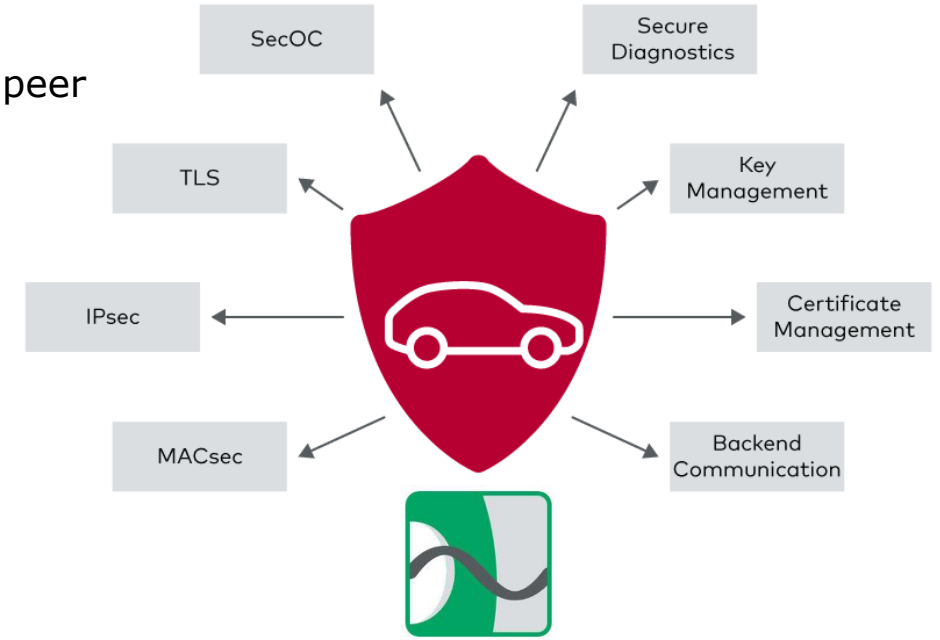
Remaining bus simulation / Stimulation

- Authenticator calculation
- Freshness management

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

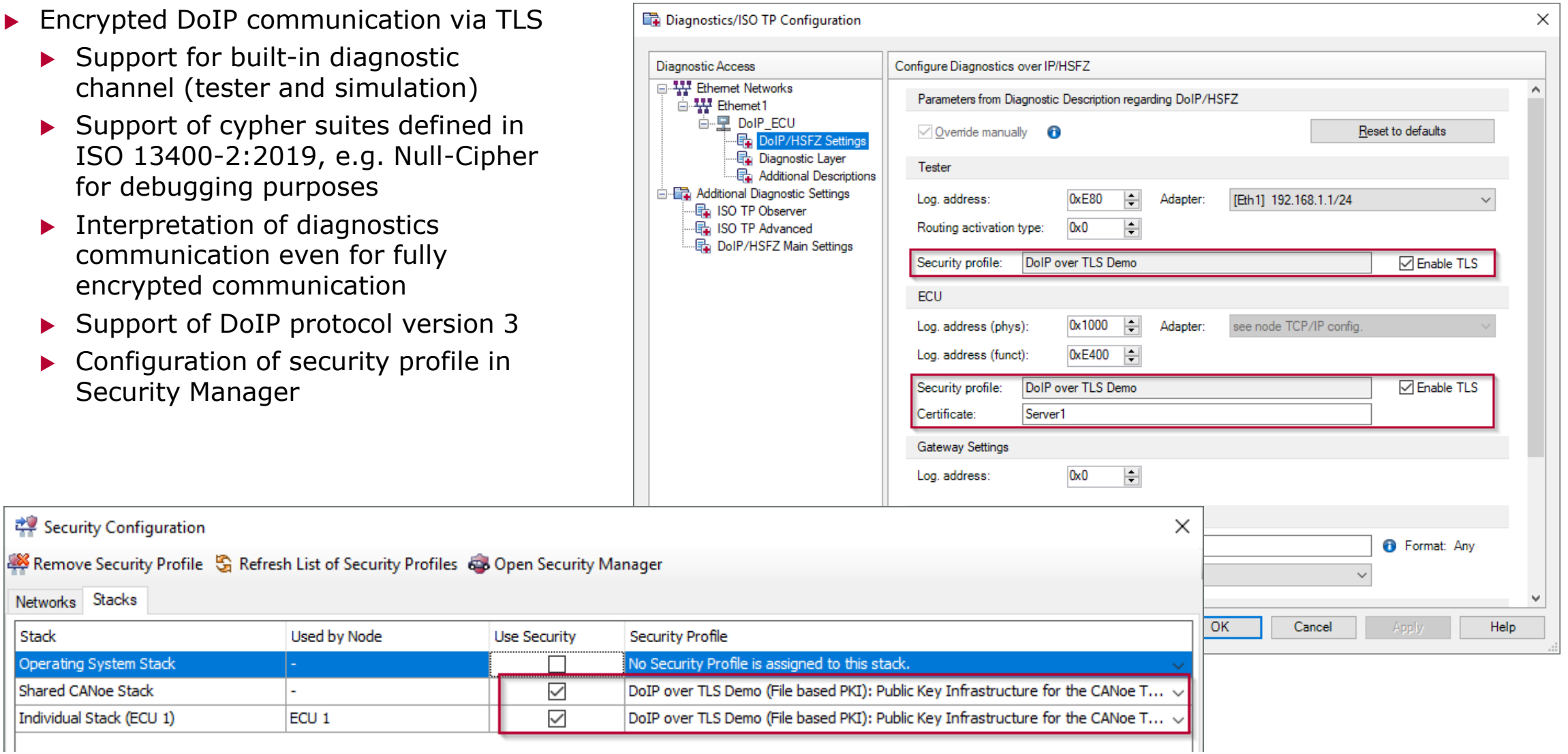
- Communication: SecOC
- Diagnostics: Authentication
- Diagnostics: Variant Coding
- TLS: Simulation of Client and Server
- TLS: TLS Observer using Master Secret
- TLS: DoIP over TLS
- IPsec: IKEv2 support for certificate based peer authentication, dead peer detection, IKE fragmentation and IKE rekeying
- 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



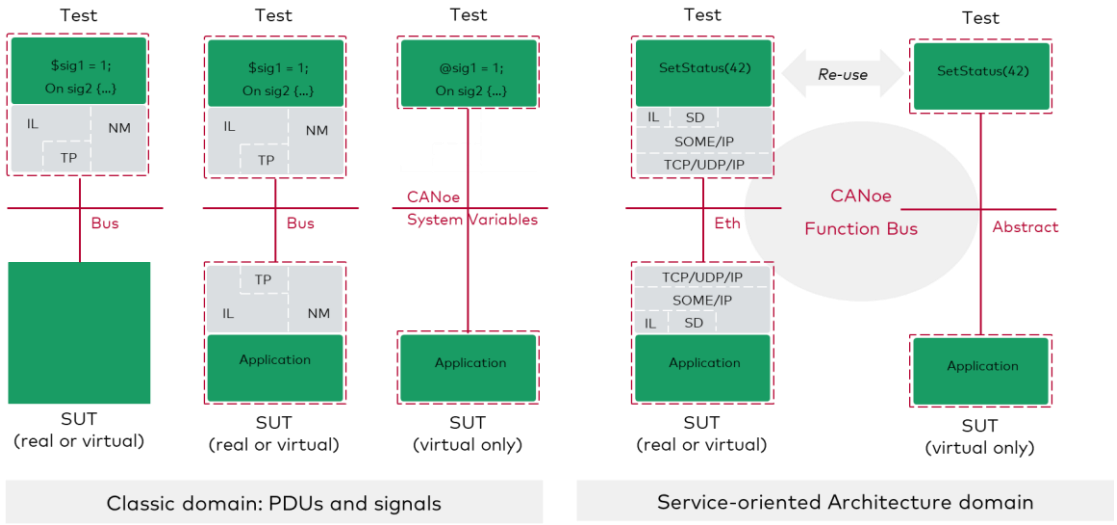
Agenda

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

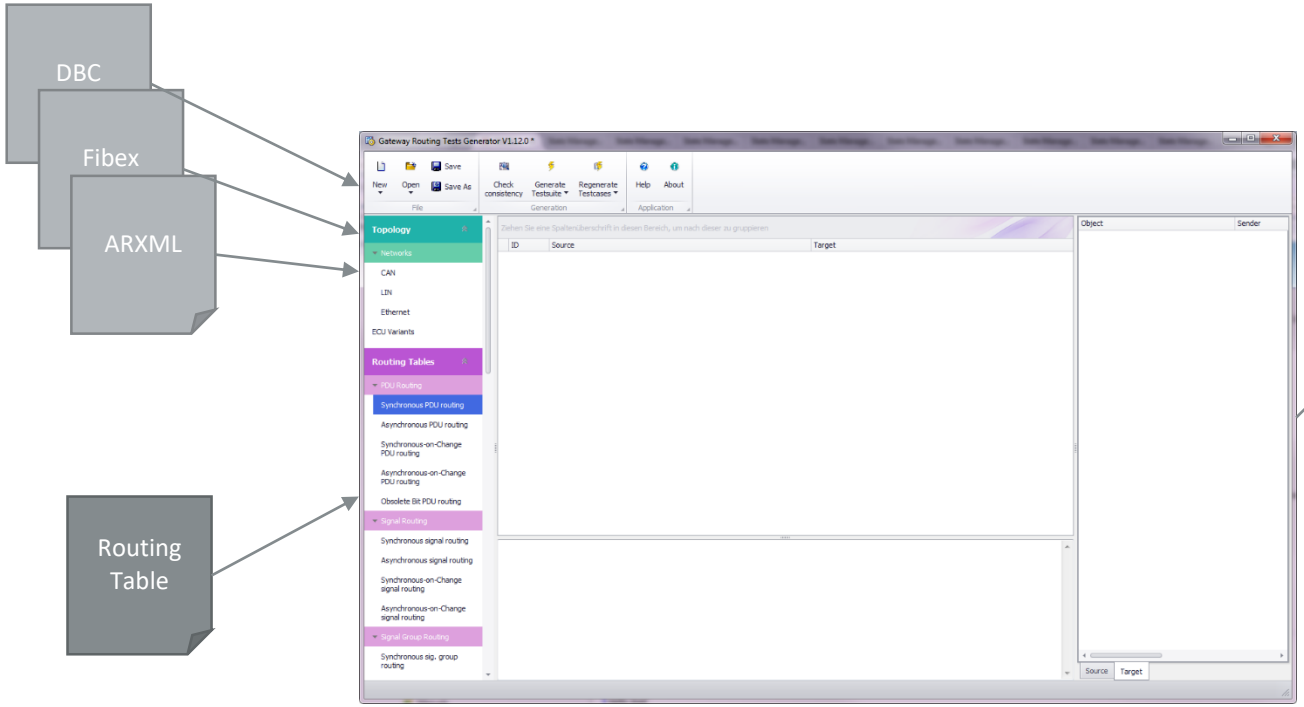
More than TC8, not only Ethernet

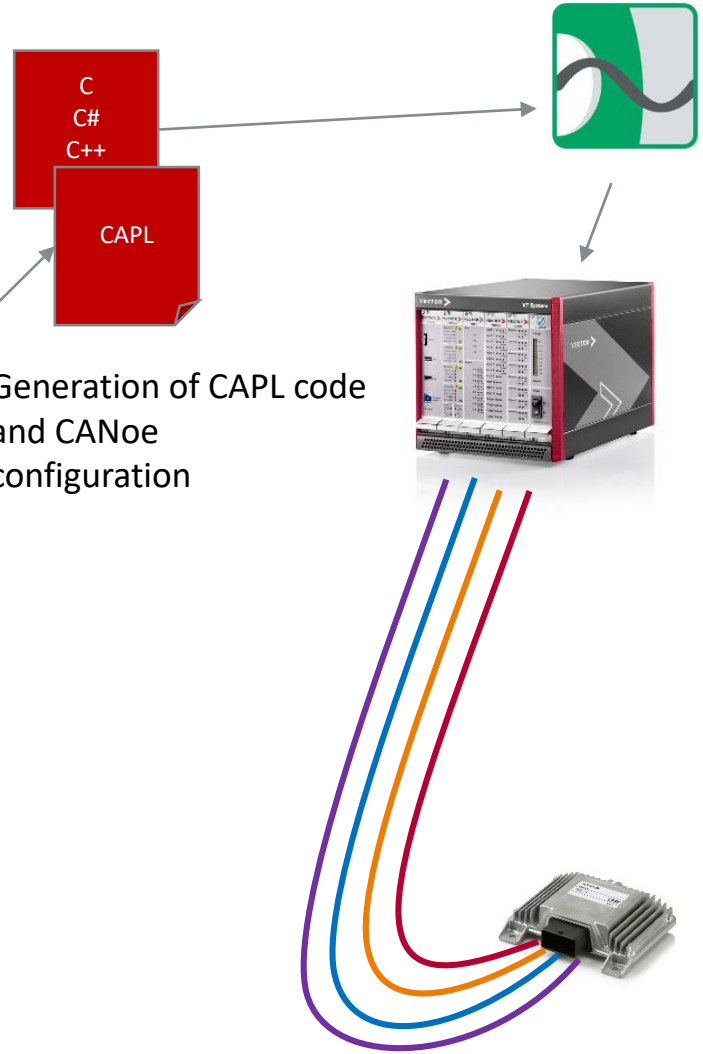
- Cars are not just another IT software
  - Testing of all production variants highly desired
  - Tests must be performed on various integration levels
    - Software component level
    - ECU level
    - Subsystem level
    - Entire vehicle network level
    - Test drive

- 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





Generation of CAPL code and CANoe configuration

Import of bus description and routing table

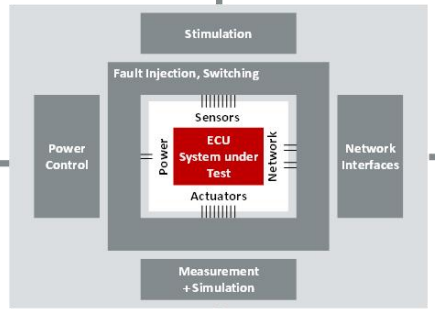
Configuration of routing (semi) automatic and manual depending on project specific implementation

HiL

Stimulation Modules
analog VT2004A
digital VT2516A

Rotation Sensor Module VT7820
Smart Charging Module
VT7970 / VT7971

Network Interface Modules
VT6104B / VT6204B
VT2710 / VT6306B



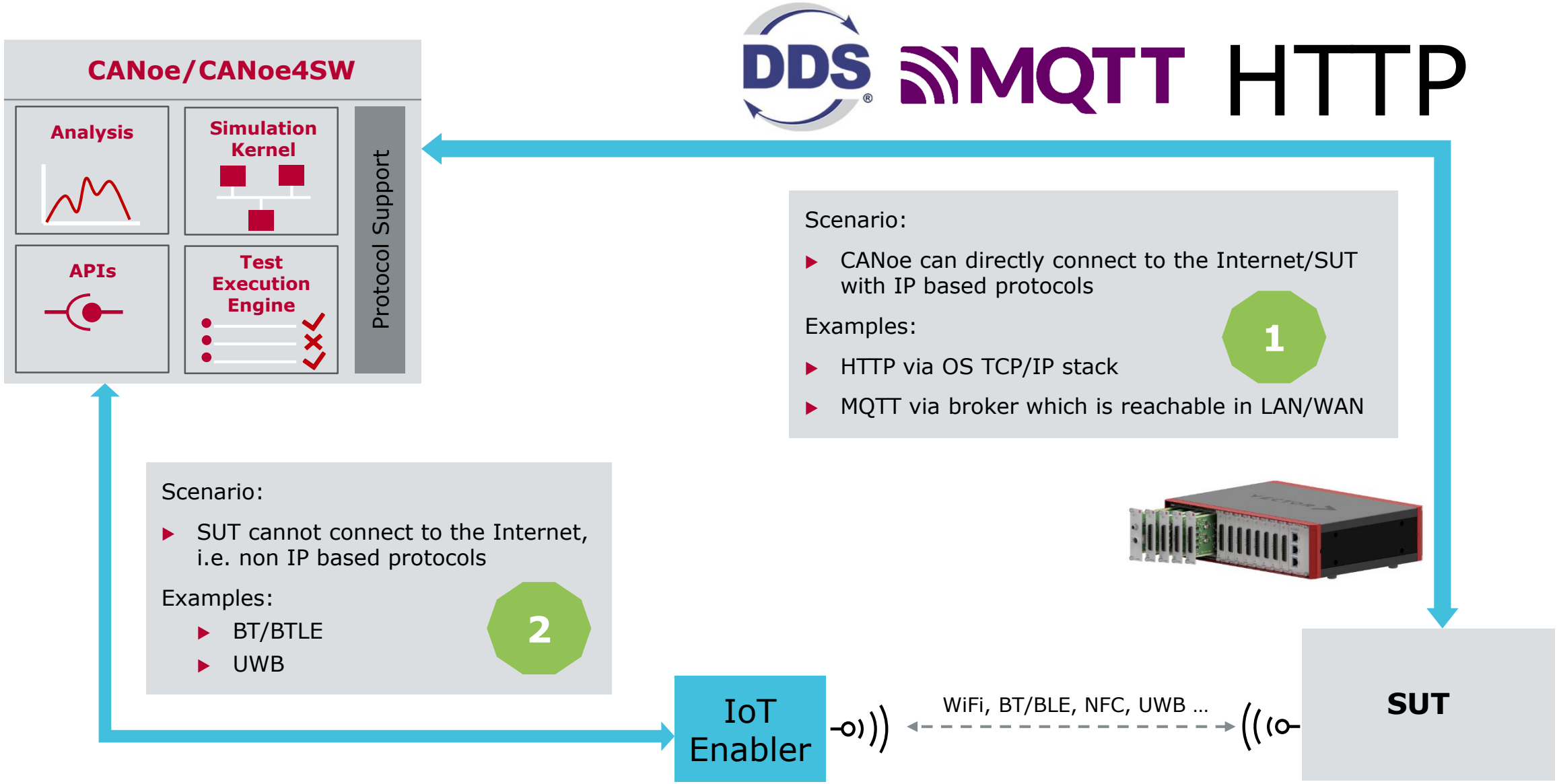
Power Modules
VT7001A (40V)
VT7101 (60V)

General Purpose Modules
current VT2808
analog VT2816A
digital VT2848
Relais VT2820
Matrix VT2832
Multi VT5838

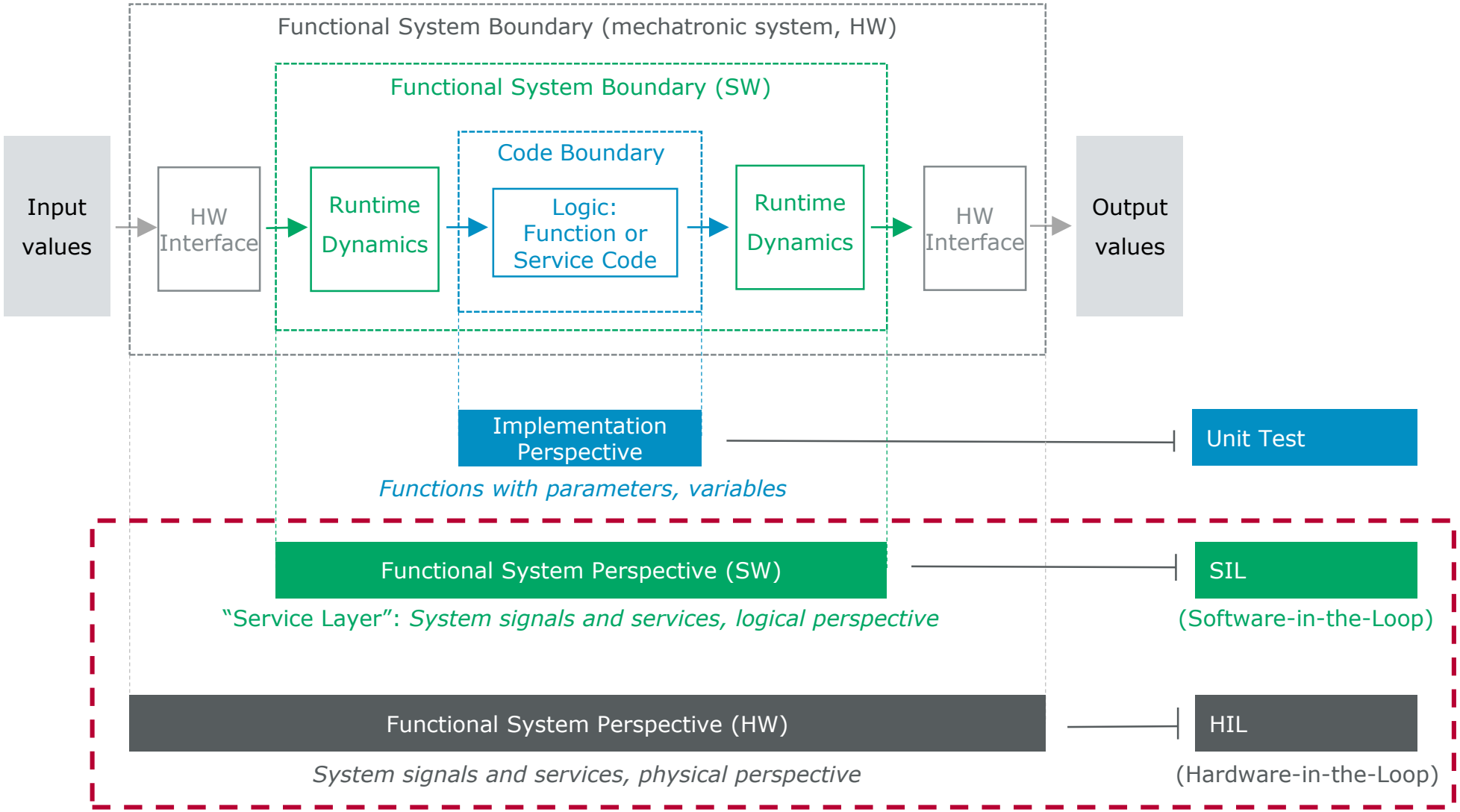
Load + Measurement Modules
VT1004A (40V)
VT1104 (60V)

Real-Time Modules
Atom VT6020
Core i7 VT6060

Connection Scenarios for SUT for connectivity

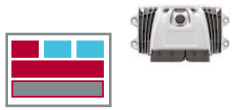
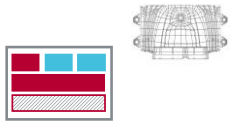
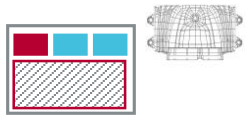


Layers of Test Interfaces



<focus of today>

Test Methods – Summary

	Unit Under Test	Authoring		Debugging		Execution
HW/SW Integration Tests (HIL)	 Real Target	Input: System Description/ DEXT	CANoe, vscode: Simulation Environment; vTESTstudio: Test Units	CANoe: Simulation Environment	IDE: Source Code	CANoe+DYNA4 (desktop); CANoe SE* (automation)
ECU Integration Tests (SIL)	 Virtual Target	Input: System Description/ DEXT	CANoe, vscode: Simulation Environment; vTESTstudio: Test Units	CANoe: Simulation Environment	IDE: Source Code	CANoe +DYNA4 (desktop); CANoe4SW+DYNA 4 SE (automation)
SWC Integration Tests (SIL)	 SWCs without BSW stack	Input: System Description/ DEXT	CANoe, vscode: Simulation Environment; vTESTstudio: Test Units	CANoe: Simulation Environment	IDE: Source Code	CANoe+DYNA4 (desktop); CANoe4SW +DYNA4 SE (automation)
SW Unit Verification	Code	PC-lint Plus: Coding Rules; VectorCAST: Unit Tests, SW/SW Integration Tests		VectorCAST: Test Case	IDE: Source Code	VectorCAST (desktop); PC-lint Plus (desktop/ automation); VectorCAST SE (automation)

* 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

Kevin.fan@vector.com

... on PC

CANoe4SW

CANoe4SW Server Edition

... on server