

Vector Test Solution From Requirements to Tests

5. Vector Congress

Agenda

> Overview Testing Portfolio

Traceability & Test Management

HIL Test Execution

Test Design

Summary & Outlook

Overview Testing Portfolio

Solution and Tools

Requirements and Test Data Management

DOORS®

Open Interface in Test Automation Editor

PREEvision.TDM
(future)

Test Design and Authoring

Solutions:

Test Automation Editor

OpenTest

C#, CAPL, C++

MATLAB Simulink®

Vector Generators:

DIVA: from CDD/ODX

CANdb++: from DBC

Custom Generators

HIL and SIL Test Execution

Test Sequencer:

CANoe as real-time execution engine

I/O access:

VT System, VN8900, IOcab, various DAQ cards

Bus access:

Various Vector Network Interfaces

Combined:

Small HIL including Network IF: VN8900

SIL:

DaVinci Component Tester

Test Evaluation and Test Management

Different Reports (e.g. with Tracability Matrix)

DOORS®

Open Interface in Test Automation Editor

PREEvision.TDM
(future)

Agenda

Overview Testing Portfolio

> **Traceability & Test Management**

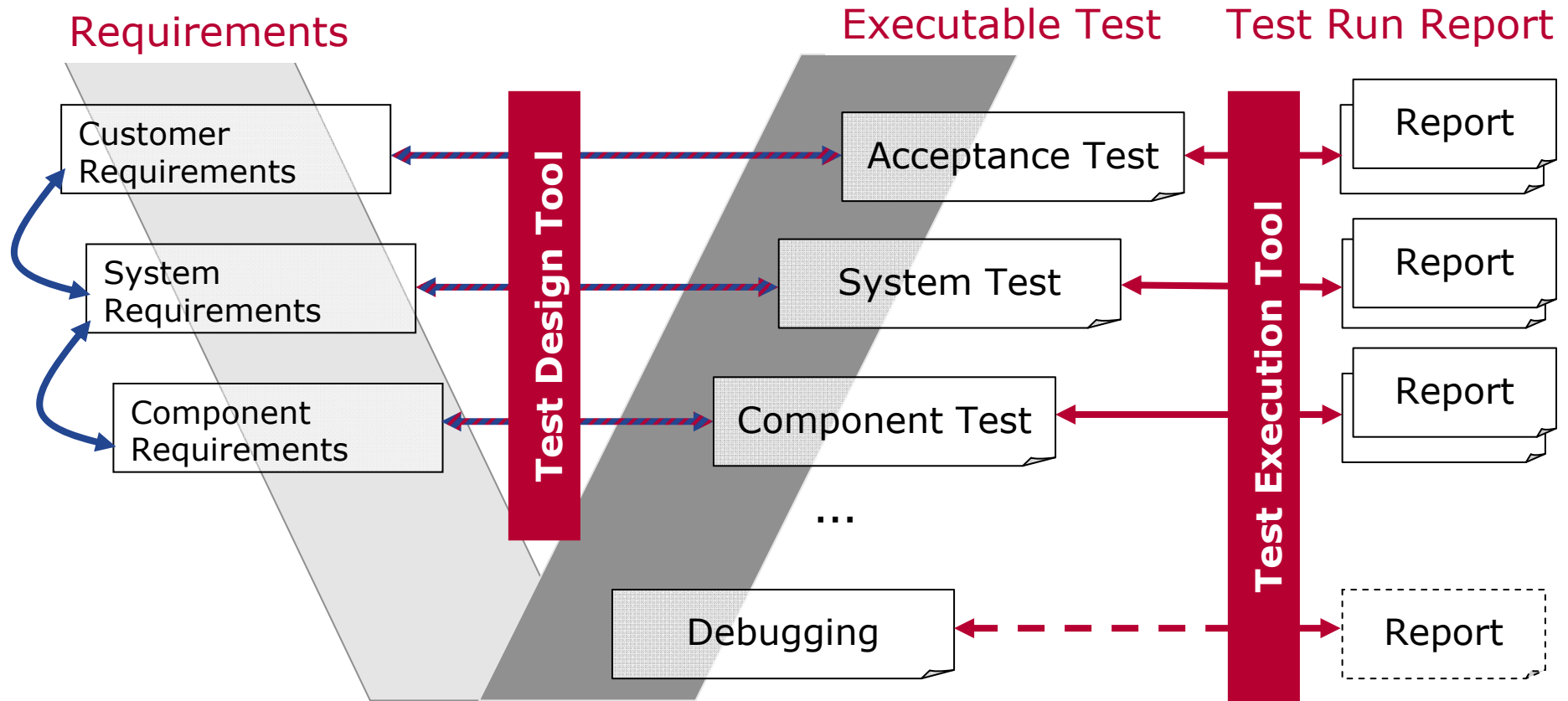
HIL Test Execution

Test Design

Summary & Outlook

Traceability & Test Management

Requirements Breakdown – Traceability on all Levels



Test Data Management & Test Management Tool

Legend:

Allocation of traceability features:



Not visualized:
System artifacts and traceability there

Traceability & Test Management

Typical Questions

Queries to check current status:

- ▶ Are tests defined for all requirements
- ▶ Which requirements are already fulfilled (tested with verdict "passed")
- ▶ Which components are finished (all associated requirements are tested "passed")

Queries to check performance of process / concept:

- ▶ Which are unstable components (recent change from "passed" to "failed")
- ▶ Does quality increase continuously (count of passed requirements increases)
- ▶ What to do on update of requirement (which tests are to be revisited)
- ▶ When will we have reached dedicated quality / integration gate

Agenda

Overview Testing Portfolio

Traceability & Test Management

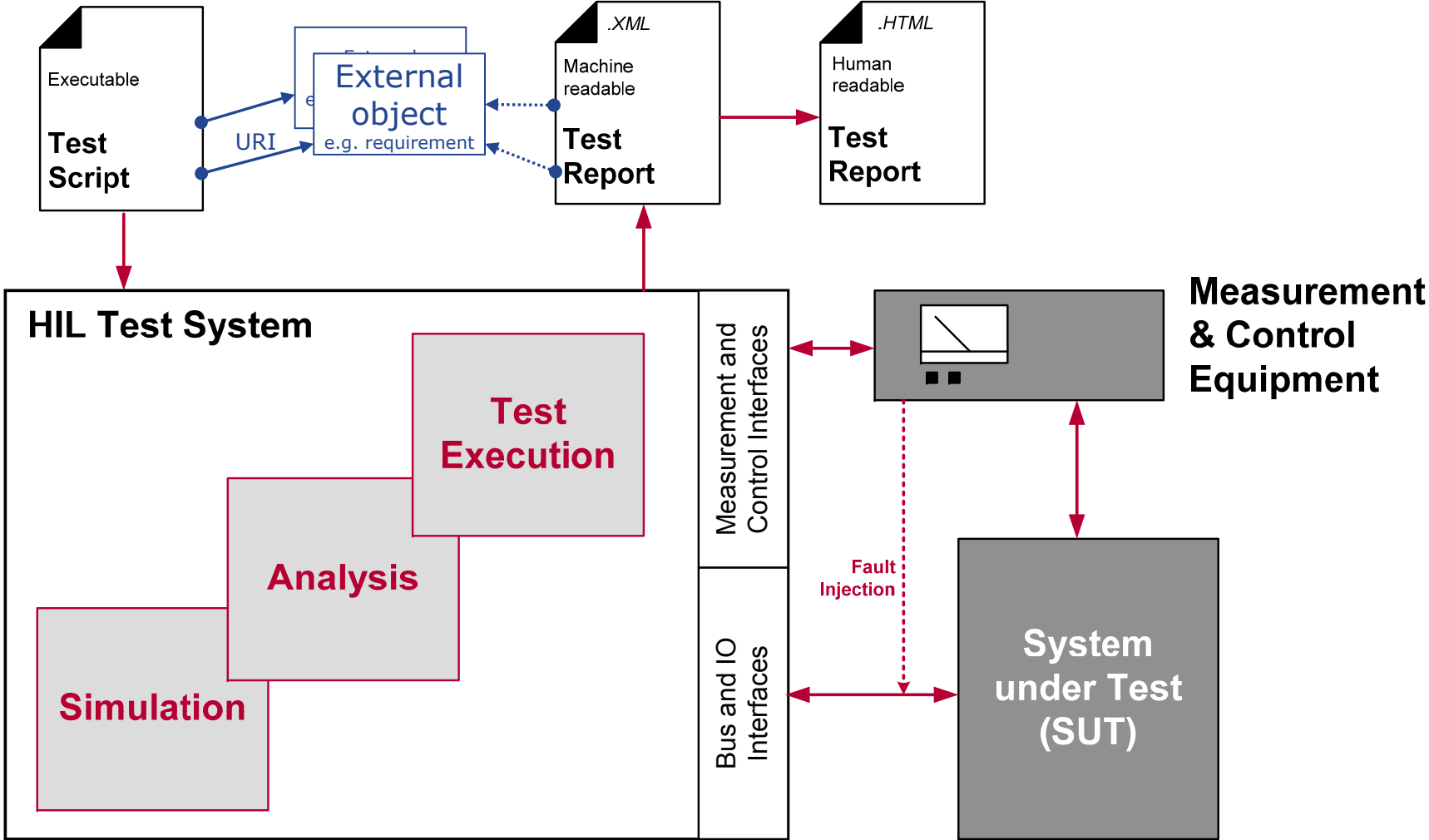
> HIL Test Execution

Test Design

Summary & Outlook

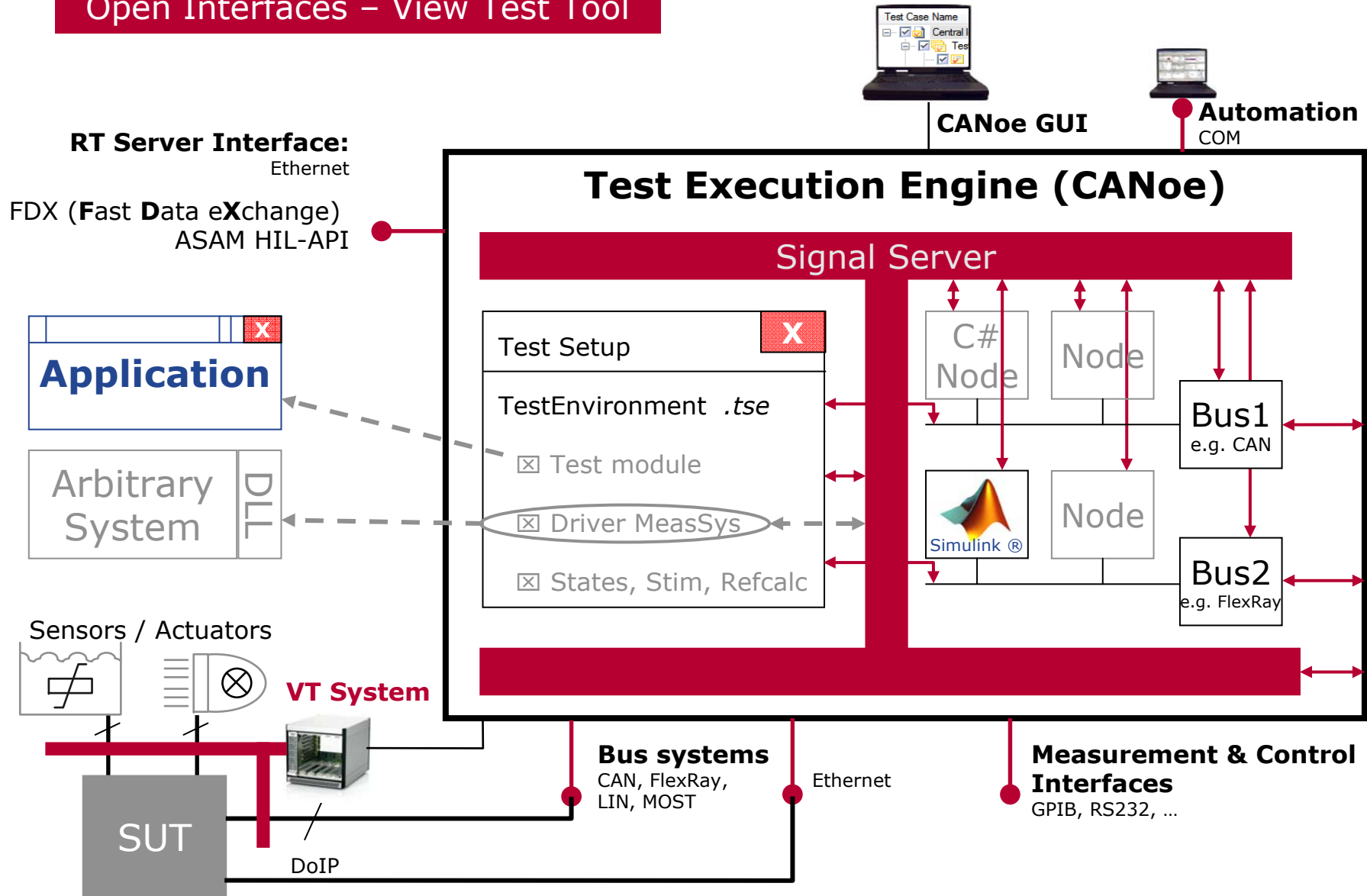
HIL Test Execution

Automated Execution of Tests



HIL Test Execution

Open Interfaces – View Test Tool



HIL Test Execution

Served Interfaces of System Under Test

Protocols

Interaction Layers

OEM specific

Network Management

AUTOSAR, OSEK, OEM specific

XCP

over CAN, IP, FlexRay

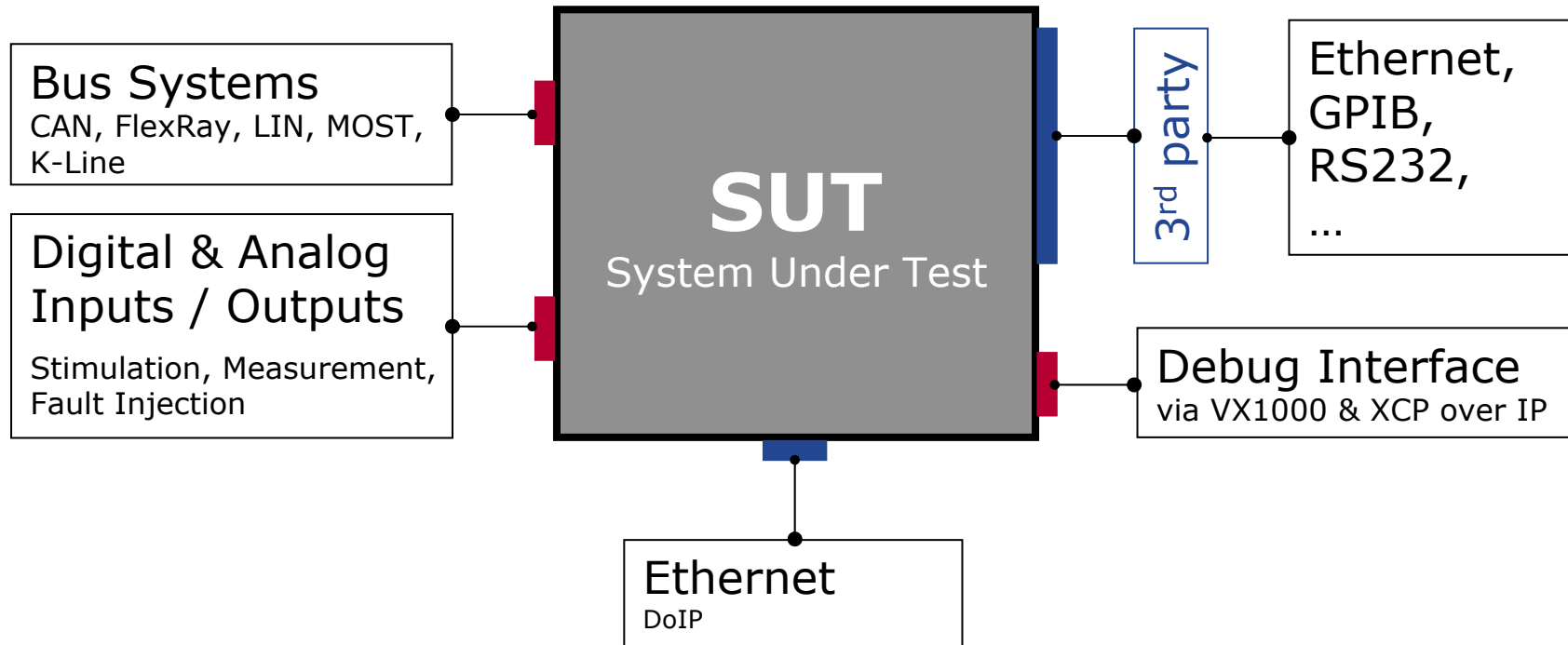
Transport Protocols

AUTOSAR, OEM specific

Diagnosis Protocols

UDS, KWP 2000

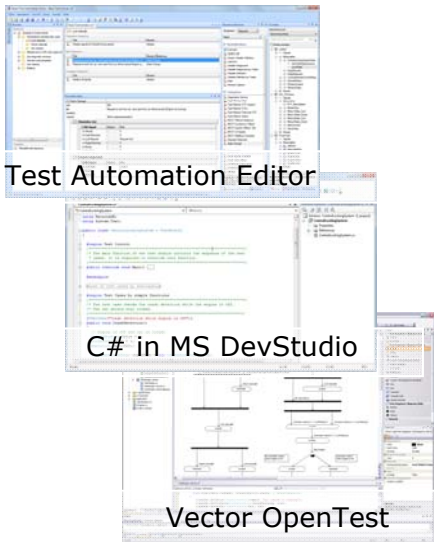
Interfaces



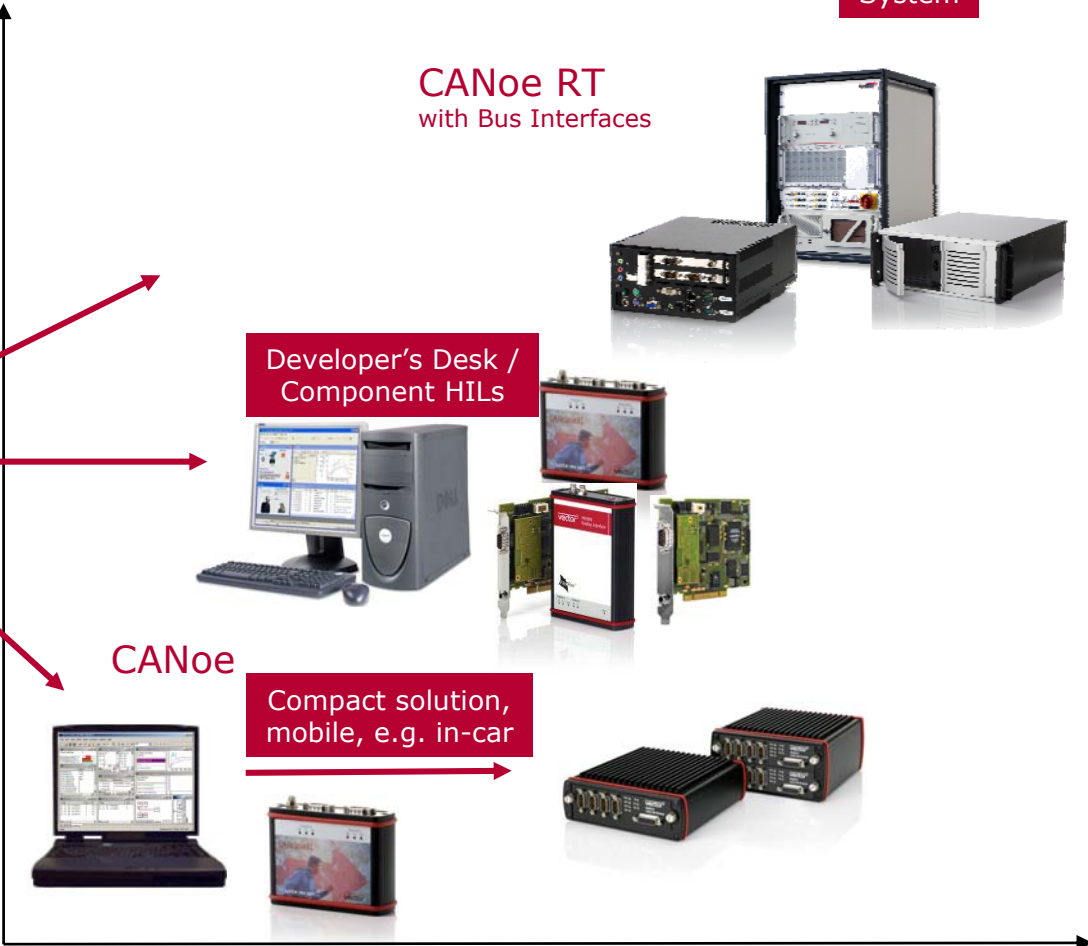
HIL Test Execution

Scalable Hardware – Same Test Design

The same
Test Design /
Implementation
can be used in all scales



Cost
of Ownership



Performance

Agenda

Overview Testing Portfolio

Traceability & Test Management

HIL Test Execution

> Test Design

Summary & Outlook

Following activities are named as “test design” in this presentation:

- ▶ Creation of test model

 - ... where test sequence can be automatically derived from

 - Vector OpenTest**

- ▶ Creation of automated test sequence

 - ... for manually implemented tests

 - Test Automation Editor; Tests in C#; Tests in CAPL**

- ▶ Test generator parameterization

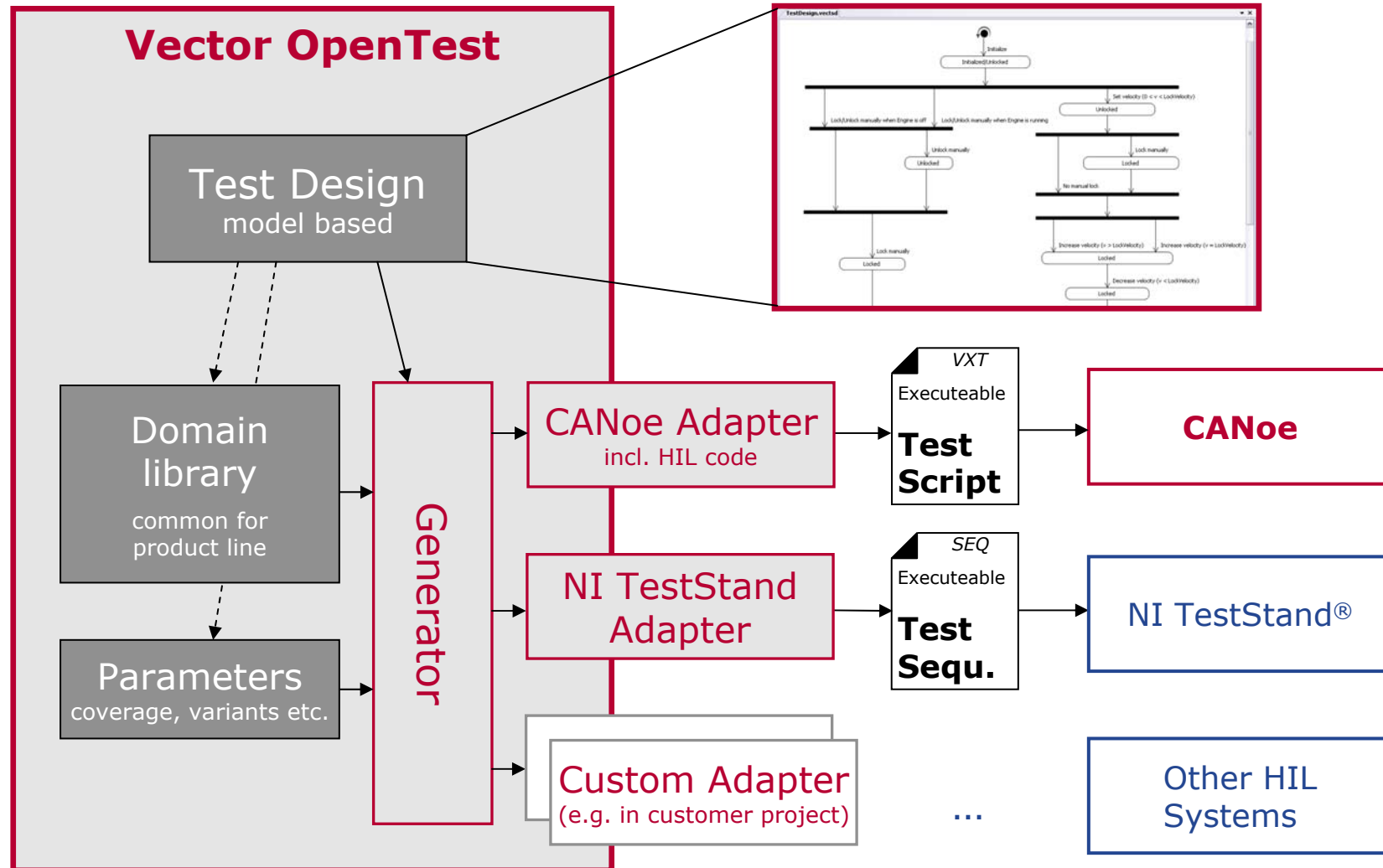
 - ... for fully generated test sequences

 - CANoe Option .DiVa**

... There are further possibilities that are not mentioned here ...

Test Design

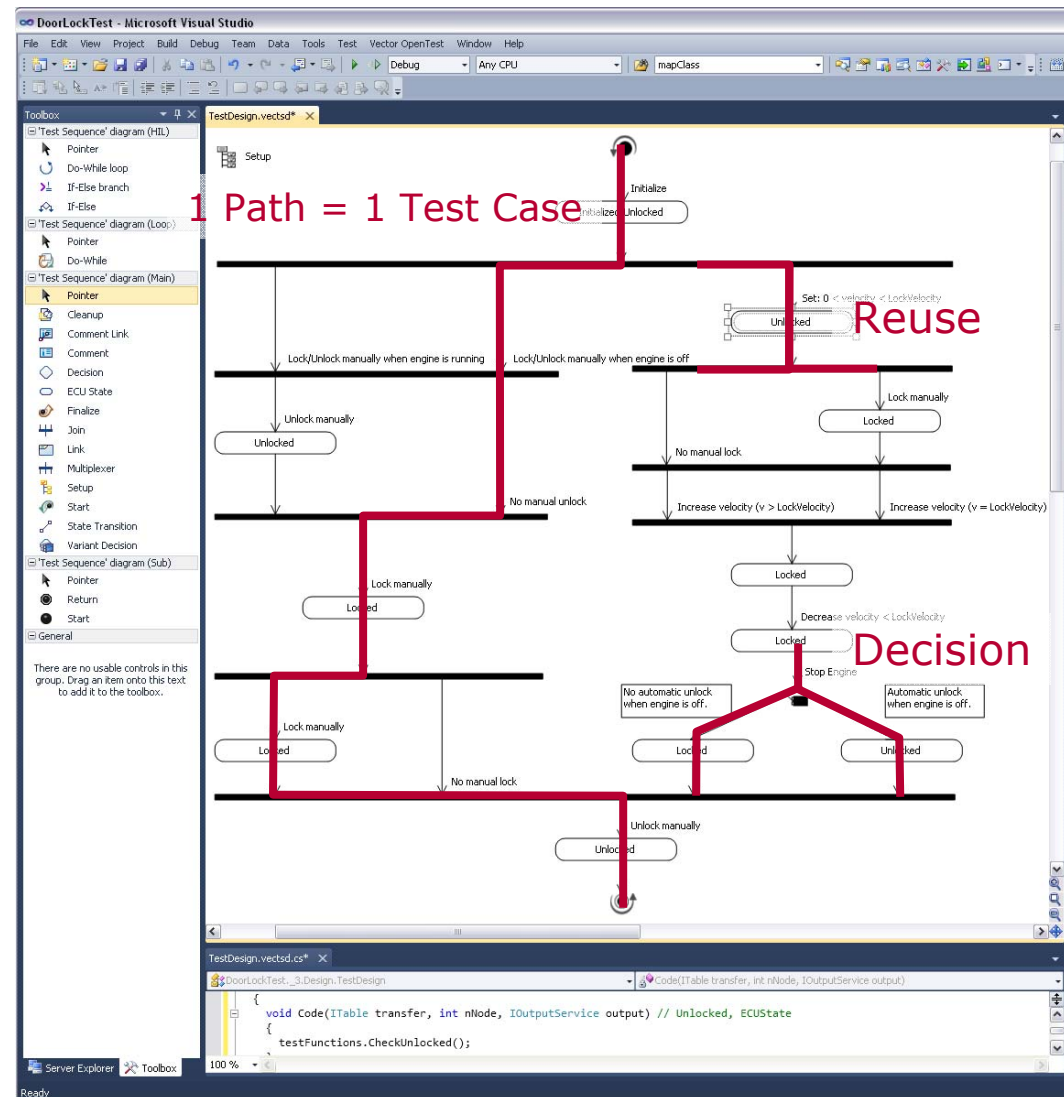
OpenTest: Design and Connection to CANoe as HIL System



Test Design

OpenTest: Model Based Test Design

- ▶ Structured, HIL system independent test design
- ▶ Graphical modeling of tests based on UML state diagrams
- ▶ Support of ECU variants with decision design elements
- ▶ Clearly arranged test designs enable reviews with customers and other stakeholders
- ▶ Easy reuse of common test sequences
- ▶ Focus of test designers is on test sequence, not on HIL implementation details



Test Design

Test Automation Editor

- ▶ Out of the box tool to specify test sequences easily by drag&drop
- ▶ Test sequence is based on prepared test functions
- ▶ User-Library test functions and test cases (.NET, CAPL) directly accessible per drag&drop
- ▶ Full access to domain symbols like bus signals, value enumerations, hardware signals
- ▶ High usability
- ▶ Tight integration to CANoe
- ▶ Integration to IBM Telelogic DOORS® to comfortably enable traceability
- ▶ Open Interface to other Requirements- and Test Management Systems

The screenshot displays the Test Automation Editor interface for a test module named 'BasicTestmodule.vxt'. The main window is divided into several panes:

- Sequence Elements:** A tree view on the right showing the test sequence structure. It includes a 'Test Primitive' section with elements like 'Await Signals Available', 'Await Value Match', 'CANstress Configuration', 'CANstress State', 'Initialize', 'Replay', 'Set', 'State Check', 'Stimulate Ramp', and 'Tester Confirmation'. A red arrow points to the 'State Change' element.
- Parameter Editor:** A central pane showing the configuration for the selected 'State Change' element. It includes fields for 'wait' (500), 'title' (Request to lock the car. Let's see if the car will b...), 'resettim' (Not variant specific), and 'variants'. A red box highlights the 'title' field, with a red arrow pointing to it from the 'Sequence Elements' pane. Below this, there are two tables: 'Stimulation (in)' and 'Expected'. The 'Stimulation (in)' table has columns for 'CAN Signal', 'relation', and 'value'. The 'Expected' table has columns for 'CAN Signal', 'relation', and 'value'. A red box highlights the 'Request lock' value in the 'Stimulation (in)' table, with a red arrow pointing to it from the 'Symbols' pane. Another red arrow points to the 'Request lock' value in the 'Stimulation (in)' table, with a red arrow pointing to it from the 'Symbols' pane.
- Symbols:** A pane on the right showing a list of symbols. It is filtered by 'Network Symbols'. A search bar is present. Below the search bar, there is a 'Display details' section showing a message tree. A red arrow points to the 'Request lock' message in the tree. Below the message tree, there are fields for 'Message Name', 'Start bit', and 'Comment'. At the bottom, there are buttons for 'Symb...', 'MOST Sym...', and 'Librai...'.

Test Design

Programming Language, Example .NET in C#

```
public class CentralLockingSystem : TestModule {
    public override void Main()
    {
        Title = "Central Locking System";           // set test case title (for test report)

        TestGroupBegin("TCX 01.23", "Check lock state"); // dynamically group test cases
        for (int i = 0; i < 8; i++)
        {
            TestCaseLockCar();           // execute test case. Arbitrary control flow can be applied
        }
        TestGroupEnd();
    }

    [TestCase("lock car, verify if locked")]
    public void TestCaseLockCar()
    {
        LockRequest.Value = 0;           // unlock car
        Execution.Wait(2000);           // wait 2s

        if(LockState.Value == 1) // derive verdict
            Report.TestStepPass("Locked expected, lock detected. OK");
        else
            Report.TestStepFail("Locked expected, lock not detected. NOK");
    }
}
```

Test Design

Overview of Test Design Possibilities

Vector OpenTest

- ▶ Model based approach with graphical test design and generation of test implementation
- ▶ Independent of HIL system: CANoe and TestStand® are integrated. Easy adaptation to new HIL systems
- ▶ Medium programming skills required
- ▶ Focus on larger user groups and test organizations
- ▶ Integrated in MS Visual Studio

Test Automation Editor

- ▶ Out-of-the box solution to get a test sequence quickly and fast
- ▶ Very easy to use - domain level, no programming skills required
- ▶ Extendable by custom test functions implemented in .NET and/or CAPL
- ▶ Tight integration to CANoe, all available test means can be used, available symbols usage per drag&drop
- ▶ Open Interface to integrate to Test Management / Requirements Management Tools

Programming in .NET (C#)

- ▶ Full Flexibility of a programming language, state-of-the-art IDE
- ▶ Intended for programming experts
- ▶ Tight integration to CANoe, many test means are available for direct usage, others through CAPL call from .NET
- ▶ Many domain symbols (signals, system variables, ...) accessible per IntelliSense

Agenda

Overview Testing Portfolio

Traceability & Test Management

HIL Test Execution

Test Design

> Summary & Outlook

Summary & Outlook

- ▶ Vector's testing solution covers many aspects from
 - ▶ **traceability** and **test management**
 - ▶ **test design**, where several design and implementation technologies / languages are available
 - ▶ **test automation** inclusive hardware integration, **test execution** and **test evaluation**
 - ▶ Vector's testing solution is open and thus can be extended
- ▶ Vector provides different approaches how to create/design tests
 - ▶ from **manual** test implementation,
 - ▶ over **model based** test development,
 - ▶ up to automatic test **generation**

Summary & Outlook

- ▶ Vector supports customers in solving their test challenges
- ▶ Vector is prepared
 - ▶ to provide custom software- and hardware solutions, and
 - ▶ to help customers to develop their own solution within customer projects

Summary & Outlook

- ▶ Details of HIL test execution platforms
 - ▶ Refer Presentation of **Dr. Stefan Bodamer**
- ▶ Details of hardware I/O and VT System
 - ▶ Refer Presentation of **Dr. Stefan Krauß**

Thank you for your attention.

For detailed information about Vector
and our products please have a look at:

www.vector.com

Author:

Siegfried Beeh

Vector Informatik GmbH

Ingersheimer Str. 24

70499 Stuttgart