

Testing of AUTOSAR Application Software Components

Vector Congress 2008, Stuttgart, 2008-10-07/08

> Introduction

Main Goals

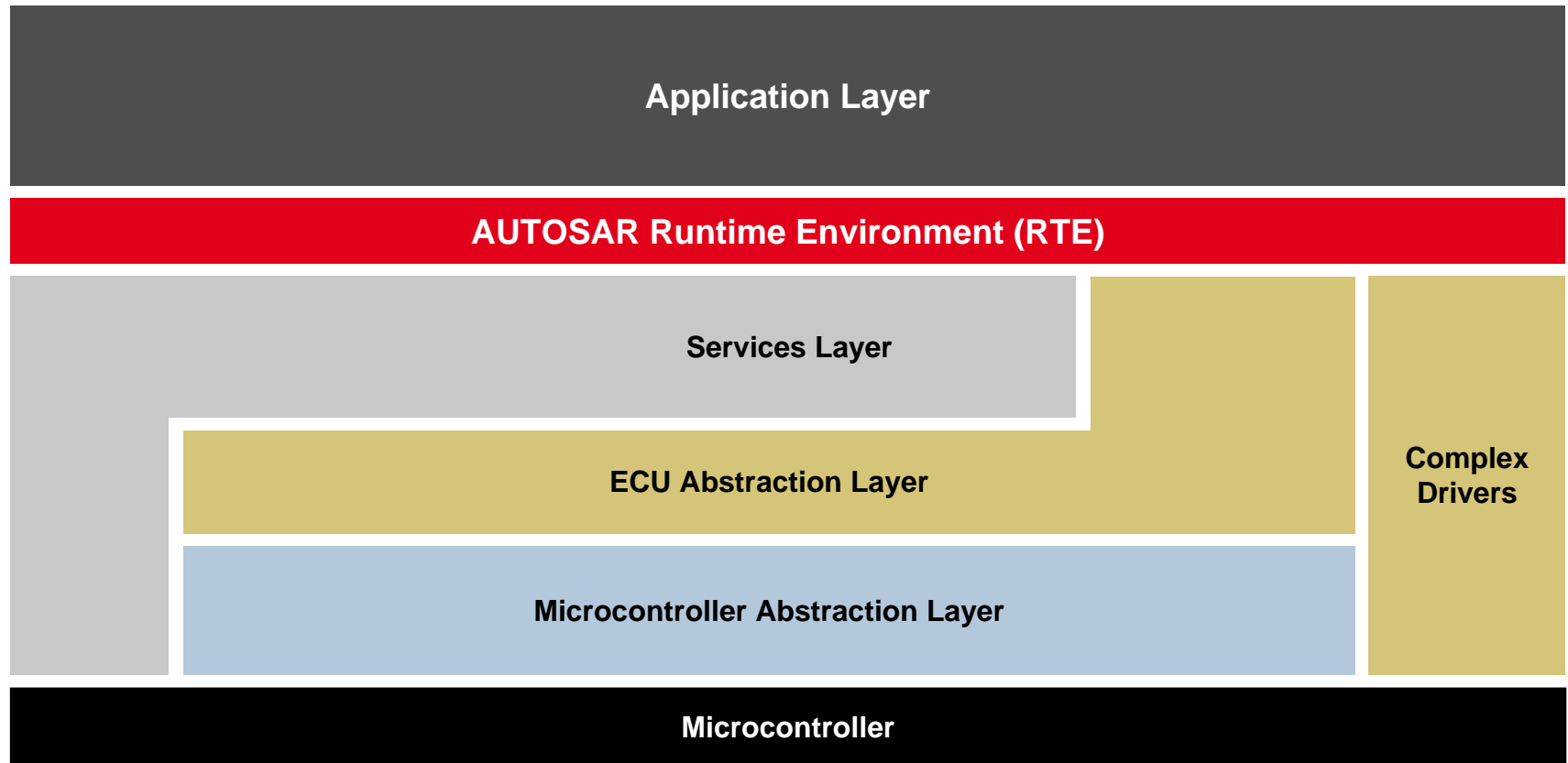
Test Environment

Conclusion

- ❑ The term “software-component” in its traditional meaning at Vector refers to **basic software**
 - ❑ e.g. CAN driver
 - ❑ e.g. network management module
- ❑ AUTOSAR coined the term with a different meaning
 - ❑ Application software
 - ❑ Running mainly above the AUTOSAR Run-Time Environment (RTE)
- ❑ This presentation refers to the AUTOSAR meaning of the term “software-component”

Introduction

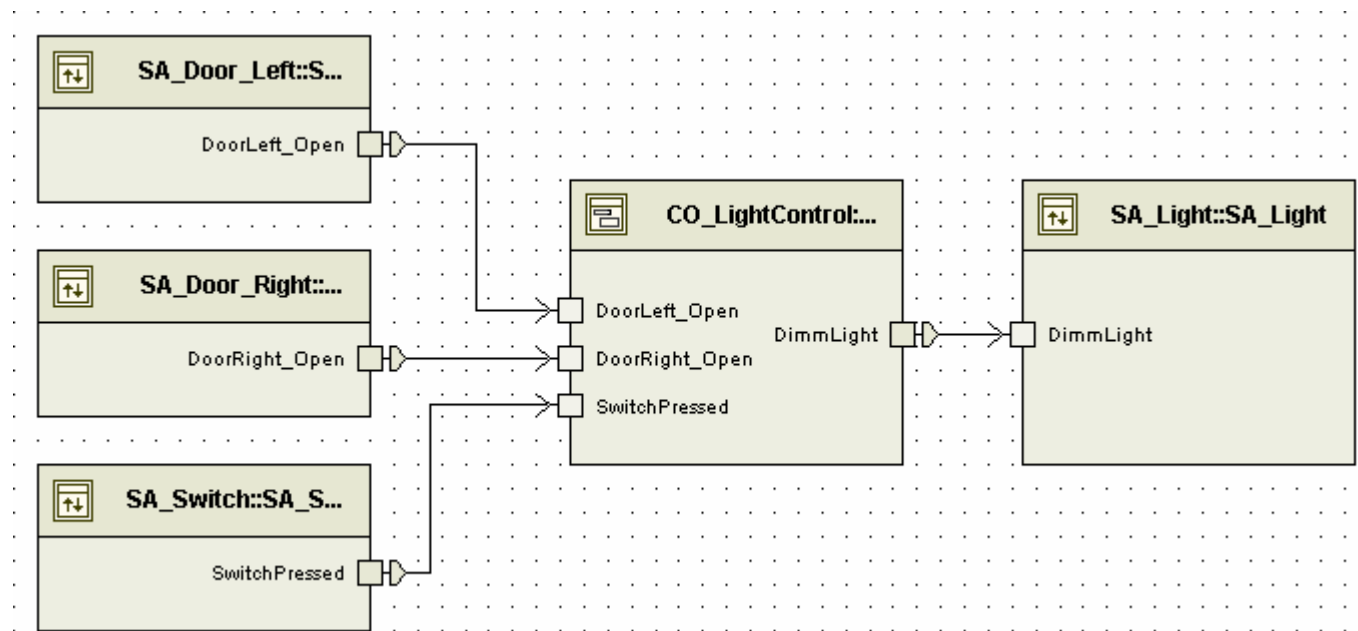
Layered Software Architecture



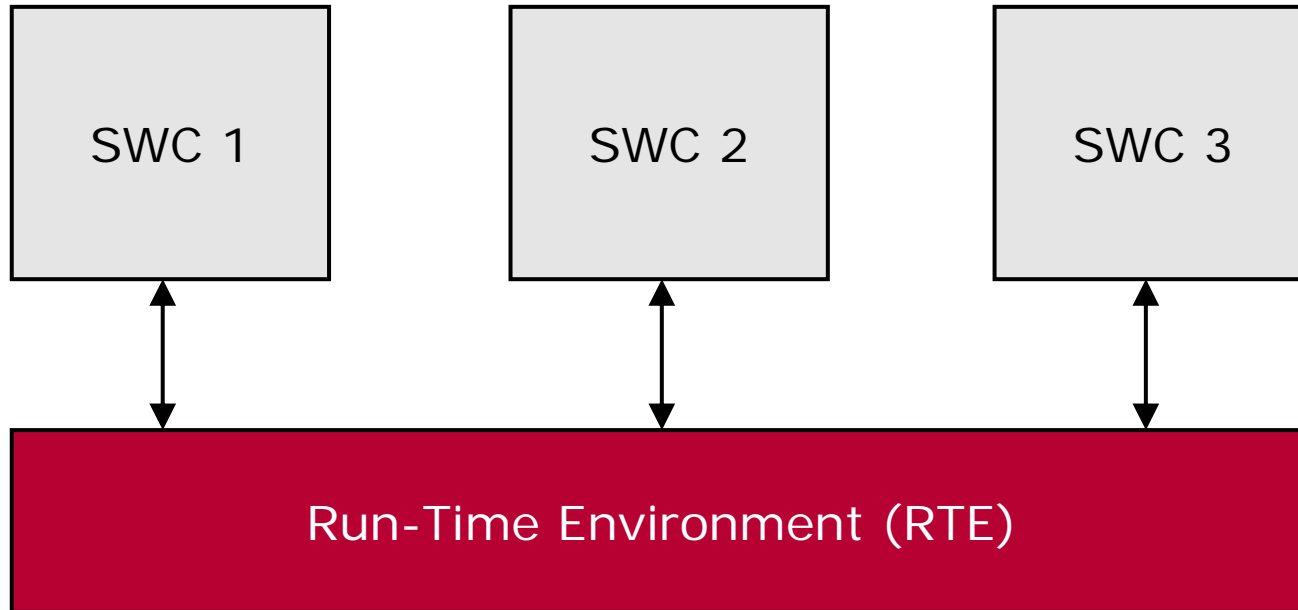
Source: AUTOSAR

Introduction

AUTOSAR Software-Components



- ❑ Application layer consists of communicating self-contained AUTOSAR software-components (SWC)
- ❑ AUTOSAR formally defines structure and constraints for SWC
- ❑ Formal definition can be serialized to XML



- ❑ RTE provides the “middleware” for all SWC on one ECU
- ❑ SWC are developed against a specific RTE API derived from the formal description that differs for each individual SWC

- ❑ According to AUTOSAR, software-components (SWC) have the following characteristics
 - ❑ Formal specification of self-contained units of software
 - ❑ “Runnable entities” represent actual execution units
 - ❑ SWC need an AUTOSAR run-time environment (RTE) to be able to execute and integrate with AUTOSAR basic software
- ❑ Wait a minute ...
 - ❑ SWC are self-contained, but need an RTE that requires comprehensive configuration to be executable?
 - ❑ How do you test (and debug) the functionality of a SWC if it cannot be executed in the absence of an RTE?
 - ❑ During development
 - ❑ Before release towards integration

Introduction

> Main Goals

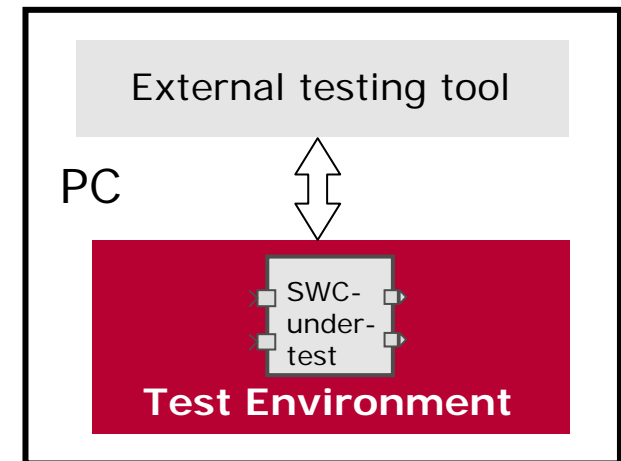
Test Environment

Conclusion

Main Goals

for the Test Environment

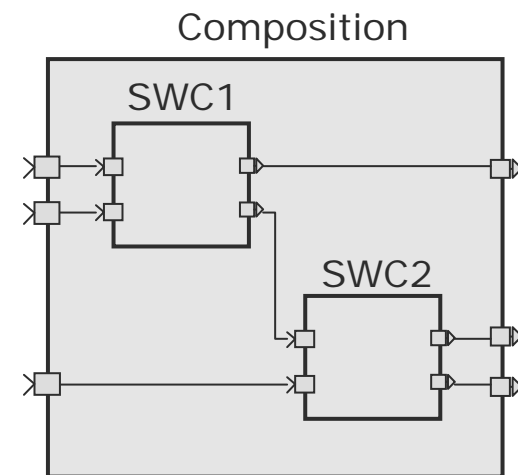
- ❑ The test environment shall require no or few RTE and basic software related configuration
 - ❑ Short turn-around times
 - ❑ Focus on application functionality rather than configuration details
- ❑ The test environment shall execute on a standard PC
 - ❑ Simple
 - ❑ Ubiquitously available
- ❑ Open
 - ❑ Test environment shall not provide yet another proprietary test frontend
 - ❑ Test environment shall be able to integrate with existing test frontends



Main Goals

for the Test Environment

- ❑ Test functional behavior
 - ❑ Execution timing is not considered
 - ❑ Test the actual implementation code
 - ❑ Robustness against specific scenarios
- ❑ Execution mode
 - ❑ Programmed
 - ❑ Interactive
- ❑ Granularity
 - ❑ Single software-components
 - ❑ Compositions of software-components



- ❑ SWC Author
 - ❑ Creates formal description of software-component
 - ❑ Creates implementation of software-component
- ❑ Test Case Author
 - ❑ Creates test cases
 - ❑ Creates environment simulation where applicable
- ❑ Tester
 - ❑ Executes test cases

- ❑ All roles can be combined arbitrarily

- ❑ Test of functional behavior during SWC development by stimulating inputs and observing outputs
 - ❑ Execution order of runnable entities
 - ❑ Supply values (test vectors) on required ports and observe the values produced by the provided ports
 - ❑ Intentionally inject faults (e.g. timeout, queue overflow, etc.) and check the robustness of the SWC against these faults
 - ❑ Check whether runnable entities call only RTE APIs for which they have got clearance
 - ❑ Check whether all available RTE APIs have actually been called
 - ❑ Check the actual order in which RTE APIs have been called
- ❑ Single-step debugging of SWC implementation with PC-based debugger

Introduction

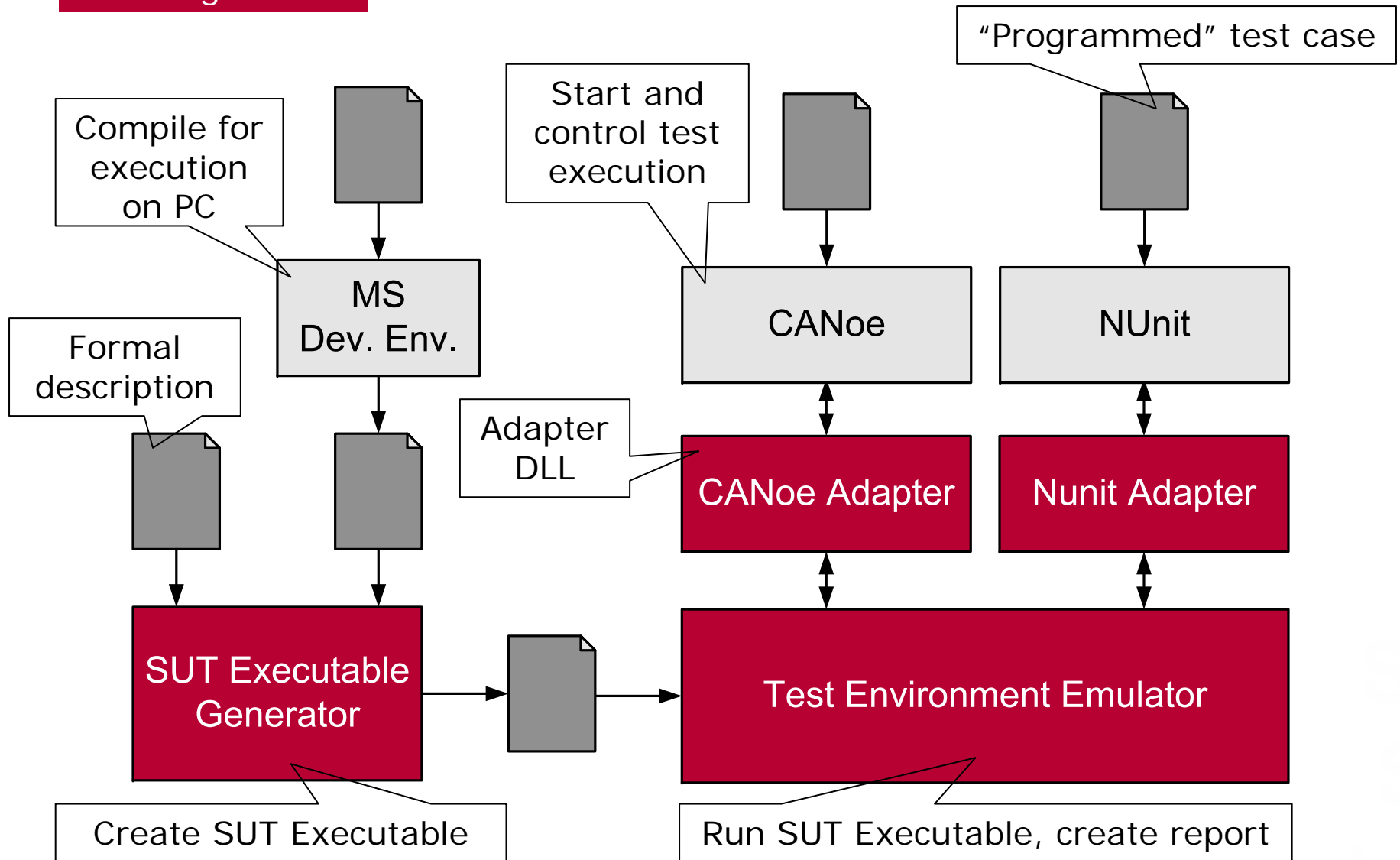
Main Goals

> Test Environment

Conclusion

Test Environment

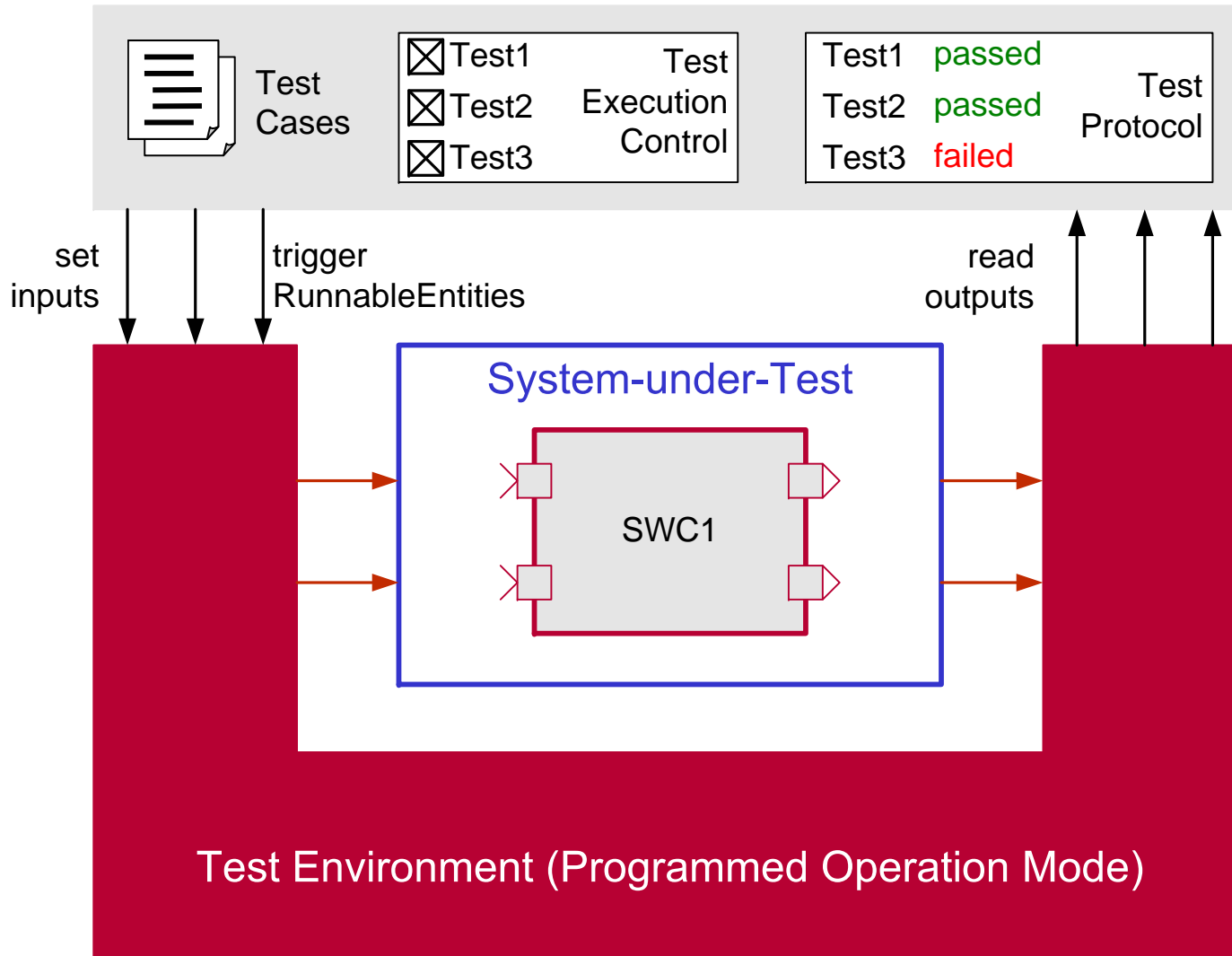
Building Blocks



- ❑ Detailed definition of test case
- ❑ Use high-level (programming) language for definition of test cases
- ❑ Pattern: stimulate – execute – check
- ❑ “Microscopic” view
 - ❑ Applicable whenever the very details matter
 - ❑ Regression testing
- ❑ “Microscopic” view is like using a debugger to track down issues
 - ❑ In fact, single-step debugging of SUT is possible
- ❑ Test tool: NUnit, (CANoe TFS)

Test Environment

Programmed Operating Mode



Test Environment

Programmed Unit Test

[TestFixture]

```
public class InputValuePermutation {
```

```
    [TestFixtureSetUp]
```

```
    public void Init() { ... }
```

Initialize

```
    [Test]
```

```
    public void Test_DoorRight_Open() {
```

```
        AP_DoorContacts.RPort.DoorRight_Open.DEP.DoorOpen.Value = true;
```

```
        AP_DoorContacts.RPort.DoorRight_Open.DEP.DoorOpen.RetVal = RTEErrors.RTE_E_OK;
```

```
        AP_DoorContacts.RPort.DoorLeft_Open.DEP.DoorOpen.Value = false;
```

```
        AP_DoorContacts.RPort.DoorLeft_Open.DEP.DoorOpen.RetVal = RTEErrors.RTE_E_OK;
```

```
        AP_DoorContacts.Runnable.AP_DoorContacts.Trigger();
```

Execute

```
        Assert.That(true == AP_DoorContacts.PPort.AnyDoor_Open.DEP.DoorOpen.Value);
```

```
    }
```

Check test result

Set stimulation
of input values

Test Environment

Test Execution Screenshot (NUnit)

The screenshot displays the NUnit application window titled "AP_DoorContact_Test.nunit - NUnit". The interface includes a menu bar (File, View, Project, Test, Tools, Help) and a tree view on the left showing the test hierarchy:

- Tests
 - D:\TACO\Test\NUnit\Demo\NUnit\AP
 - AP_DoorContacts_Tests
 - InputValuePermutation
 - Test_BothDoors_Closed
 - Test_BothDoors_Open
 - Test_DoorLeft_Open
 - Test_DoorRight_Open

The main area contains a "Run" button, a "Stop" button, and a progress bar. Below the progress bar, the summary statistics are displayed:

Test Cases: 4 Tests Run: 4 Failures: 0 Ignored: 0 Skipped:

The console window shows the following log entries:

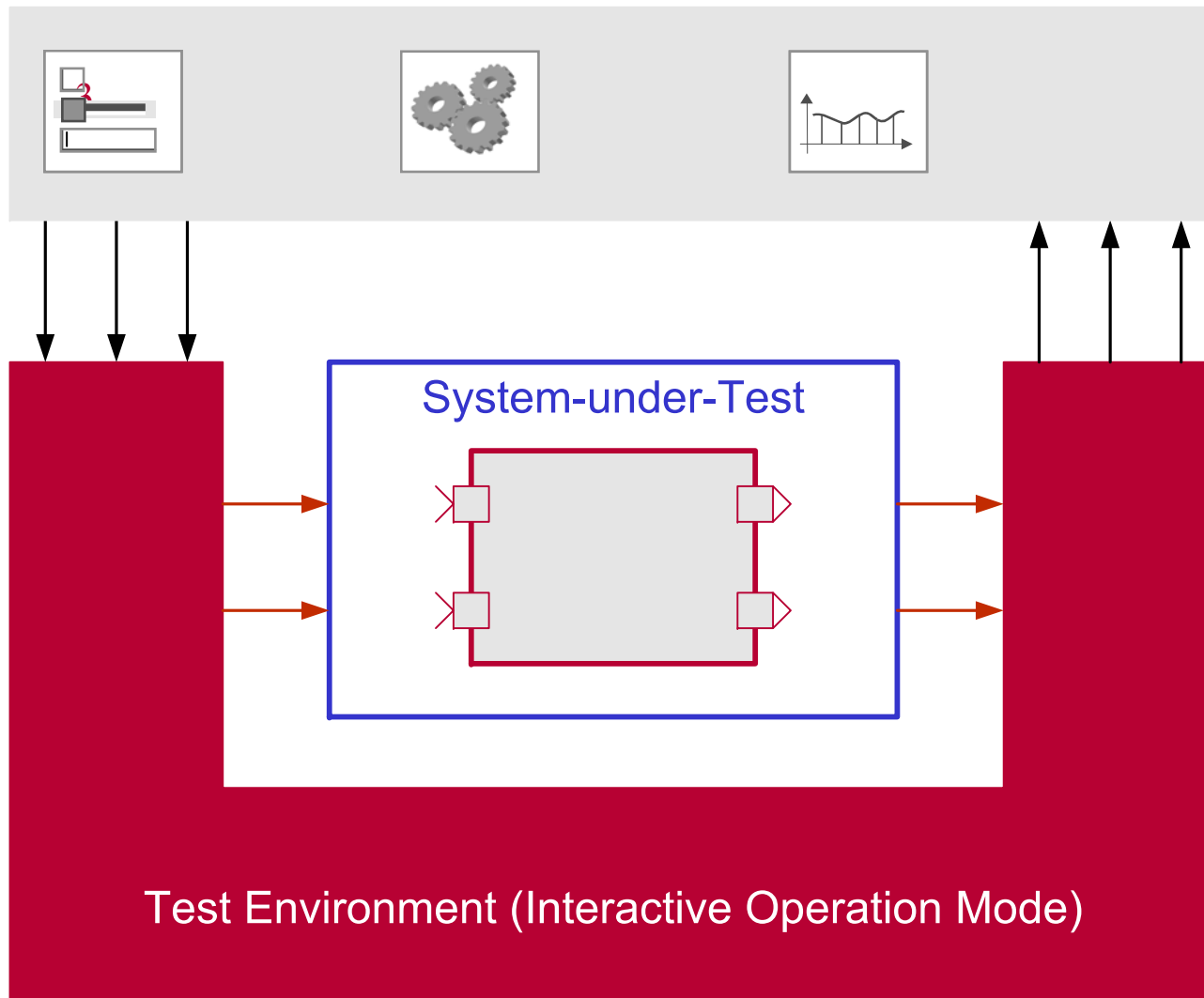
```
15:55:34.0865658 - LoadSUT, path=D:\TACO\Test\NUnit\Demo\NUnit\AP_DoorCo
15:55:35.7271068 - Start
15:55:35.8052278 - SetReturnValue_Read DoorRight_Open.DoorOpen: RTE_E_OK
15:55:35.8364762 - Write DoorRight_Open.DoorOpen: False
15:55:35.8364762 - SetReturnValue_Read DoorLeft_Open.DoorOpen: RTE_E_OK
15:55:35.8364762 - Write DoorLeft_Open.DoorOpen: False
15:55:35.8364762 - >>> Trigger AP_DoorContacts.AP_DoorContacts
15:55:35.8364762 - Read DoorLeft_Open.DoorOpen: False
15:55:35.8364762 - Read DoorRight_Open.DoorOpen: False
15:55:35.8521004 - Write AnyDoor_Open.DoorOpen: False
15:55:35.8521004 - <<< Trigger AP_DoorContacts.AP_DoorContacts
15:55:35.8521004 - Read AnyDoor_Open.DoorOpen: False
15:55:35.8677246 - SetReturnValue_Read DoorRight_Open.DoorOpen: RTE_E_OK
15:55:35.8677246 - Write DoorRight_Open.DoorOpen: True
15:55:35.8677246 - SetReturnValue_Read DoorLeft_Open.DoorOpen: RTE_E_OK
15:55:35.8677246 - Write DoorLeft_Open.DoorOpen: True
15:55:35.8677246 - >>> Trigger AP_DoorContacts.AP_DoorContacts
15:55:35.8677246 - Read DoorLeft_Open.DoorOpen: True
15:55:35.8677246 - Read DoorRight_Open.DoorOpen: True
15:55:35.8677246 - Write AnyDoor_Open.DoorOpen: True
15:55:35.8677246 - <<< Trigger AP_DoorContacts.AP_DoorContacts
15:55:35.8677246 - Read AnyDoor_Open.DoorOpen: True
```

At the bottom, the status bar shows the current path and test count: "D:\TACO\Test\NUnit\Demo\NUnit\AP_DoorContacts Test Cases : 4".

- ❑ Study behavior against complex interactions
- ❑ “Macroscopic” view
 - ❑ Applicable where large amounts of data create a context
 - ❑ Applicable where testers need to toy around with a SWC in order to properly understand or verify its behavior
 - ❑ Might be taken as an input for programmed regression tests
 - ❑ Applicable where continuous changes over a period in time occur
- ❑ Emulator is synchronized with the simulation time of the test tool
- ❑ Test tool: CANoe

Test Environment

Interactive Operating Mode



Sti

Test Environment

Test execution screenshot (CANoe)

The screenshot displays the Vector CANoe software interface during a test execution. The main window is titled "Vector CANoe" and features a menu bar (File, View, Start, Mode, Configuration, Window, Help) and a toolbar. The interface is divided into several sections:

- gBlinkerOutput:** A section with a "Name" field and a scale from 0 to 8. A checkbox for "Signal_CmdEL" is checked.
- gEnvironment:** A section with a "Name" field.
- gRequest:** A section with a "Name" field.
- Log Window:** A window titled "x" showing a list of messages with columns for "Source" and "Message".
- Vehicle Model:** A 3D model of a car, viewed from the top, with a black interior.

The log window contains the following messages:

Source	Message
15:55:34.0865658	- LoadSUT, path=D:\TACO\Test\Unit\Demo\Unit\VAP_DoorContact
15:55:35.7271068	- Start
15:55:35.8052278	- SetReturnValue_Read DoorRight_Open.DoorOpen: RTE_E_OK
15:55:35.8364762	- Write DoorRight_Open.DoorOpen: False
15:55:35.8364762	- SetReturnValue_Read DoorLeft_Open.DoorOpen: RTE_E_OK
15:55:35.8364762	- Write DoorLeft_Open.DoorOpen: False
15:55:35.8364762	- >>> Trigger AP_DoorContacts.AP_DoorContacts
15:55:35.8364762	- Read DoorLeft_Open.DoorOpen: False

The status bar at the bottom shows "Ready" and "Desktop1".

Agenda

Introduction

Main Goals

Test Environment

> Conclusion

- ❑ AUTOSAR SWC represent reusable, self-contained pieces of the application software.
- ❑ Testing of SWC within an integrated AUTOSAR environment requires high configuration effort
- ❑ Solution: test environment
 - ❑ Zero configuration
 - ❑ Runs on standard PC
- ❑ Open/customizable test tool interface
 - ❑ CANoe
 - ❑ NUnit
- ❑ Operation modes allow for different test approaches
 - ❑ “microscopic” vs. “macroscopic”

- ❑ The test environment will be made available as a stand-alone product
- ❑ Development in cooperation with pilot customer
- ❑ Release in Q1/2009

Thank you for your attention.

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

www.vector-informatik.com

Author: Dr. Uwe Honekamp

Vector Informatik GmbH

Ingersheimer Str. 24

70499 Stuttgart