Reuse of Hardware Independent Test Sequences across MiL-, SiL- and HiL-Test Scenarios

Testing Expo 2008 Stuttgart

Berner & Mattner Systemtechnik GmbH
Contents

• Test methods in the automotive industry
• Problems / challenges
• Solution: signal abstraction
• Model Integration
• Hardware Integration
• Test integration
• Integration into the model based development
• Discussion
Test methods in the automotive industry (1)

MiL (Model in the Loop)

• Test object: model
• Input signals are simulated
• Output signal values will be saved / logged and can be compared to the expected values
• Automatic test execution through:
  – The development environment used for modeling
  – External tools using the appropriate interface of the modeling environment (e.g.: automation interface of MATLAB/Simulink)
Test methods in the automotive industry (2)

SiL (Software in the Loop)

• Test object: generated code
• Environment is simulated
• The inputs and outputs of the test object are connected to the test system
• The generated code is executed on a PC or on an evaluation board
• Automatic test execution through:
  – The used development environment (e.g.: MATLAB/Simulink with Realtime Workshop)
  – Interfaces to external tools
Test methods in the automotive industry (3)

HiL (Hardware in the Loop)
- Test object: real ECU
- Environment simulation through environment models (e.g.: MATLAB/Simulink)
- Inputs and Outputs are connected to the HiL-Simulator
- Stimuli is generated by the HiL-Simulator
- Comparison of the ECU output values to the expected values
- Automatic test execution through the control software of the HiL-Simulator
Test methods in the automotive industry - Summary
Challenges / Problems

- Test cases of the early development phases can not be used in later development phases (MiL test cases can not be used for SiL- or HiL-Tests)
- Test cases for the ECU can be created only when the ECU is available
- High occupancy of the HiL-Simulators for the creation of the tests
- Test cases have to be adapted or recreated if the HW is replaced or modified
- Reuse of models in the later development phases
- Verification of the ECU against the specification model
Solution: Signal abstraction

• The signal pool contains all signals of the system
• Every signal has
  – a name
  – a type
  – a length
  – a signal-ID
• Signals are generated on the basis of a configuration
• All components of the system communicate via the signal pool
Model Integration

- Models can access the signal pool via ports
- The mapping, port to signal can be configured
- Adapters can register themselves for changes on the signal pool
- Adapters notify the model about the changes on the signal pool
- Models have no information about:
  - The used HW
  - Other communication partners
  - The test platform
Linking of different model types
HW Integration (e.g.: CAN)

- CAN signals are mapped to ports
- The connection, port to signal pool signal can be configured
- Adapter registers himself for changes on the signal pool
- Adapter sends the appropriate CAN message automatically on changes
- Adapter “disassembles” the incoming CAN messages and sets the signal values on the signal pool accordingly
Test integration

- Test cases communicate via the signal pool
- Signals
  - WiperPoti
  - WiperLever
  - WiperStatus
- Test case can be reused through parametrization
- Parameters
  - INTERVAL
  - TIME

```java
interval_wiping() {
    // set wiping interval
    WiperPoti.sendValue(INTERVAL);
    // set wiperlevel
    WiperLever.sendValue(1);
    // wait for wiper to start
    ASSERT(WiperStatus.wait(true, 1000));
    // wait for wiper to stop
    ASSERT(WiperStatus.wait(false, 2000));
    // wait for 2nd interval
    ASSERT(WiperStatus.wait(true, TIME));
    // wait for wiper to stop
    ASSERT(WiperStatus.wait(false, 2000));
    // switch off wiper
    WiperLever.sendValue(0);
}
```
Variant diversity using the classification tree method

- Definition of the parameters (eventually derived from the test cases)
- Definition of the possible parameter values
- Configuration of the variants
- Generation of the variants
- Definition of the expected results
- Used for parameterization of the test cases
Integration into the model based development (1)

Car manufacturer…
• creates specification models
• develops test cases
• tests models “stand-alone” or in combination with
  – other models
  – generated code
  – ECUs
• provides models and test cases for its suppliers
Integration into the model based development (2)

Supplier…

- makes models more detailed
- generates code
- develops test cases
- tests code “stand-alone” or in combination with
  - models
  - other generated code
  - ECUs
- builds the ECU
- tests the ECU “stand-alone” or in combination with
  - models
  - generated code
  - other ECUs
- delivers the ECU and test cases to the car manufacturer
Integration into the model based development (3)

Car manufacturer…

- tests the ECU alone or in combination with
  - generated code
  - specification models
  - other ECUs
Summary

- Test cases are independent of
  - the used environment
  - the used HW
- Test cases can be parameterized
- Test cases can be created already in the early development phases and reused in later development phases
- Derivation of the test cases from the models used
- Models can be used for the verification of the ECUs