Embedded Success

dSPACE

Combining Two Worlds: Precise Real-Time-Based and Convenient PC-Based Testing

Dr.-Ing. Rainer Rasche

dSPACE GmbH · Rathenaustrasse 26 · 33102 Paderborn

automotive testing expo · June 22nd, 2010

Agenda

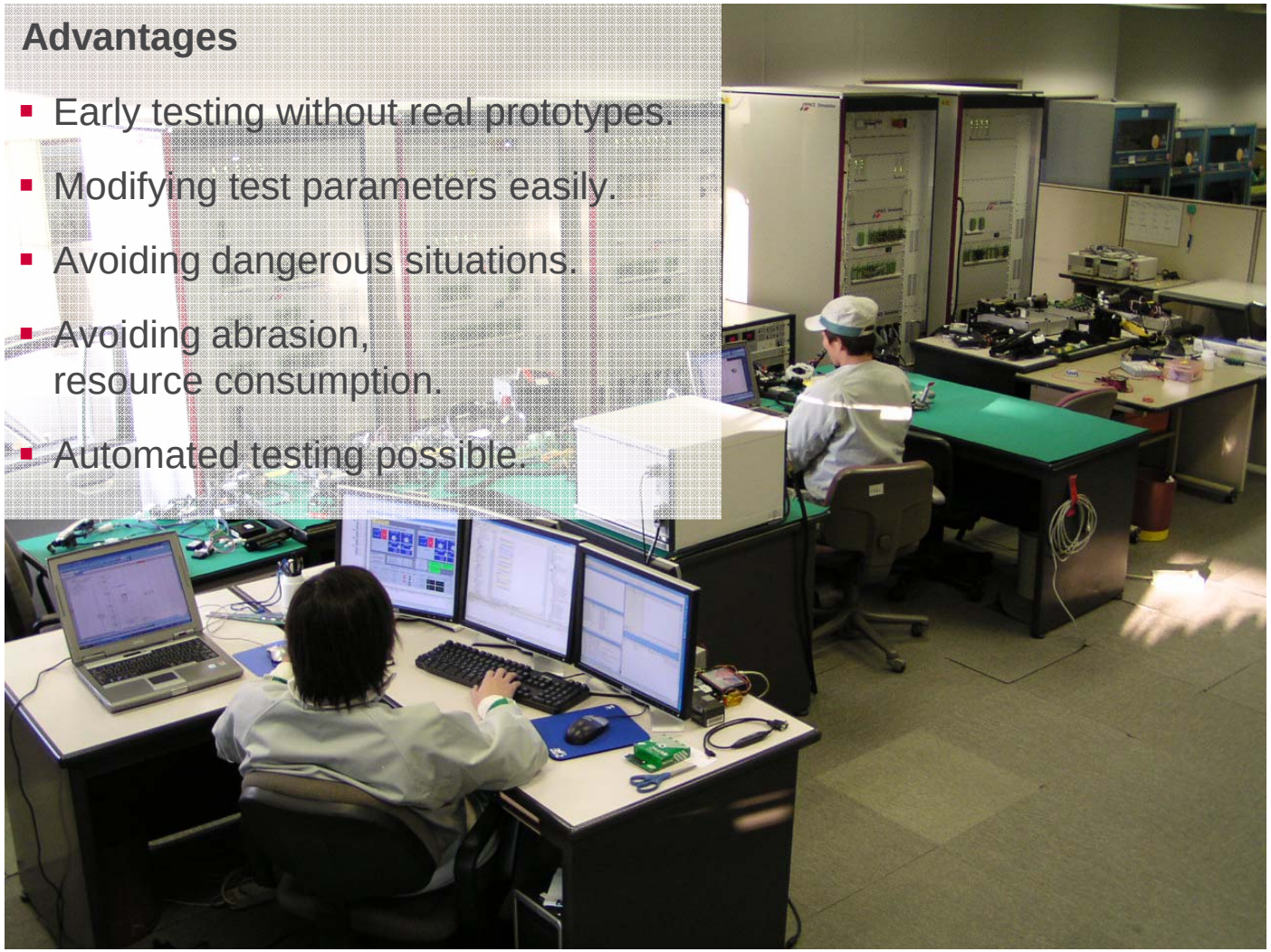
- **Testautomation**
  - Hardware-in-the-Loop (HIL) Simulation
  - Automated PC-based ECU Testing
- **Real-Time Testing (RTT)**
  - Python scripting
  - Basics on Executing RTT Sequences
- **Examples**
- **Summary**

Hardware-in-the-Loop (HIL) Simulation

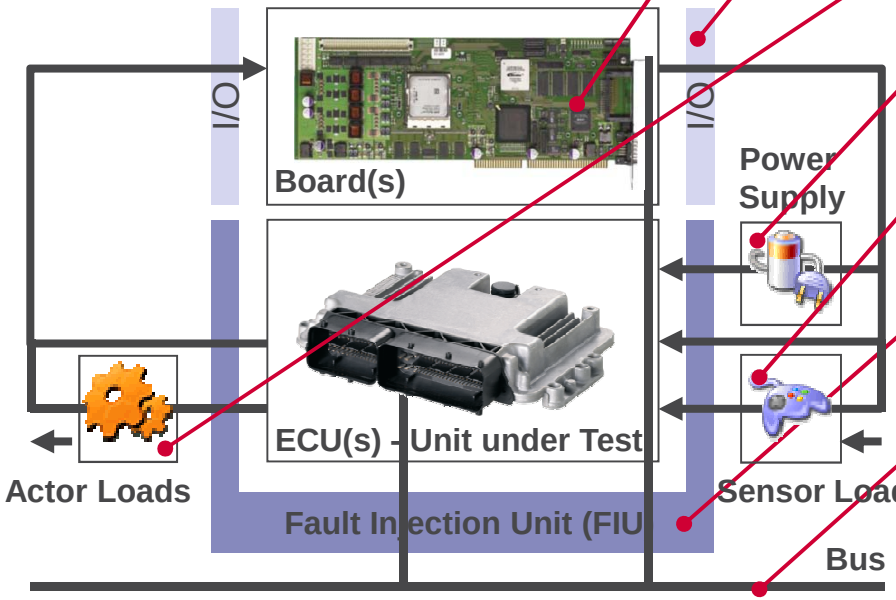
dSPACE

Advantages

- Early testing without real prototypes.
- Modifying test parameters easily.
- Avoiding dangerous situations.
- Avoiding abrasion, resource consumption.
- Automated testing possible.



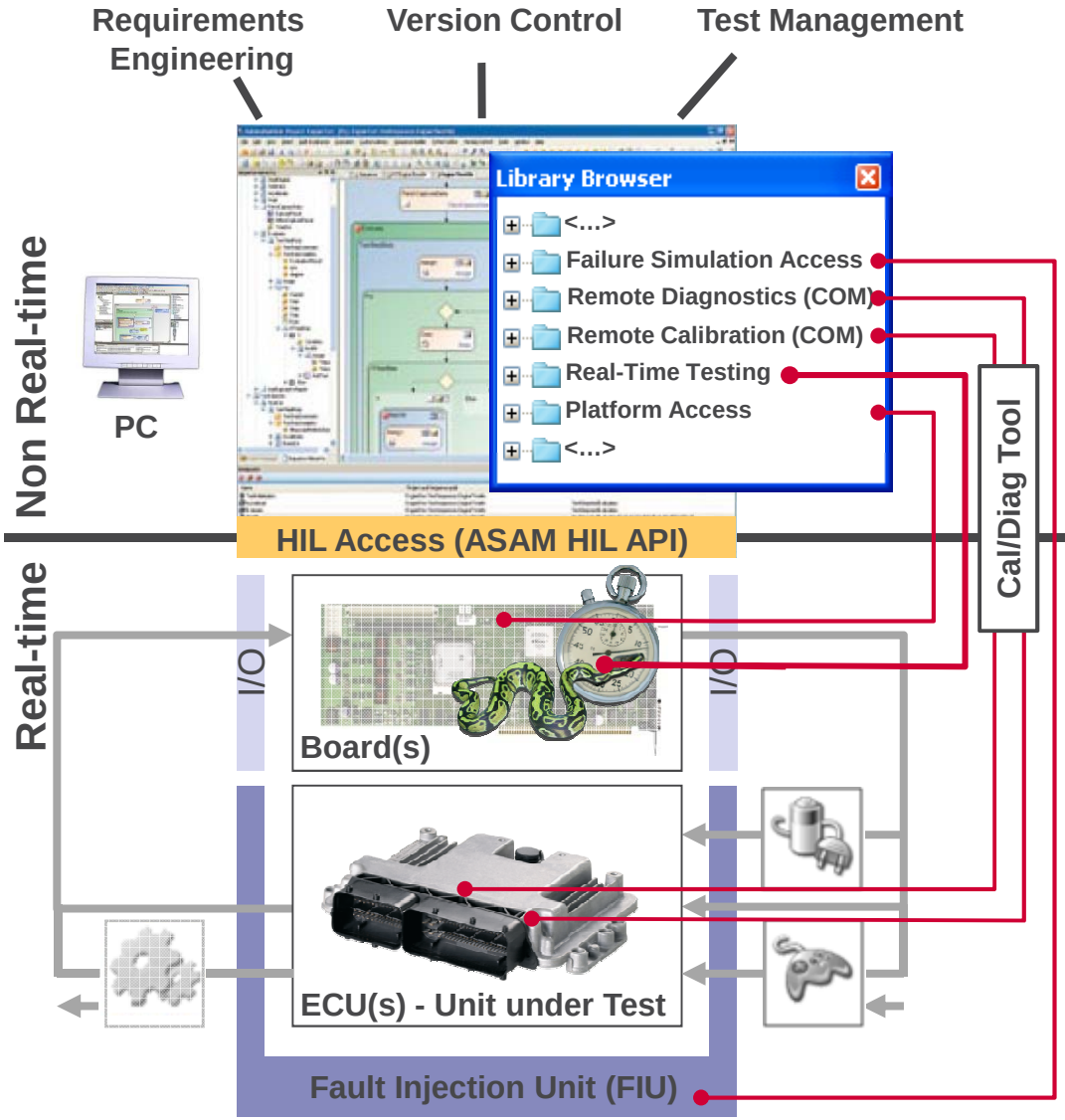
Overview HIL Simulation



HIL Simulator

- Processor board(s) run models in real-time, e. g. vehicle dynamics, lighting.
- Connected to ECU(s) via I/O and signal conditioning.
- Real actor loads.
- Power supply replaces vehicle battery.
- Real sensor loads.
- Failure Simulation on ECU pins.
- Connected to bus systems e. g. CAN, LIN, FlexRay.

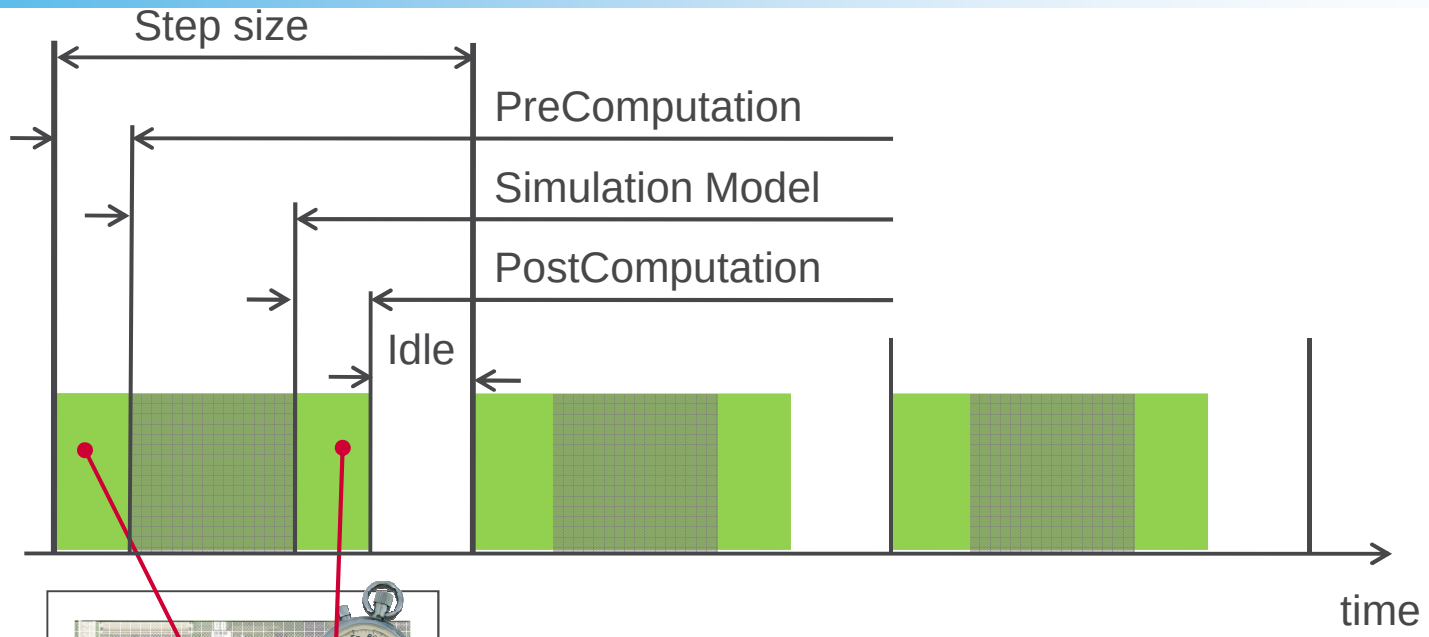
Automated ECU Testing



Automated ECU Testing

- Repeating tests precisely and automatically as often as required.
- Access to all relevant test interfaces.
- State of the art:
  Convenient PC-based
  test development and execution.

Computation of Real-Time Tests and Simulation Model



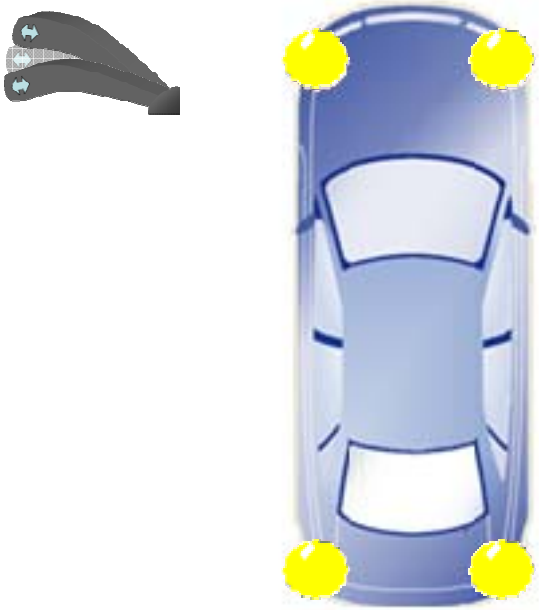
Real-Time Testing and Simulation Model

- Python interpreter is part of the real-time application.
- Real-time tests can be hooked in before or after model execution.
- Synchronized execution of simulation model and tests.
  ➔ every model change can be observed by real-time tests (e. g. concurrent watchdogs)
  ➔ real-time test can access the model in every step (e. g. for reactive stimulus)

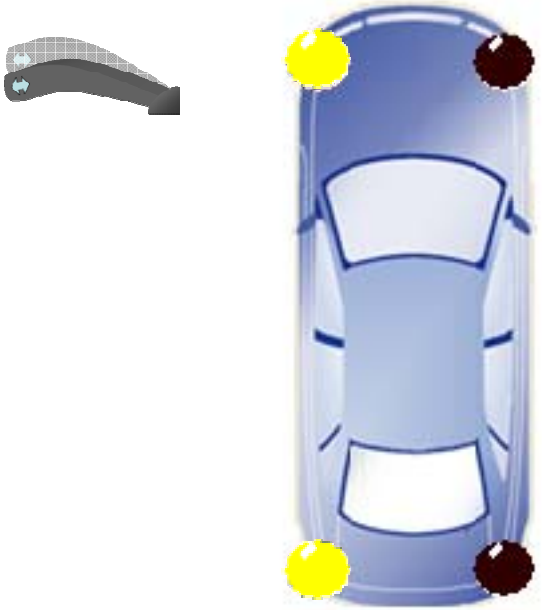
Why Python for Describing Real-Time Tests?

- **High-level programming** concepts result in **compact and readable** code.
- **Easy** to learn, easy to use, easy to extend libraries.
- Various standard **libraries available** out of the box.
- Functionality can easily be **extended** by user libraries.
- **Successfully** in use for test automation of dSPACE simulators for several years (AutomationDesk).
- **Python** can now also be used for **real-time** test programming.
- Existing test know-how can be **reused**.
- Python **objects** can easily be passed between **host and target**.

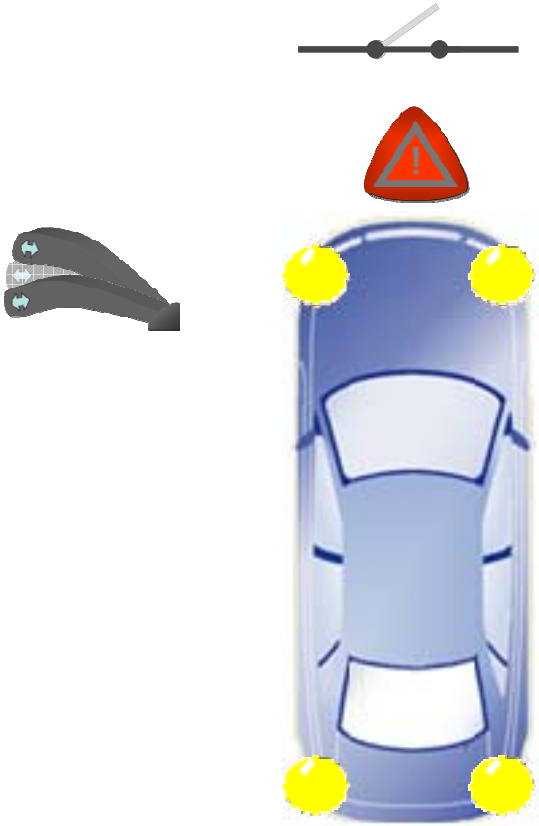
Example: Flashing (ok)



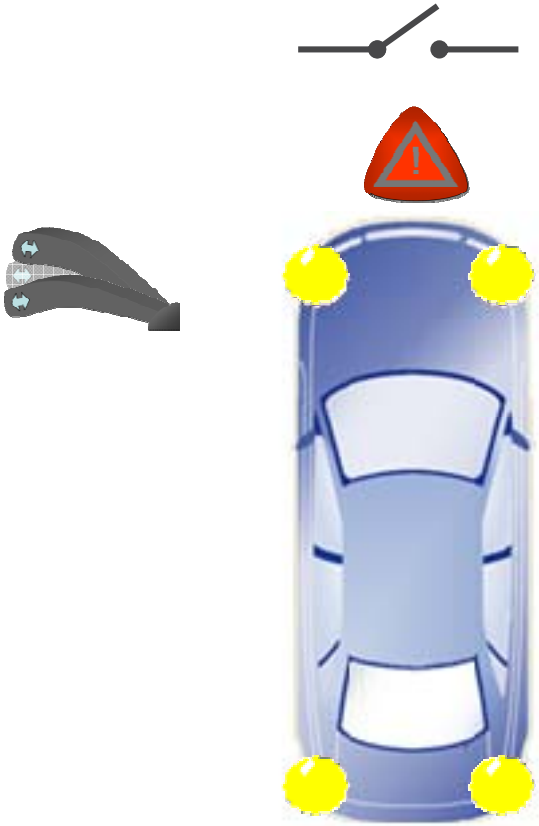
Example: Flashing (not synchronous)



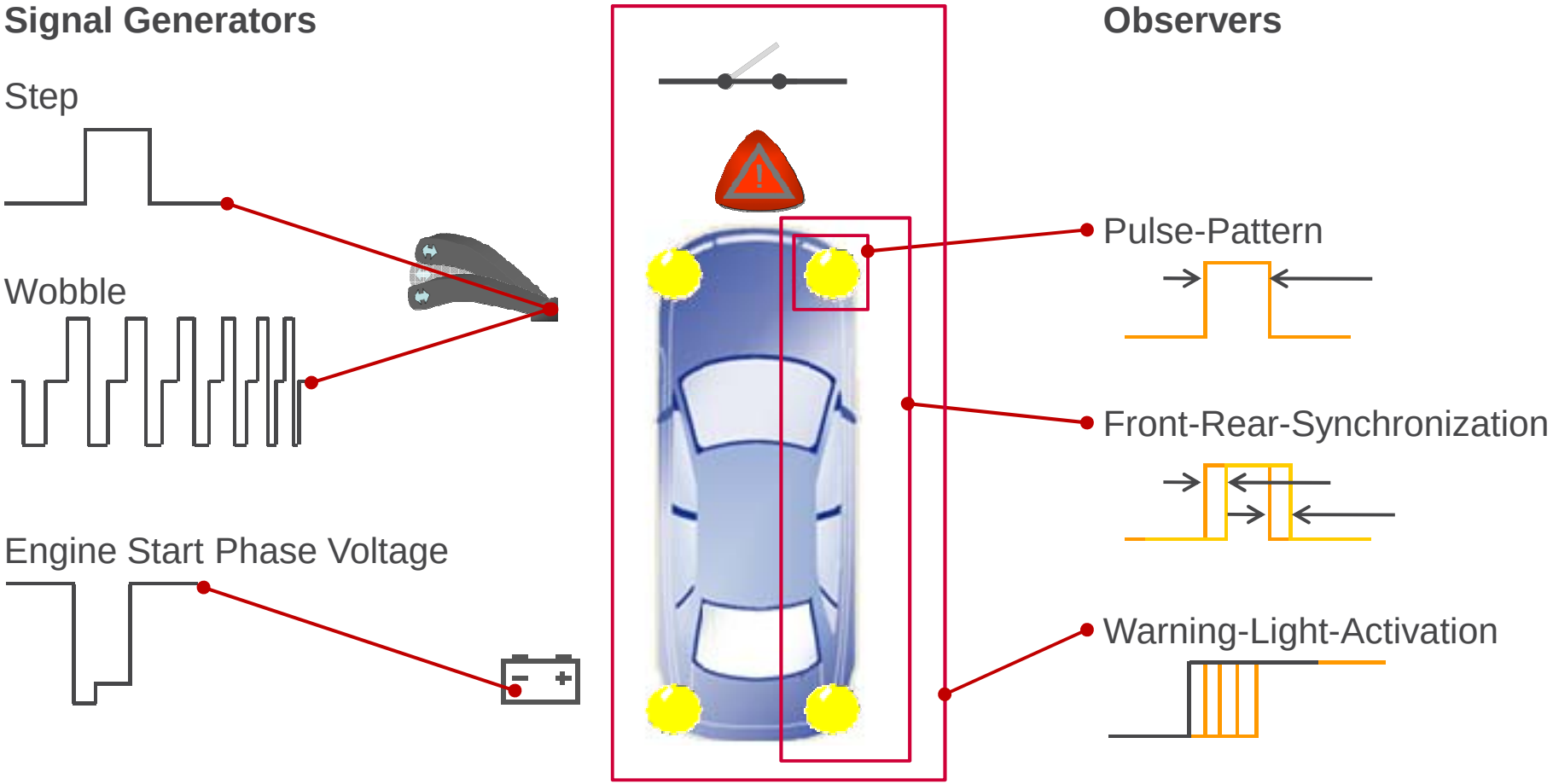
Example: Warning-light activated correctly



Example: Warning-light activated by accident

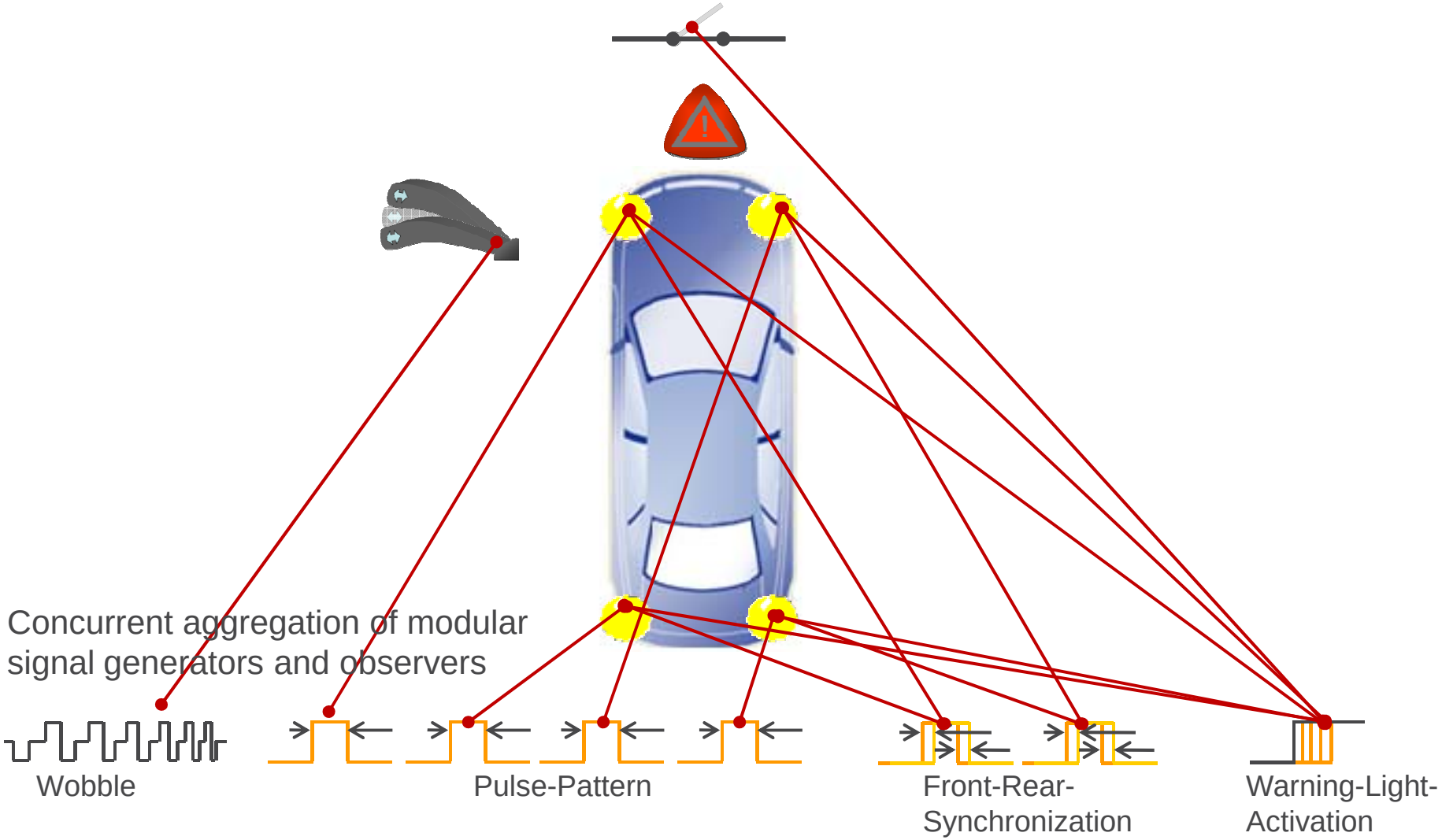


Generators and Observers

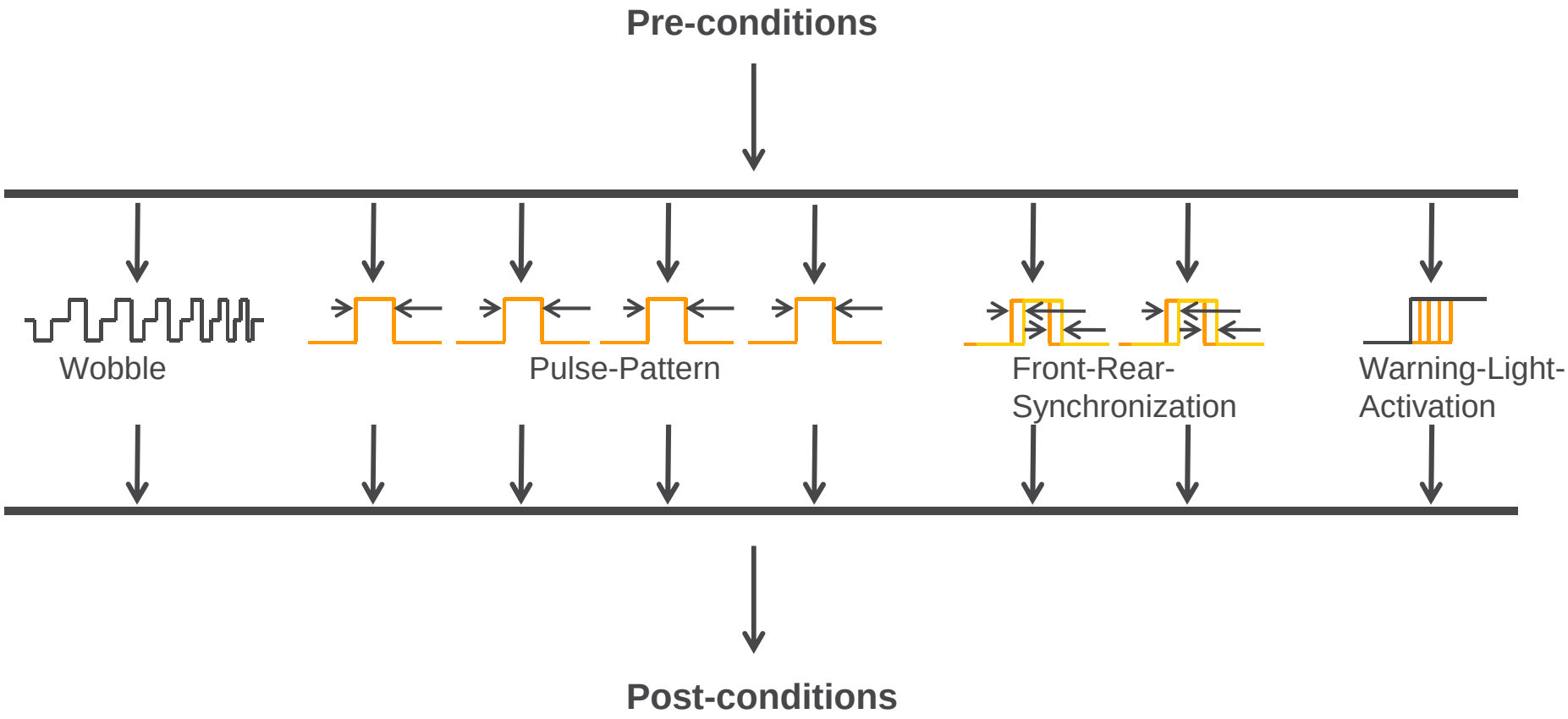


Select data (e. g. error event logging)
React (start additional test, abort the test, etc.)
Run concurrently (e. g. watchdog)

Implementing A Real-Time Test Sequence



Implementing A Real-Time Test Sequence



Python Example

dSPACE

```
def T070_MonkeyTestTurnSignalLeverGen(Result):

    # Parameters for the wobble generator.
    WobbleStartTime = 2.0
    WobbleTimeDelta = -0.25
    WobbleEndTime   = 0.25

    # Other Parameters.
    Timeout         = 30.0

    yield None
```

Pre-conditions

```
    yield scheduler.ParallelRace(
        scheduler.Parallel(nStateWobbleGenerator( mv.TurnSignalLever, WobbleStartTime, WobbleTimeDelta, WobbleEndTime),
                           PulsePatternObservatorGen(mv.SignalFrontLeft, Result),
                           PulsePatternObservatorGen(mv.SignalRearLeft, Result),
                           PulsePatternObservatorGen(mv.SignalFrontRight, Result),
                           PulsePatternObservatorGen(mv.SignalRearRight,  Result),
                           FrontRearSynchronicityObservatorGen(mv.SignalFrontLeft, mv.SignalRearLeft, Result),
                           FrontRearSynchronicityObservatorGen(mv.SignalFrontRight, mv.SignalRearRight,  Result),
                           WarningLightActivationObservatorGen(Result)),
        WaitGen(Timeout))
```

```
    yield None

    # Set signal lever position to "off".
    mv.TurnSignalLever.Value = turnsignalutilities.SIGNAL_LEVER_OFF

    # Evaluate the test result.
    Result.Evaluate()

    yield None
```

Post-conditions

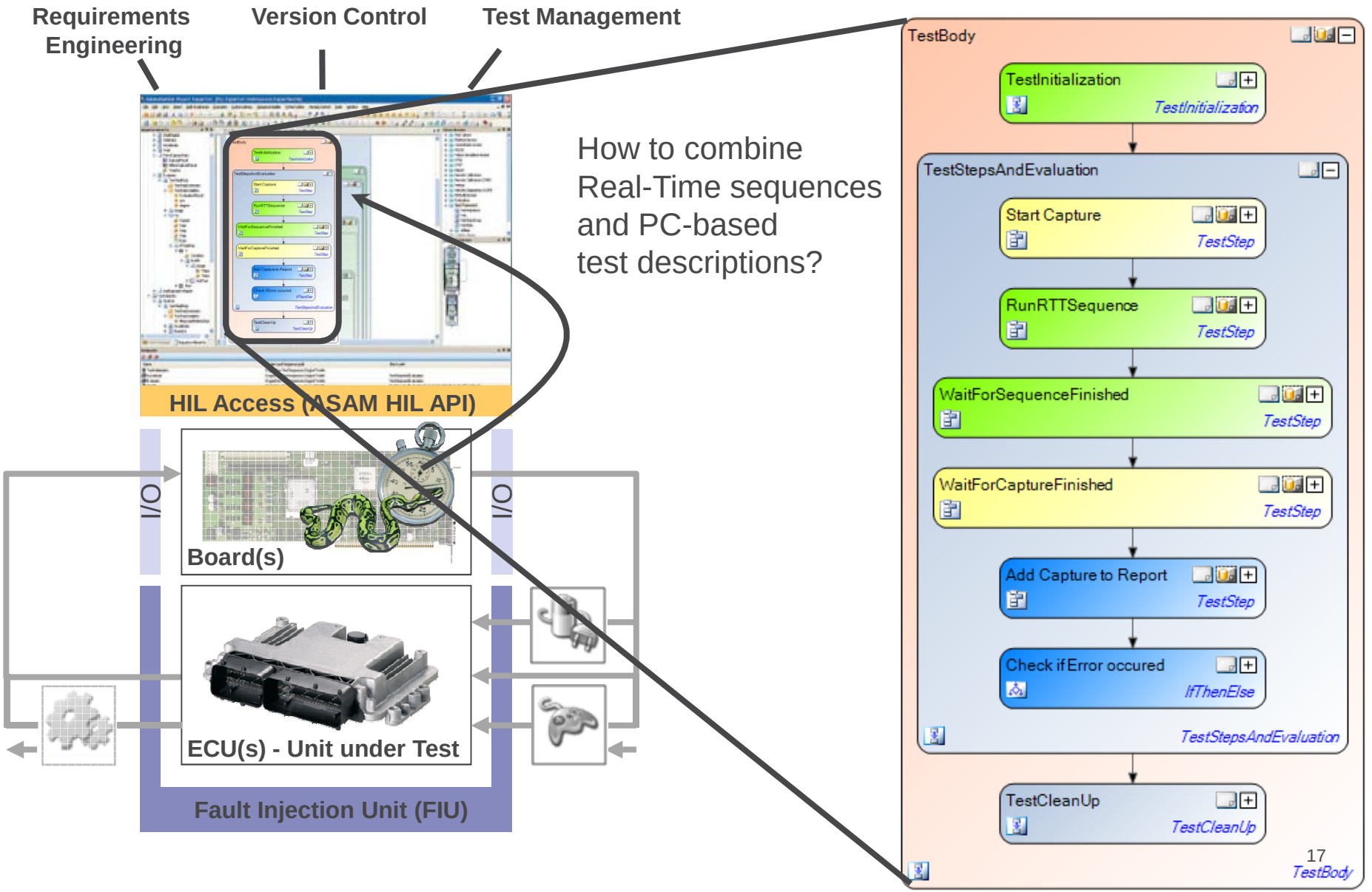
What Real-Time Testing allows

ECU Testing for **Tougher Requirements**

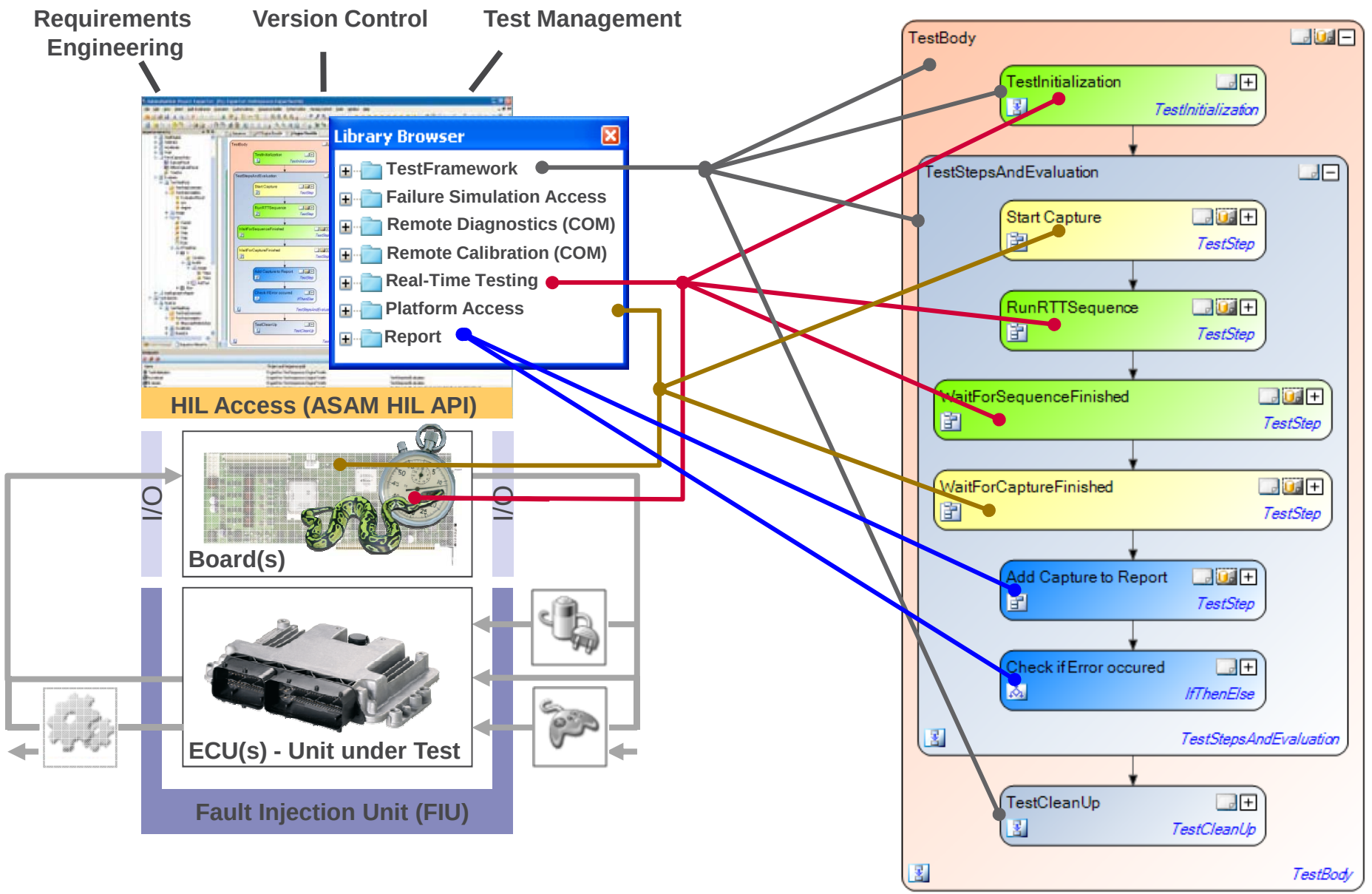
Tougher Requirements for **ECU Testing**

- Timing precision (sample time precise).
- Reproducability (100%).
- Test reactivity (in same sample step).
- Data selection.
- Concurrent watchdogs.
- Flexible, powerful test programming language (Python).

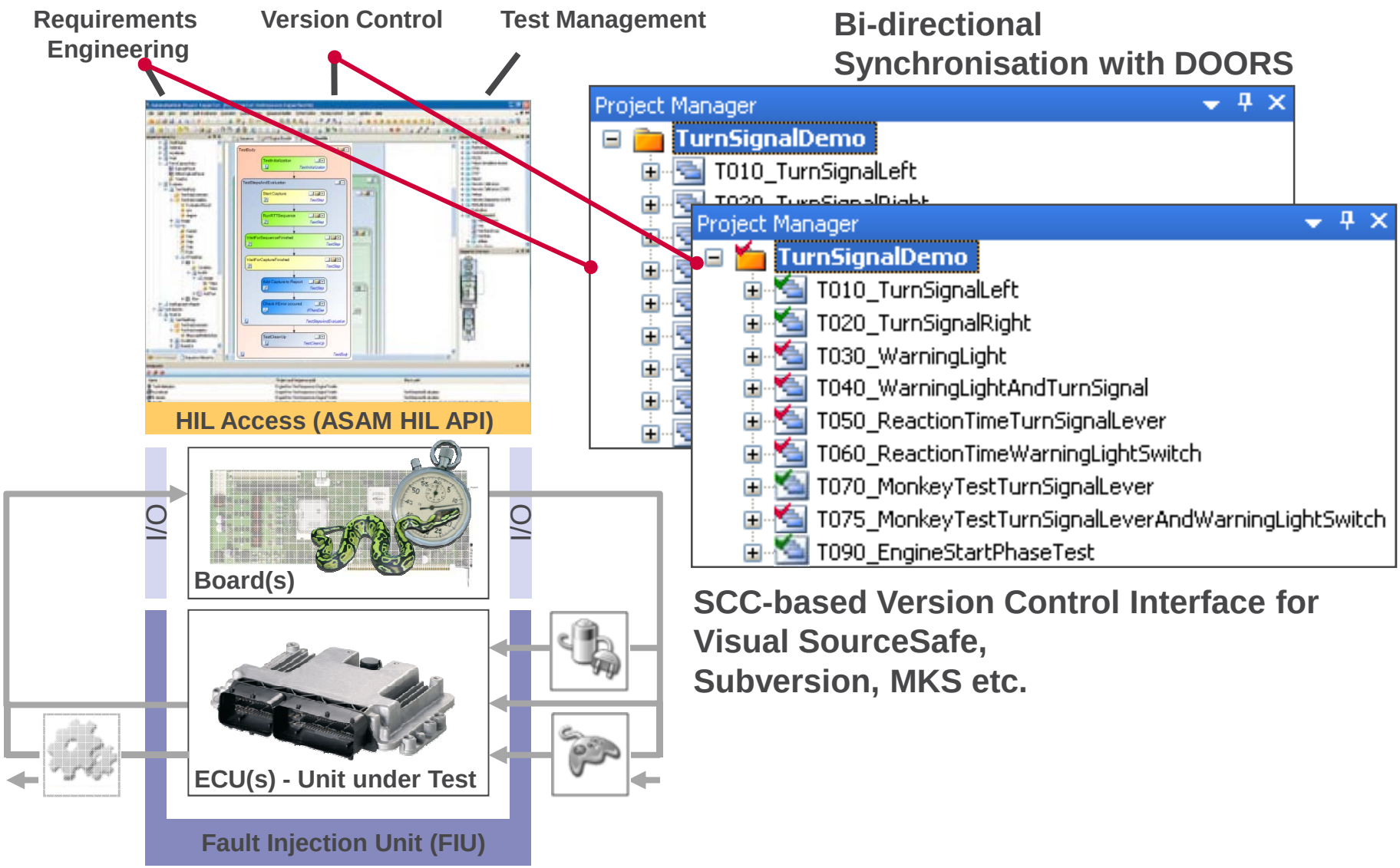
Graphical Test Description



Graphical Test Description

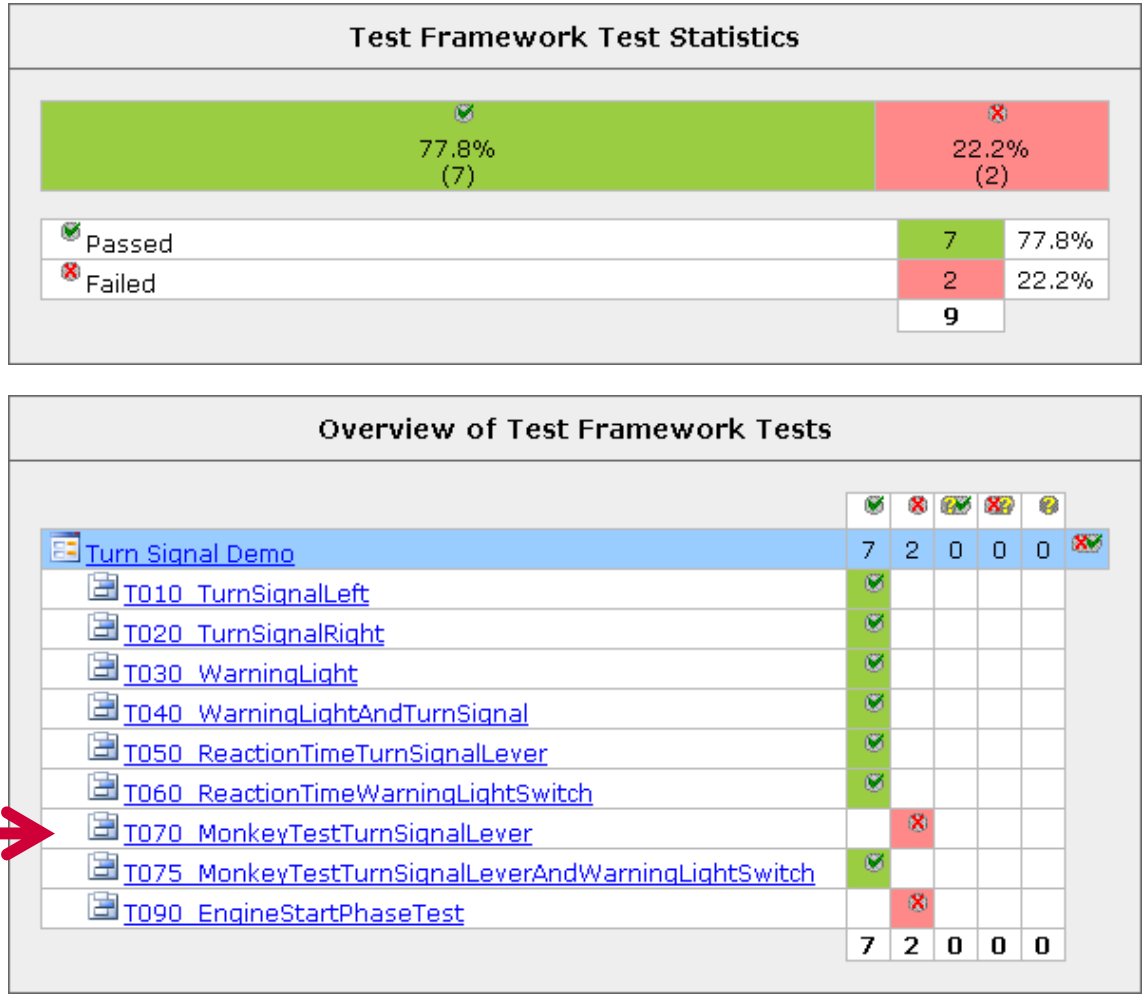


Test Process Integration



Test Report (Overview)

dSPACE



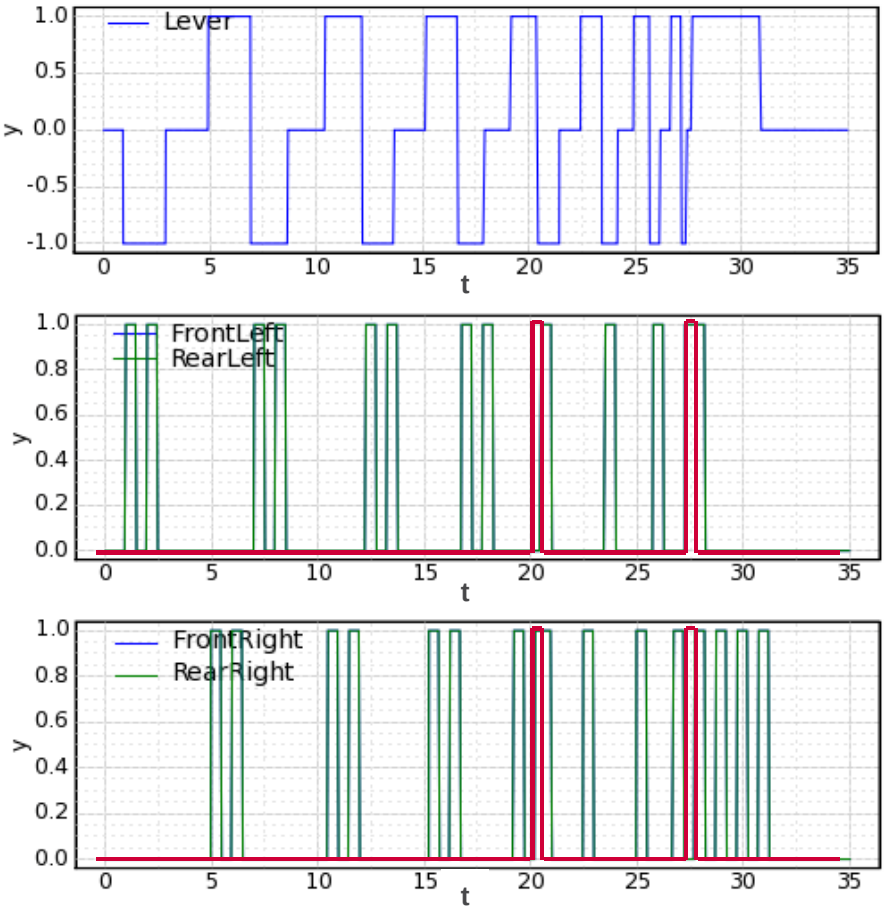
Test Framework Test Statistics

	77.8% (7)	22.2% (2)
Passed	7	77.8%
Failed	2	22.2%
	9	

Overview of Test Framework Tests

Turn Signal Demo	7	2	0	0	0	
T010_TurnSignalLeft	✓					
T020_TurnSignalRight	✓					
T030_WarningLight	✓					
T040_WarningLightAndTurnSignal	✓					
T050_ReactionTimeTurnSignalLever	✓					
T060_ReactionTimeWarningLightSwitch	✓					
T070_MonkeyTestTurnSignalLever		✗				
T075_MonkeyTestTurnSignalLeverAndWarningLightSwitch	✓					
T090_EngineStartPhaseTest		✗				
	7	2	0	0	0	

Detailed Test Report: Monkey Test TurnSignalLever

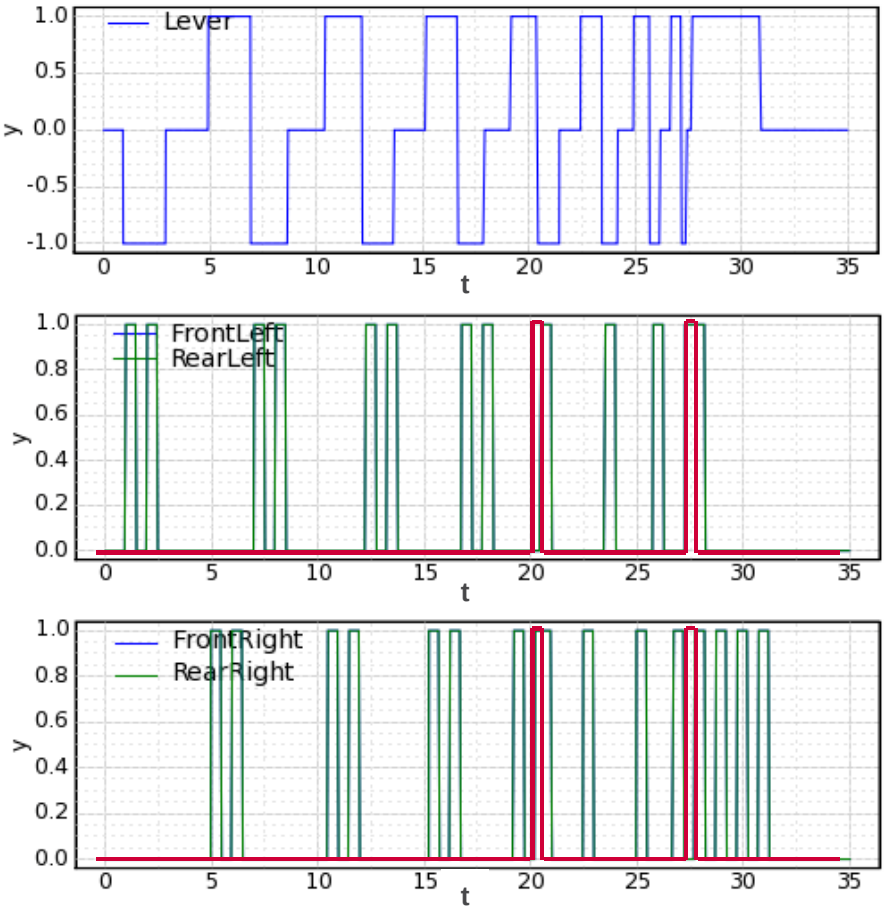


Results logged by
real-time observers
and transferred to host PC.

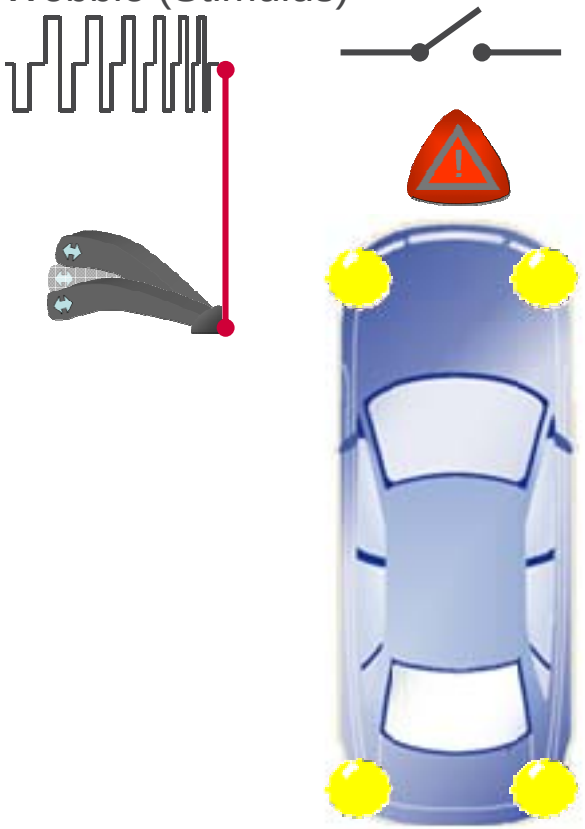
RTTTestFailureException occured.

Detailed information:
WarningLightActivationObservator [O040]: All signals activated, but warning light not switched on.
PulsePatternObservator [O010]: Pulse longer than 0.55 sec.
PulsePatternObservator [O010]: Pulse longer than 0.55 sec.

Detailed Test Report: Monkey Test TurnSignalLever



Wobble (Stimulus)



RTTTestFailureException occured.

Detailed information:
WarningLightActivationObservator [O040]: All signals activated, but warning light not switched on.
PulsePatternObservator [O010]: Pulse longer than 0.55 sec.
PulsePatternObservator [O010]: Pulse longer than 0.55 sec.

Summary

- **Timing precision, reactive and concurrent real-time tests in Python allow**

  ECU Testing for **Tougher Requirements**
  Tougher Requirements for **ECU Testing**

- **Embedded into convenient PC-based graphical test development and execution.**
- **Test Framework**
  Initialization, Evaluation, Reporting, CleanUp, Error Handling, etc.
- **Easy combination with other other HIL access types**
  Diagnostics, Calibration, FIU etc.
- **Easy integration into the test process**
  Requirements Engineering, Version Control, Test Management etc.

Thank you very much for your attention.

Contact: rrasche@dspace.de