

VETESS : MDE, Testing approaches and SysML

F. Fondement (UHA)

May 2010 - NEPTUNE

Project organization

- ▶ Vérification de systèmes embarqués
VEhicules par génération automatique de
TESts à partir des Spécifications

Checking automotive embedded systems
with automatic test case generation from
specifications

- Started in 07/2008
- End in 08/2010



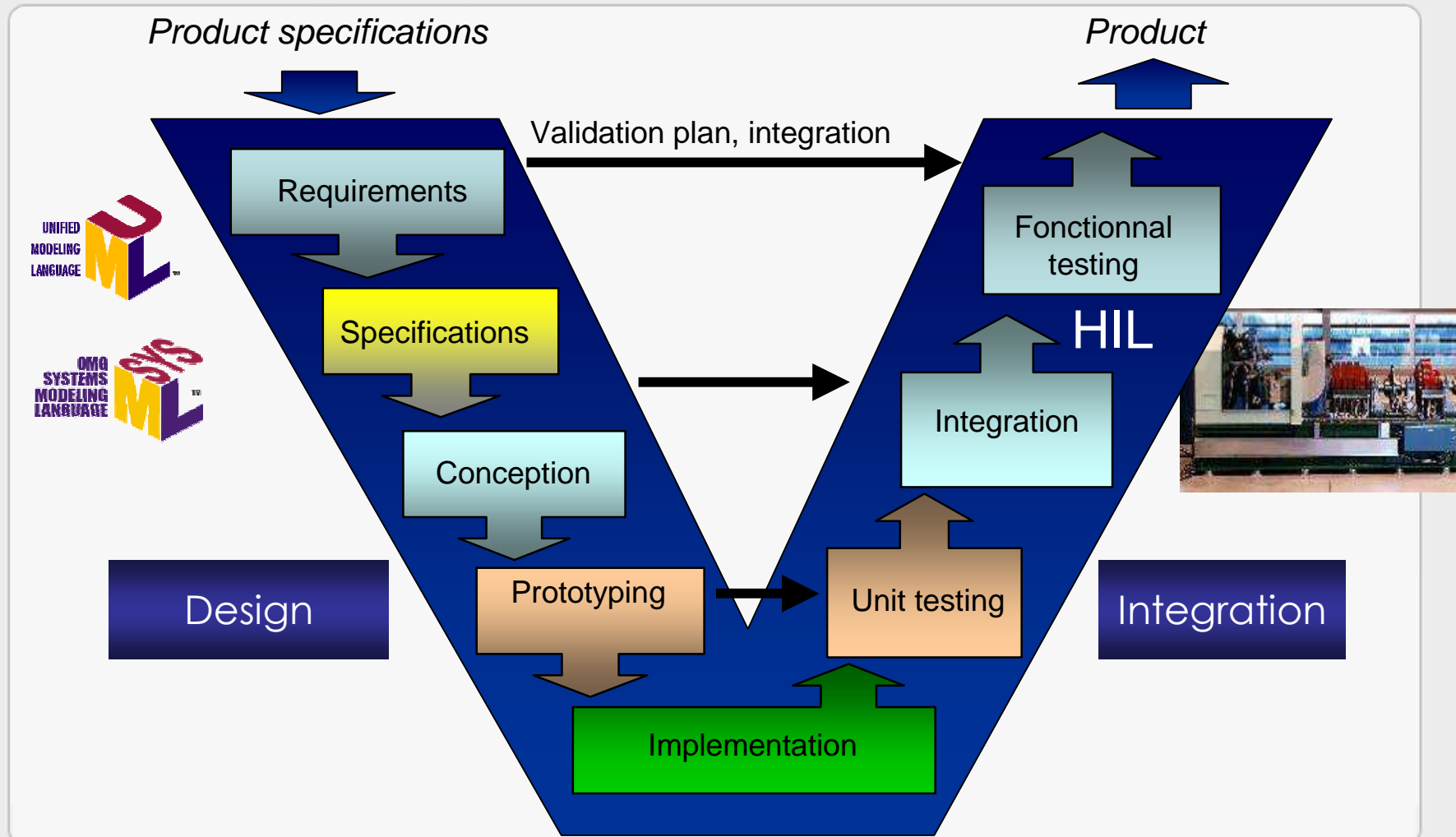
Plan

- ▶ Test in system engineering
- ▶ Model Based Testing
- ▶ The VETESS tool chain
- ▶ The front wiper case study
- ▶ Outlook

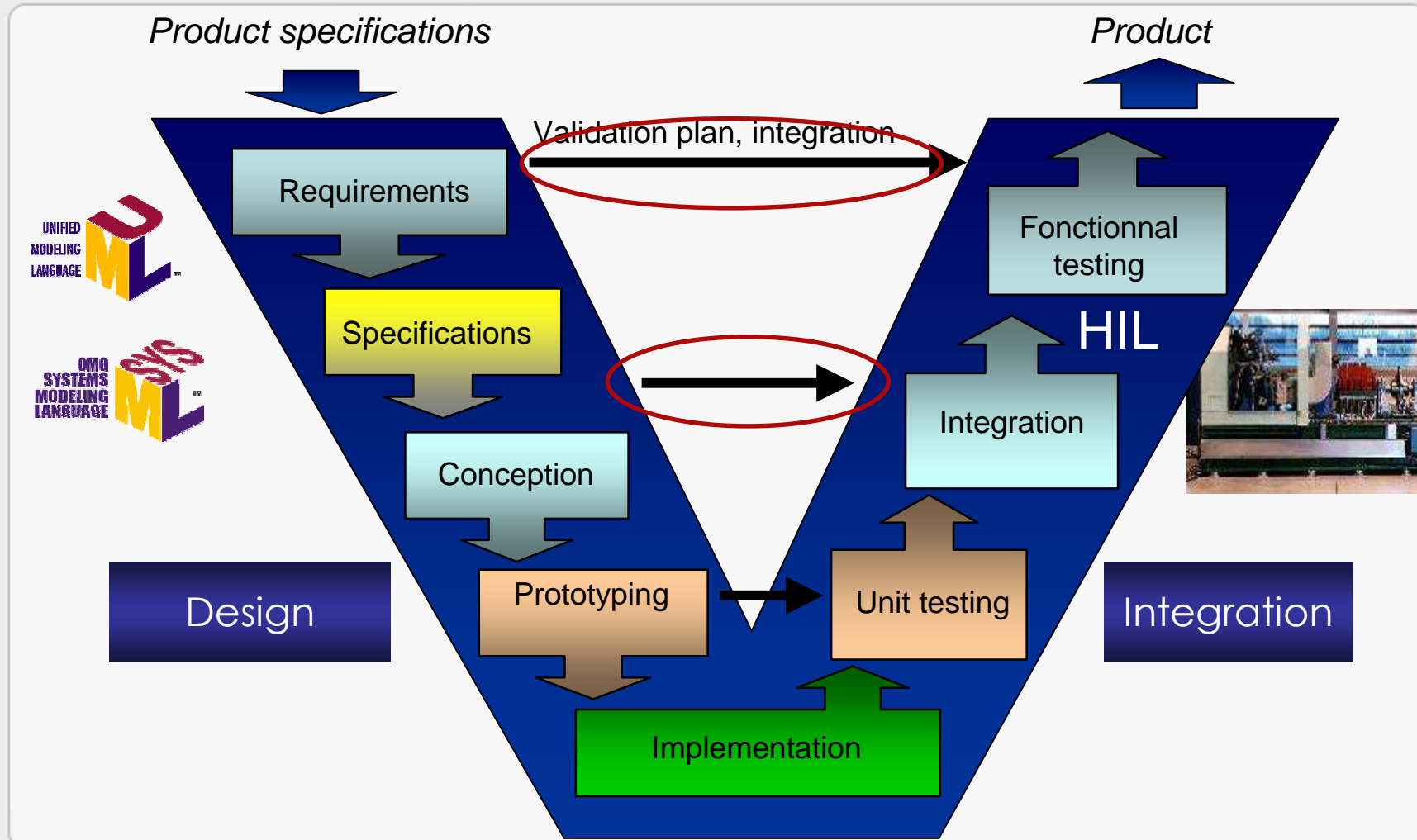
Plan

- ▶ Test in system engineering
- ▶ Model Based Testing
- ▶ The VETESS tool chain
- ▶ The front wiper case study
- ▶ Outlook

Test in system engineering



Problem: sharing specifications



Project objectives

- ▶ **Checking** automotive embedded **systems** with automatic test case generation from specifications
 - The purpose is to *generate test cases* directly from the *models* representing system *specification*

Project objectives

- ▶ Checking **automotive embedded systems** with automatic test case generation from specifications
 - Specification models:
 - UML or SysML
 - Test cases:
 - Enter test beds

Plan

- ▶ Test in system engineering
- ▶ **Model Based Testing**
- ▶ The VETESS tool chain
- ▶ The front wiper case study
- ▶ Outlook

Testing

- ▶ Verification: checking coherence between
 - A model as engineered from requirements to represent the system
 - Functional behavior
 - Non-functional properties (performance...)
 - The system
 - At runtime
 - Alone or in situation

Testing

▶ Static verification

- Code reviews
- Model checking / proof

▶ **Dynamic** verification

- System stimulation and behavior control
 - Can't cover all possible cases
 - Most representative test cases have to be selected
 - Often more expensive than the system itself
 - Execute tests
 - Produce / maintain tests
 - Analyze results

Testing

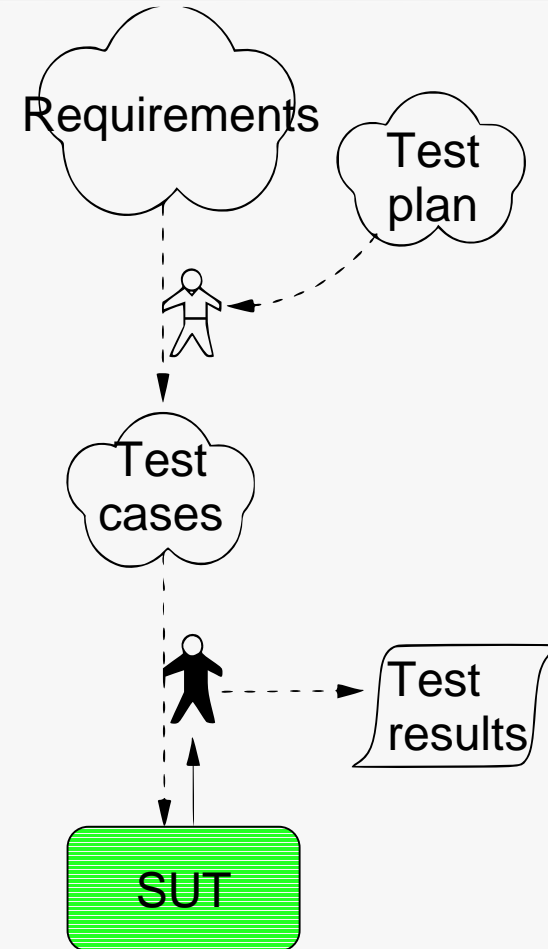
► Questions:

- Do I have enough tests ?
- Are my test covering all possibilities offered by the specification ?
- Do my tests execute in a reasonable time ?
- Did my tests pass ?
(i.e. How to observe ?)

Test approaches

▶ Manual approach

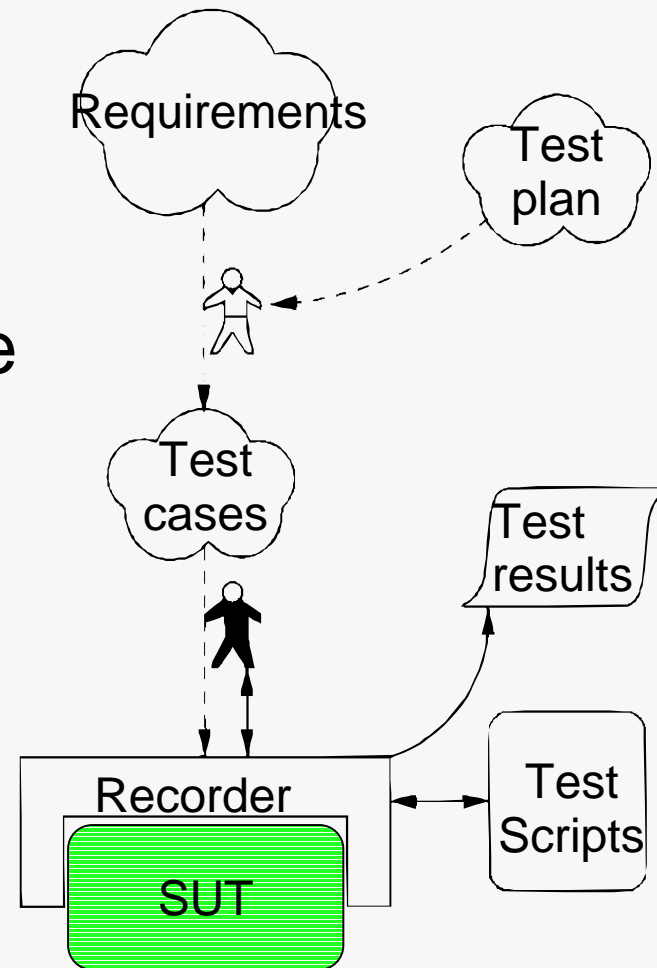
- Scenarii describe system stimulation and responses
- Human execute scenarii and check results



Test approaches

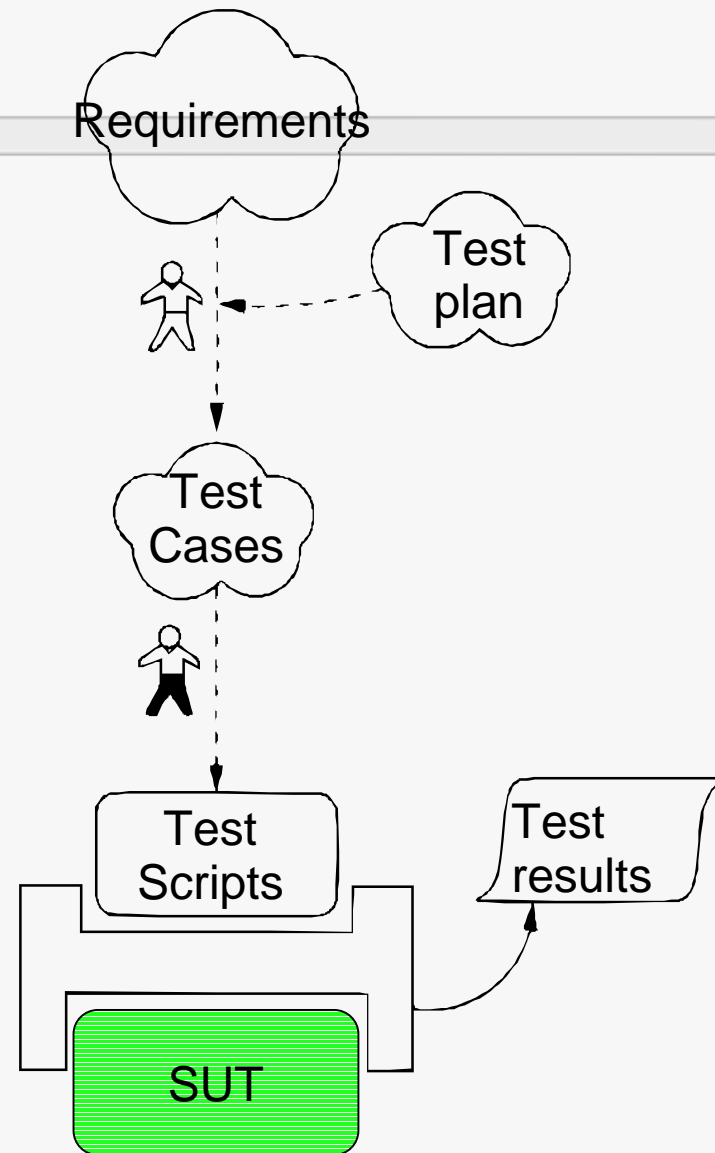
▶ Capture/replay approach

- Stimuli and responses are recorded and can be replayed automatically



Test approaches

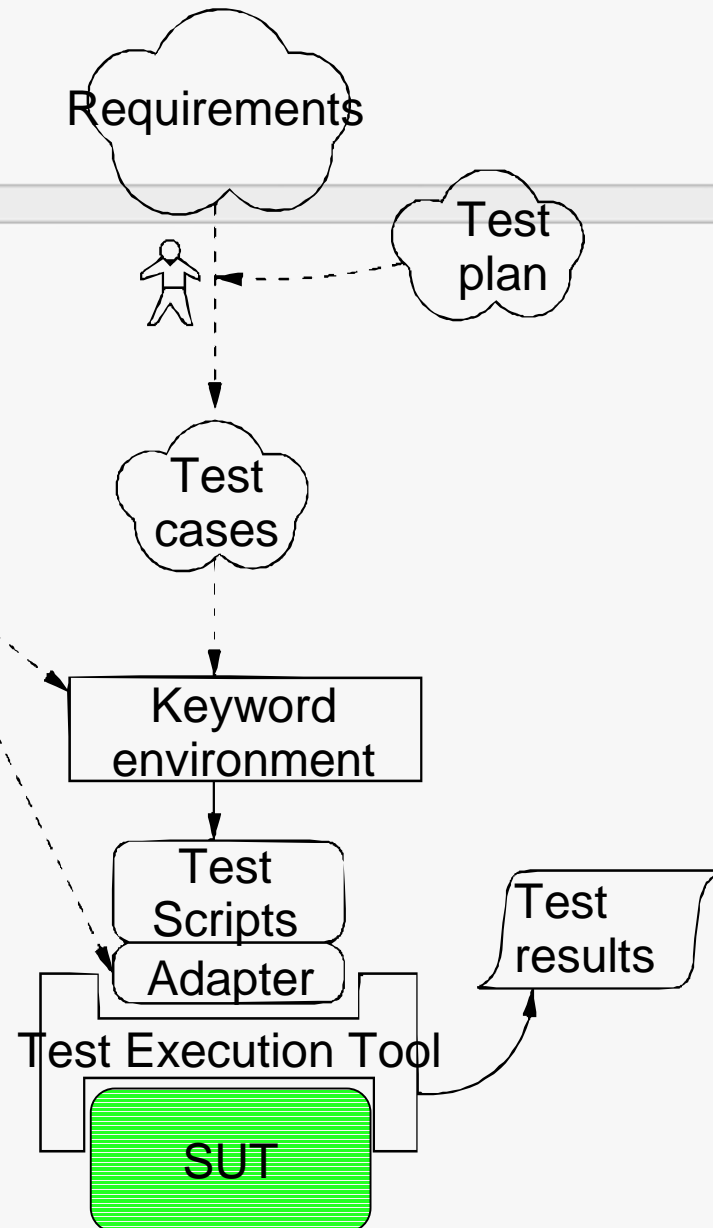
- ▶ Scripted approach
 - Scripts are written
 - JUnit, TestNG,...



Test approaches

▶ Keyword-based approach

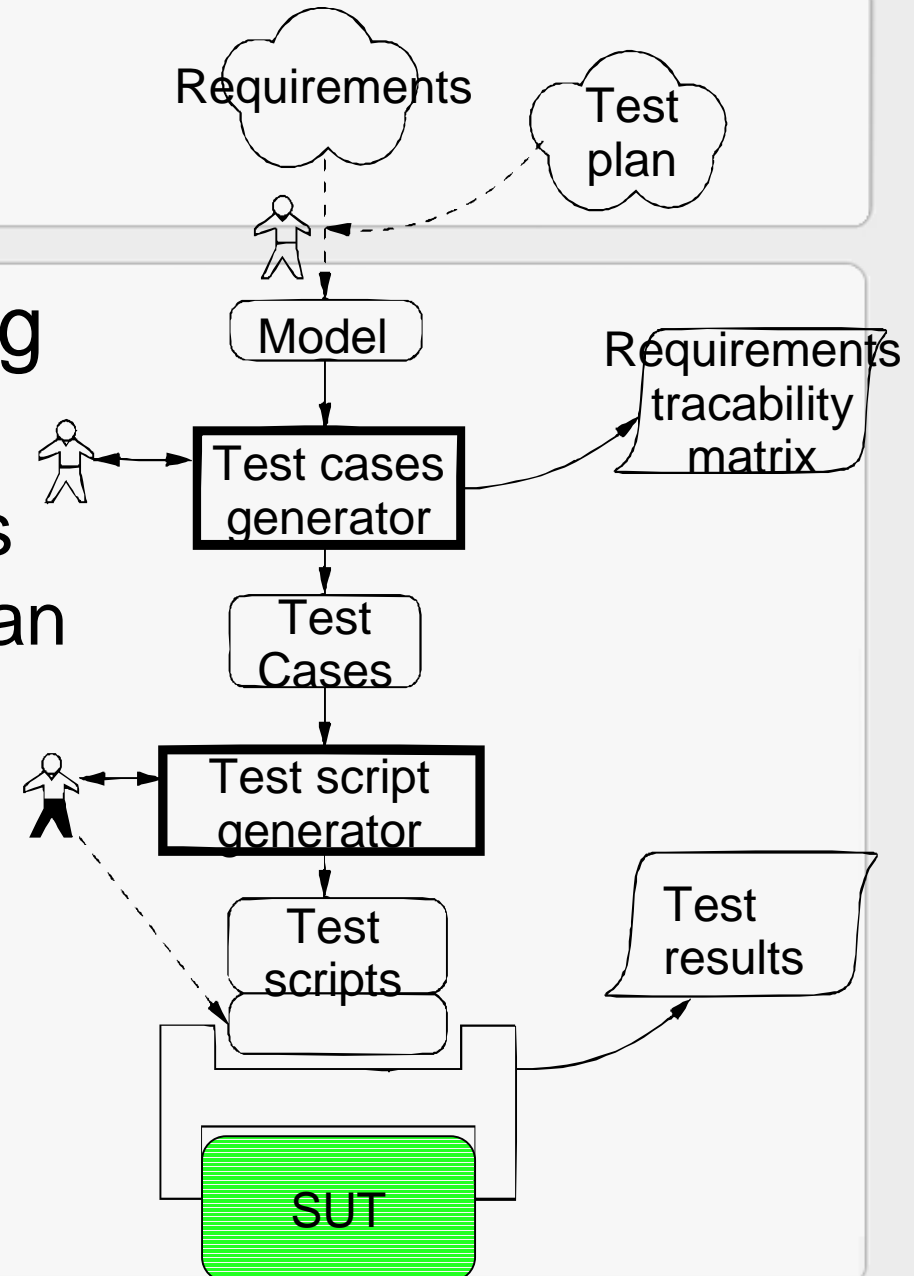
- Scripts are written in some pseudo-code
- Concretization matches keywords with real-life messages



Test approaches

► Model Based Testing

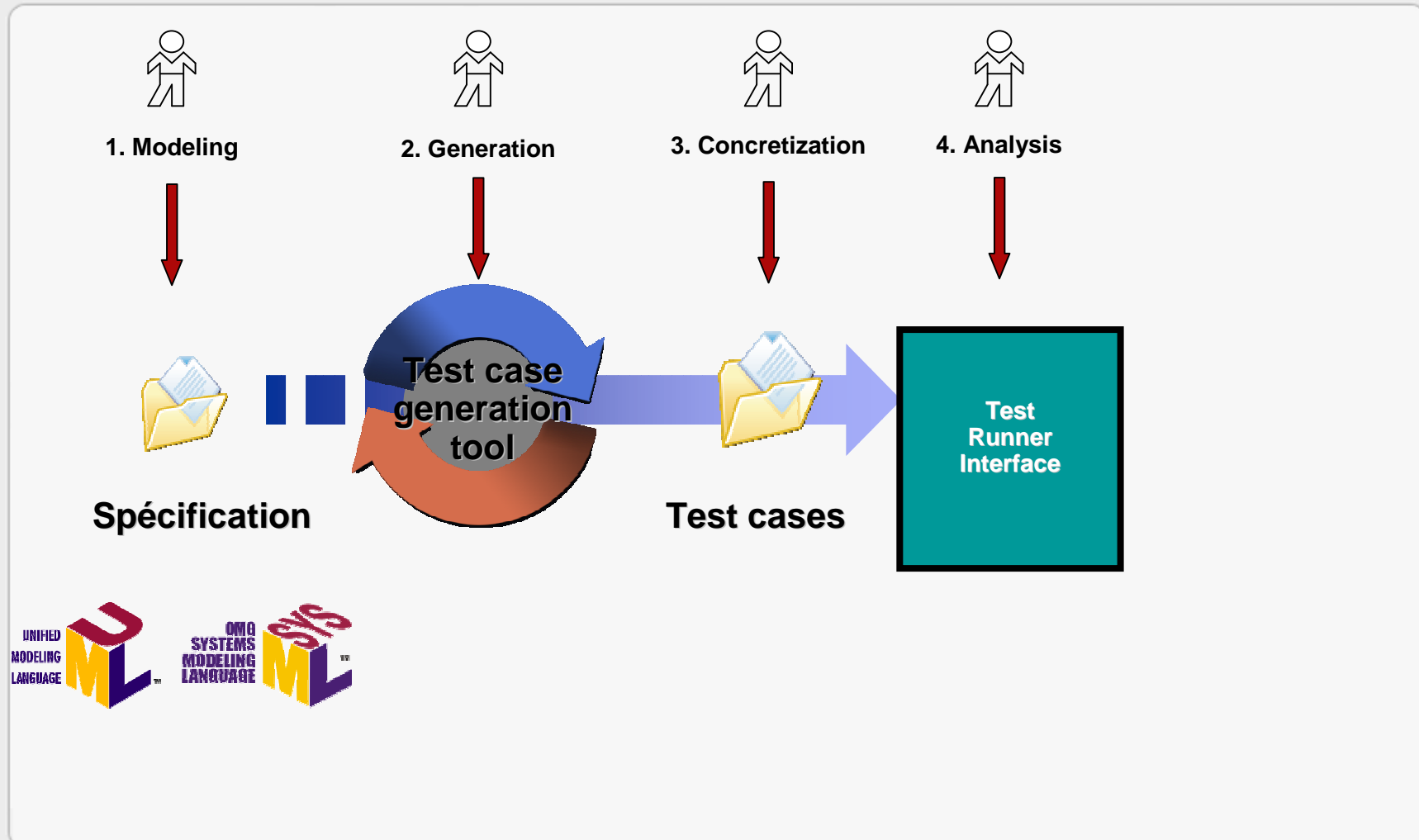
- A large number of keyword-based tests are generated from an executable model



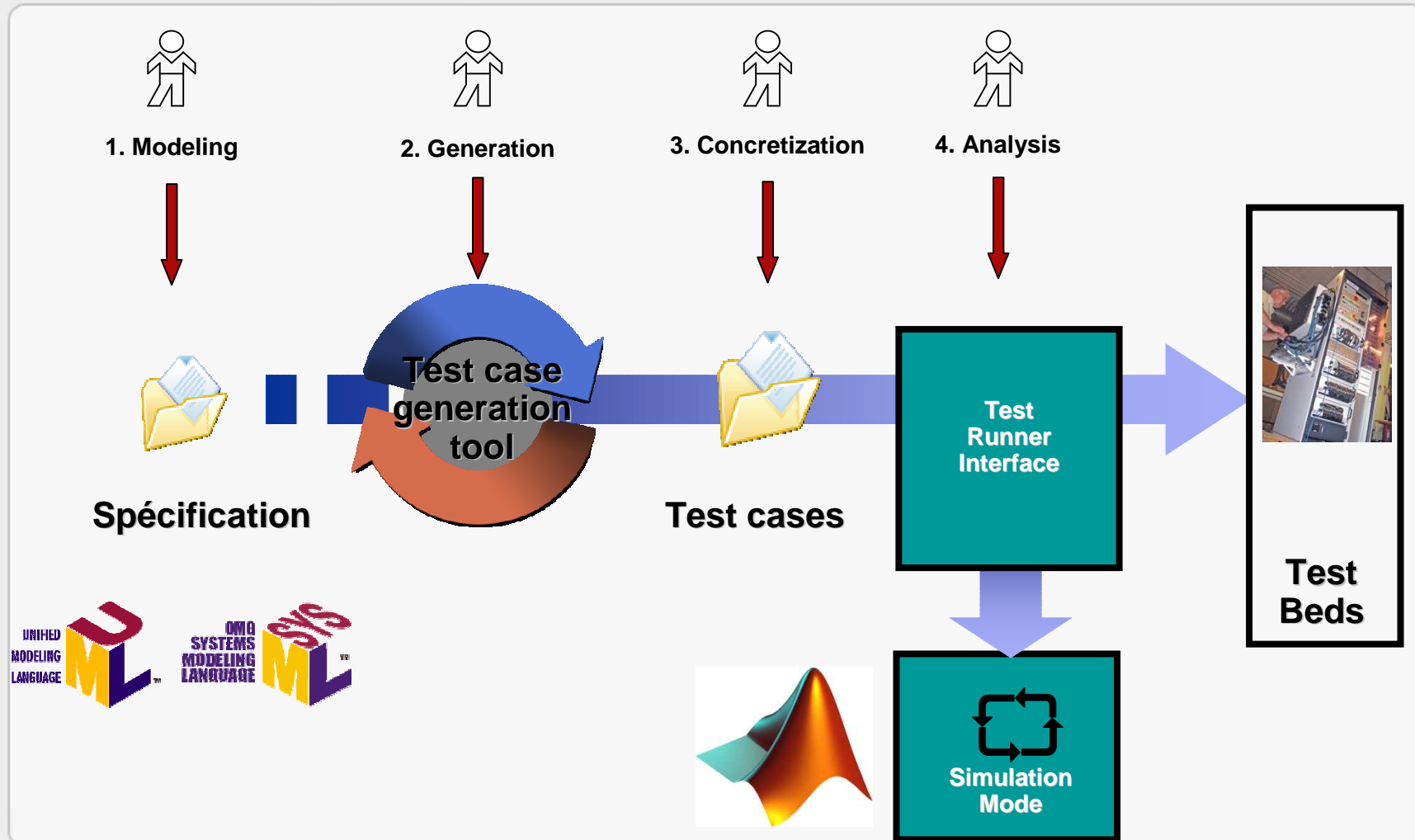
Advantages

- ▶ Automatically covers the specification model as much as possible
 - Coverage criteria (metrics)
- ▶ Test case mutualization
 - 1 test case for covering different parts of the model
- ▶ Predicts system behavior (oracle)

Model Based Testing



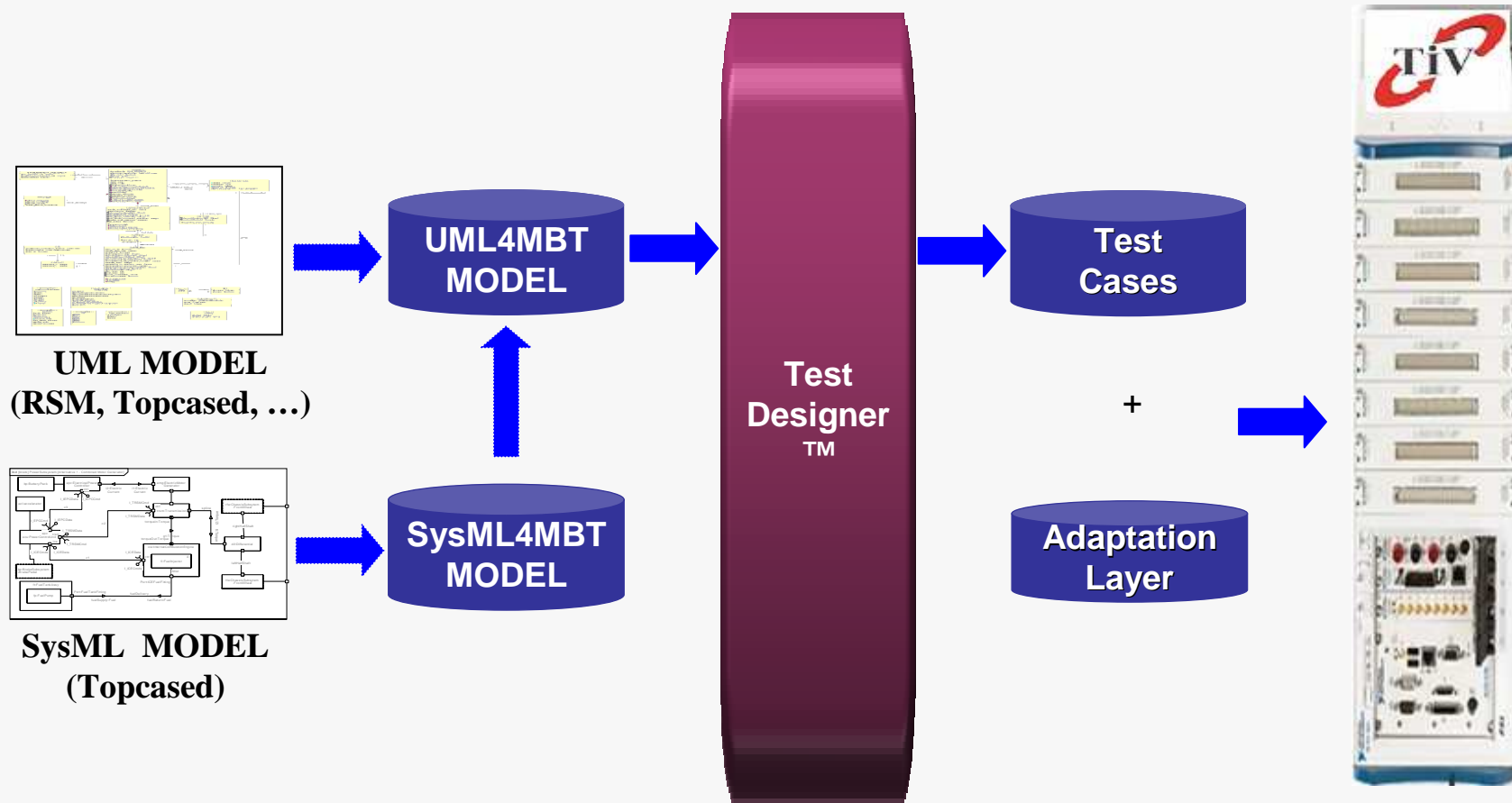
Simulation vs. test beds



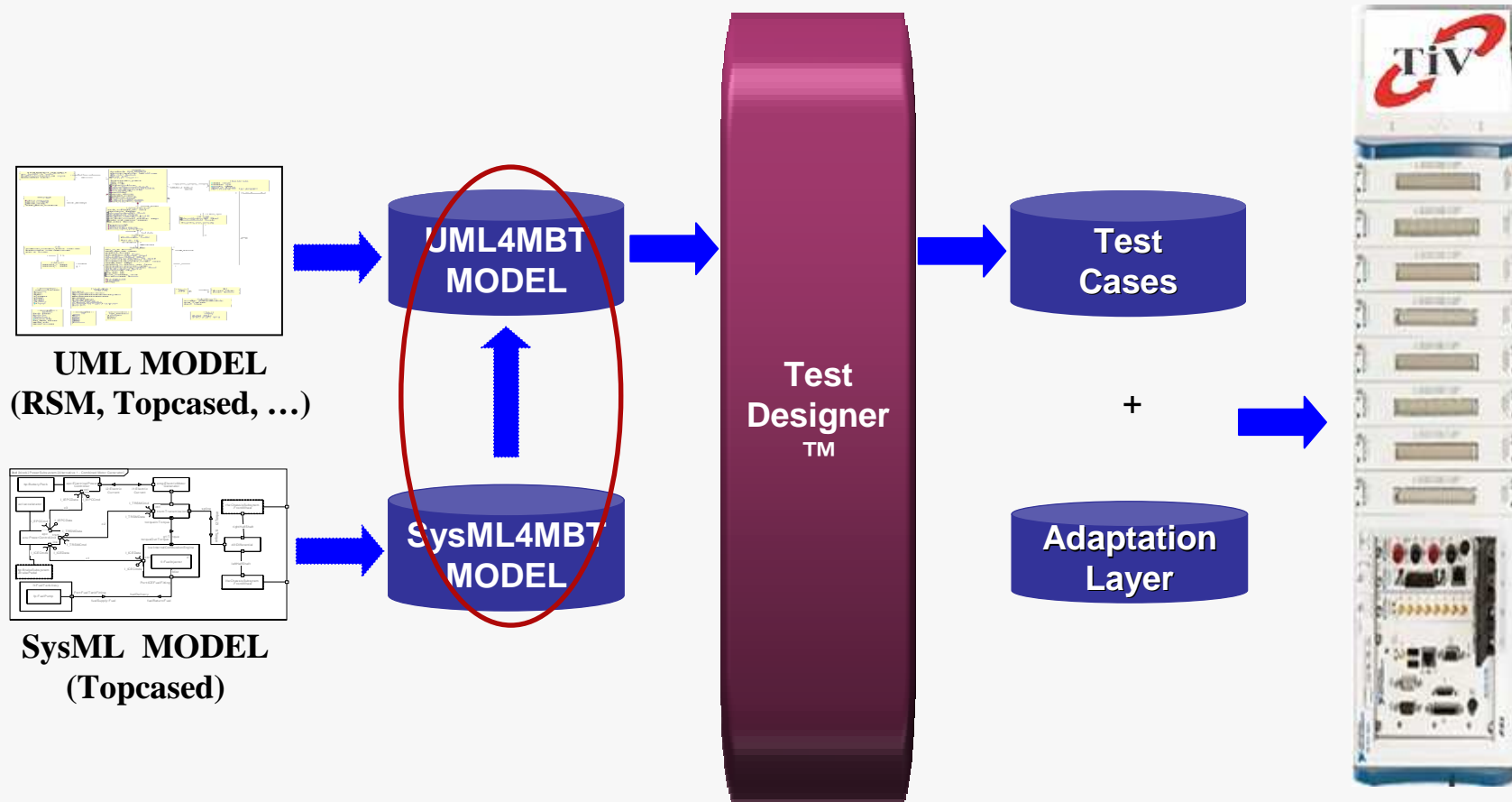
Plan

- ▶ Test in system engineering
- ▶ Model Based Testing
- ▶ The VETESS tool chain
- ▶ The front wiper case study
- ▶ Outlook

Tool chain



Tool chain



UML4MBT / SysML4MBT

▶ UML4MBT:

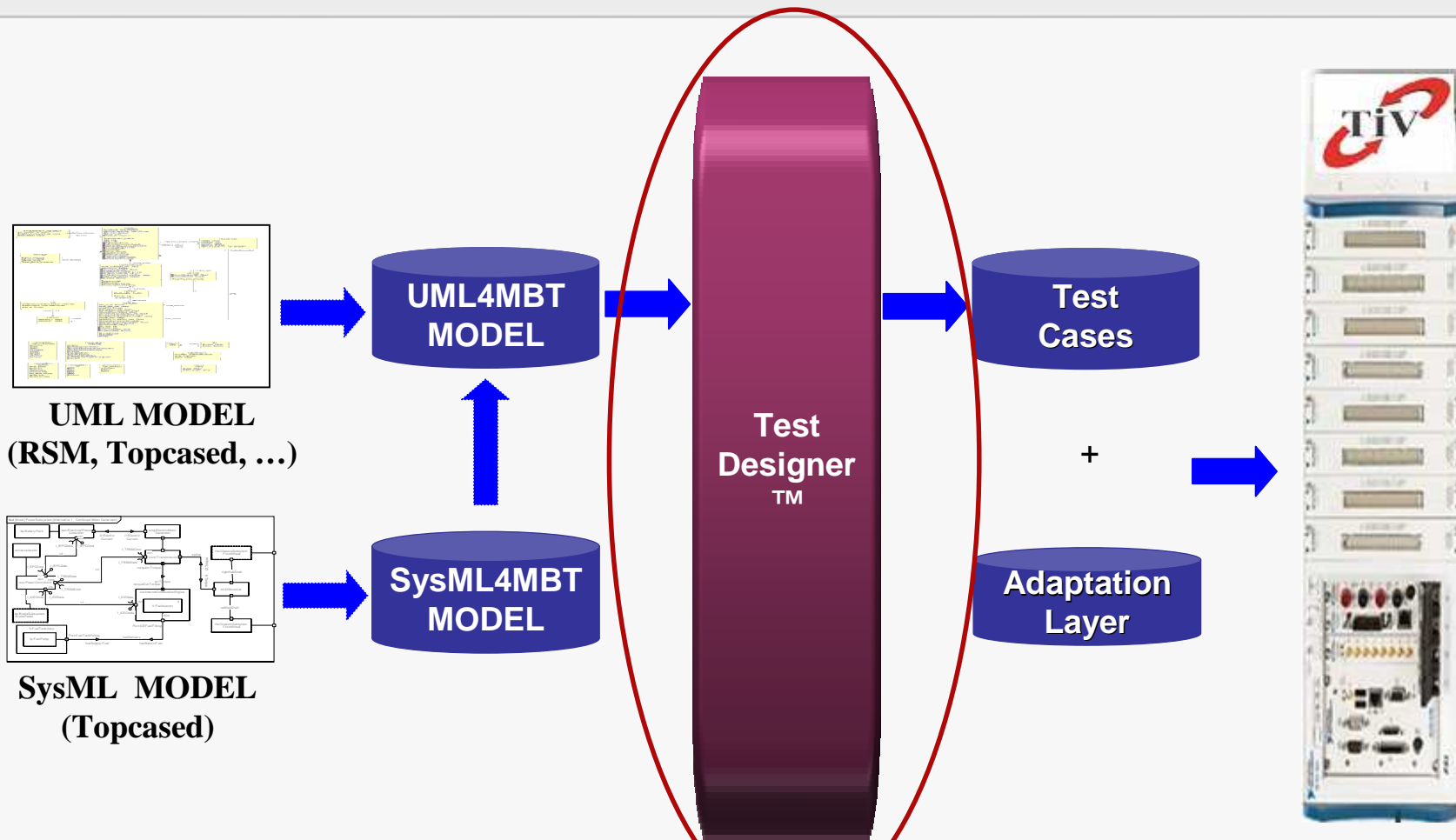
Accepted input data

- Formalized as a metamodel
- Subset of UML
- API (required interface) for Test Designer

▶ SysML4MBT

- Subset of SysML
- Model transformation
from SysML4MBT to UML4MBT

Tool chain



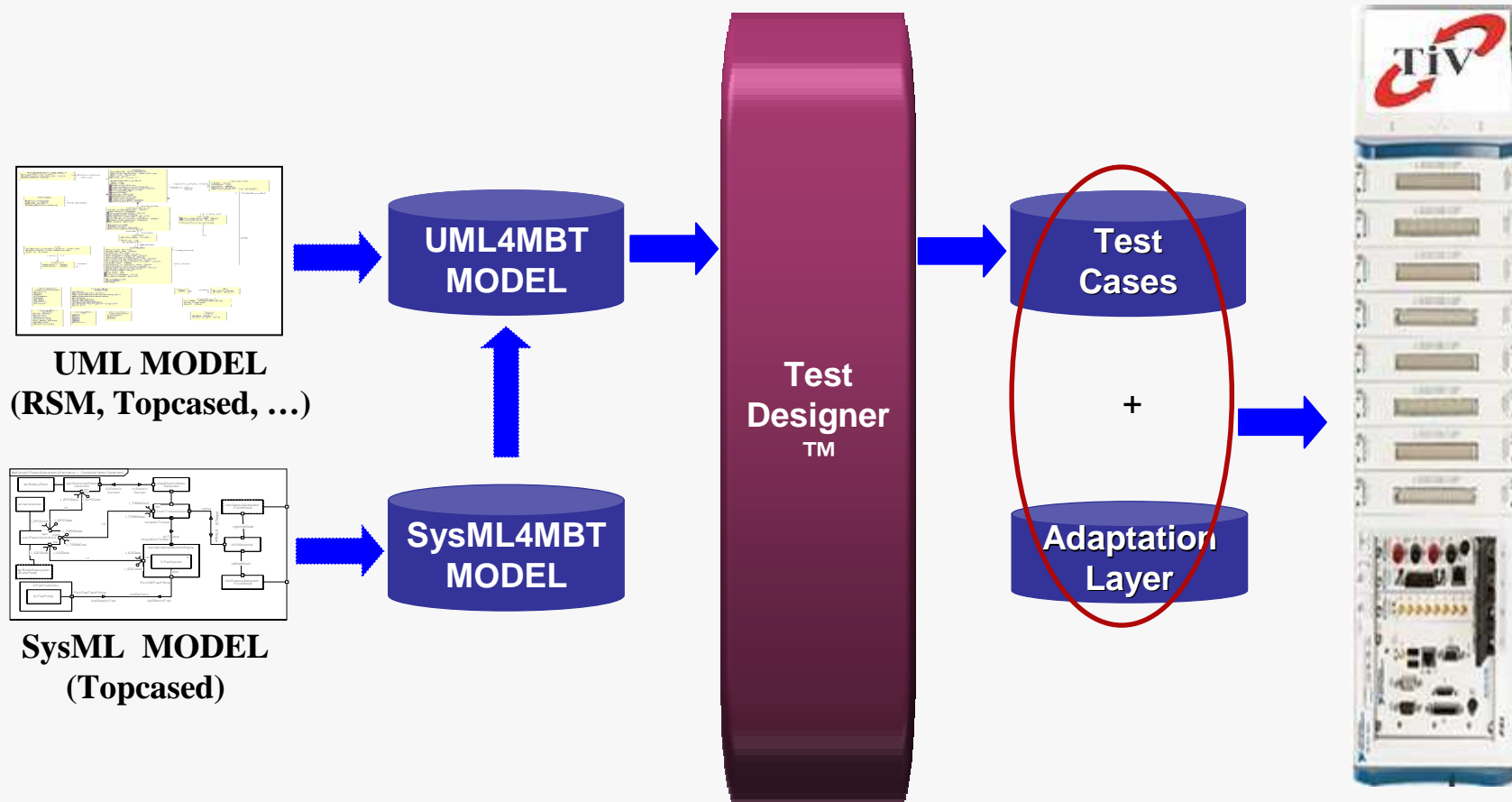
Test Designer



▶ Model-based testing tool for discrete systems

- The model is a behavioral specification of the system
 - UML model
- Test can be exported
 - Keyword
 - JUnit
 - HP Quality Center
 - ...

Tool chain

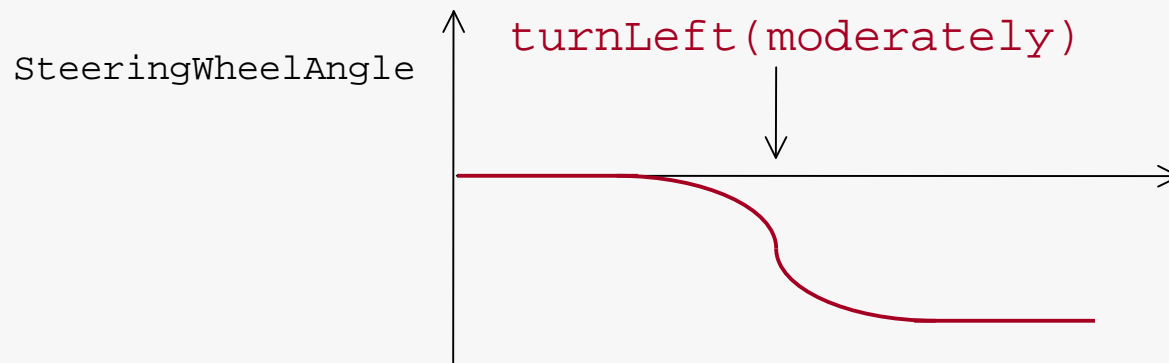


Adaptation Layer

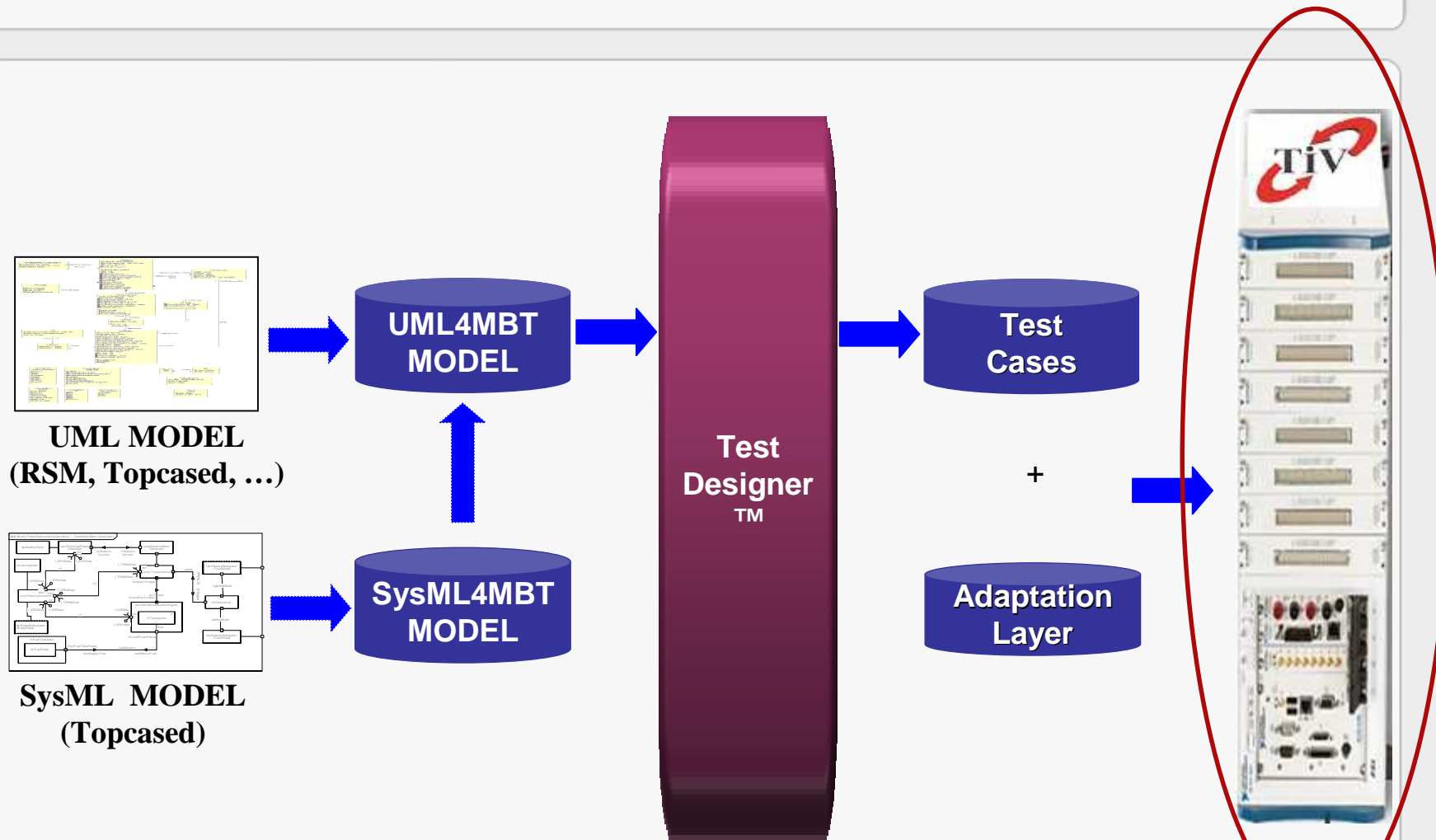
- ▶ Generated test cases
 - Have to be concretized
 - Messages : *discrete nature*
- ▶ Test bed inputs
 - Have to be mapped to reality
 - I/O Signals: *continuous nature* (sampled)
- ▶ Automatic matching
 - Generate variables
 - Generate signals

Adaptation Layer

- ▶ Discrete messages generate continuous signals
 - Example: `turnLeft(moderately)` message changes the signal to -90° to be sent on the `SteeringWheelAngle` variable
 - Curves are smoothed

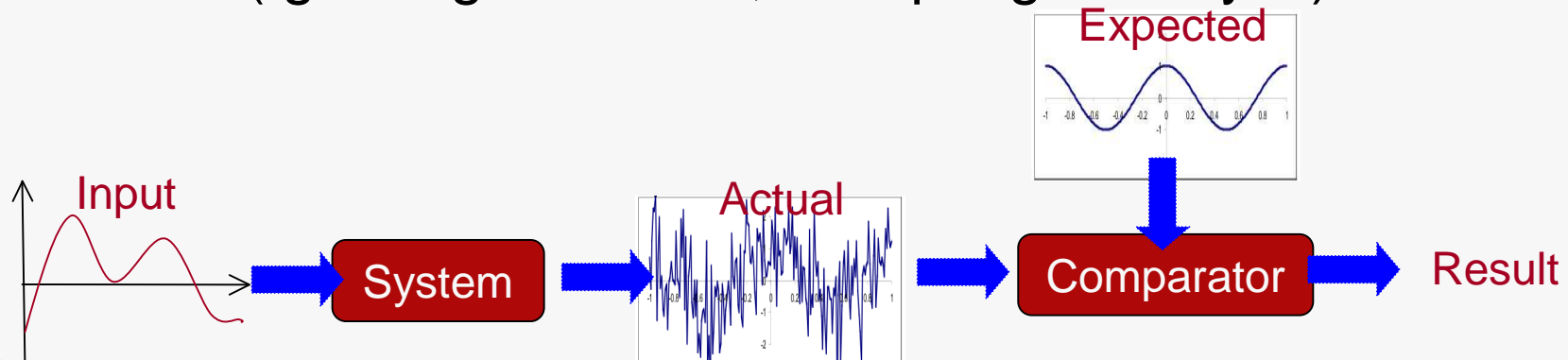


Tool chain



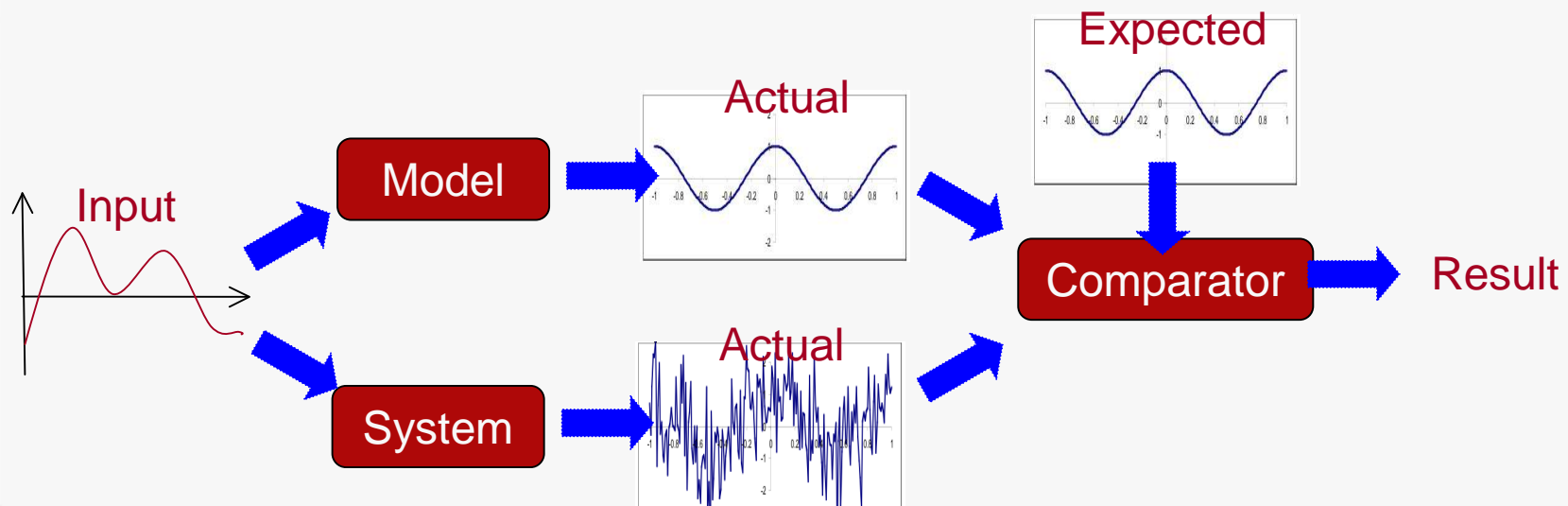
Test-In-View

- ▶ Testing continuous systems
- ▶ Input and outputs are sampled signals
 - Expected signals are compared to actual signals
 - Specific comparators (Ignoring threshold, accepting latency...)



Test-In-View

- ▶ Test may be exercised on
 - A system on a test bed
 - A simulation model (Matlab, .NET...)



Testing systems in 4 steps

- ▶ Create specification model in SysML
 - States discrete system behavior
(Could also state environment behavior)
- ▶ Generate tests from model
 - Take the shape of sampled signals sent to / read from variables
- ▶ Map variables to reality
 - CAN messages, input pins...
 - Select comparators
- ▶ Run the tests

Refining expected signals

- ▶ Additional step:
provide a continuous mathematical model
 - Generated test are run against the mathematical model
 - Output signals are promoted to expected signals

Plan

- ▶ Test in system engineering
- ▶ Model Based Testing
- ▶ The VETESS tool chain
- ▶ The front wiper case study
- ▶ Outlook

FrontWiper case study

- ▶ Industrial applications of windscreen wiper controller
- ▶ The user can turn the windscreen wiper on or off using the control lever (manual mode). Sensors can automatically drive this management (automatic mode).
- ▶ The windscreen wiper engine has three states:
 - stopped,
 - intermittent
 - activated.
- ▶ When the wiper engine is intermittent or activated, the wiper speed depends on the control level and the car speed.

Plan

- ▶ Test in system engineering
- ▶ Model Based Testing
- ▶ The VETESS tool chain
- ▶ The front wiper case study
- ▶ Outlook

Outlook

- ▶ SysML integrated by mean of a model transformation
 - UML equivalences (UML4MBT)
 - Discrete subset
 - Additional mathematical formula
- ▶ Modeling system vs. modeling environment
 - “Unsampling” continuous behavior
 - Practical experiments with a car steering column case study