VETESS

VETESS : MDE, Testing approaches and SysML

May 2010

smartesting®
Optimize your Test Center

CLEMESSY

mips

LIFC

PSA PEUGEOT CITROËN

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
  - En in 08/2010

Pôle Véhicule du Futur®
Solutions pour véhicules & mobilités du futur

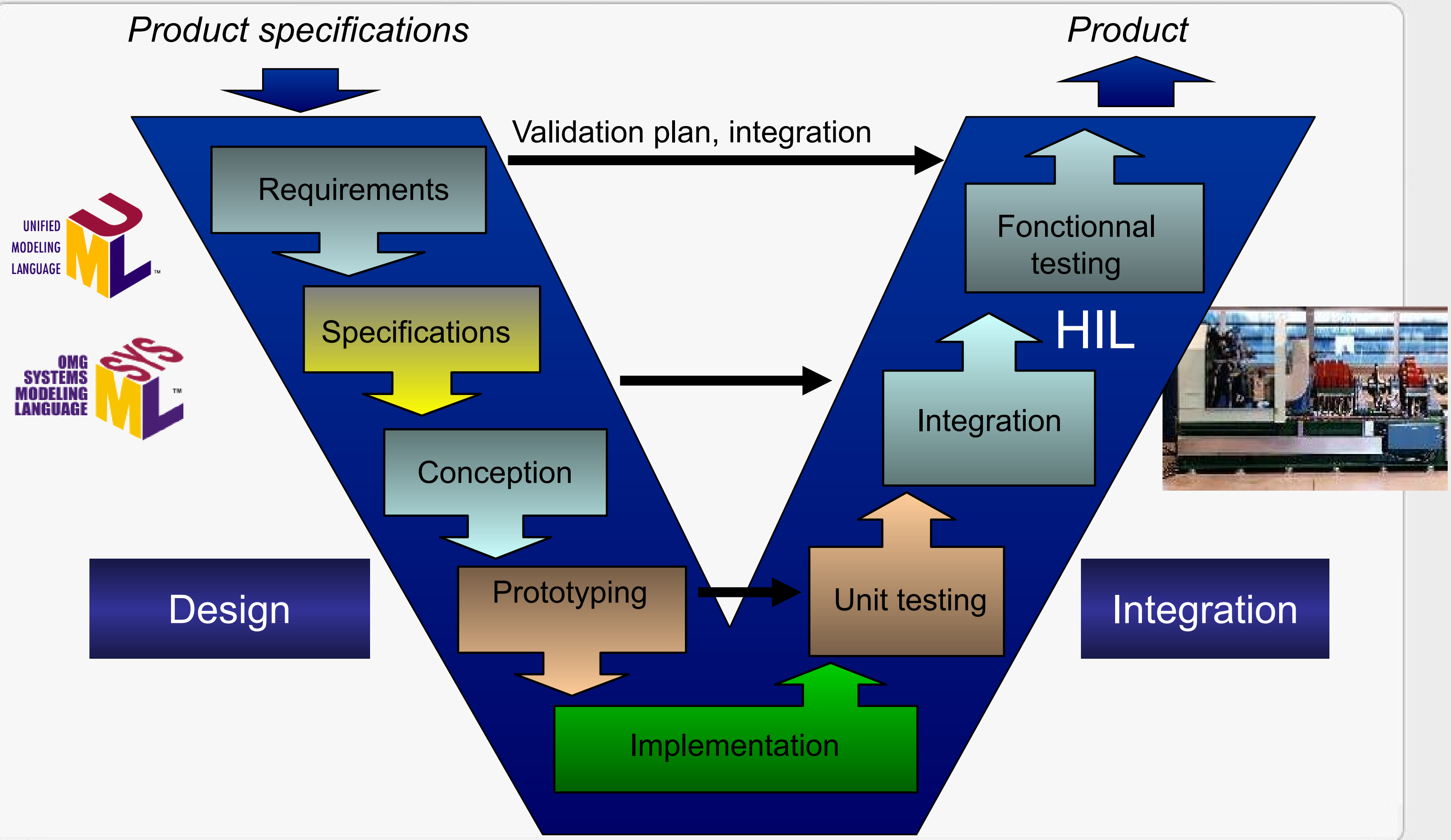
Plan

- 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



Product specifications

Product

Validation plan, integration

Requirements

Specifications

Conception

Prototyping

Implementation

Unit testing

Integration

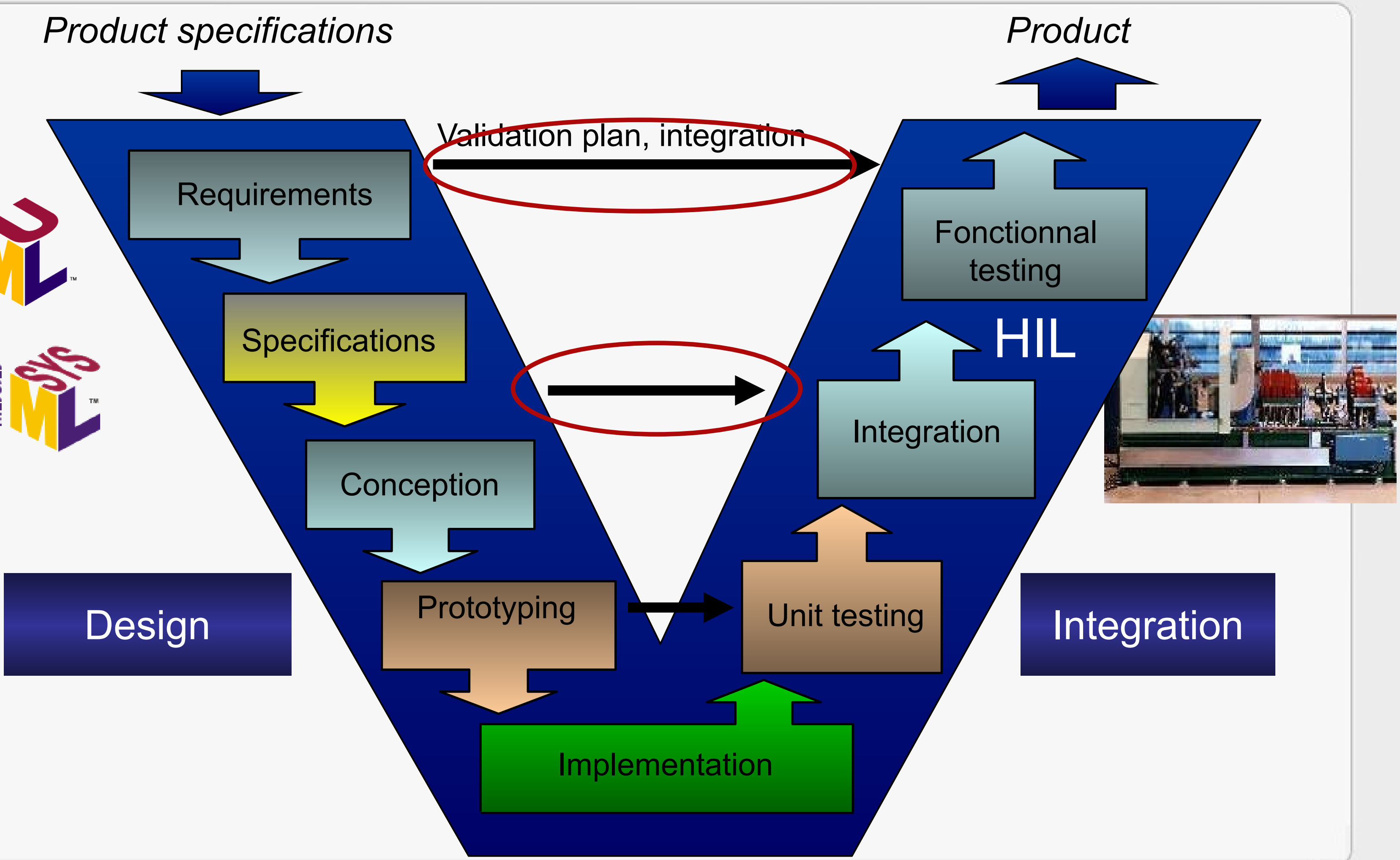
Fonctionnal testing

HIL

Design

Integration

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

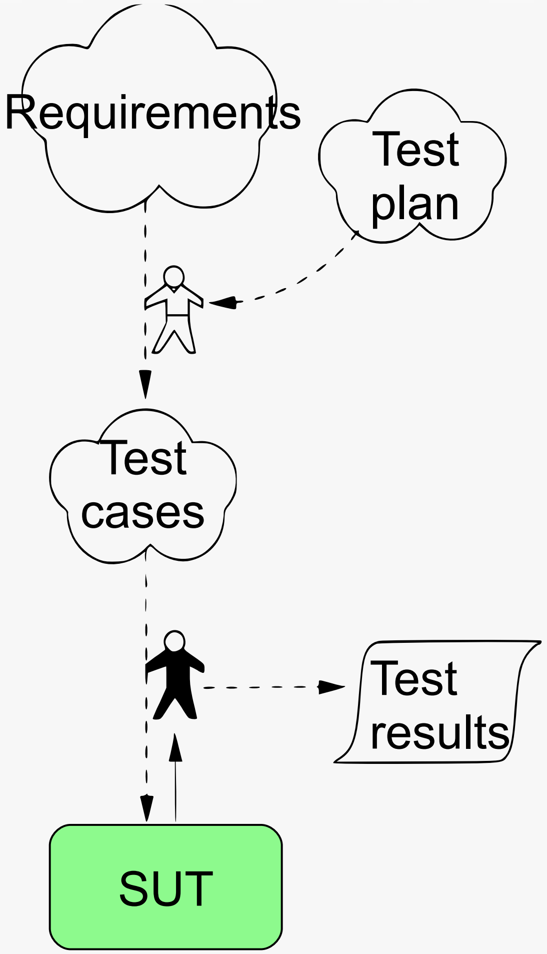
- Static verification
  - Code reviews
  - Model checking / proof
- Dynamic verification
  - System stimulation and behavior control
    - Can’t cover all possible cases
      - Most representative test cases have to be selected
    - Often more expensive that the system itself
      - Execute tests
      - 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 ?

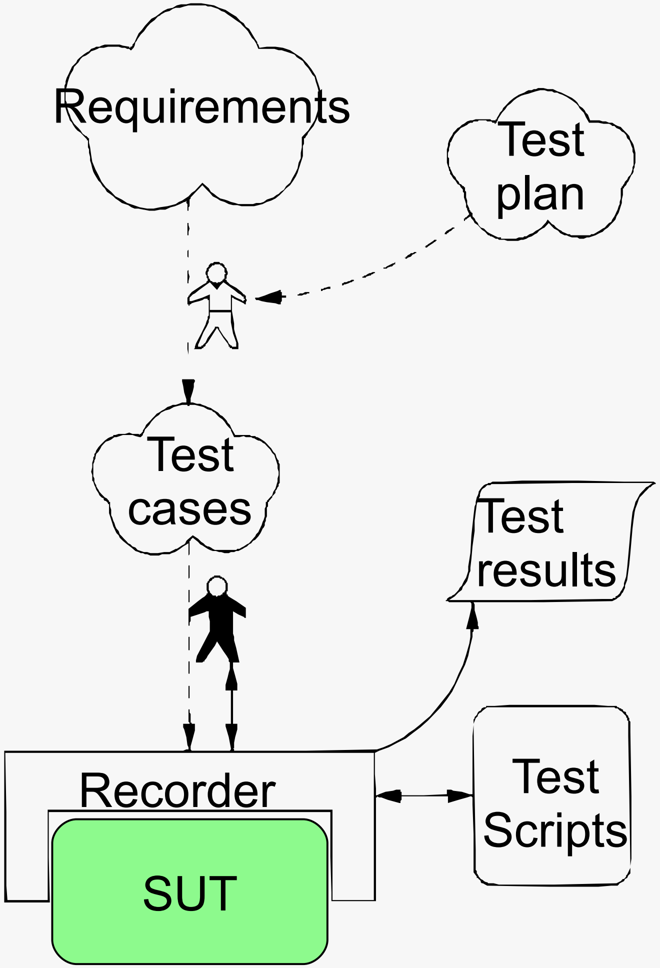
Test approaches

- Manual approach
  - Scenarios describe system stimulation and responses
  - Human execute scenarios and check results



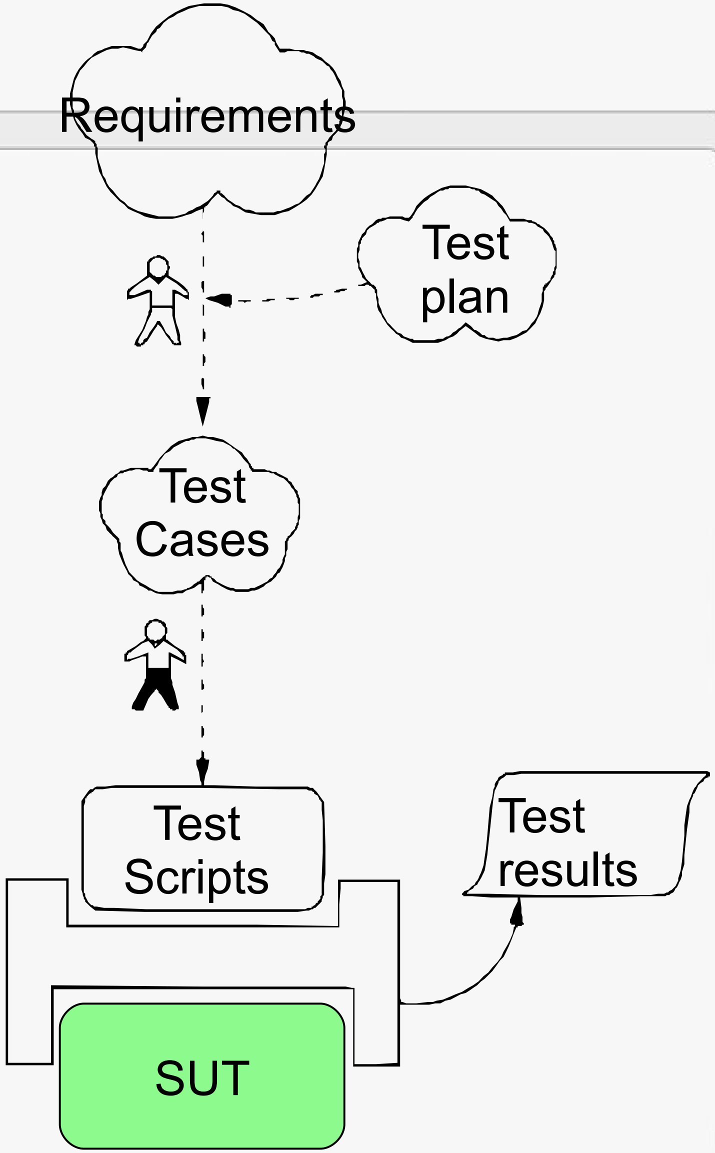
Test approaches

- Capture/replay approach
  - Stimuli and responses are recorded and can be replayed automatically



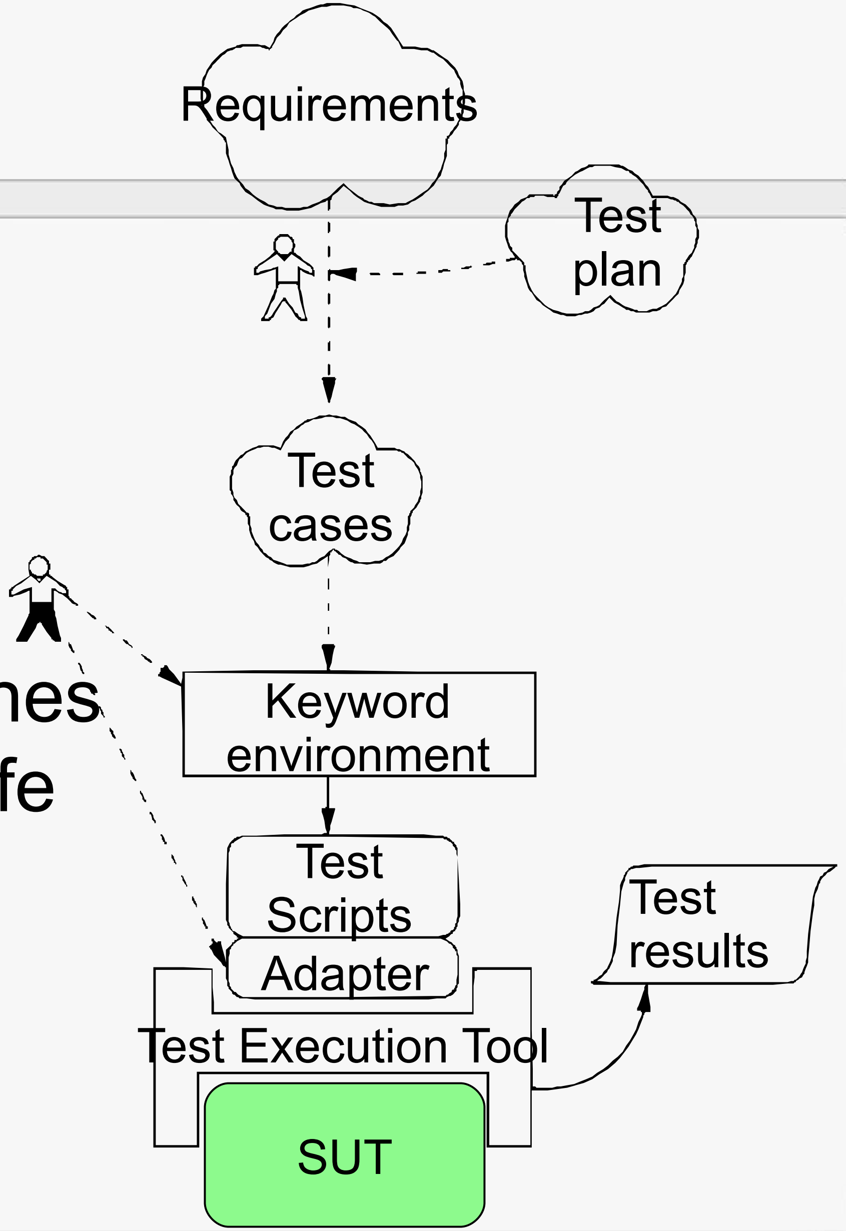
Test approaches

- Scripted approach
  - Scripts are written
    - JUnit, TestNG,…



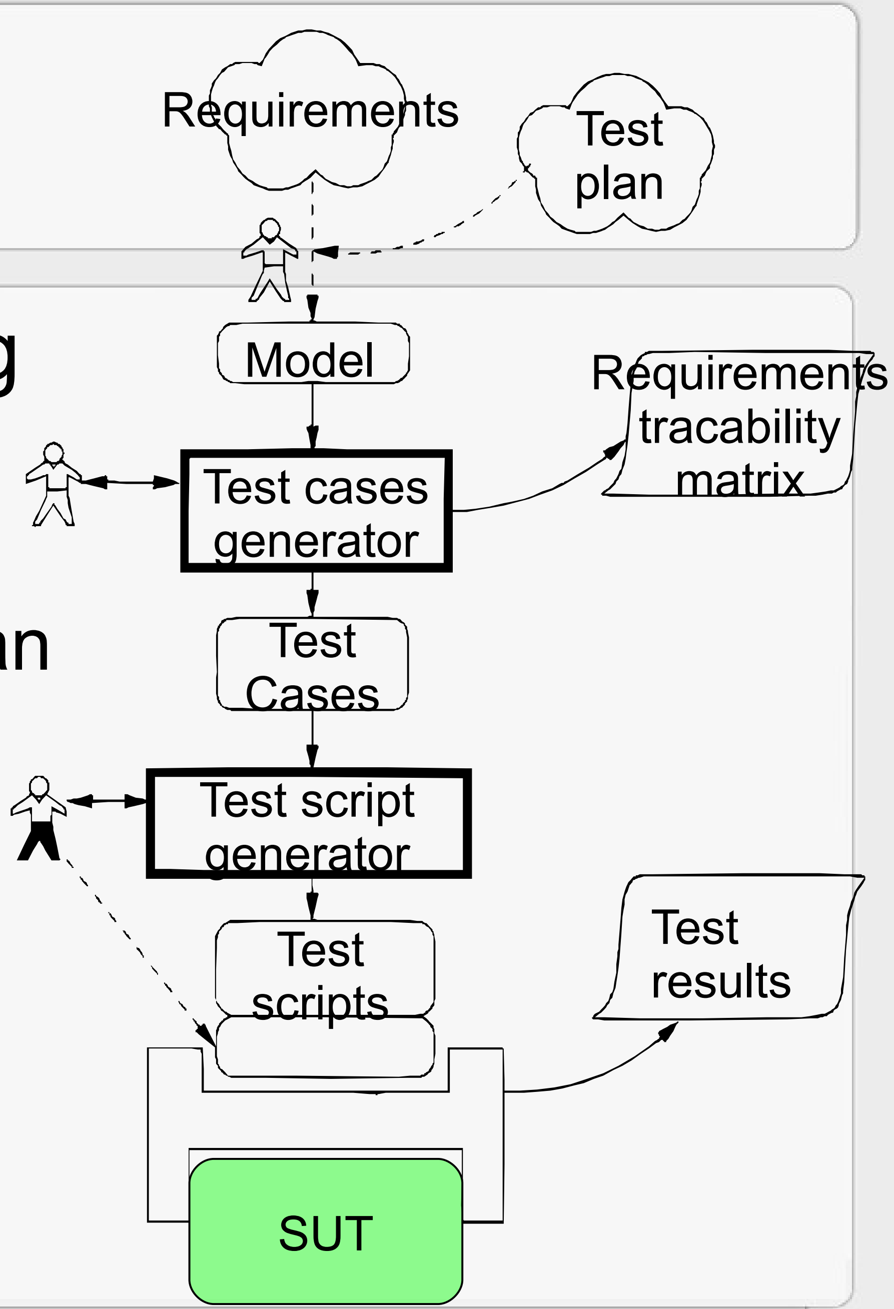
Test approaches

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



Test approaches

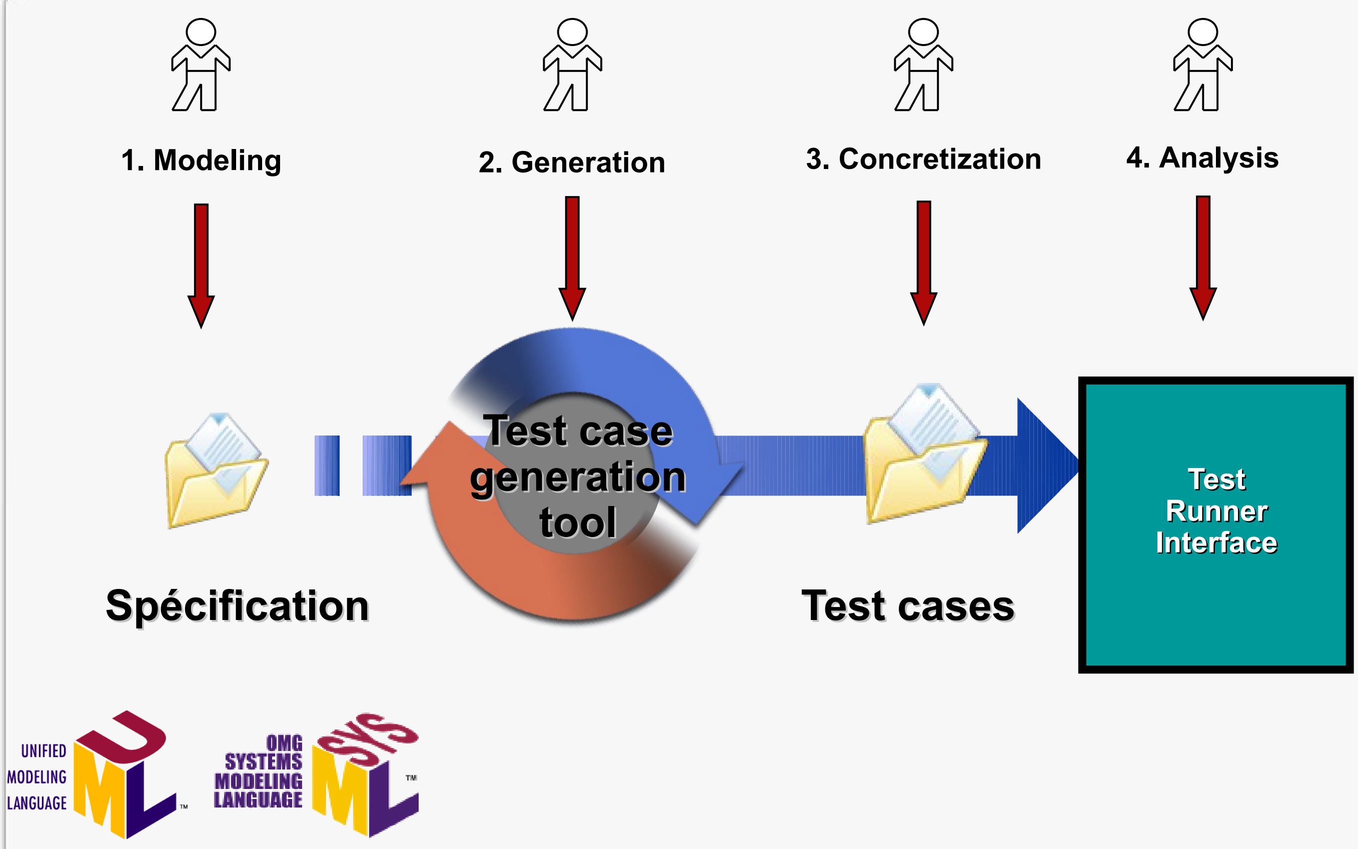
- Model Based Testing
  - A large number of keyword-based tests are generated from an executable model



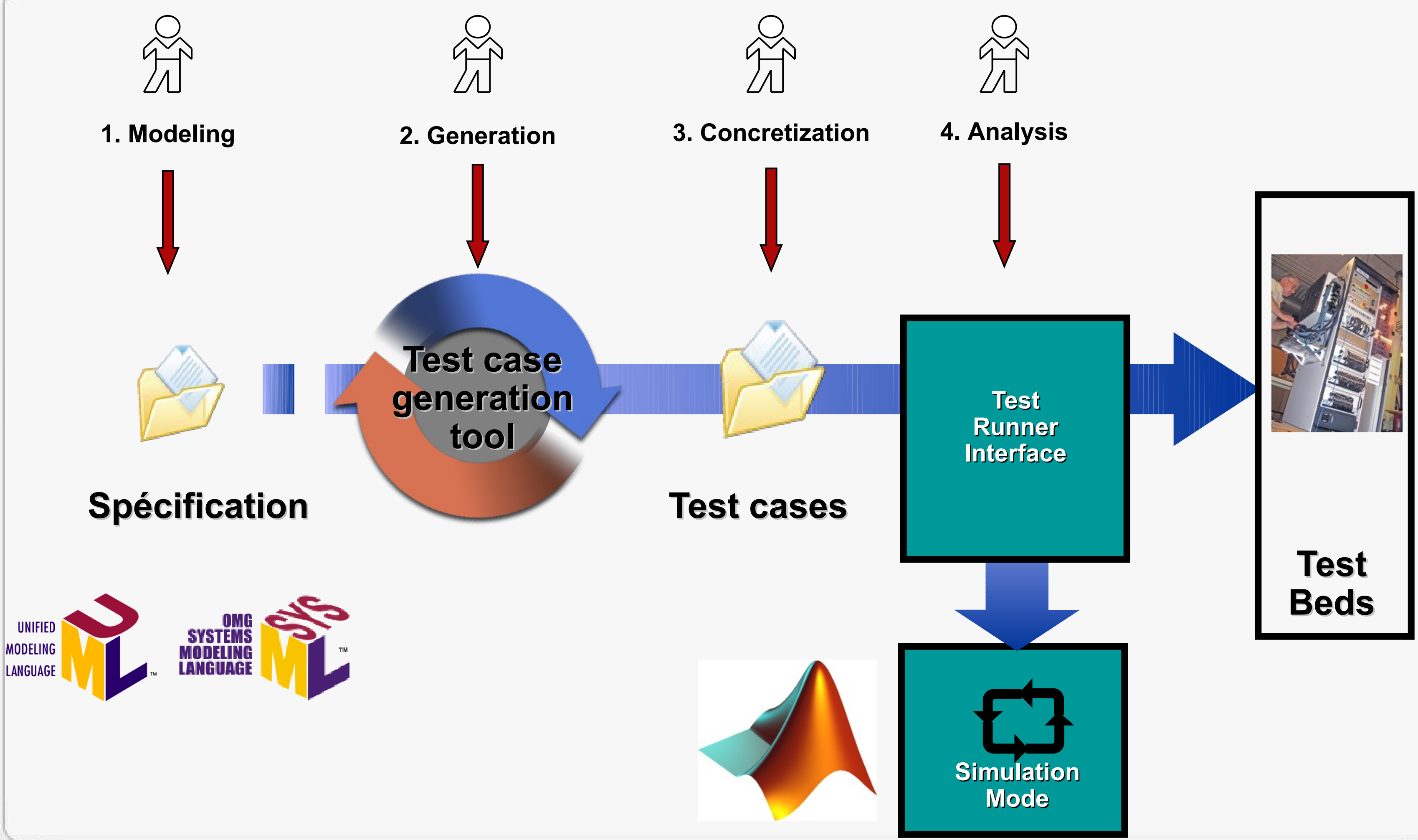
Advantages

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

Model Based Testing



Simulation vs. test beds



1. Modeling

2. Generation

3. Concretization

4. Analysis

Test case generation tool

Spécification

Test cases

Test Runner Interface

Simulation Mode

Test Beds

Plan

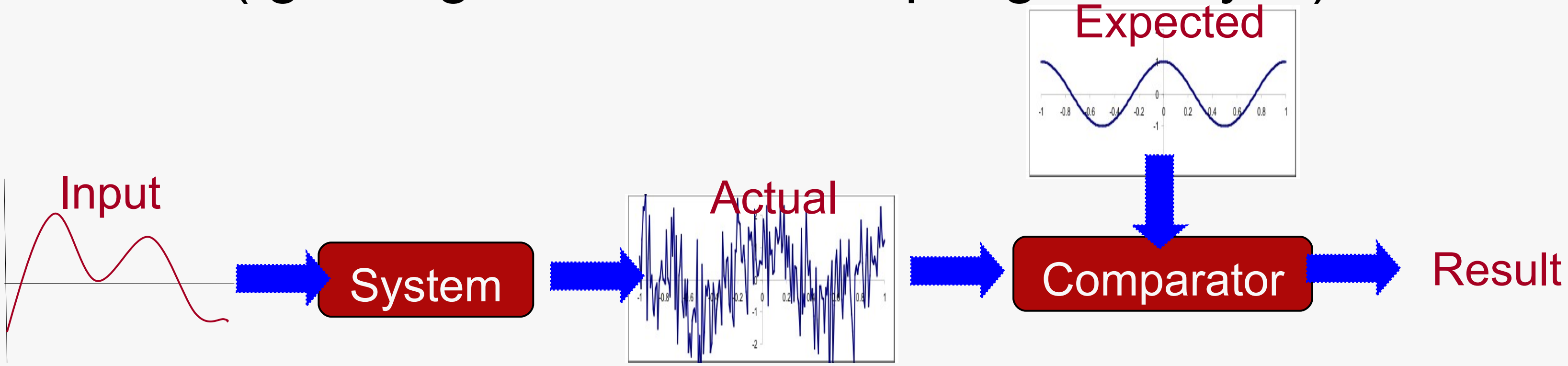
- Test in system engineering
- Model Based Testing
- The VETESS tool chain
- The front wiper case study
- Outlook

Test Designer

- Model-based testing tool for discrete systems
  - The model is a behavioral specification of the system
    - UML model or SysML models
  - Test can be exported
    - Keyword
    - JUnit
    - HP Quality Center

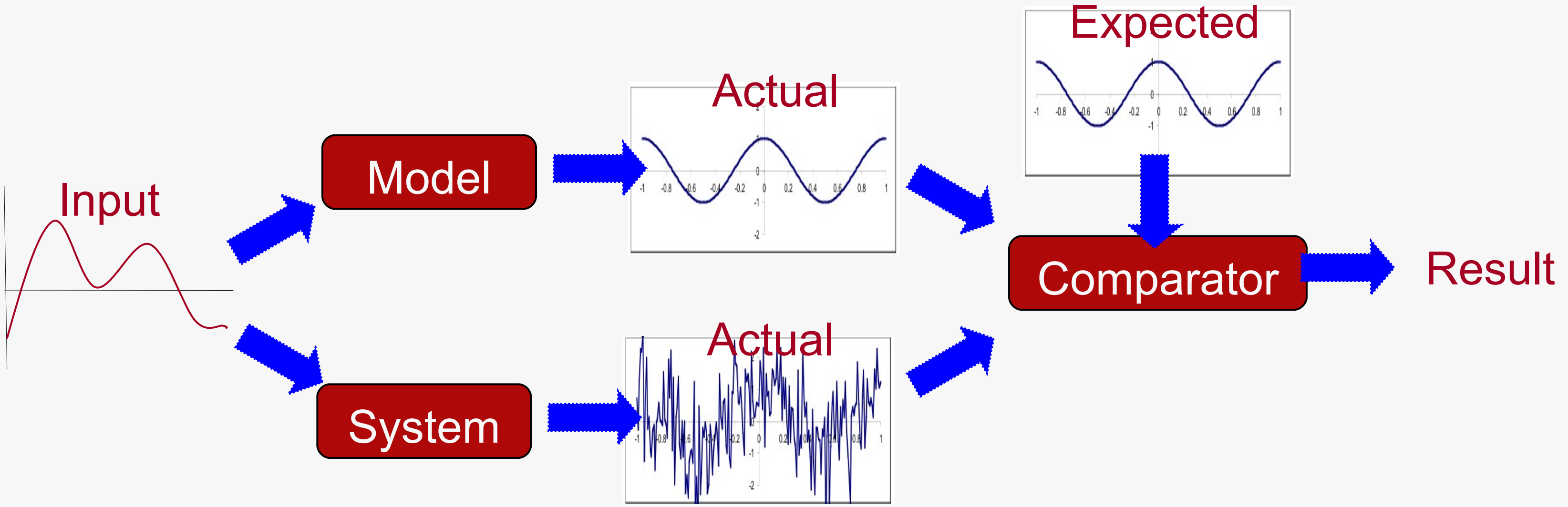
Test-In-View

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

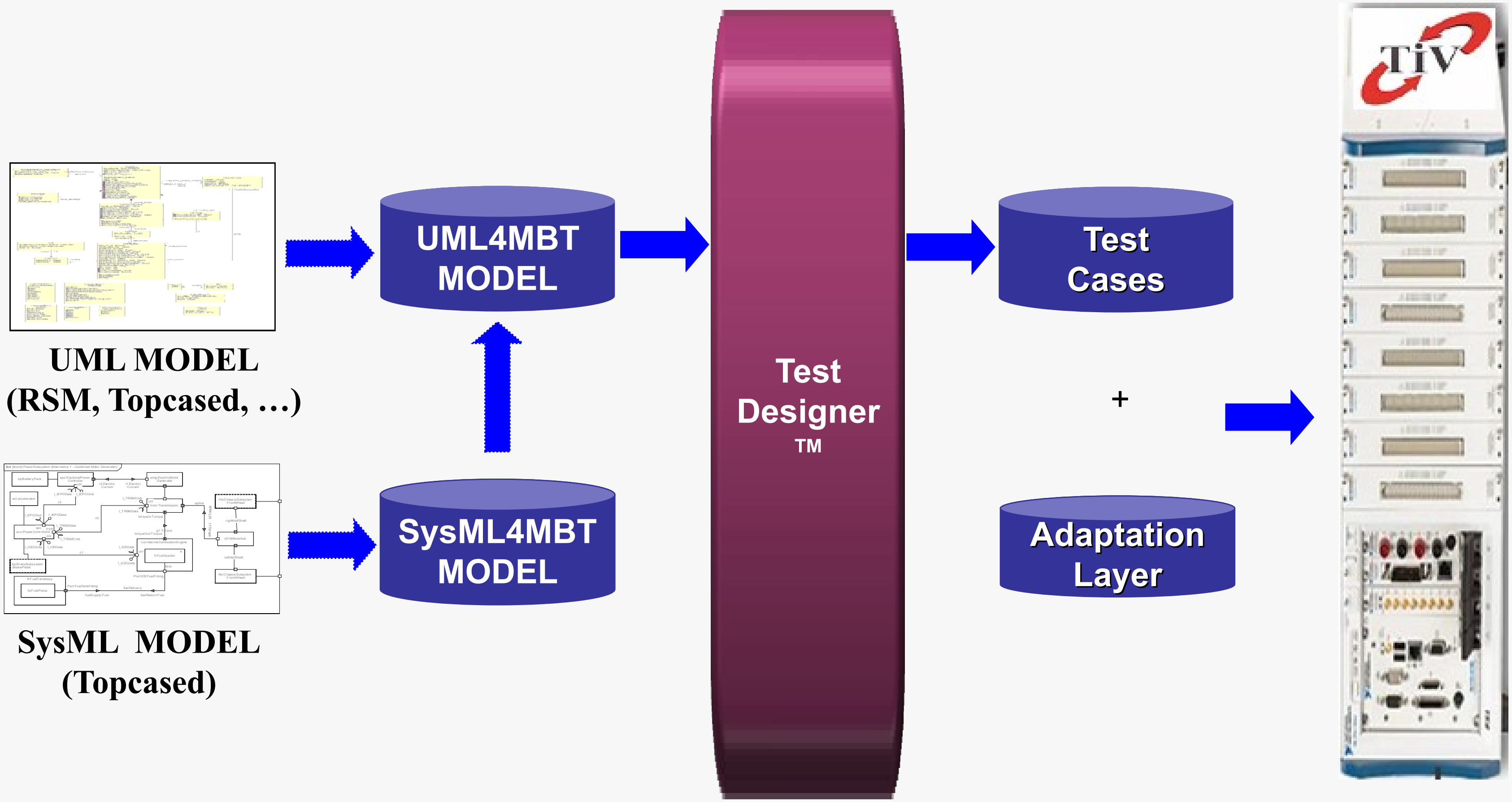


Test-In-View

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



Tool chain



UML MODEL
(RSM, Topcased, …)

SysML MODEL
(Topcased)

UML4MBT
MODEL

SysML4MBT
MODEL

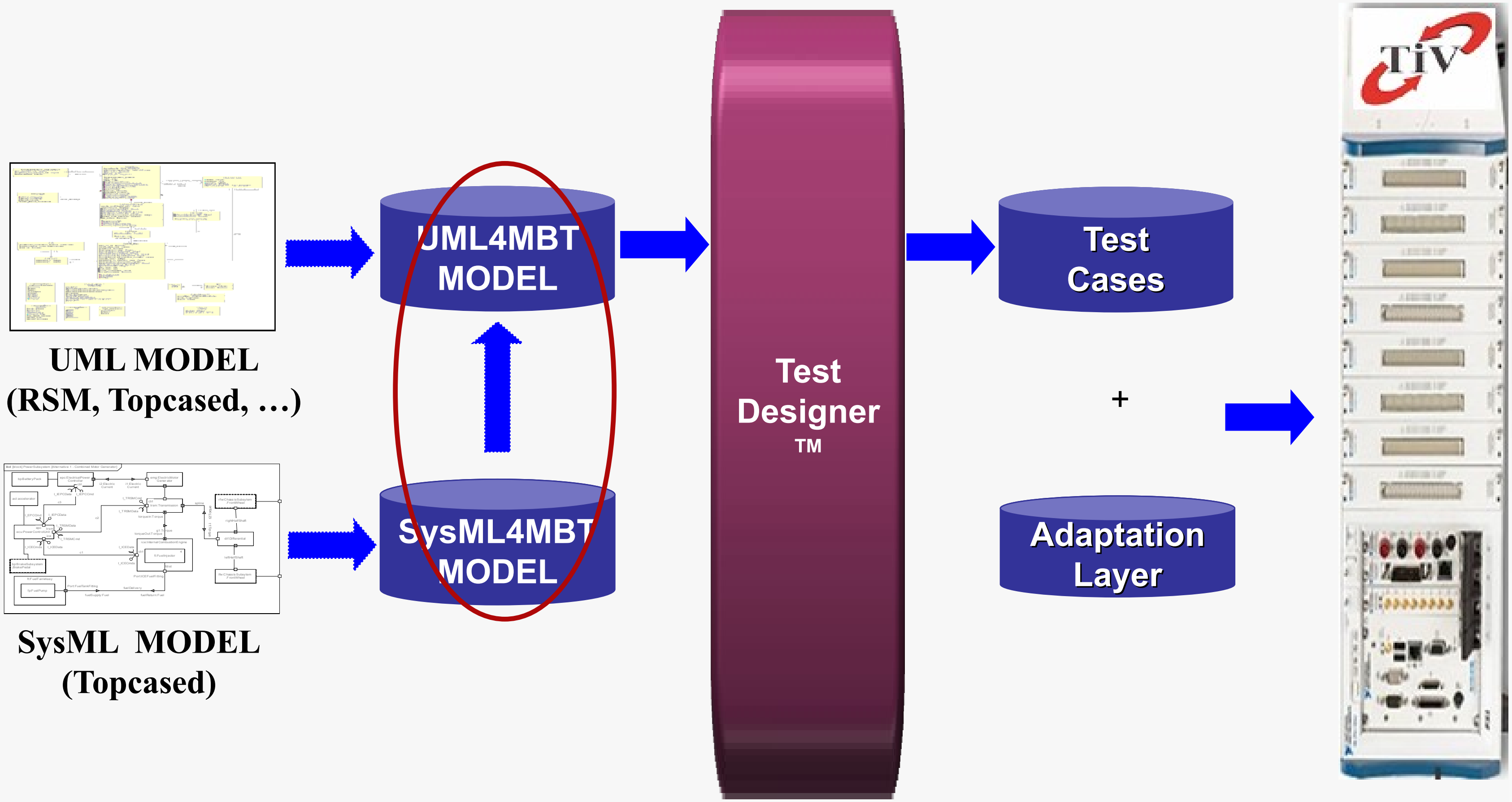
Test
Designer
TM

Test
Cases

+

Adaptation
Layer

Tool chain



UML MODEL
(RSM, Topcased, …)

SysML MODEL
(Topcased)

UML4MBT
MODEL

SysML4MBT
MODEL

Test
Designer
TM

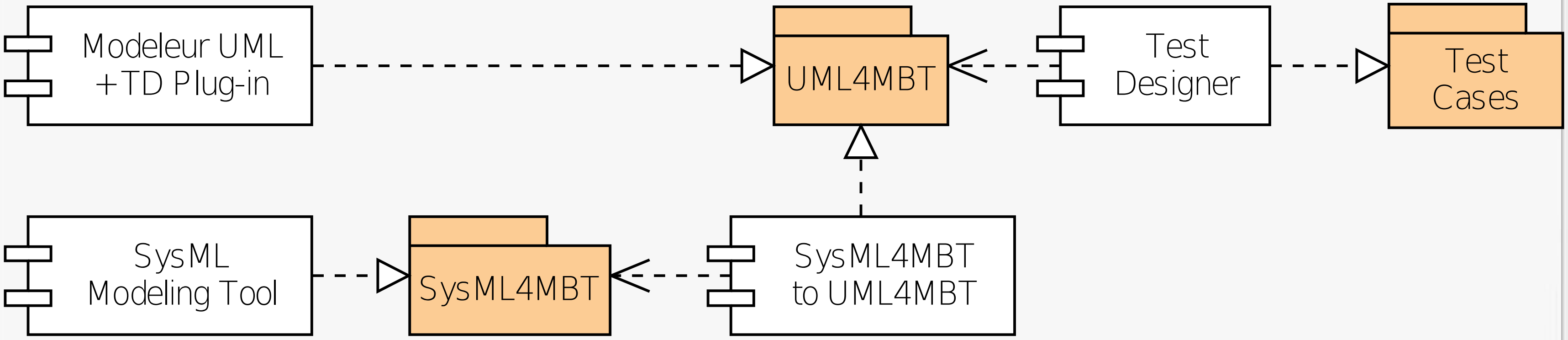
Test
Cases

+

Adaptation
Layer

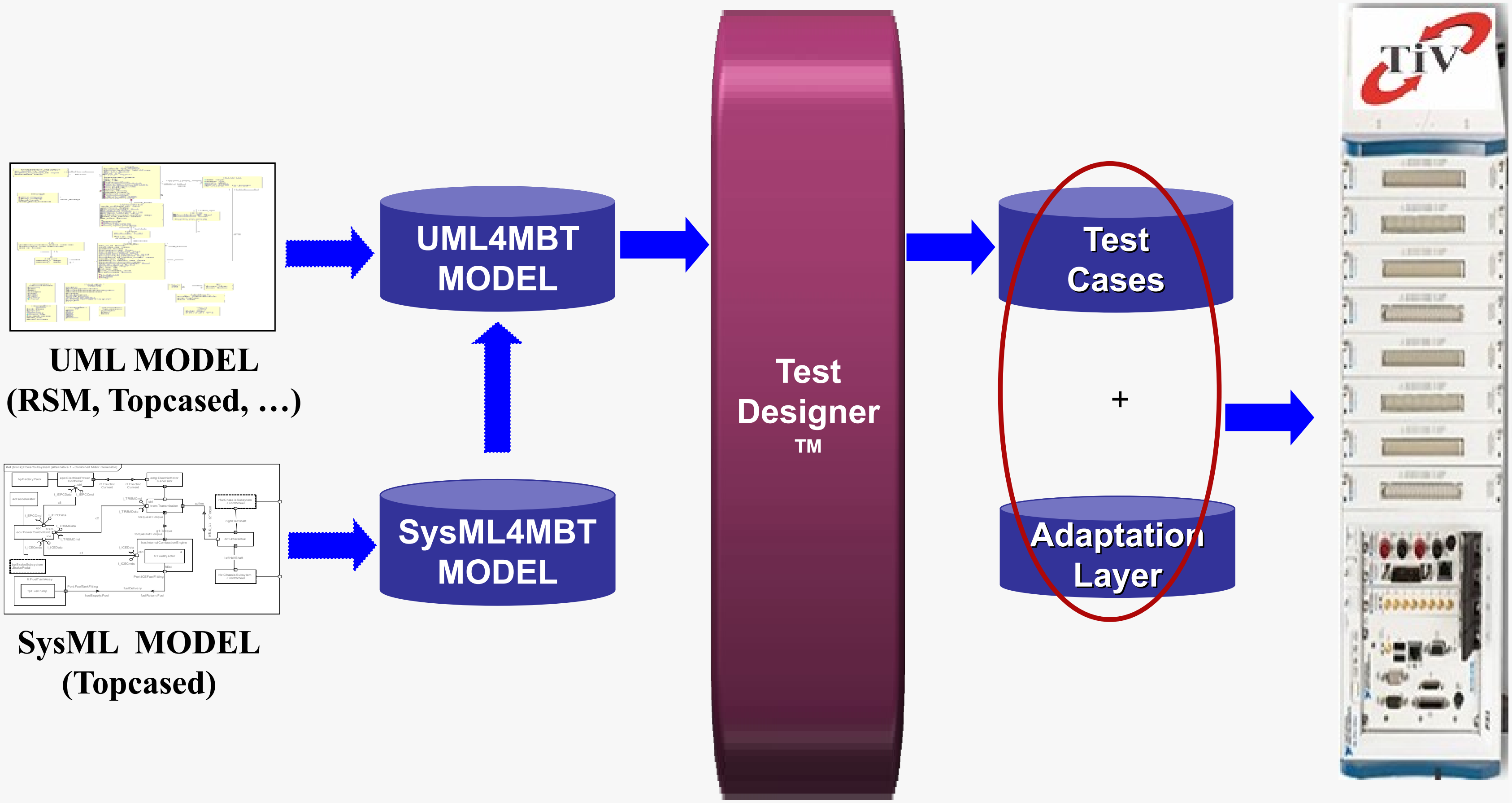
UML4MBT / SysML4MBT

- 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



- Other languages can be supported through model transformation

Tool chain



UML MODEL
(RSM, Topcased, …)

SysML MODEL
(Topcased)

UML4MBT MODEL

SysML4MBT MODEL

Test Designer TM

Test Cases

+

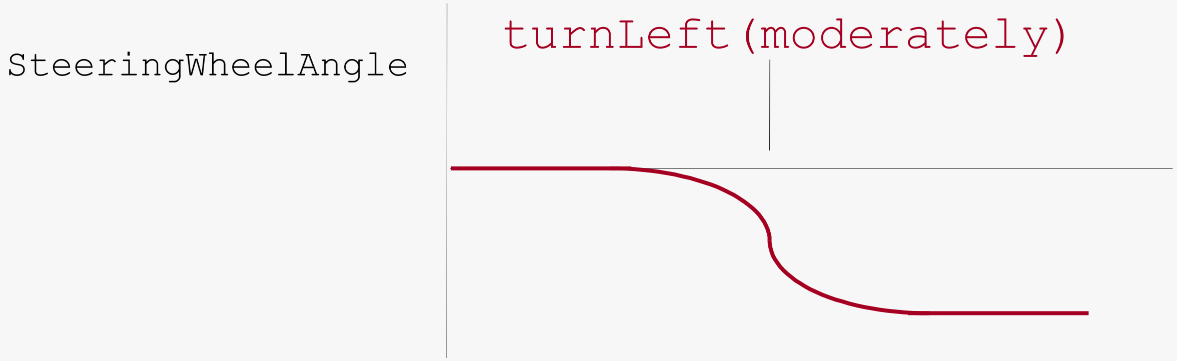
Adaptation Layer

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



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

Plan

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