D-MINT
Deployment of Model-Based
Technologies to Industrial Testing

TUTORIAL: *TTCN-3 and its role and usage in MBT from the D-MINT perspective*

Thomas Bauer, Axel Rennoch
Fraunhofer IESE & FOKUS, Germany

Presentation Overview

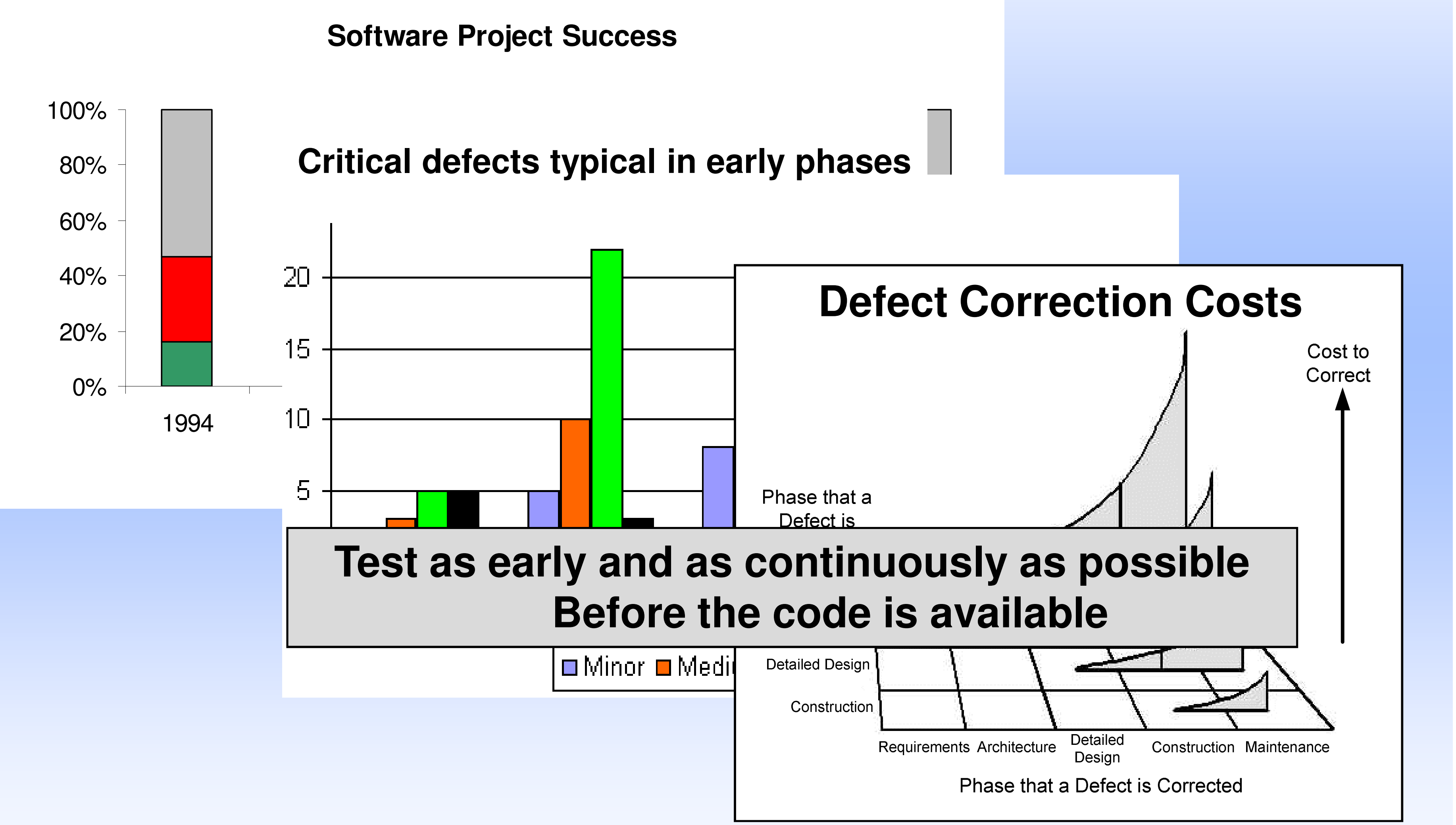
- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Current Situation for Software-Based Systems

Number of successful software projects still less than 1/3



Software Project Success

Critical defects typical in early phases

Defect Correction Costs

Test as early and as continuously as possible
Before the code is available

Example: Ariane 5

- Ariane 5 Flight 501 on 4 June 1996 failed
- Weight: 740 t, Payload: cluster satellites
- Rocket self-destructing 37 seconds after launch because of a malfunction in the control software
- Most expensive computer bug in history: **370 Mio $**
- Causes
  - Reused software from Ariane 4
  - Data conversion from 64-bit float to 16-bit signed integer → overflow / not caught
  - ADA software with 2 channels (redundancy), but identifical implementation!
  - 1st channel had same problem 72ms before
  - Software handler got exceptions from both channels, no Plan B for such situations
  - Main computer interpreted horizontal velocity and sent strange control command
  - Self-destruction due to safety issues

ADA Code of 2nd channel

```
...
declare
  vertical_veloc_sensor: float;
  horizontal_veloc_sensor: float;
  vertical_veloc_bias: integer;
  horizontal_veloc_bias: integer;
  ...
begin
  declare
    pragma suppress(numeric_error,
horizontal_veloc_bias);
  begin
    sensor_get(vertical_veloc_sensor);
    sensor_get(horizontal_veloc_sensor);
    vertical_veloc_bias :=
integer(vertical_veloc_sensor);
    horizontal_veloc_bias :=
integer(horizontal_veloc_sensor);
    ...
  exception
    when numeric_error => calculate_vertical_veloc();
    when others => use_irs1();
  end;
end irs2;
.
```

Horizontal velocity > 32786.0 internal unit

Unclassified Exception caught → Control transfer to 1st channel

* source: http://www-aix.gsi.de/~giese/swr/ariane5.html

Make Sure You Have the Right Method, Technology, Tool!

Borrowed from M. Berglund, Ericsson, T3UC 2007

Make Sure You Have the Right Method, Technology, Tool!

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Make Sure You Have the Right Method, Technology, Tool!

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Make Sure You Have the Right Method, Technology, Tool!

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Make Sure You Have the Right Method, Technology, Tool!

Make Sure You Have the Right Method, Technology, Tool!

Make Sure You Have the Right Method, Technology, Tool!

Make Sure You Have the Right Method, Technology, Tool!

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Make Sure You Have the Right Method, Technology, Tool!

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Make Sure You Have the Right Method, Technology, Tool!

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Model-based Testing

- **Model-based testing = test generation from models**
- *”Model-based testing* is concerned with comparing models with realizations using automatically generated and executed test cases.” – Tretmans
- ”Model-based testing is a variant of testing that relies on explicit behaviour models that encode the intended behaviour of a system and possibly the behaviour of its environment.” – Utting, Pretschner, Legeard

- **Data Models / Input domain models**
  - System structure/interface models
  - e.g. equivalence class partitioning

- **System behavior models**
  - e.g. state machines
  - can be used as test oracles

- **Environment Models**
  - (Probabilistic) descriptions of the stimulation by the system environment
  - e.g. Markov chains

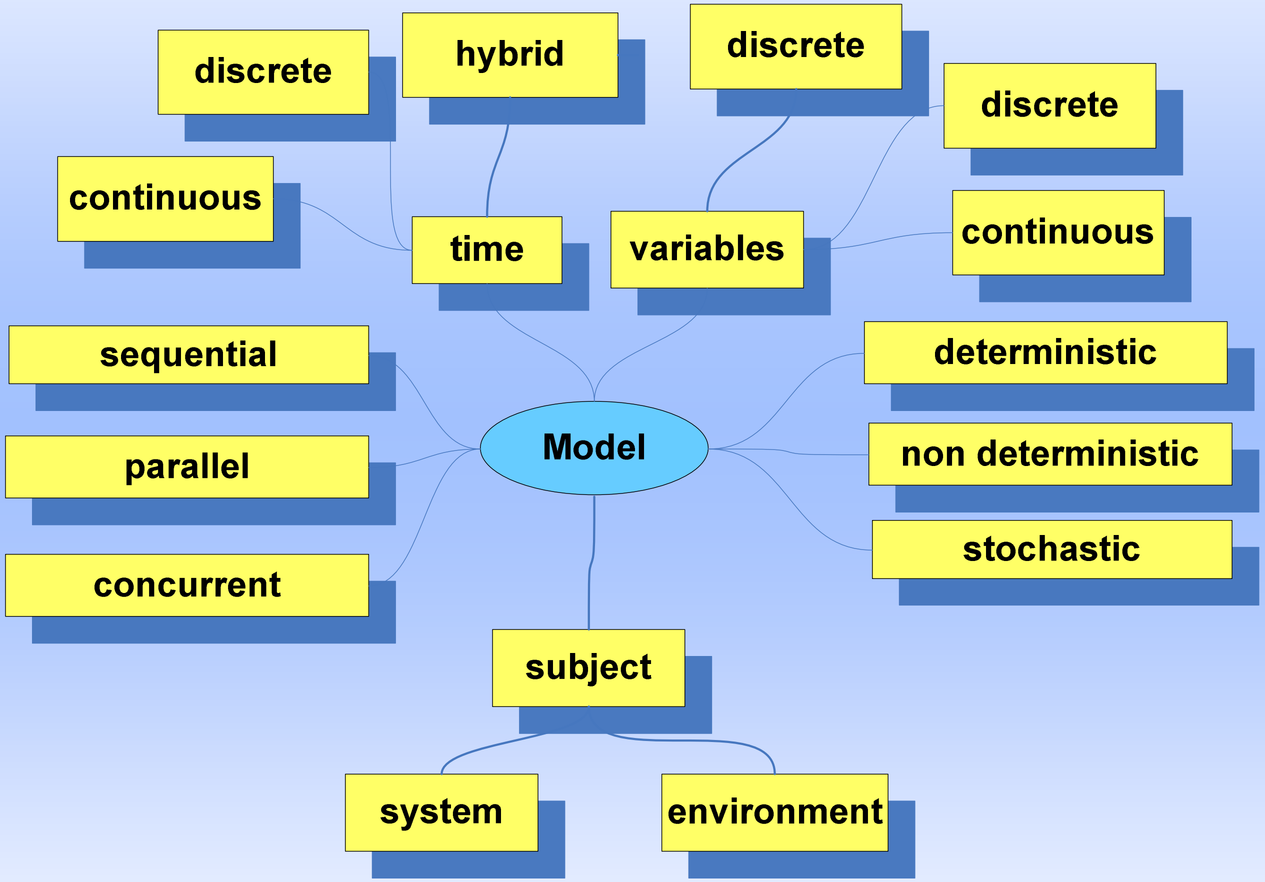
Test automation

•Test generation

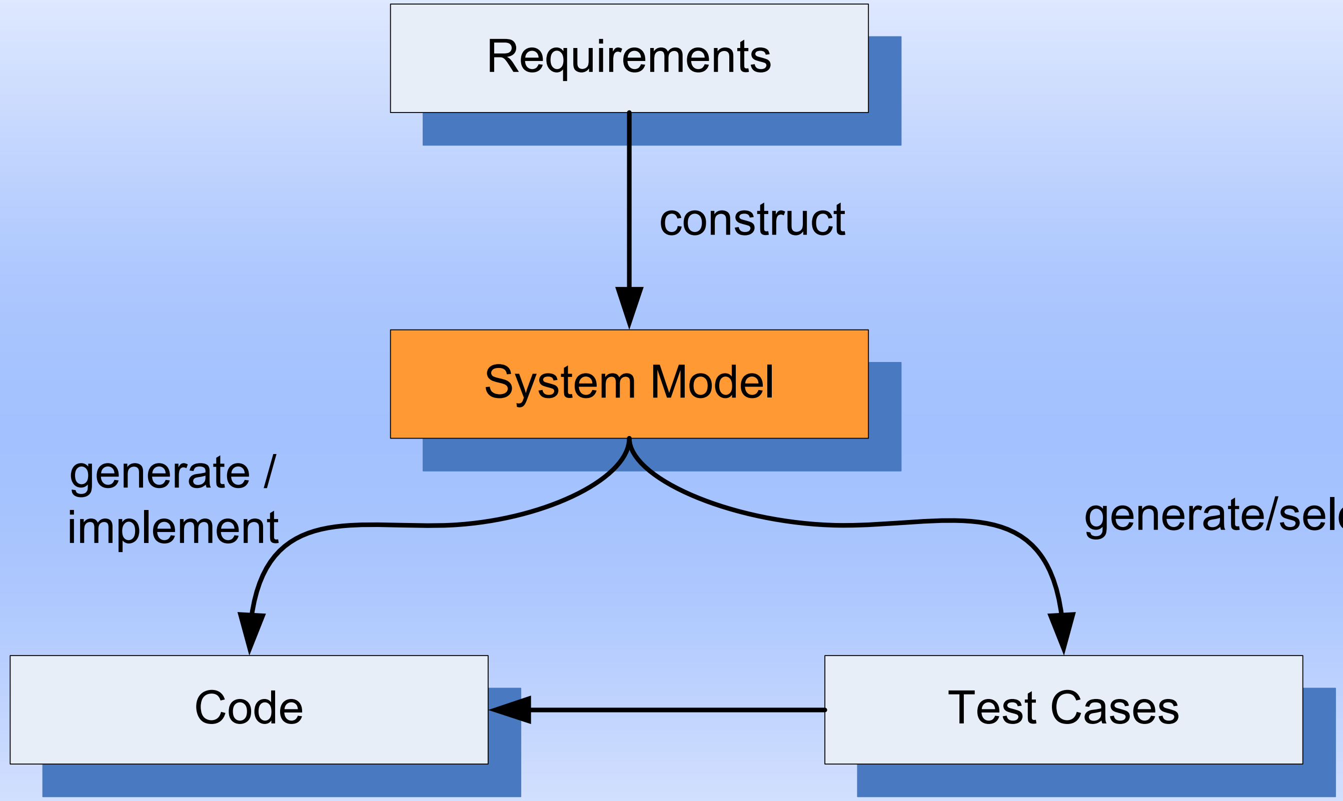
•Test execution (platform)

•Test evaluation

Model: Characteristics



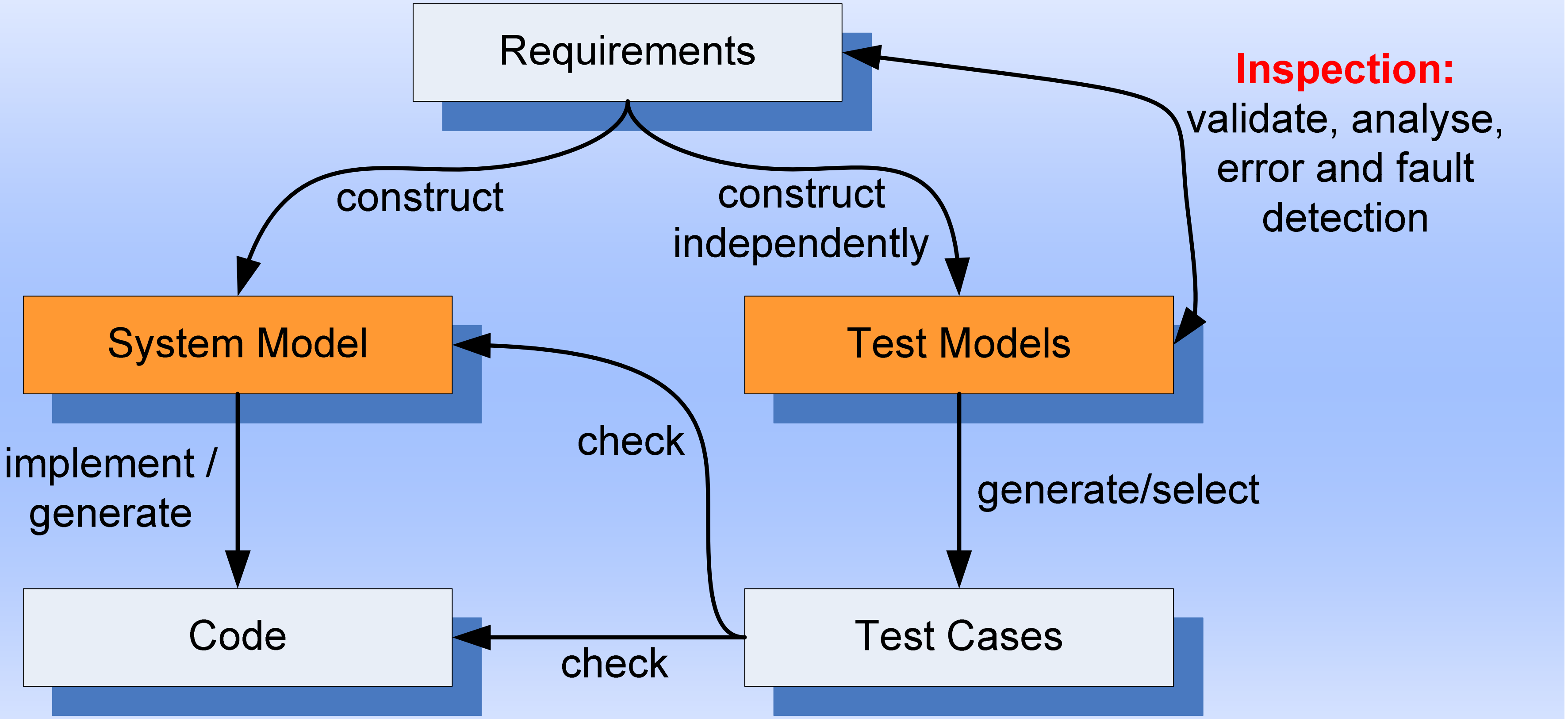
Model: Reuse existing Development Models



Check code generator, test case generator, environment assumptions

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Model: Construct separate Test Models



Check system model and code against requirements

Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Model-based Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Make Sure You Have the Right Tool!

→ A test modelling and test implementation language

Test Technology

- The Testing and Test Control Notation

- A *standardized alternativ* to propri y test systems
  - Develope by a large gr *tes* *rts*
  - Used by a ing *com*
  - Proven by
  - Maintain

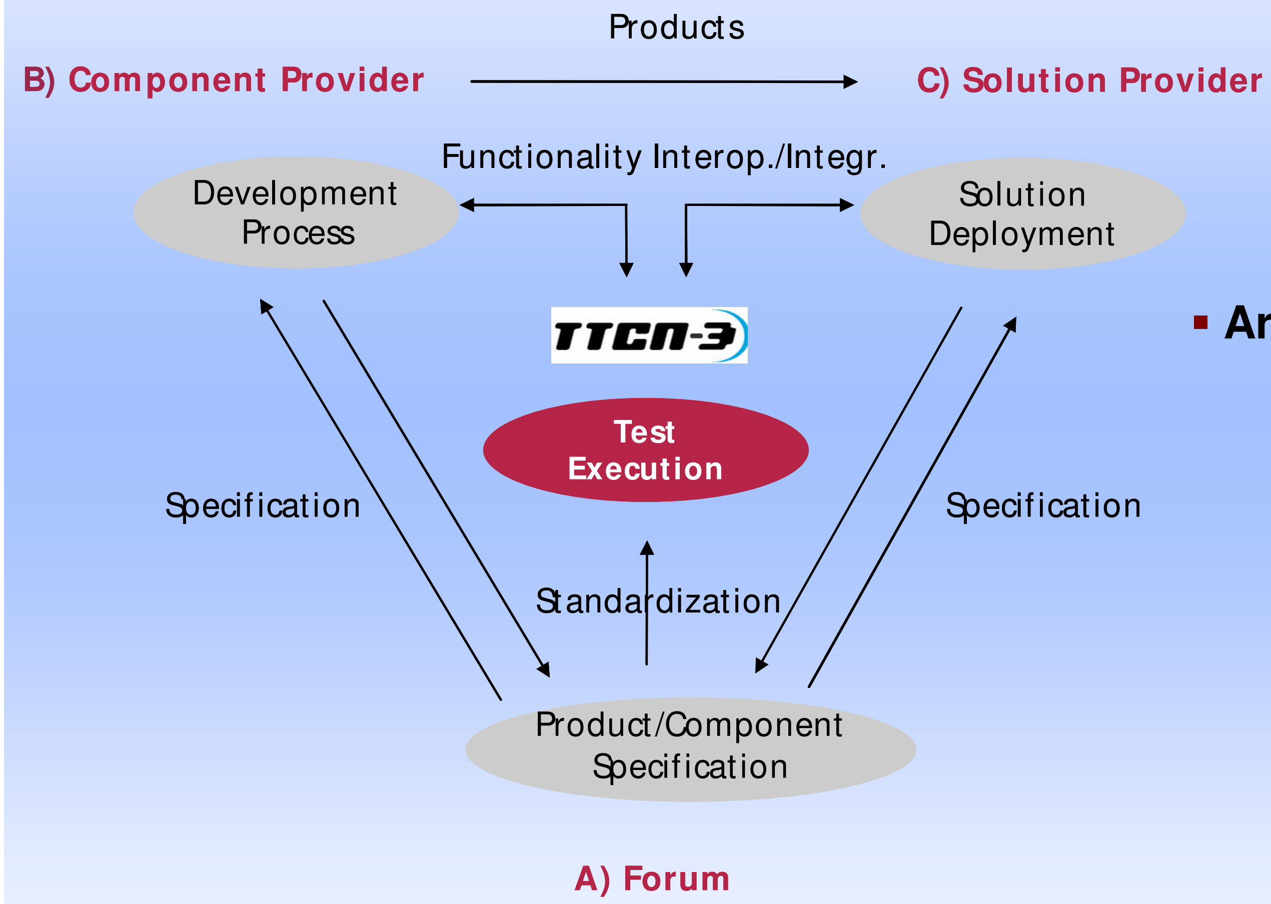
- A test s
  - A
    - ✓ g
    - ✓ *lang* CN .g. for IDL

Enabling a *testing middleware*

- **unifying** methods, tools,
  test infrastructure,
  documentation & training

- by **domain-specific** profiles

Placement of TTCN-3

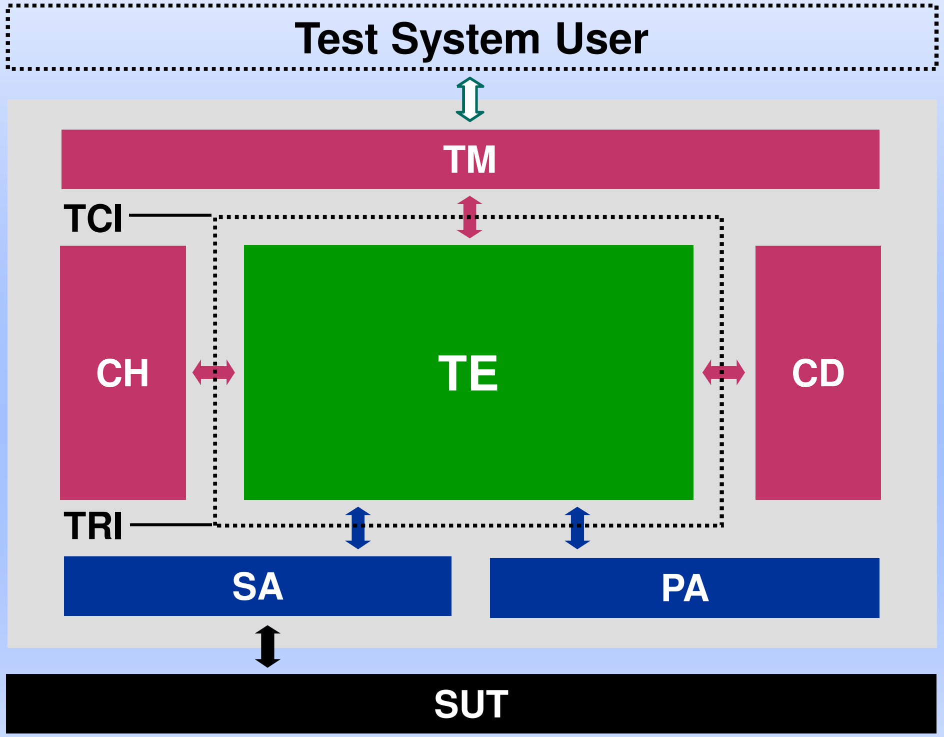


- **Areas of Testing**
  - Regression Testing
  - Conformance/Functionality Testing
  - Interoperability/Integration Testing
  - Load/ Stress Testing

Main Capabilities of TTCN-3

- Dynamic concurrent test configurations
- Synchronous and asynchronous communication mechanisms
- Encoding information
- Data and signature templates with powerful matching mechanism
- Assignment and handling of test verdicts
- Testcase selection mechanisms
- Test suite and test data parameterization

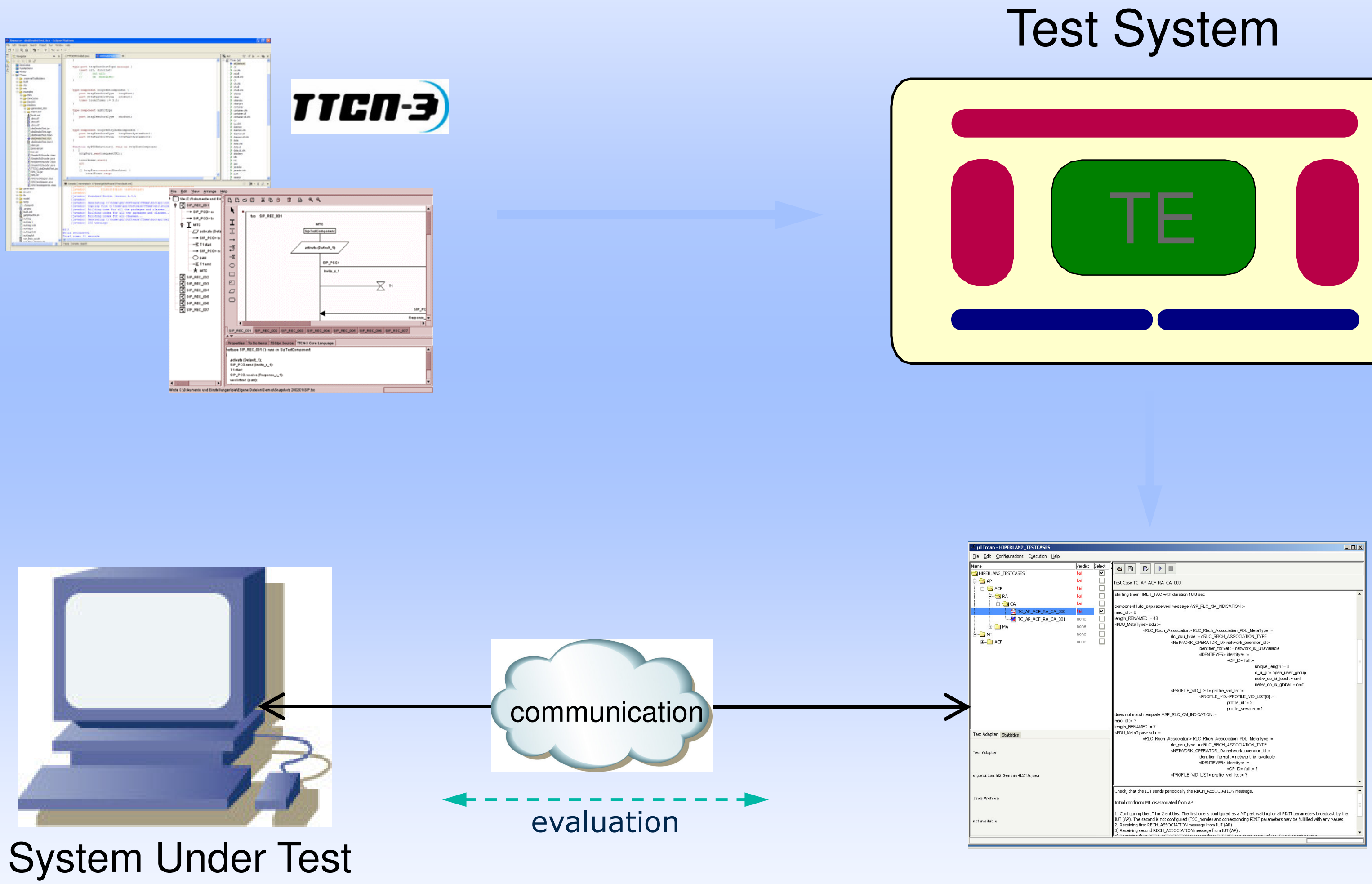
A TTCN-3 Test System



TE	–	TTCN-3 Executable
SA	–	System Adapter
PA	–	Platform Adapter
CD	–	Codec
TM	–	Test Management
CH	–	Component Handling
SUT	–	System Under Test

ETSI ES 201 873-1	TTCN-3 Core Language (CL)
ETSI ES 201 873-5	TTCN-3 Runtime Interface (TRI)
ETSI ES 201 873-6	TTCN-3 Control Interfaces (TCI)

Test Execution with TTCN-3



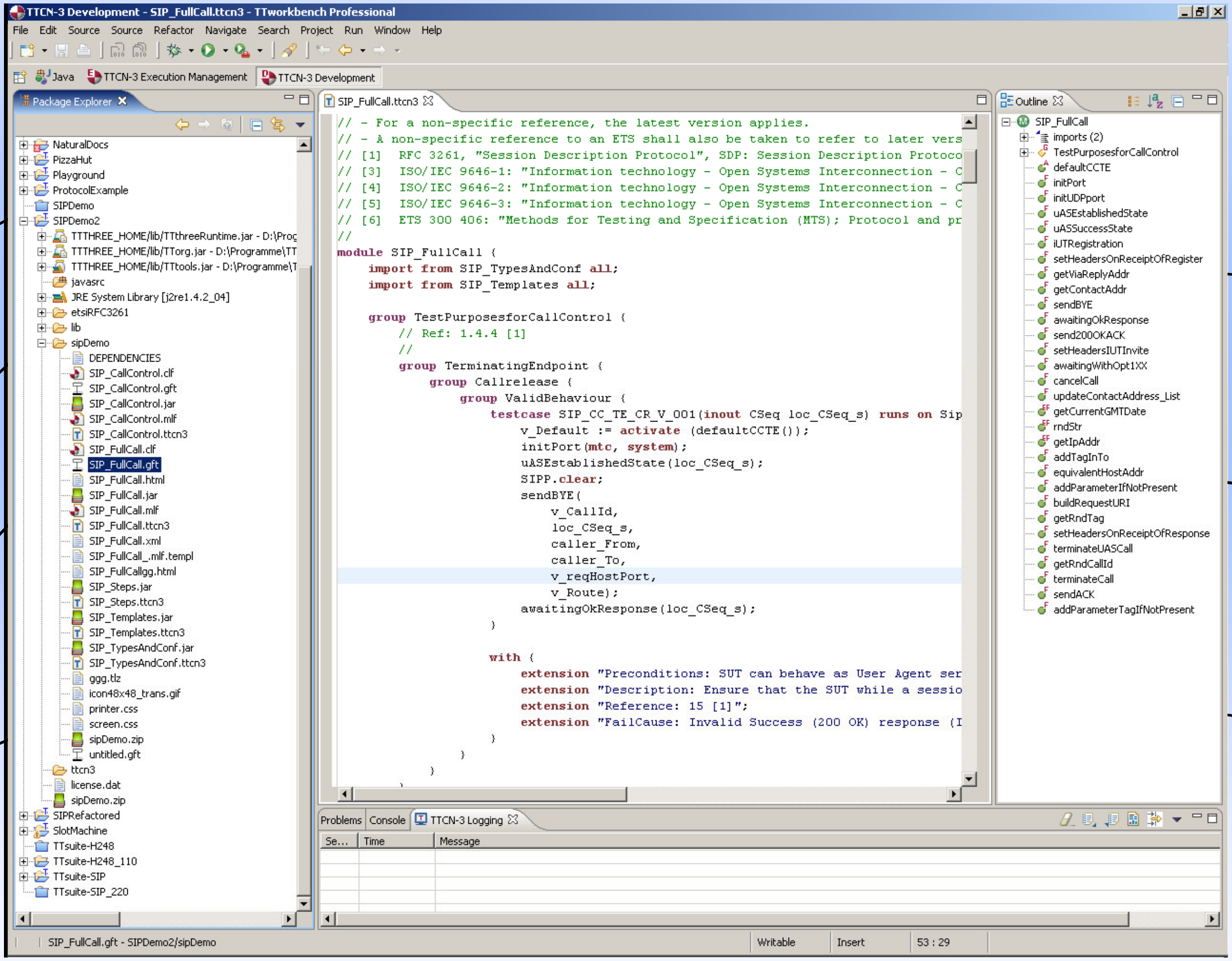
An Impression of TTCN-3 Tooling

Developers Perspective for Modification

Test Execution

Test Campaign Designer (Test Automation)

Test Parametrization



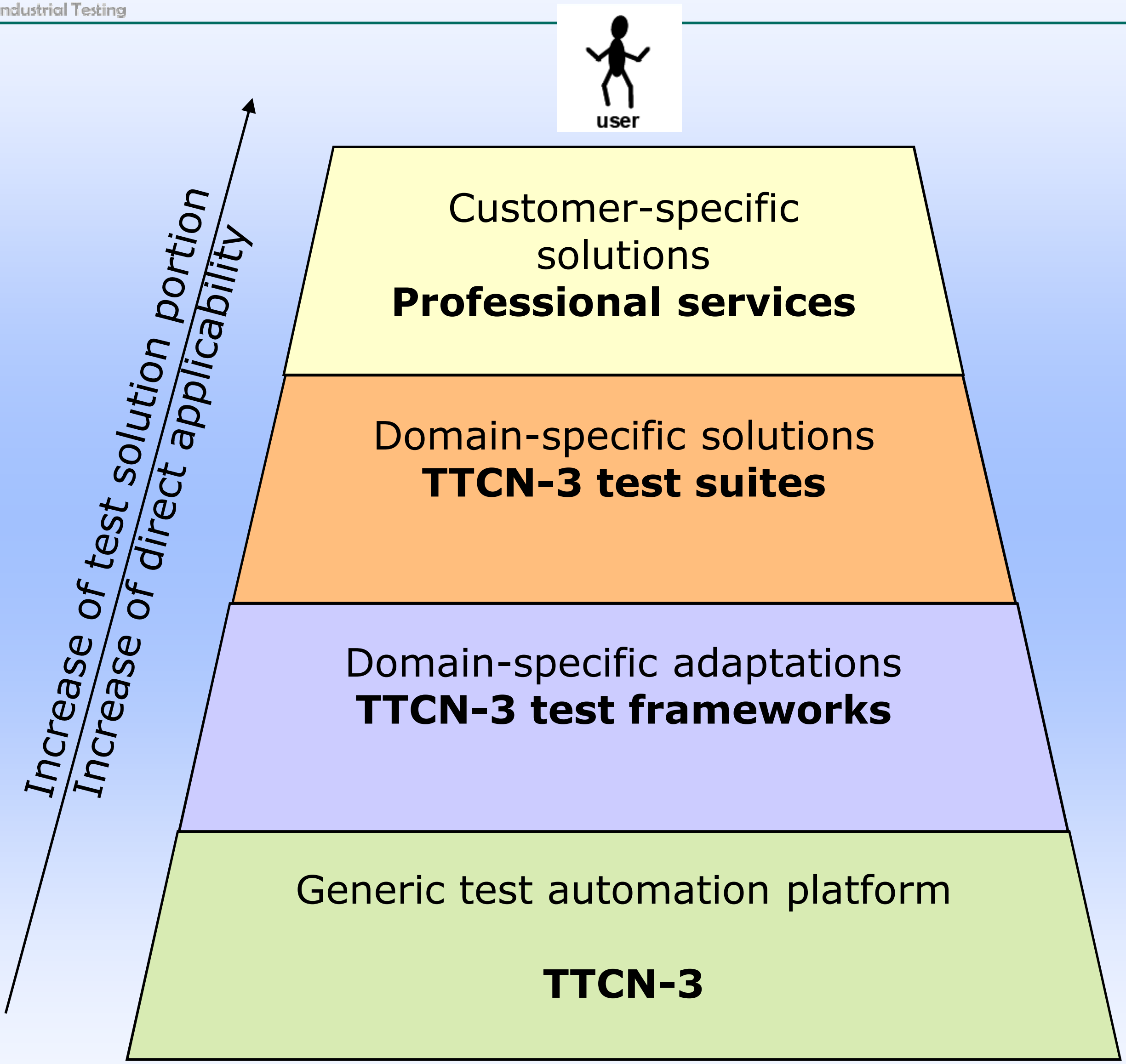
```
// - For a non-specific reference, the latest version applies.
// - A non-specific reference to an ETS shall also be taken to refer to later vers
// [1]  RFC 3261, "Session Description Protocol", SDP: Session Description Protoco
// [3]  ISO/IEC 9646-1: "Information technology - Open Systems Interconnection - C
// [4]  ISO/IEC 9646-2: "Information technology - Open Systems Interconnection - C
// [5]  ISO/IEC 9646-3: "Information technology - Open Systems Interconnection - C
// [6]  ETS 300 406: "Methods for Testing and Specification (MTS); Protocol and pr
//
module SIP_FullCall {
    import from SIP_TypesAndConf all;
    import from SIP_Templates all;

    group TestPurposesforCallControl {
        // Ref: 1.4.4 [1]
        //
        group TerminatingEndpoint {
            group Callrelease {
                group ValidBehaviour {
                    testcase SIP_CC_TE_CR_V_001(inout CSeq loc_CSeq_s) runs on Sip
                        v_Default := activate (defaultCCTE());
                        initPort(mtc, system);
                        uASEstablishedState(loc_CSeq_s);
                        SIPP.clear;
                        sendBYE(
                            v_CallId,
                            loc_CSeq_s,
                            caller_From,
                            caller_To,
                            v_reqHostPort,
                            v_Route);
                        awaitingOkResponse(loc_CSeq_s);
                    }

                    with {
                        extension "Preconditions: SUT can behave as User Agent ser
                        extension "Description: Ensure that the SUT while a sessio
                        extension "Reference: 15 [1]";
                        extension "FailCause: Invalid Success (200 OK) response (I
                    }
                }
            }
```

Result Analyzer

Test Report

Online Logging, Filter, Reporting

The TTCN-3 pyramid


user
Increase of test solution portion
Increase of direct applicability

Customer-specific solutions
Professional services

Domain-specific solutions
TTCN-3 test suites

Domain-specific adaptations
TTCN-3 test frameworks

Generic test automation platform

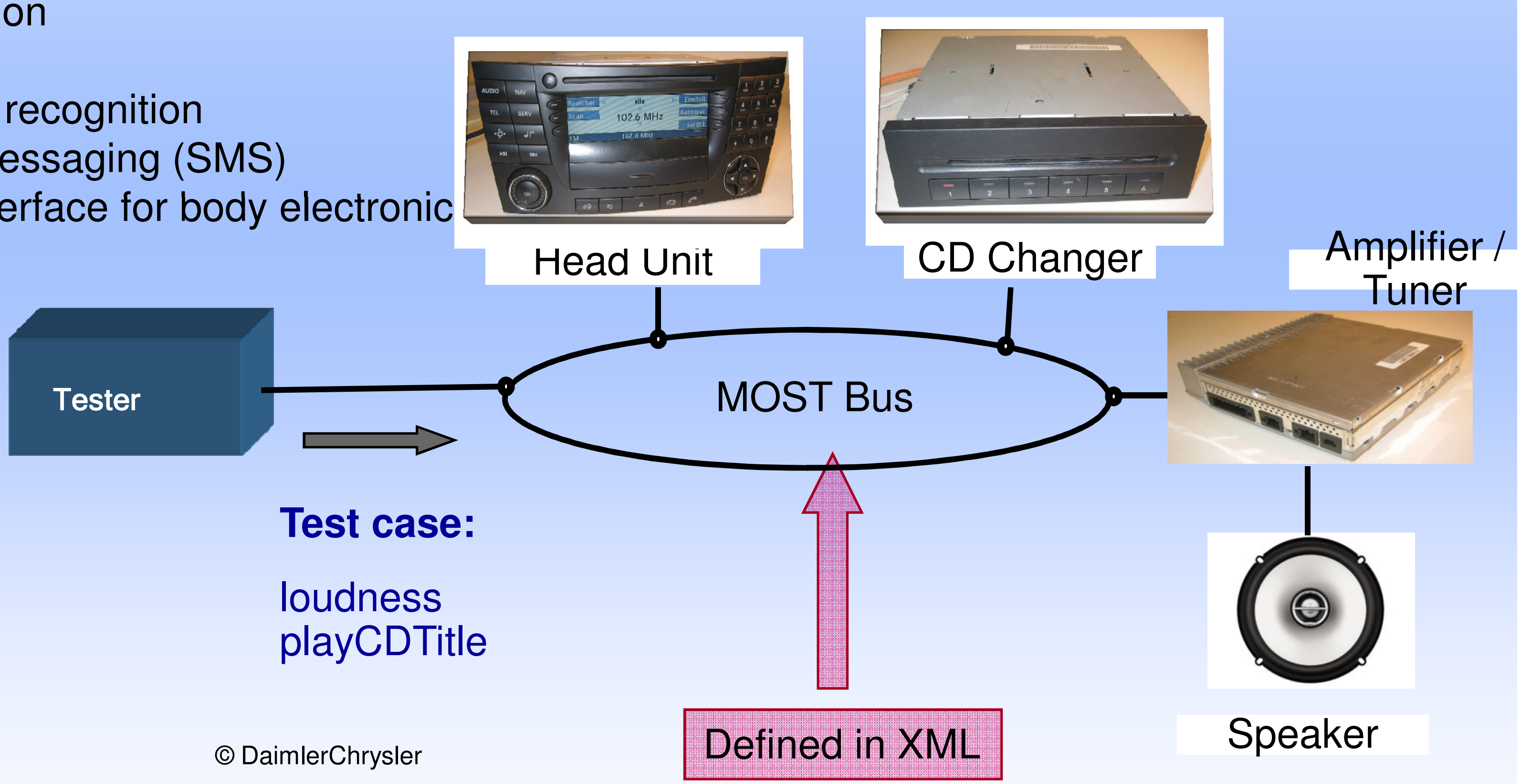
TTCN-3

Example Usage in Automotive

Work in ITEA project TTmedal

Telematics Applications

- Audio (CD / Radio)
- Telephone
- Navigation
- Video
- Speech recognition
- Short messaging (SMS)
- User interface for body electronic



Head Unit

CD Changer

Amplifier / Tuner

Tester

MOST Bus

Speaker

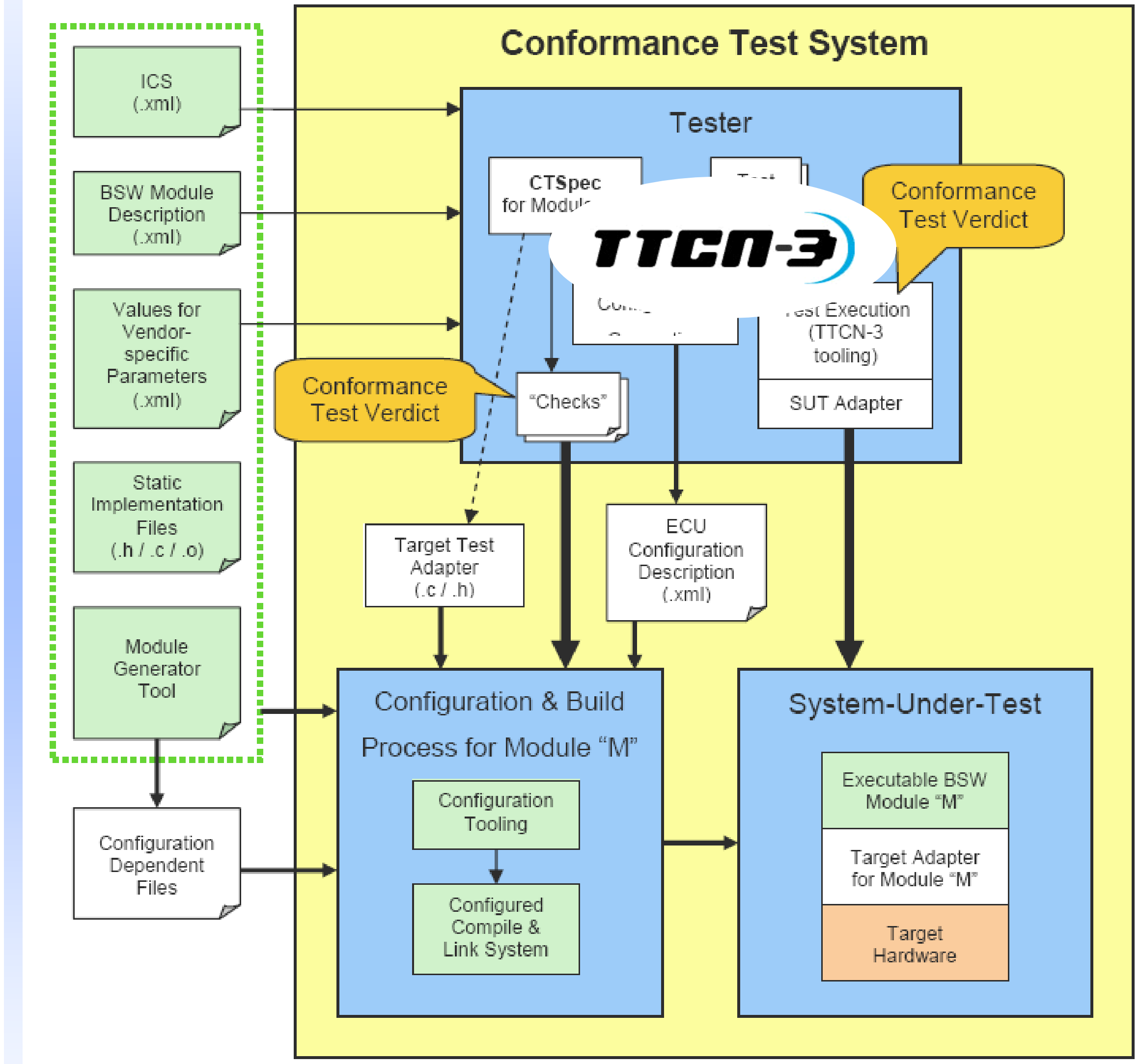
Test case:

loudness
playCDTitle

Defined in XML

© DaimlerChrysler

AUTOSAR adopted TTCN-3


Conformance Test System
ICS (.xml)
BSW Module Description (.xml)
Values for Vendor-specific Parameters (.xml)
Static Implementation Files (.h / .c / .o)
Module Generator Tool
Configuration Dependent Files
Tester
CTSpec for Modul
TTCN-3
Conformance Test Verdict
Test Execution (TTCN-3 tooling)
SUT Adapter
Conformance Test Verdict
"Checks"
Target Test Adapter (.c / .h)
ECU Configuration Description (.xml)
Configuration & Build Process for Module "M"
Configuration Tooling
Configured Compile & Link System
System-Under-Test
Executable BSW Module "M"
Target Adapter for Module "M"
Target Hardware

- Other usages
  - telecommunication
  - cockpit applications – MOST Forum
  - avionics systems – ESA
  - medical devices – HL7
  - power tranmission systems
  - smart cards
  - transport
  - financial systems
  - ...

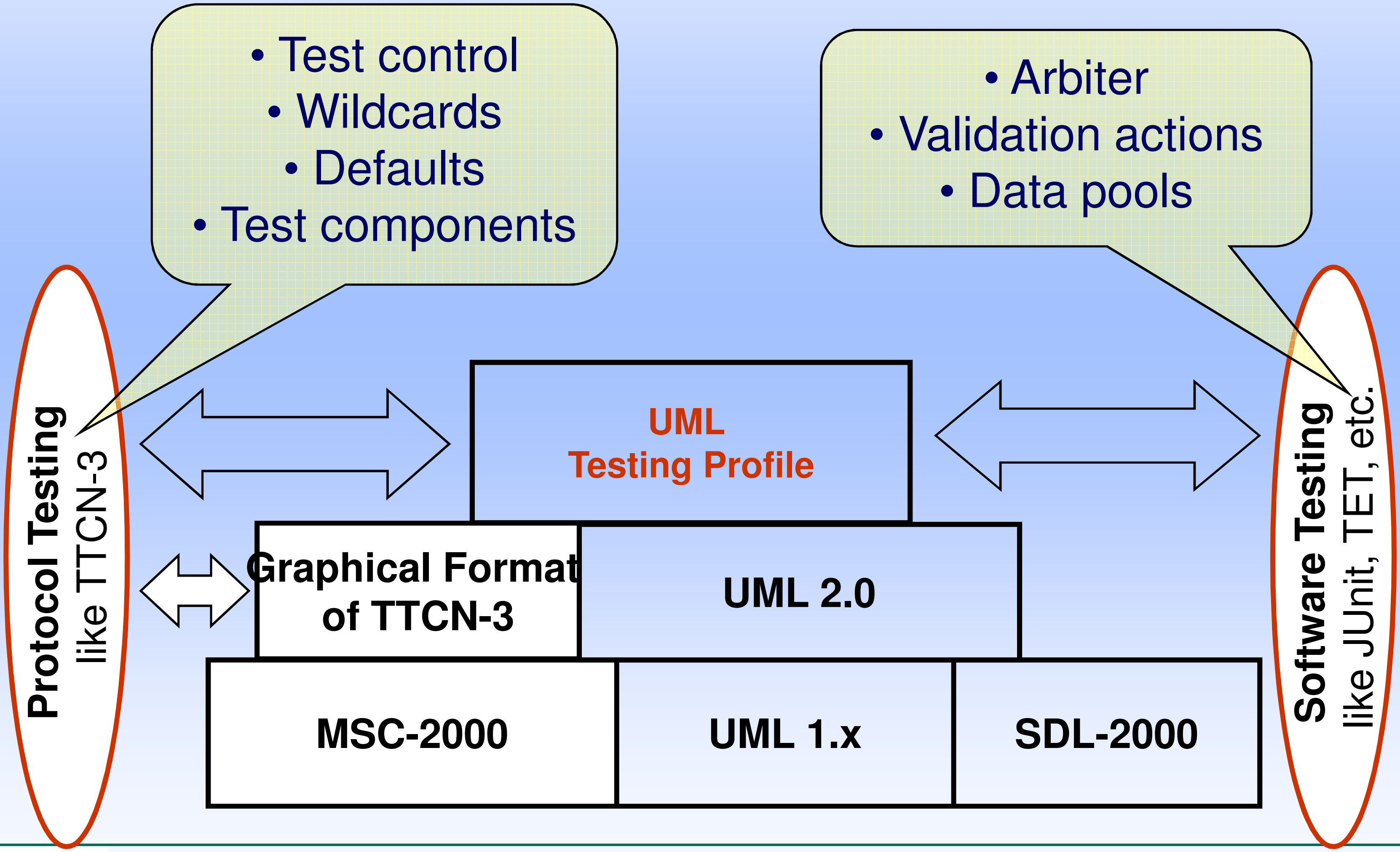
TTCN-3 today

- A successful testing technology
  - Used in telecommunication, software industry, automotive
- A textual and graphical test scripting language
  - Human readable
- A test implementation language
  - Automated test execution is built-in
- A test realization framework
  - A variety of ready-to-use tools and test assets provided by an agile community
- A philosophy
  - Specifically made for testers

Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

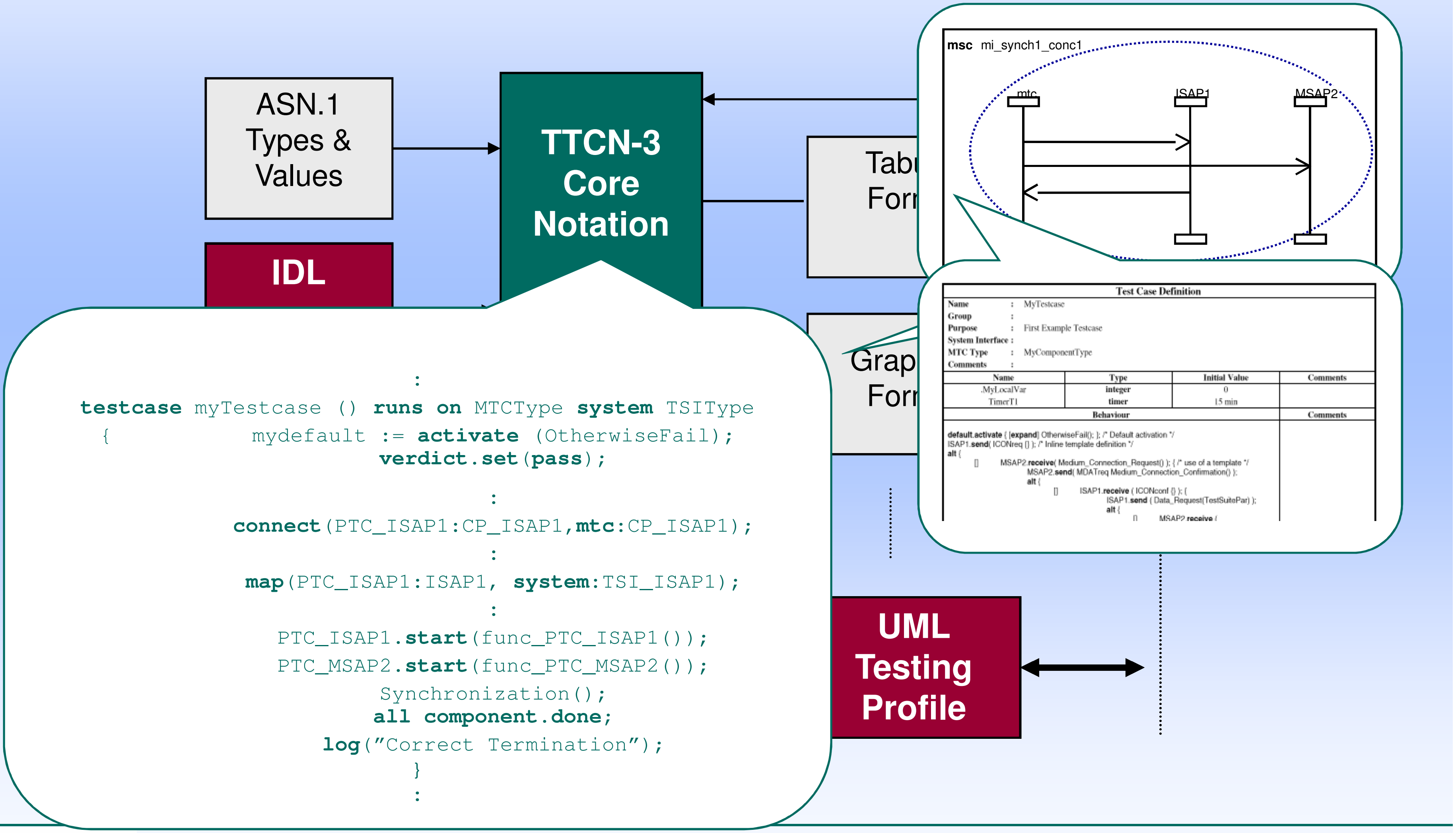
The Testing Profile Roots



1st Root: TTCN-3

- The new standardised test specification and test implementation language
  - Developed from 1999 – 2002 at the European Telecommunications Standards Institute (ETSI)
- Developed based on experiences from previous TTCN editions
  - Removal of OSI specific concepts; Improvement of concepts; Introduction of new concepts
- Applicable for all kinds of black-box testing for reactive and distributed systems, e.g.,
  - Telecom systems (ISDN, ATM, GSM, UMTS); Internet (IP, IP based protocols and applications); Software systems (Java, XML); Middleware platforms and component-based systems (CORBA, .Net, EJB)

TTCN-3



```
                     :
testcase myTestcase () runs on MTCType system TSIType
  {            mydefault := activate (OtherwiseFail);
                         verdict.set(pass);
                                   :
            connect(PTC_ISAP1:CP_ISAP1,mtc:CP_ISAP1);
                                   :
             map(PTC_ISAP1:ISAP1, system:TSI_ISAP1);
                                   :
               PTC_ISAP1.start(func_PTC_ISAP1());
               PTC_MSAP2.start(func_PTC_MSAP2());
                   Synchronization();
                   all component.done;
              log("Correct Termination");
                     }
                     :
```

Test Case Definition

Name : MyTestcase
Group :
Purpose : First Example Testcase
System Interface :
MTC Type : MyComponentType
Comments :

Name	Type	Initial Value	Comments
.MyLocalVar	integer	0	
TimerT1	timer	15 min	

Behaviour	Comments
default.activate ([expand] OtherwiseFail();); /* Default activation */ ISAP1.send(ICONreq []); /* Inline template definition */ alt ([] MSAP2.receive(Medium_Connection_Request()); { /* use of a template */ MSAP2.send(MDATreq Medium_Connection_Confirmation()); alt ([] ISAP1.receive (ICONconf {}); (ISAP1.send (Data_Request(TestSuitePar)); alt { [] MSAP2.receive (	

2nd Root: UML 2.0

- Developed by OMG (Object Management Group) 1999-2004, adoption June 2003, available 2004
  - UML 2.0 Infrastructure RFP
    - metamodel restructuring in order for Core to be reusable by other OMG languages
  - UML 2.0 Superstructure RFP
    - new and improvement/extension of UML concepts
  - UML 2.0 OCL RFP
    - defining an OCL metamodel
  - UML 2.0 Diagram Interchange RFP
    - ensuring diagram interchange between different tools

UML 2.0 Improvements

- More unified conceptual base
  - Parts in Internal structure, Collaborations, Use cases and indirectly in Interactions
- More unified semantics
  - Higher precision
- Improved expressiveness
  - Structured Classes, Sequence Diagrams and Statemachines
  - Activities merged with actions
  - Collaborations aligned with structured classes
  - Patterns (templates) and frameworks support

➢ More powerful and expressive than UML 1.4

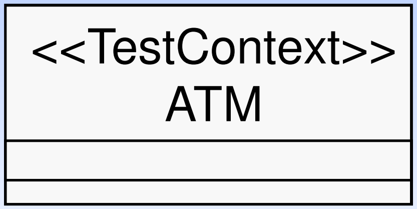
➢ Tighter and more consistent than UML 1.4

➢ Executable UML becomes possible

UML 2.0 Profiles

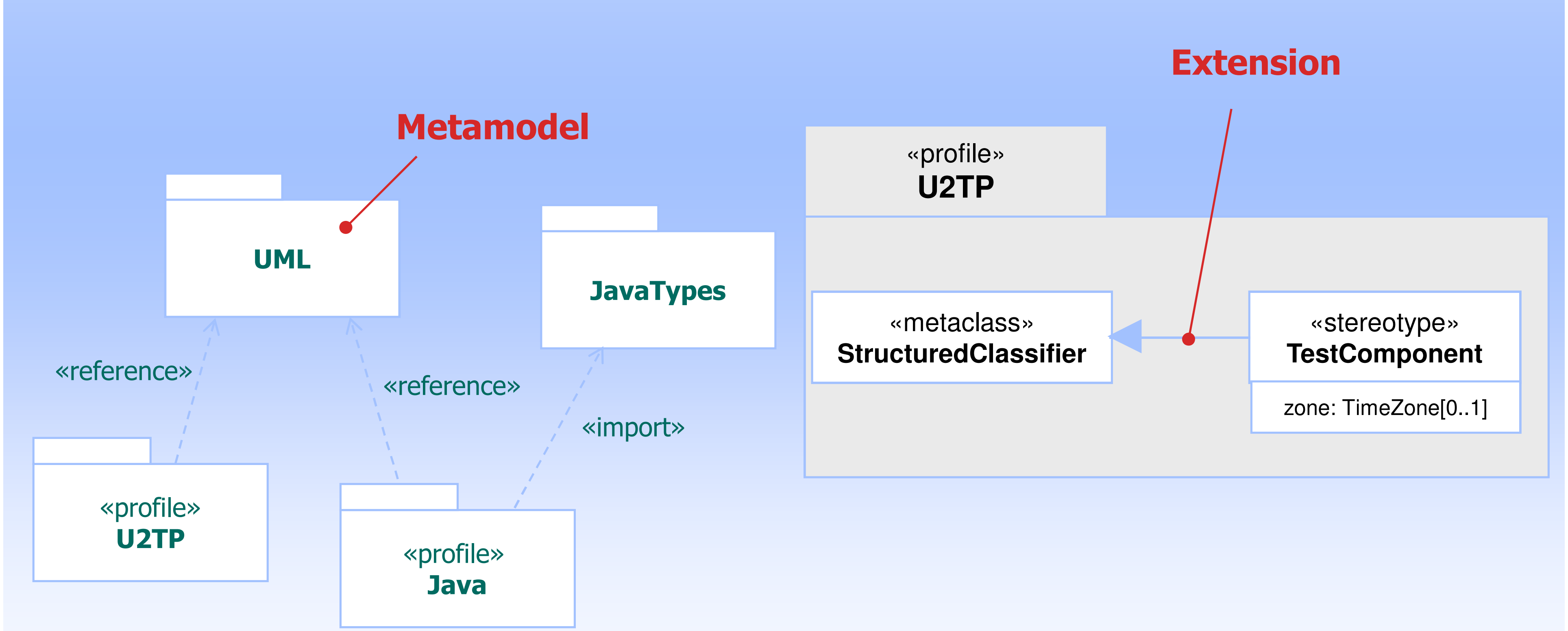
- Use of UML in
  - Analysis
  - Design/implementation
  - Directly executable notation (eg xUML)
  - Architecture description
  - Process engineering, workflow
  - Website structures
  - Data Modeling
- with obviously different (and inconsistent) semantics

- UML has many “semantic-free zones”, so called “semantic variation points”
  - E.g. detailed semantics of state machines, ...
- ➢ Profiles
  - Specializations of UML by stereotypes, providing special semantics


<<TestContext>>
ATM

UML 2.0 Profile Walkthrough (1)

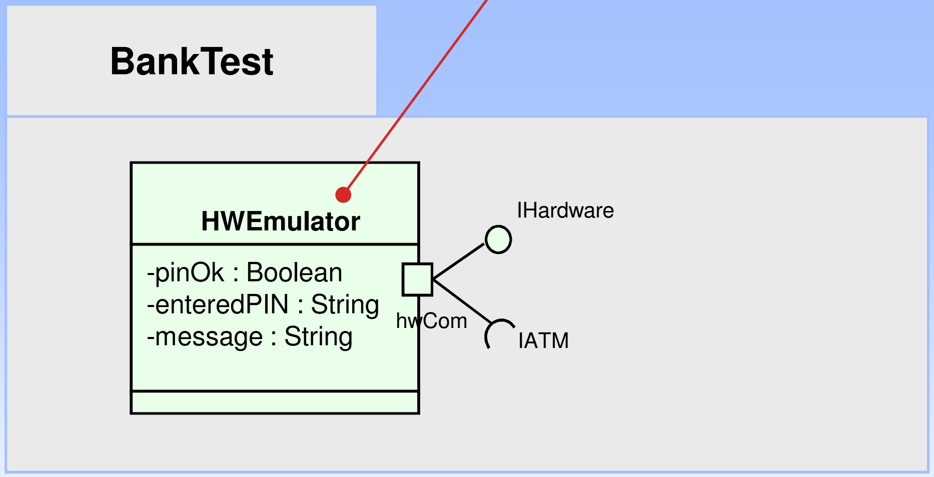
- Define profile(s)
  - based on reference metamodel
  - may use other packages for its definition



UML 2.0 Profile Walkthrough (2)

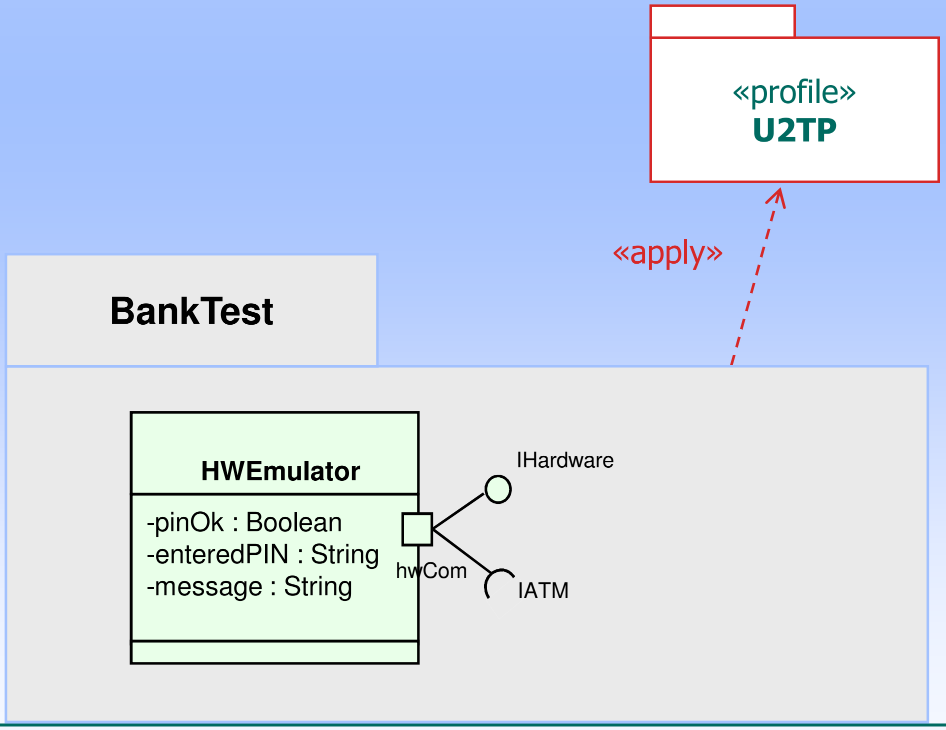
- Specify model
  - based on UML metamodel

A class definition



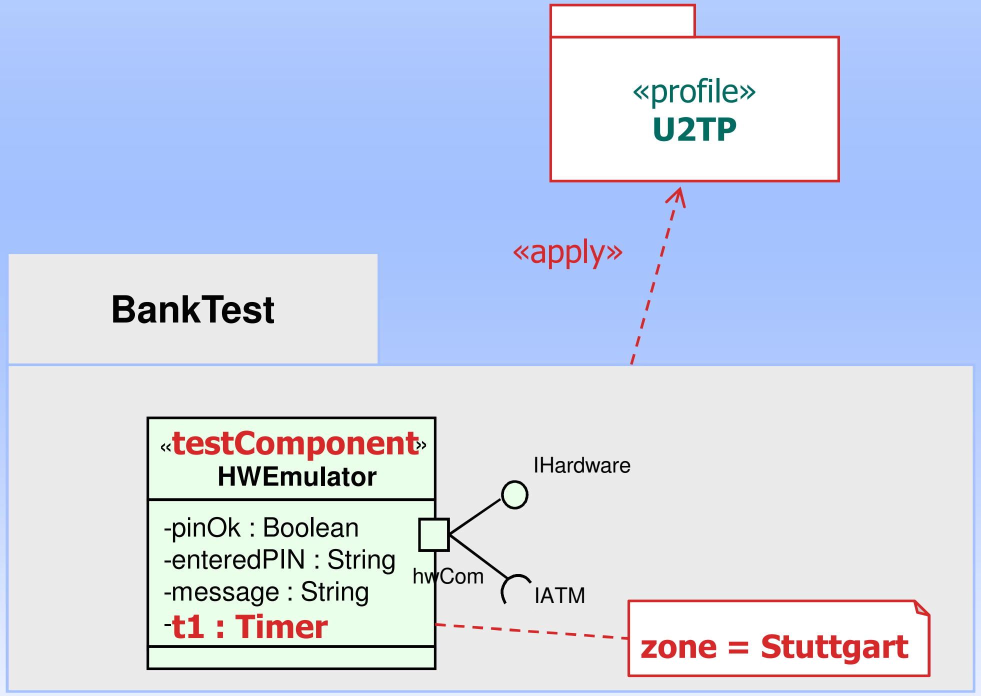
UML 2.0 Profile Walkthrough (3)

- Apply profile(s) to model
  - make it possible to apply stereotypes of the profile to the model elements



UML 2.0 Profile Walkthrough (4)

- Apply stereotypes to model elements as desired



Concepts of the Testing Profile

- Test architecture
    - Test structure, test components and test configuration
- Test data
    - Data and templates used in test procedures
- Test behavior
    - Dynamic aspects of test procedures
- Test time
    - Time quantified definition of test procedures

Test Architecture Realization

- System Under Test (*SUT*)
- *Test components*
- *Test context* with test configuration and test cases
- Test verdict arbitration with *arbiter*
- Test coordination with *scheduler*

Test Data Realization

- Individual *coding rule* definition
- *Wildcards ** and ***?***
- Concrete test data with *data pool, data partition and data selector*

Test Behavior Realization

- *Test objectives*
- *Test cases*
- Test *verdicts:* pass, fail, inconclusive
- *Defaults* behaviors on different levels
- *Utility* part

Test Time Realization

- *Clock*
- *Timezone* definition for synchronizing test components
- *Timer* operations

Concepts beyond TTCN-3

- Unification of test cases:
  - Test case as a composition of test cases
  - Test behavior defines the execution of a test case
- Separation of test behavior and verdict handling
  - Arbiter is a special component to evaluate the verdict
  - Validation actions are used to set the verdict
- Abstract test cases that work on data partitions rather than individual data
  - Data partitions to describe value ranges for observations and stimuli
- Test architecture with test deployment support
  - Part of the test specification is the definition of deployment requirements for a test case

Concepts beyond UML

- Defaults within test behavior
  - Concentration on main flow of test behavior
  - Default hierarchy to handle different concerns
- Wildcards within test data
  - Flexible definition of value sets
- Timers and time constraints
  - Time controlled test behavior
- Arbitration and verdicts
  - Assessment of test behavior
- Coding attributes
  - Encoding/decoding for data exchange with the SUT

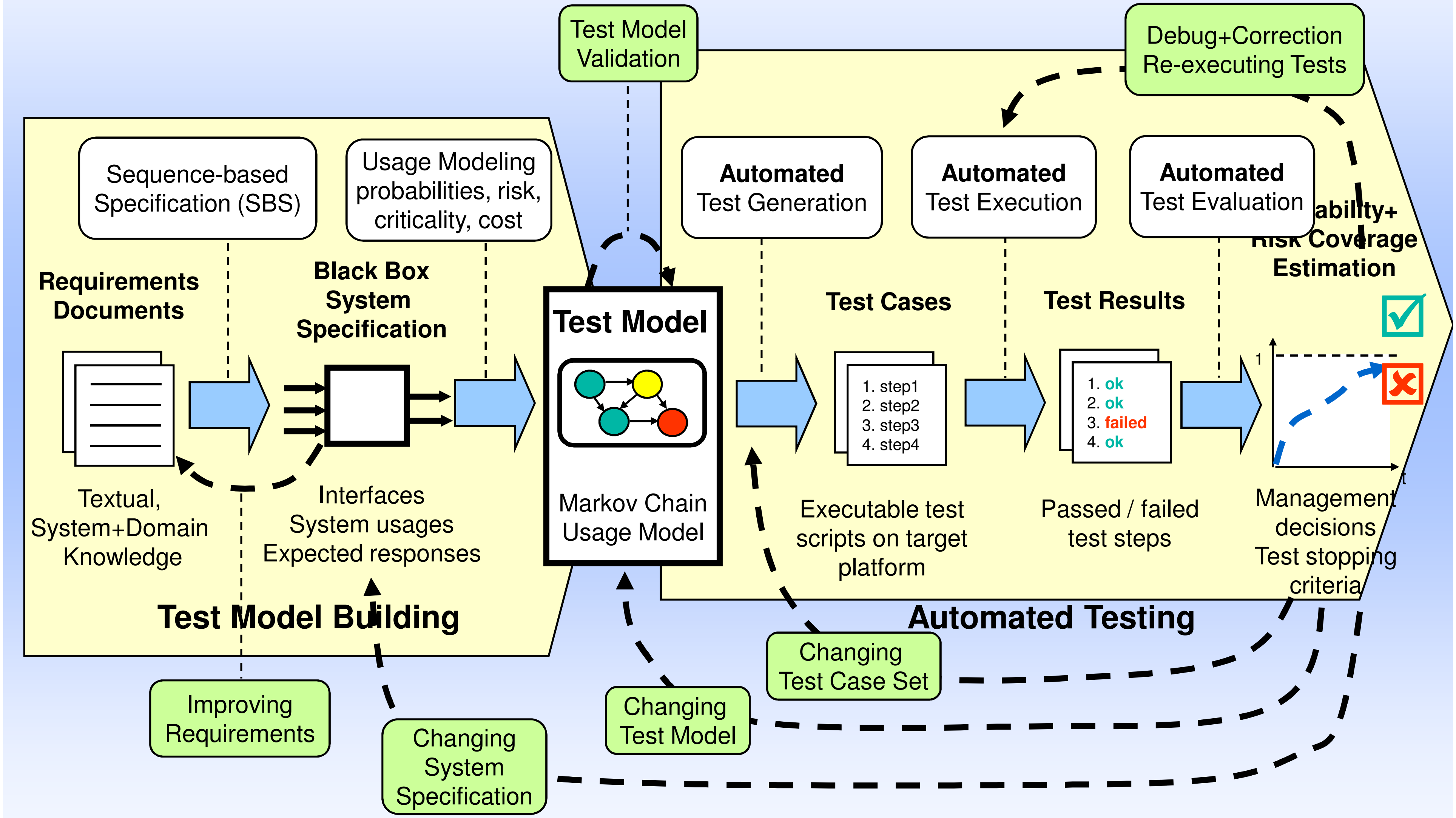
Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Model-based Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Model-based Statistical Testing (MBST)

- Definition
  - **Usage-oriented black box** testing
  - Testing = **statistical experiment**
  - Selection, execution, and evaluation of a representative subset of software input/output trajectories
  - Analysis of sample to produce **reliability estimates**
- Approach
  - Focus: Model Construction (not manual test case generation)
  - Building a **Test Model** based on the requirements
  - Considering **operational** / **usage profiles**
  - **Test automation** (=automated test generation, execution, evaluation)
- Applying statistics for
  - Test model building and model analysis
  - Generating Test Cases (test model paths)
  - Test analysis and reliability estimation

Steps of MBST



Test Model Validation

Debug+Correction Re-executing Tests

Sequence-based Specification (SBS)

Usage Modeling probabilities, risk, criticality, cost

Requirements Documents

Black Box System Specification

Textual, System+Domain Knowledge

Interfaces System usages Expected responses

Test Model

Markov Chain Usage Model

Test Model Building

Automated Test Generation

Automated Test Execution

Automated Test Evaluation

ability+ Risk Coverage Estimation

Test Cases

Executable test scripts on target platform

Test Results

Passed / failed test steps

Management decisions Test stopping criteria

Automated Testing

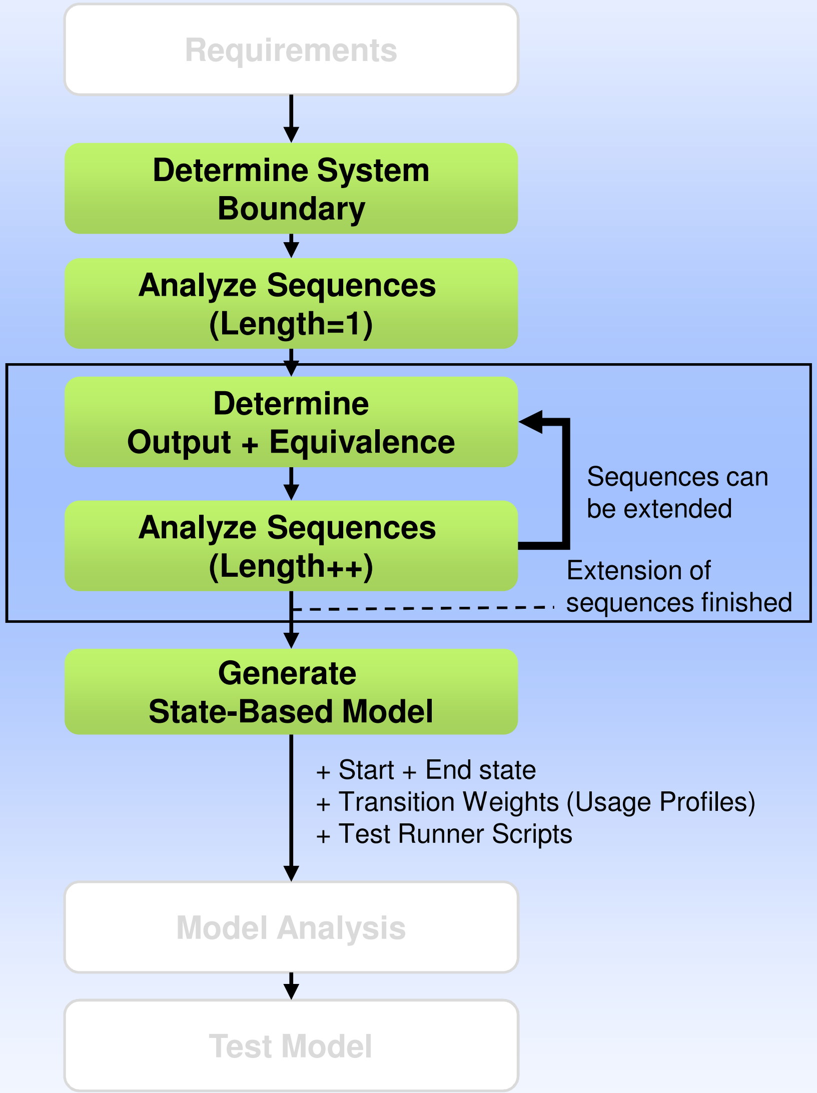
Improving Requirements

Changing System Specification

Changing Test Model

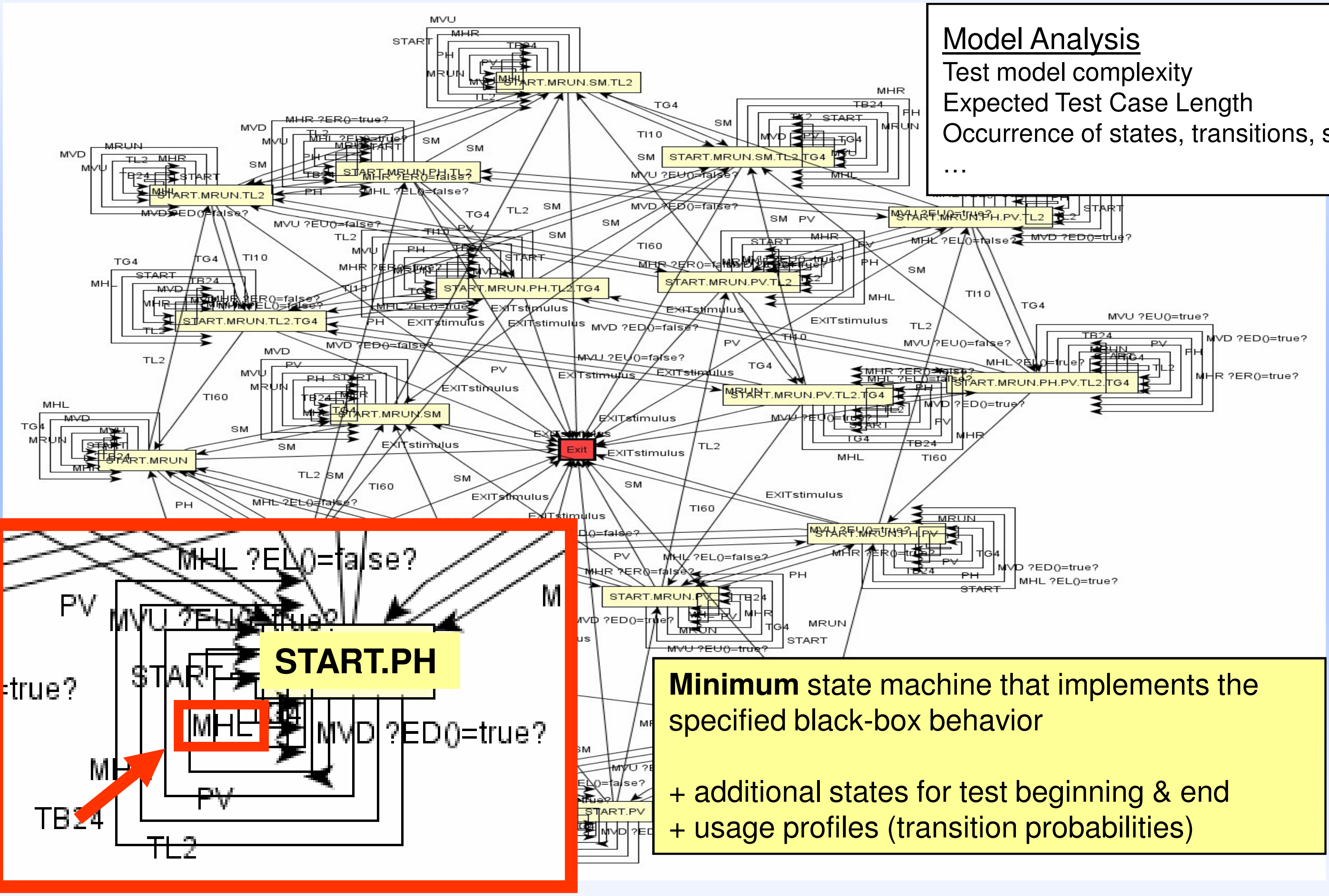
Changing Test Case Set

Introduction to Sequence-based Specification (SBS)


Requirements
Determine System Boundary
Analyze Sequences (Length=1)
Determine Output + Equivalence
Analyze Sequences (Length++)
Sequences can be extended
Extension of sequences finished
Generate State-Based Model
+ Start + End state
+ Transition Weights (Usage Profiles)
+ Test Runner Scripts
Model Analysis
Test Model

- Systematic inspection of requirements to develop a complete and consistent specification
- Finding relevant input sequences (stimuli) and expected responses (test oracle)
- Mapping: Stimulus sequences -> Responses
- Development of a state-based model that implements the mapping

Test Model



Model Analysis

Test model complexity

Expected Test Case Length

Occurrence of states, transitions, stimuli

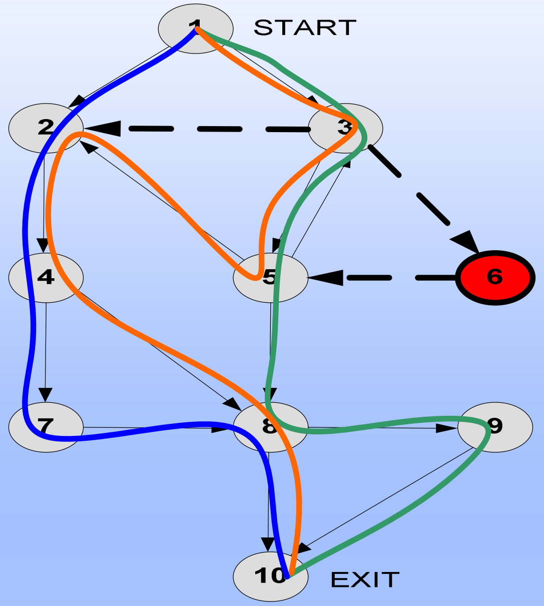
…

Minimum state machine that implements the specified black-box behavior

+ additional states for test beginning & end

+ usage profiles (transition probabilities)

Automated Test Case Generation



Test Plan

Model Coverage (here: 23 test cases)

Random tests to achieve desired reliability (here: 1000 test cases)

Test Cases Recorded	1,023 cases / ~16000 stimuli
Node coverage	23 nodes (100%)
Arc coverage	304 arcs (100%)
Stimulus coverage	24 stimuli (100%)

- Model coverage
  - Coverage of model elements (states, transitions)
  - Minimum number of test cases and test steps
- Random tests
  - Considering transition probability
  - Generation of representative test case due to usage profile
- Weighted tests
  - Considering probability, cost, value of transitions
  - Generation of test cases with minimized or maximized sums or products of transitions attributes
- Manual tests
  - Required by standards, guidelines

Abstract vs concrete tests

•paths in the model (=sequence of stimuli)

•executable test scripts

Reliability Estimations

- Reliability in statistical testing: Probability of failure-free operation / use (0..1)
- Input parameters:
  - Number of failures
  - Number of test cases
  - Prior information about system reliability in the past
- Reliability estimations for
  - Model elements (stimuli, transitions)
  - System/test object usages (input sequences)

Stimulus Reliabilities	Gen	Exec	Fail	Actual Reliability	Optimum Reliability
MHL	841	841	6	0.979191	0.986127
PH	1,028	1,028	0	0.981273	0.981273

- **Single Event Reliability**
  - Probability that next randomly selected stimulus will not produce a failure
- **Single Use Reliability**
  - Probability that next randomly generated test case (system use) will not produce a failure

System Reliabilities	Actual Reliability	Optimum Reliabiliy
Single Event Reliability	0.981500	0.982900
Single Use Reliability	0.892600	0.900700

Presentation Overview

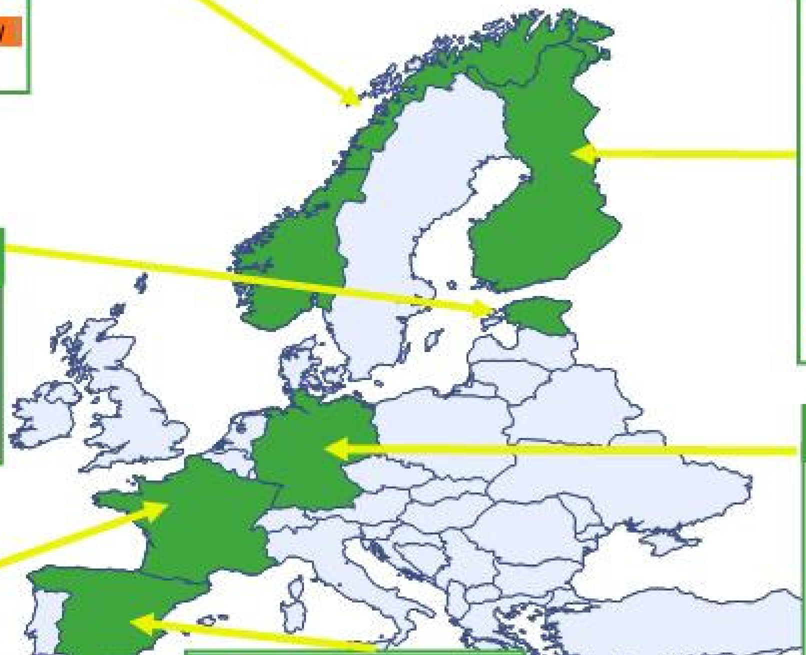
- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Project Goals

- To develop the methodologies, tools and industrial experience to enable European industry to test more effectively and more efficiently
- To drive the deployment of Model-based testing technology into European industry

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Project Partners



Norway

TANDBERG

simula research laboratory

Estonia

ELVIOR

eliko

1918 TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

France

ETSI

Spain

DATAPIXEL

Trimek

ideko

innovalia ASSOCIATION

SORALUCE

SQS

Finland

Nokia Siemens Networks

VTT

NetHawk

CONFORMIQ

ÅBO AKADEMI

Germany

DAIMLER

PIKETEC

ABB

Testing Technologies

FOKUS
Fraunhofer Institute for Open Communication Systems

Nokia Siemens Networks

INSPIRE

IXtronics

IESE
Fraunhofer Institut Experimentelles Software Engineering

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

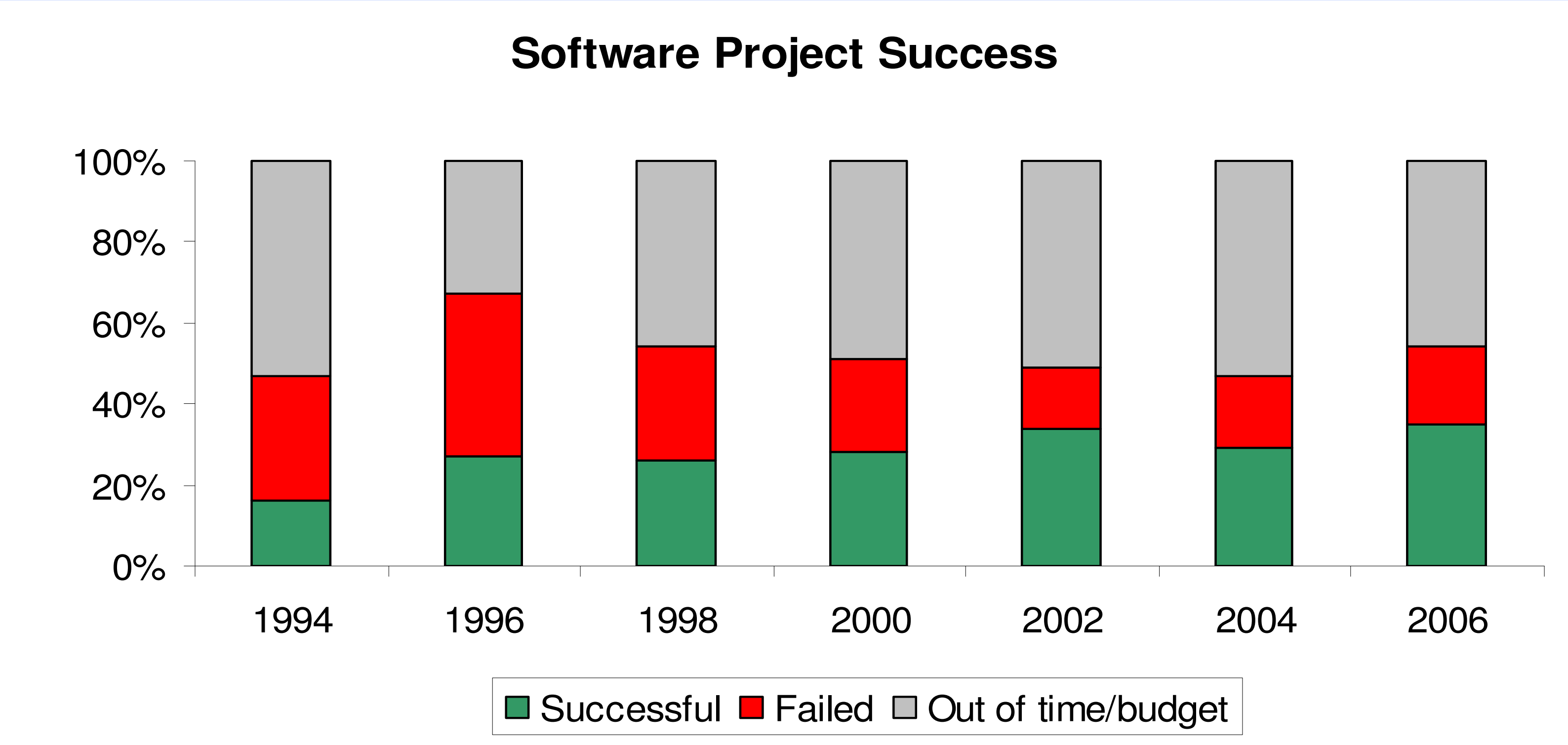
Project Relevance

- The importance of software in product development is increasing
- 40-60% of product development costs goes in testing
- New testing technology has the potential to save 25-50% of testing costs
  - "The use of models pays off when it comes to detecting failures by means of model-based tests"[1]
- Improving testing will directly impact European Industrial competitiveness

[1]One Evaluation of ModelBased Testing and its Automation; A. Pretschner et al ICSE 2005

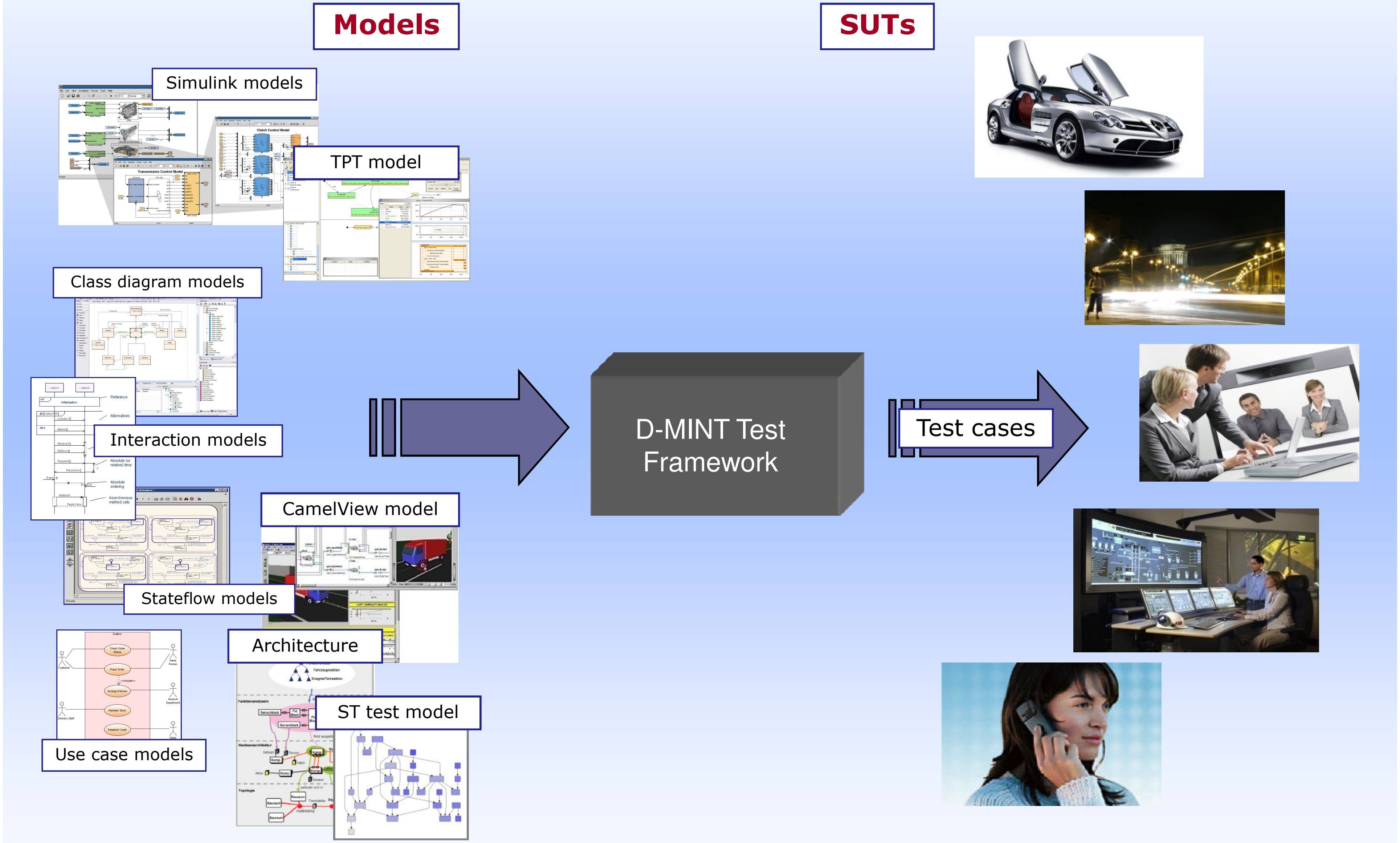
Project Relevance

Number of successful software projects still less than 1/3[1]



[1] The Standish Group 2006; The Chaos Report

Vision



Models

- Simulink models
- TPT model
- Class diagram models
- Interaction models
- CamelView model
- Stateflow models
- Architecture
- ST test model
- Use case models

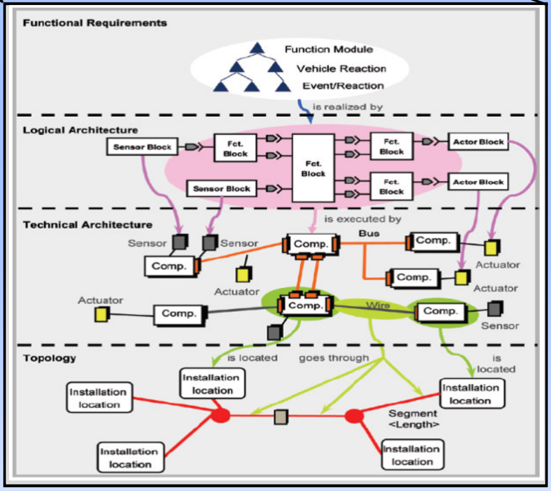
D-MINT Test Framework

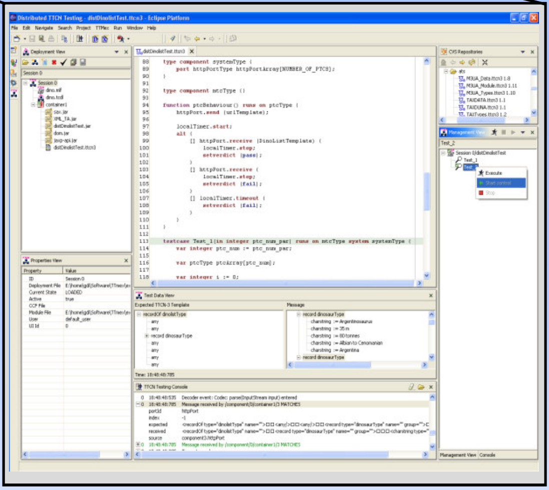
Test cases

SUTs

Project structure

WP1 Industrial Case Studies → WP2 Test Principles & Methods → WP3 Test Tool Chain → WP4 Exploitation → WP5 Dissemination



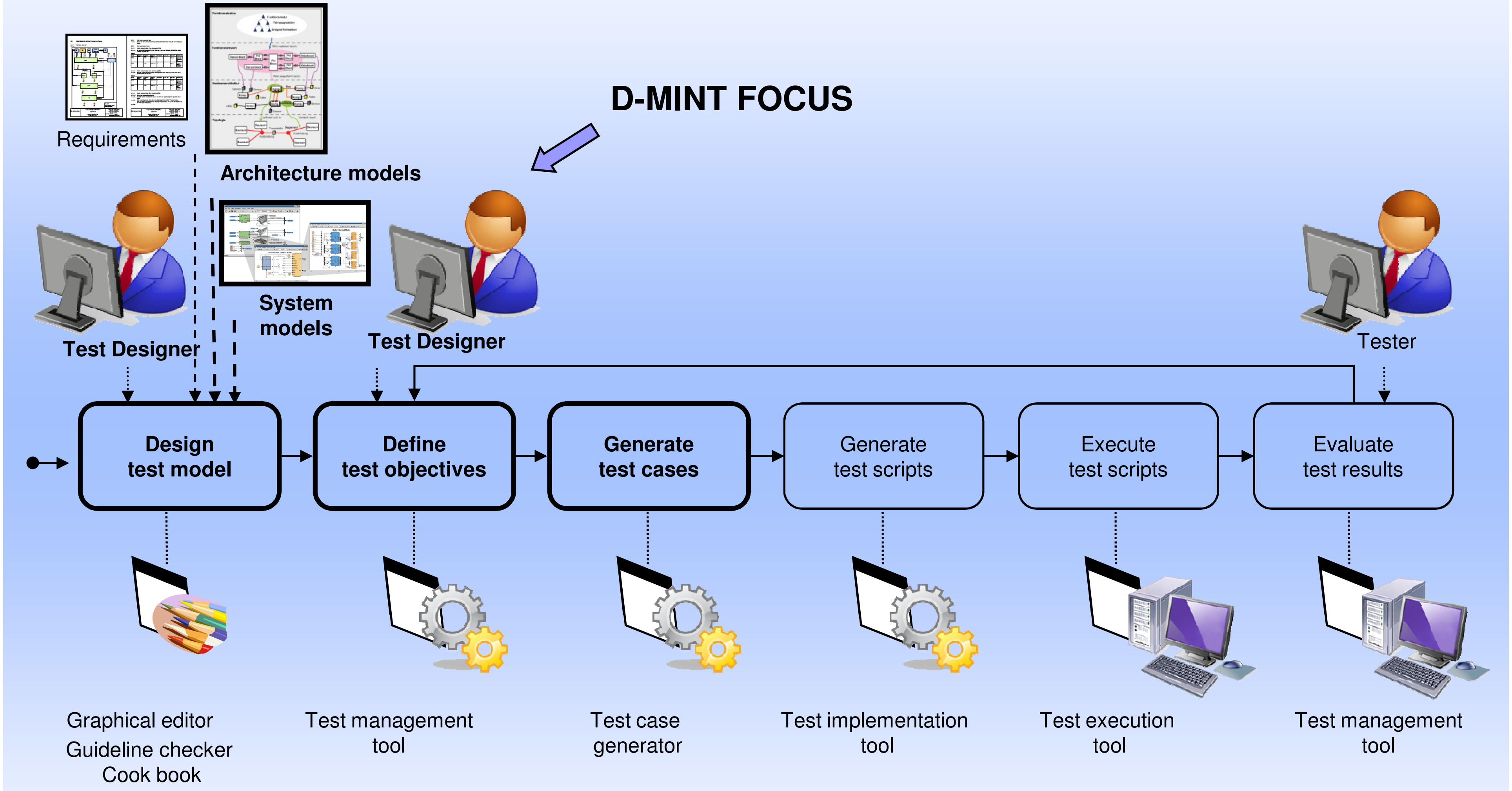


```
type component systemType {
    port httpPortType httpPortArray[NUMBER_OF_PTCS];
}

type component mtcType {}

function ptcBehaviour() runs on ptcType {
    httpPort.send (urlTemplate);

    localTimer.start;
    alt {
        [] httpPort.receive (DinoListTemplate) {
          localTimer.stop;
          setverdict (pass);
        }
        [] httpPort.receive {
          localTimer.stop;
          setverdict (fail);
        }
        [] localTimer.timeout {
          setverdict (fail);
        }
    }
}

testcase Test_1(in integer ptc_num_par) runs on mtcType system systemType {
    var integer ptc_num := ptc_num_par;

    var ptcType ptcArray[ptc_num];

    var integer i := 0;
```

```
recordOf dinolistType
  any
  any
  record dinosaurType
  any
  any
  any
  any
```

```
record dinosaurType
  charstring := Argentinosaurus
  charstring := 35 m
  charstring := 80 tonnes
  charstring := Albian to Cenomanian
  charstring := Argentina
record dinosaurType
```

Model-Based Testing



D-MINT FOCUS

Requirements

Architecture models

System models

Test Designer

Test Designer

Tester

Design test model

Define test objectives

Generate test cases

Generate test scripts

Execute test scripts

Evaluate test results

Graphical editor
Guideline checker
Cook book

Test management tool

Test case generator

Test implementation tool

Test execution tool

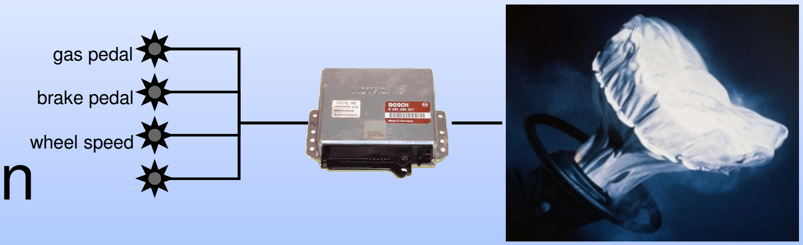
Test management tool

Test modelling techniques

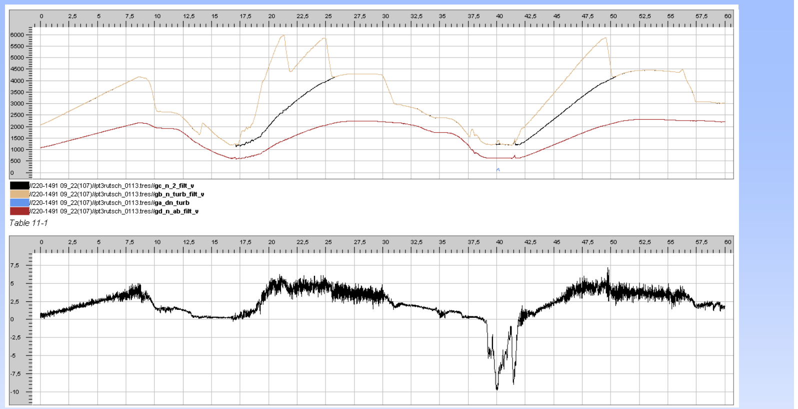
- Modelling techniques used and *developed*:
  - Continuous systems:
    - Time-Partition-Testing (TPT)
    - *Model-in-the-Loop for Embedded System Test (MiLEST)*
  - Hybrid and discrete systems:
    - UML Testing Profile (UTP)
    - *UML Testing Profile for Embedded Systems (UTPes)*
    - *UML Test Modelling Language (UTML)*

MBT for Testing Embedded Controllers

Systems under test are



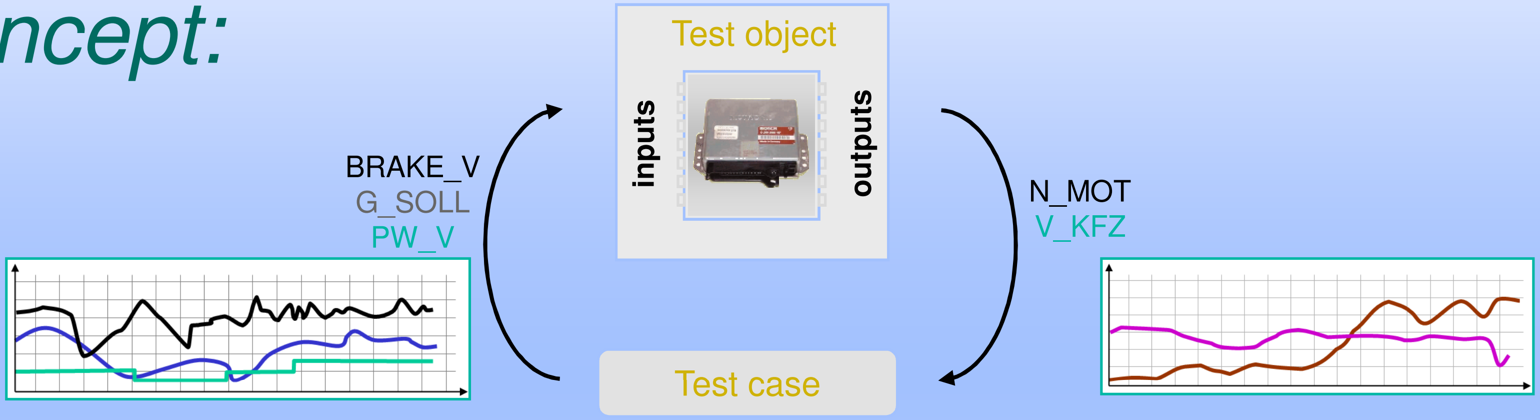
- signal driven and/or event driven
- large interfaces
- timing complexity (sequences, temporal conditions, signal processing etc.)
  - Noise
  - Monotony
  - Sequences (off ⇨ on ⇨ off)
  - Duration
- hybrid systems (mixture of continuously changing, discrete quantities, events and messages)



⇨ Difficult to cope with conventional test methods

MBT for Embedded Controllers

Modeling Concept:



- Test cases stimulate the test object by continuously affecting system quantities (inputs).
- Test cases can react to system behavior by observing system quantities (outputs).
- Interface test object ↔ TPT test case is based on named variables

TPT – Test modeling

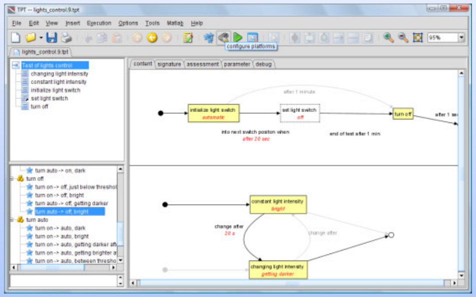
TPT
Time Partition Testing

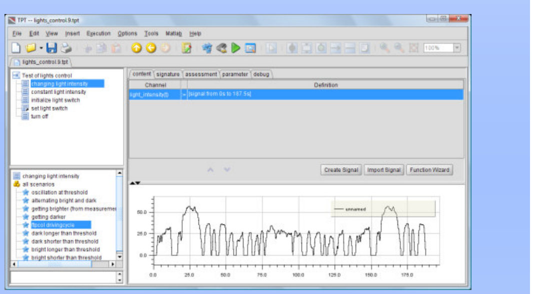
Language properties

- Graphical test case modeling
- Based on automata (hybrid, hierarchical, parallel)
- Support a natural way of continuous signal definitions
- Usage of natural language for description
- Formal details are hidden behind graphics

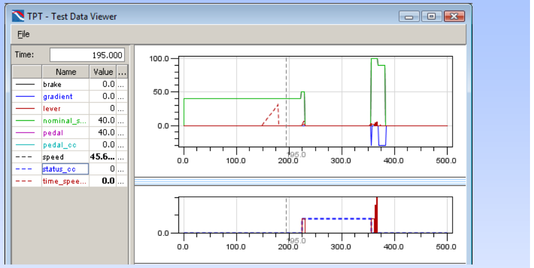
Advantages

- Clear structured and easy to learn
- Easy to read even for non-programmers
- Compact (complexity of test cases is comparatively low)



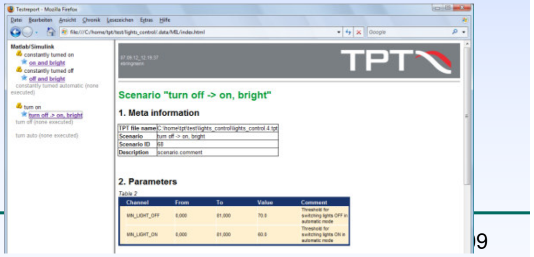


Channel		Definition
light_intensity(t)	▾	[signal from 0s to 187.5s]



Time: 195.000

	Name	Value	...
———	brake	0.0	...
———	gradient	0.0	...
———	lever	0	...
———	nominal_s...	40.0	...
———	pedal	40.0	...
———	pedal_cc	0.0	...
- - -	speed	**45.6...**	...
- - -	status_cc	0	...
- - -	time_spee...	**0.0**	...


Testreport - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
file:///C:/home/tpt/test/lights_control/.data/MiL/index.html
Google
Matlab/Simulink
constantly turned on
on and bright
constantly turned off
off and bright
constantly turned automatic (none executed)
turn on
turn off -> on, bright
turn off (none executed)
turn auto (none executed)
07.09.12_12.19.37
TPT

Scenario "turn off -> on, bright"

1. Meta information

TPT file name	C:\home\tpt\test\lights_control\lights_control.4.tpt
Scenario	turn off -> on, bright
Scenario ID	68
Description	scenario.comment

2. Parameters

Table 2

Channel	From	To	Value	Comment
MIN_LIGHT_OFF	0,000	81,000	70.0	Threshold for switching lights OFF in automatic mode
MIN_LIGHT_ON	0,000	81,000	60.0	Threshold for switching lights ON in automatic mode

TPT Features

Requirements:

1. Automation
2. Consistency
3. Systematic testing
4. Readability
5. Reactive tests
6. Real-time and continuous behavior

Features:

Automated tests (from test execution to test report)

Platform independent

Consistency from model to assessment and report

Abstract test language

Systematic test case definition

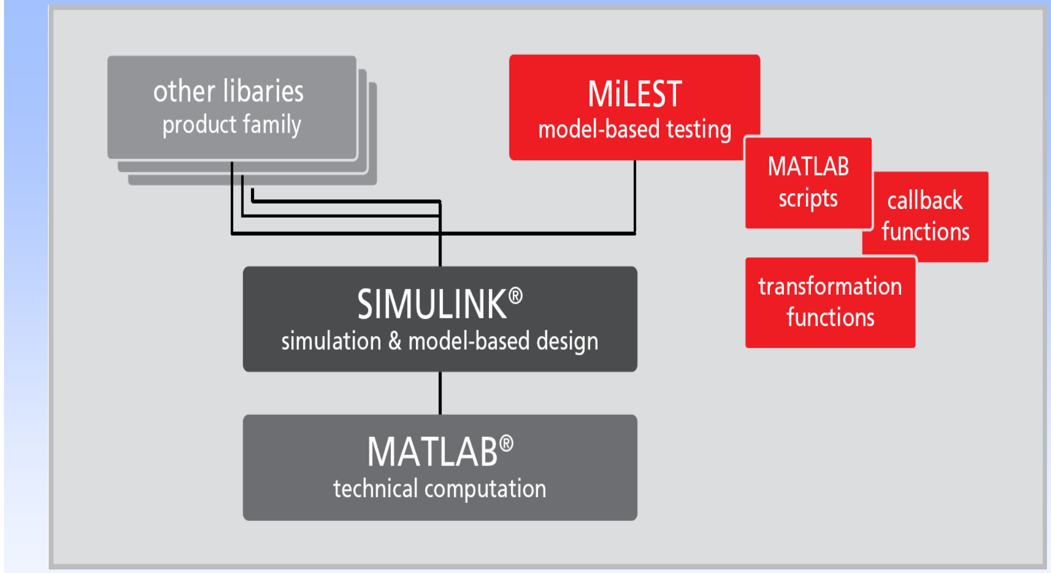
Intuitive graphical models

Reactive tests supported

Continuous behavior testing

Test generation

- Example *MiLEST*:
  - Continuous and discrete signal flows
  - Test harness generation and execution
  - Realization in ML/SL



other libaries
product family

MiLEST
model-based testing

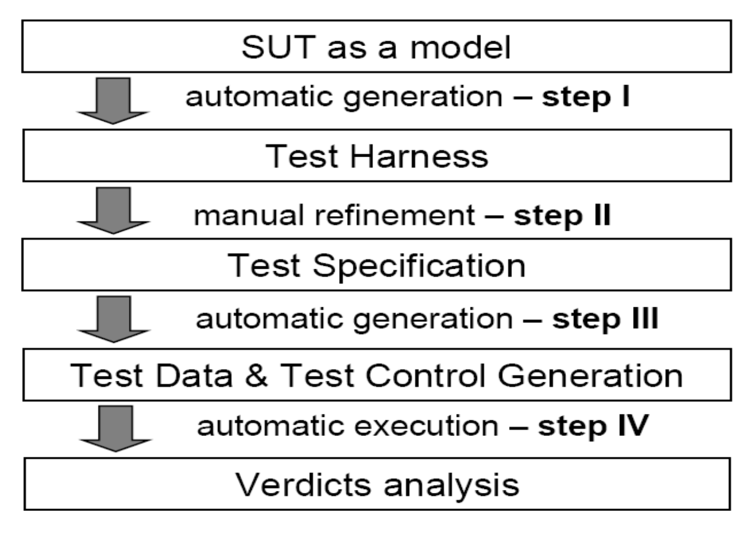
MATLAB
scripts

callback
functions

transformation
functions

SIMULINK®
simulation & model-based design

MATLAB®
technical computation



SUT as a model

automatic generation – **step I**

Test Harness

manual refinement – **step II**

Test Specification

automatic generation – **step III**

Test Data & Test Control Generation

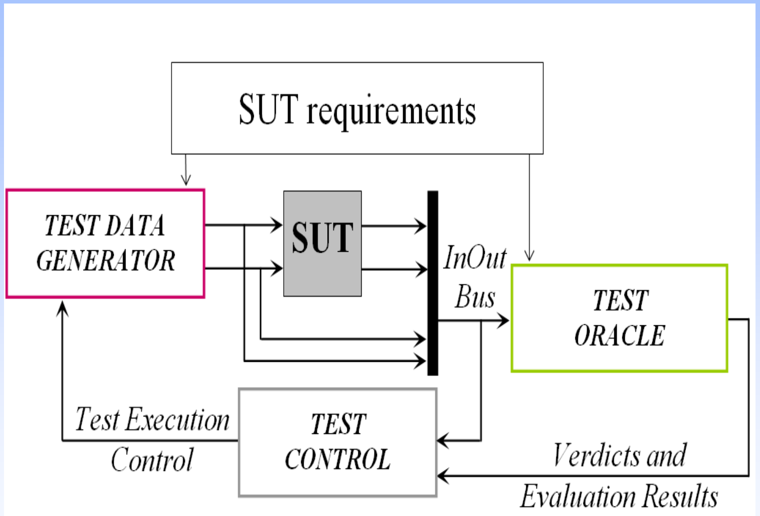
automatic execution – **step IV**

Verdicts analysis

MiLEST Summary

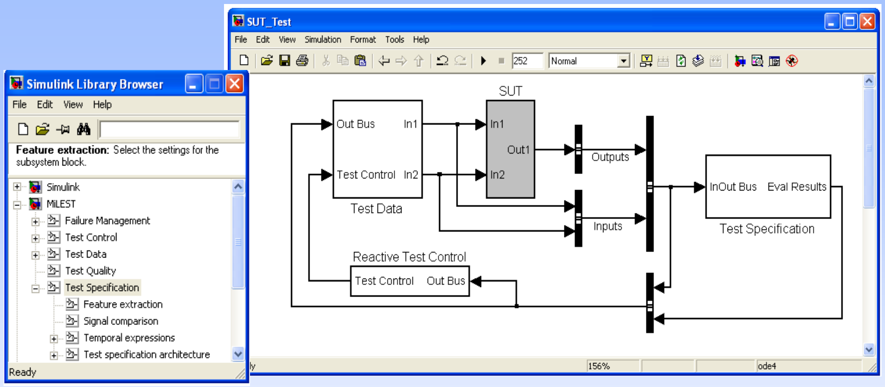
Features:

- **Systematic**, consistent **functional** test specification
- **Signal's feature** - oriented paradigm
- **Graphical** test design
- Test process **automation**
  - systematic and automatic test data generation
  - online automatic test evaluation
- **Model-in-the-Loop** test execution
- Reusable **test patterns**
- Abstract and concrete **views**



Benefits:

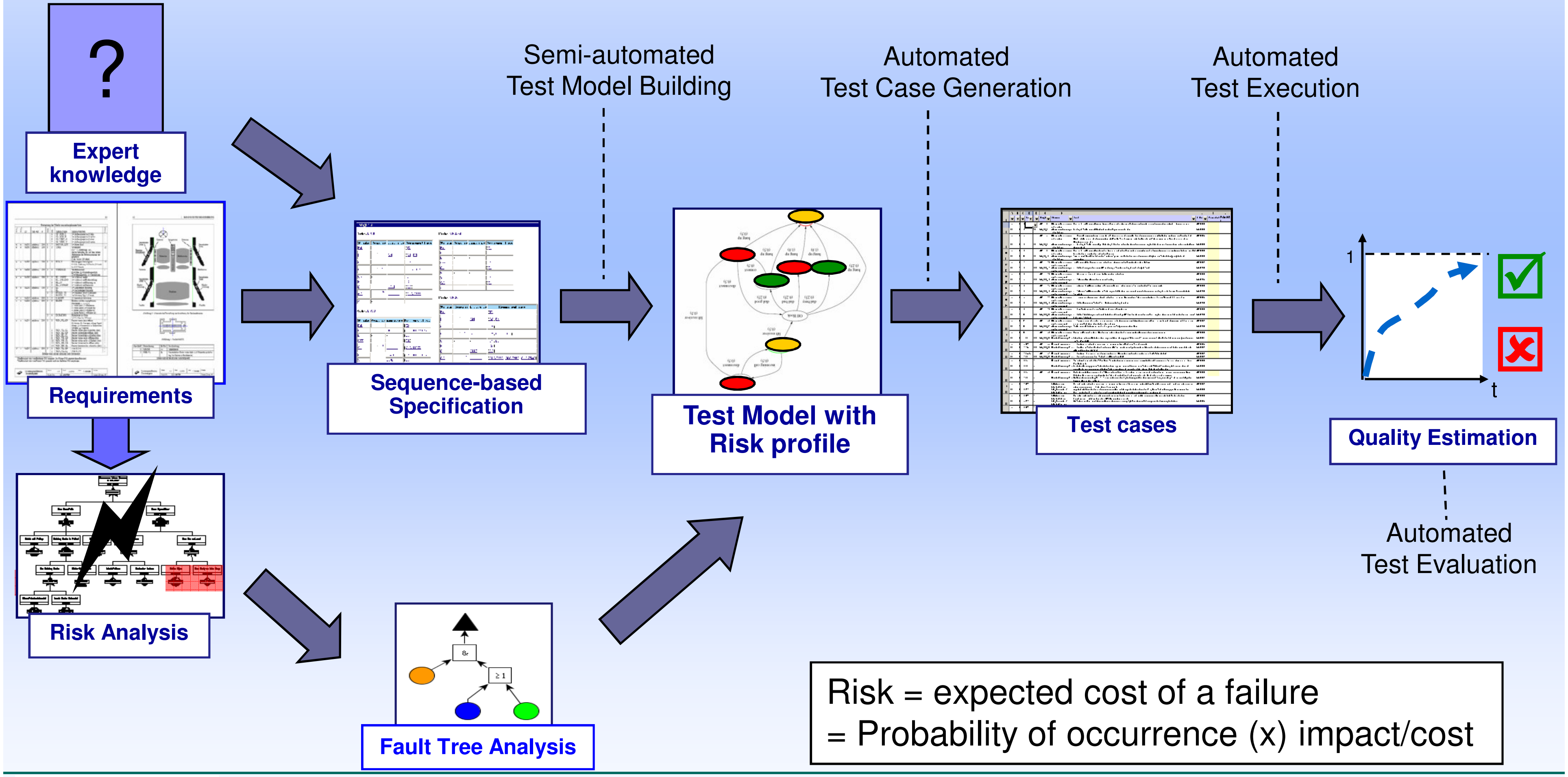
- Testing in **early design** stages
- Test of **hybrid** systems including **temporal** and logical dependencies
- **Traceability** of test cases to the requirements; verdicts to root faults
- Increased **test coverage** and **test completeness**
- Assured **test quality** of the test specification



Feature extraction: Select the settings for the subsystem block.

Test prioritization and selection

- Extension of statistical testing with risk-based considerations



Semi-automated
Test Model Building

Automated
Test Case Generation

Automated
Test Execution

Expert knowledge

Requirements

Sequence-based Specification

Test Model with Risk profile

Test cases

Quality Estimation

Automated
Test Evaluation

Risk Analysis

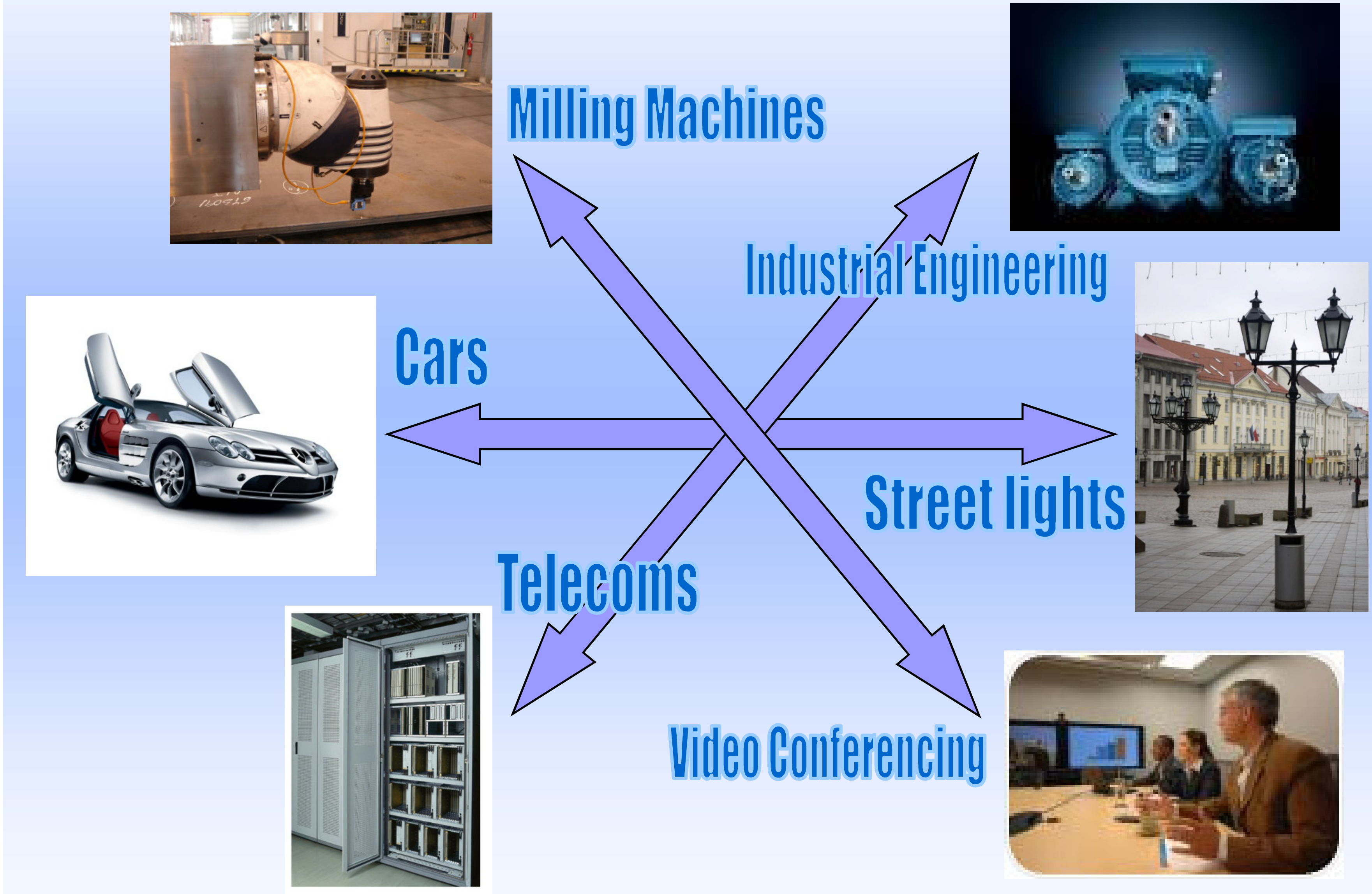
Fault Tree Analysis

Risk = expected cost of a failure
= Probability of occurrence (x) impact/cost

Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Application Areas



Milling Machines

Industrial Engineering

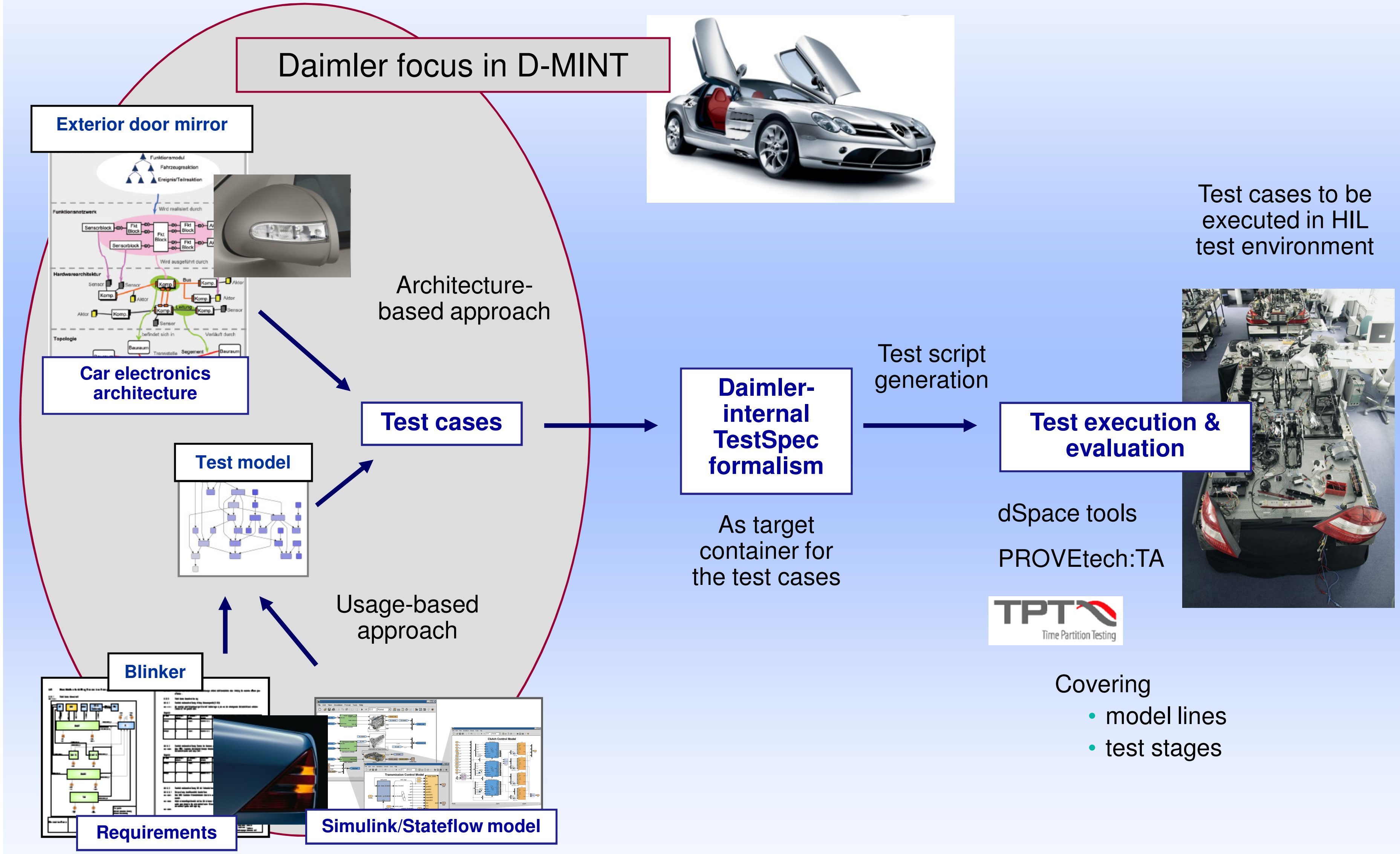
Cars

Street lights

Telecoms

Video Conferencing

Daimler automotive case study



Daimler focus in D-MINT

Exterior door mirror

Car electronics architecture

Architecture-based approach

Test model

Test cases

Usage-based approach

Blinker

Requirements

Simulink/Stateflow model

Daimler-internal TestSpec formalism

As target container for the test cases

Test script generation

Test execution & evaluation

Test cases to be executed in HIL test environment

dSpace tools

PROVEtech:TA

TPT Time Partition Testing

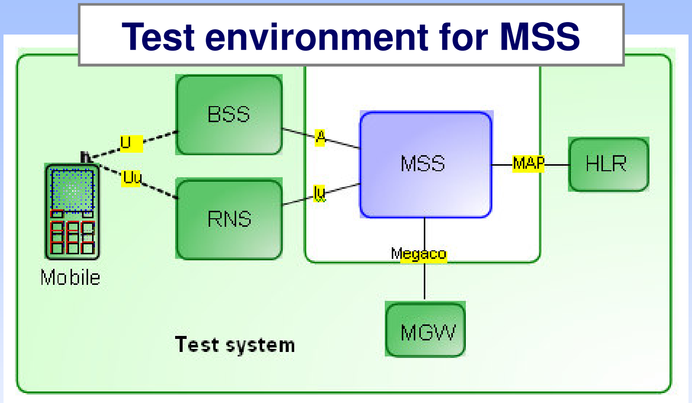
Covering

- model lines
- test stages

NSN telecom case study

- Focus is on model-based test case design,
  but test execution and evaluation is also taken into account
- Goal is to *reduce costs* for test case design by means of model-based test approach
- Network element under test is the *Mobile Switching Server* (MSS):
  responsible for establishing calls and to control the handover of mobiles among different cells
- Three MSS *features will be tested*: location update, voice call, handover
- Models in use: *UML state charts*, the MSS is described with this

➢ Models are built and test cases are generated with *QTronic tool*


Test environment for MSS

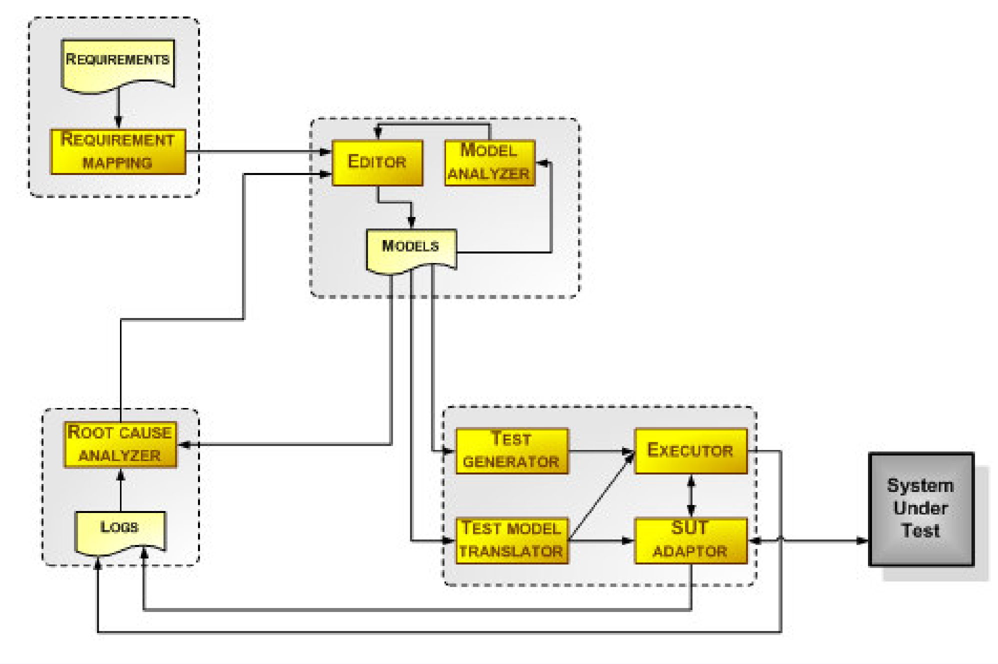
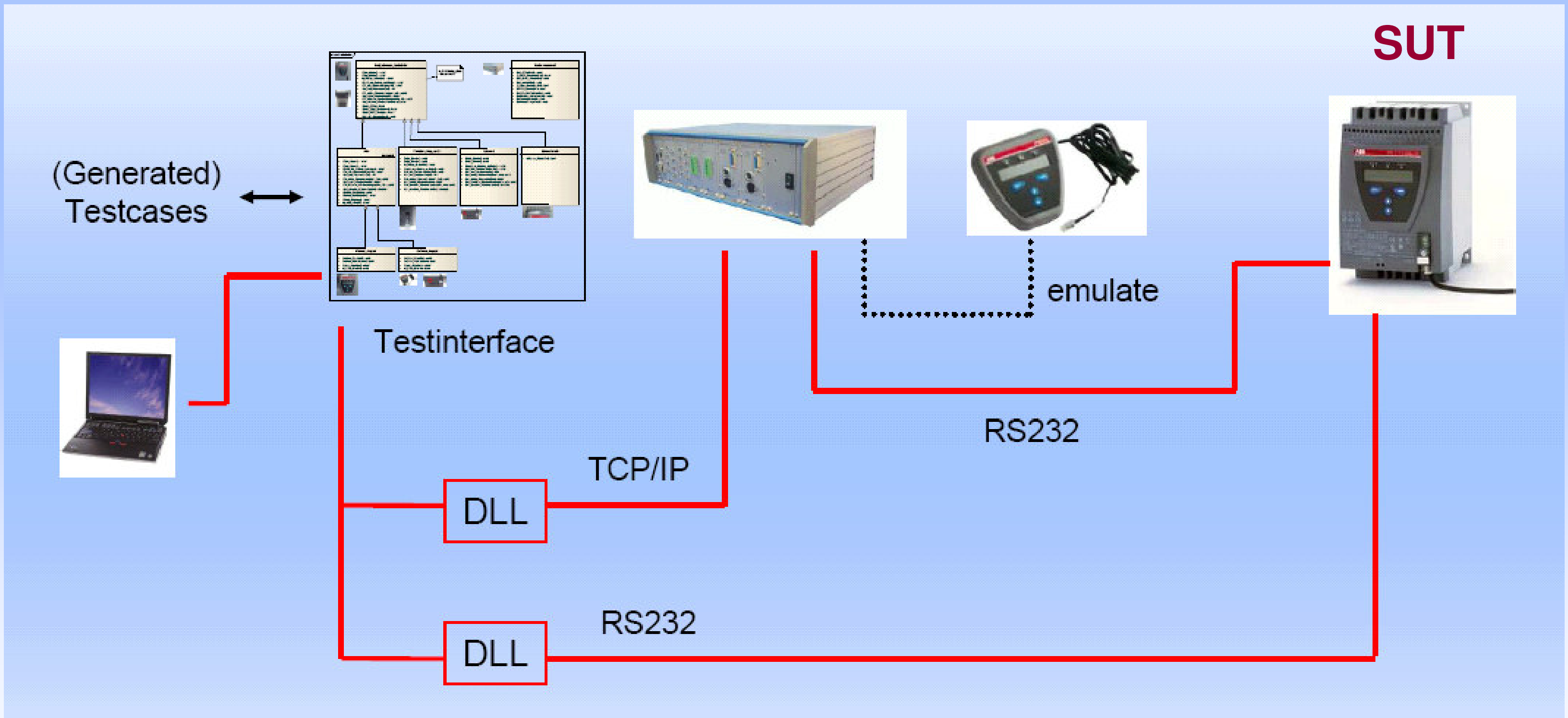


ABB production engineering case study

- System under test is a *soft starter*
  (a device to smoothly start and stop an electrical motor)
- Design models in use: *UML use cases and class diagrams*
- Test model in use: *usage model*

➢ Test model derived from requirements and UML models,
then test cases are derived from test model and exectuted



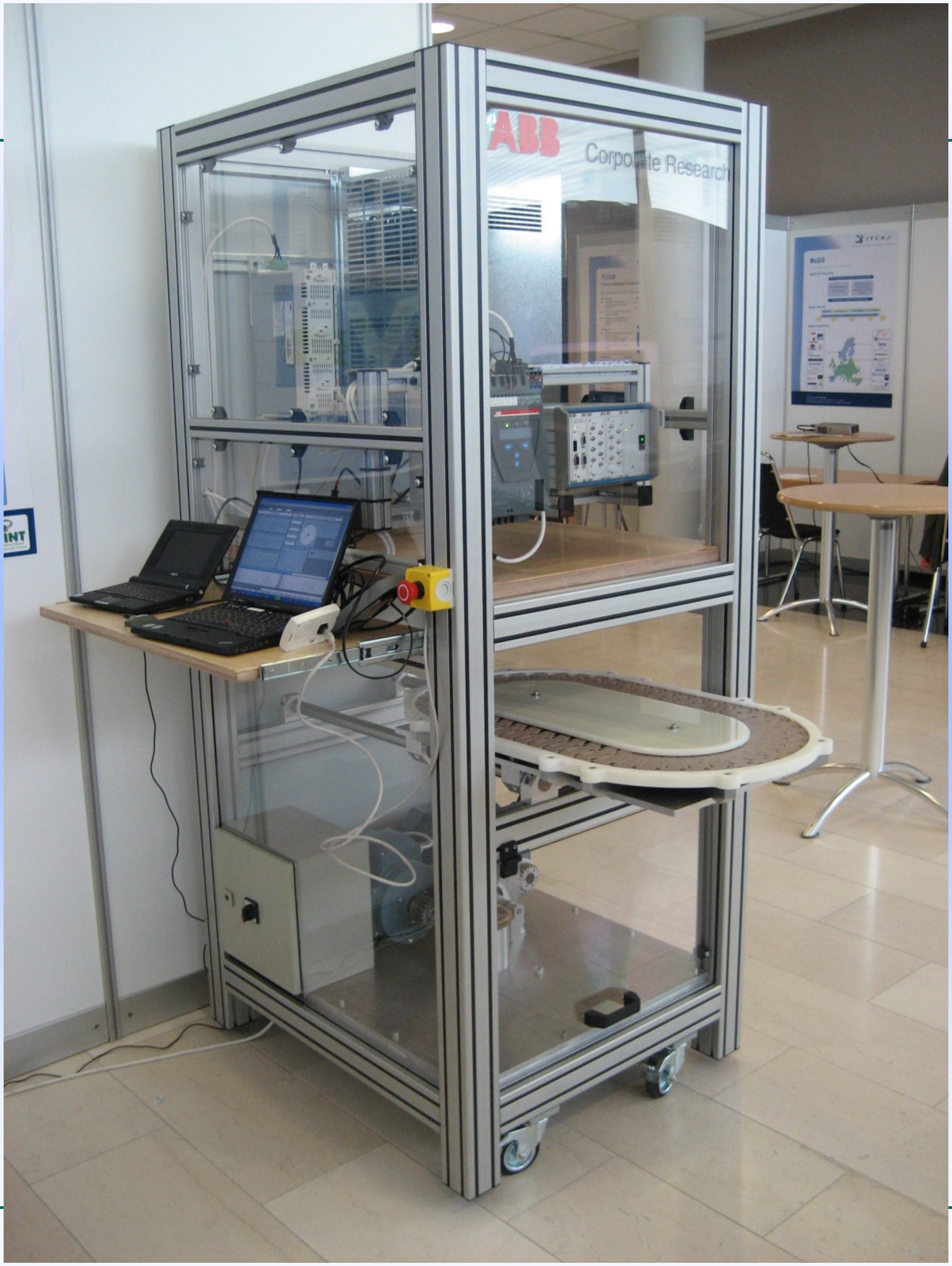
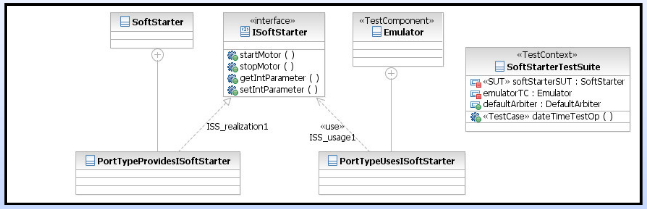
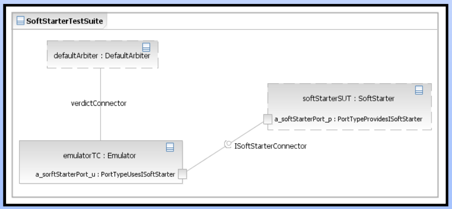


ABB production engineering demonstrator

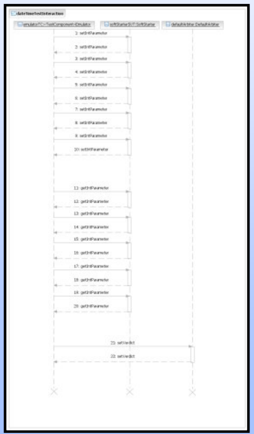
ITEA
Symposium
2008
Rotterdam

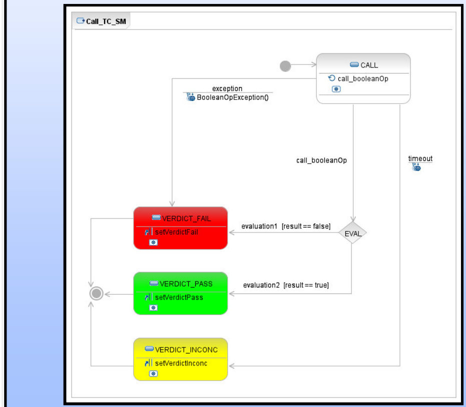
Test generation




SoftStarterTestSuite
defaultArbiter : DefaultArbiter
verdictConnector
softStarterSUT : SoftStarter
a_softStarterPort_p : PortTypeProvidesISoftStarter
ISoftStarterConnector
emulatorTC : Emulator
a_sorftStarterPort_u : PortTypeUsesISoftStarter

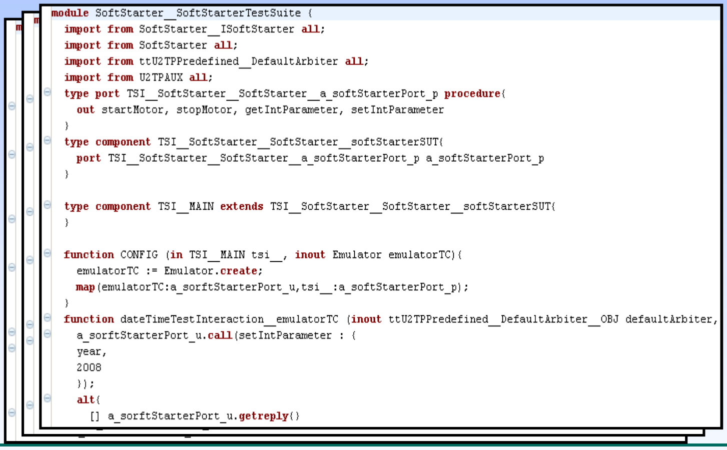
Configuration





Behavior

TTCN-3
Test cases



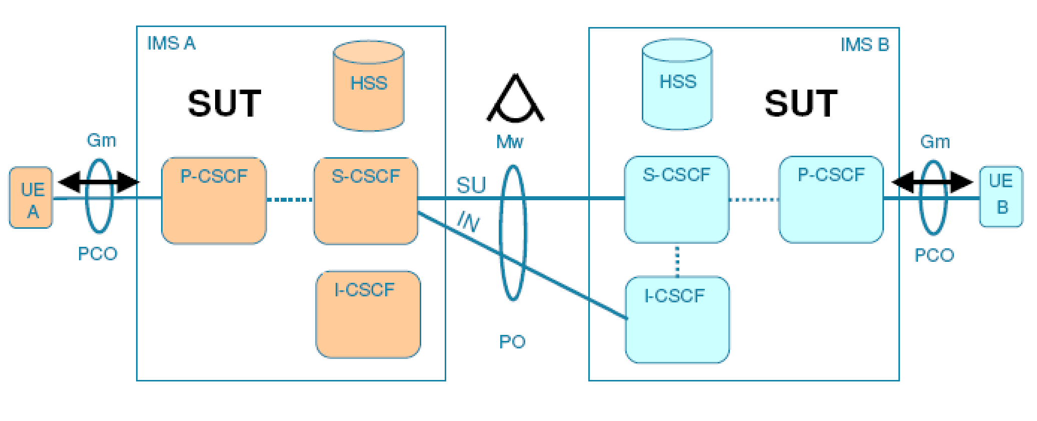
```
module SoftStarter__SoftStarterTestSuite {
  import from SoftStarter__ISoftStarter all;
  import from SoftStarter all;
  import from ttU2TPPredefined__DefaultArbiter all;
  import from U2TPAUX all;
  type port TSI__SoftStarter__SoftStarter__a_softStarterPort_p procedure{
    out startMotor, stopMotor, getIntParameter, setIntParameter
  }
  type component TSI__SoftStarter__SoftStarter__softStarterSUT{
    port TSI__SoftStarter__SoftStarter__a_softStarterPort_p a_softStarterPort_p
  }

  type component TSI__MAIN extends TSI__SoftStarter__SoftStarter__softStarterSUT{
  }

  function CONFIG (in TSI__MAIN tsi__, inout Emulator emulatorTC){
    emulatorTC := Emulator.create;
    map(emulatorTC:a_sorftStarterPort_u,tsi__:a_softStarterPort_p);
  }
  function dateTimeTestInteraction__emulatorTC (inout ttU2TPPredefined__DefaultArbiter__OBJ defaultArbiter,
    a_sorftStarterPort_u.call(setIntParameter : {
    year,
    2008
    });
    alt{
      [] a_sorftStarterPort_u.getreply()
```

ETSI telecom case study

- The interoperability of *IP Multimedia Subsystem (IMS) networks* will be tested
- The case study focuses on the assessment of *interoperability of basic services* (such as basic Voice over IP (VoIP) call and instant messaging between two distinct IMS networks)
- Both functional and conformance tests

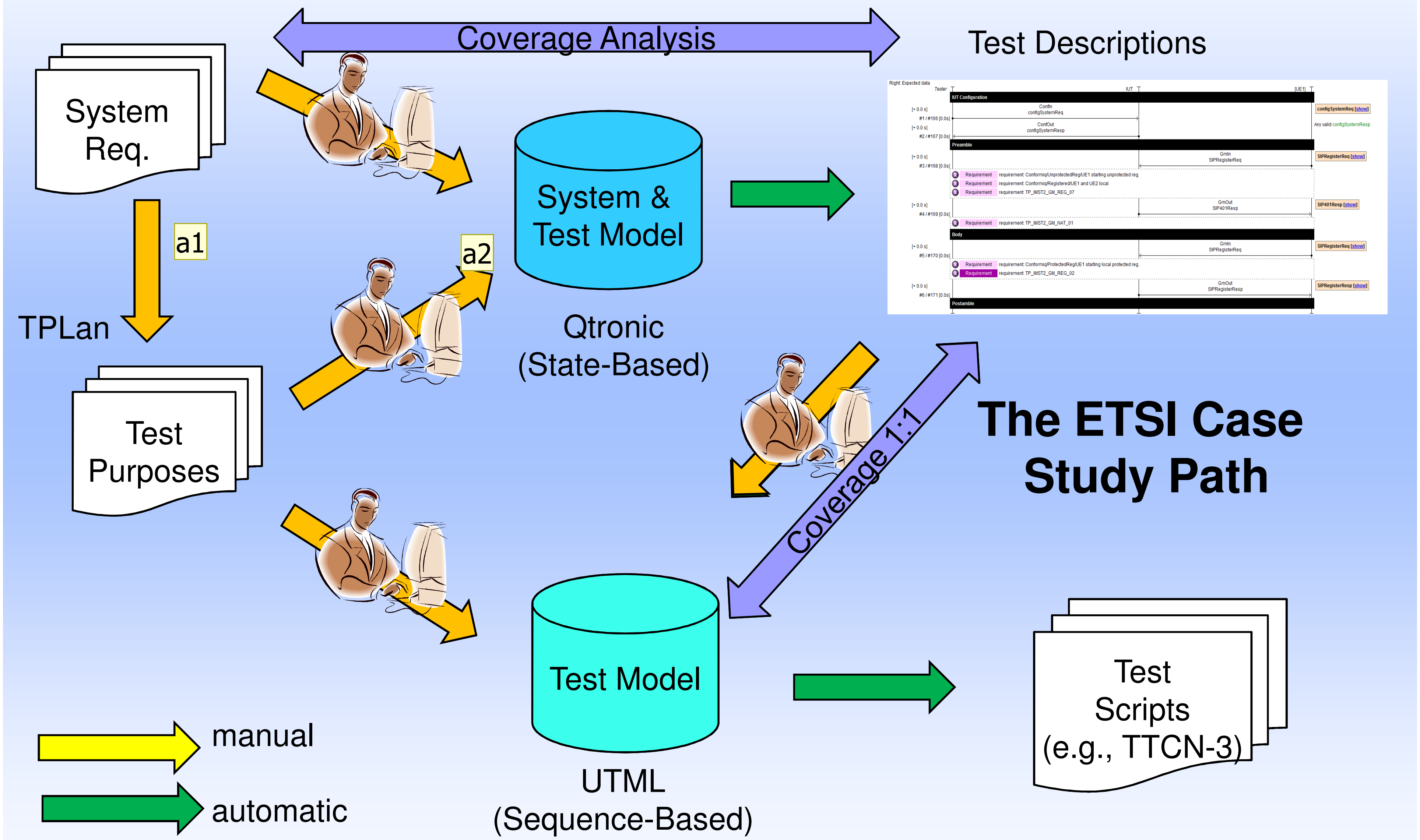
➢ *UML state charts* are used to model the SUT, test cases are derived from this

SUT: System of 2 IP Multimedia networks



UE = User Equipment (Terminal)
CSCF = Call Session Control Function (~Proxy)
HSS = Home Subscriber Function (~User Database)

ETSI Case Study: Overview



Right: Expected data

Tester | IUT | [UE1]

IUT Configuration

[+ 0.0 s] #1 / #166 [0.0s] — ConfIn configSystemReq — configSystemReq [show]

[+ 0.0 s] #2 / #167 [0.0s] — ConfOut configSystemResp — Any valid configSystemResp

Preamble

[+ 0.0 s] #3 / #168 [0.0s] — GmIn SIPRegisterReq — SIPRegisterReq [show]

- Requirement requirement: Conformiq/UnprotectedReg/UE1 starting unprotected reg.
- Requirement requirement: Conformiq/Registered/UE1 and UE2 local
- Requirement requirement: TP_IMST2_GM_REG_07

[+ 0.0 s] #4 / #169 [0.0s] — GmOut SIP401Resp — SIP401Resp [show]

- Requirement requirement: TP_IMST2_GM_NAT_01

Body

[+ 0.0 s] #5 / #170 [0.0s] — GmIn SIPRegisterReq — SIPRegisterReq [show]

- Requirement requirement: Conformiq/ProtectedReg/UE1 starting local protected reg.
- Requirement requirement: TP_IMST2_GM_REG_02

[+ 0.0 s] #6 / #171 [0.0s] — GmOut SIPRegisterResp — SIPRegisterResp [show]

Postamble

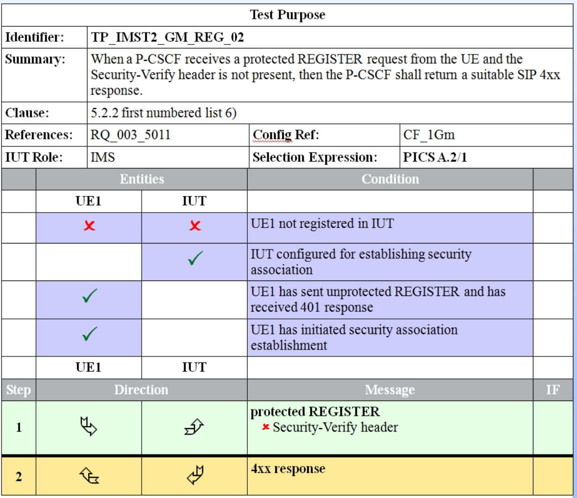
Folie 79

a1 Not sure, if this arrow is needed. Is this a transformation or are these two items equivalent ?
alain; 27.01.2009

a2 I was told (by Axel), that this option has been used exceptionally in this case study and that the "normal" way was to use system requirements as input for Qtronic modelling. Does it mean, that the vision is to generate TPs (TPLan) from Qtronic test descriptions and that the manual step of deriving TPs from requirements will not be required anymore?
alain; 27.01.2009

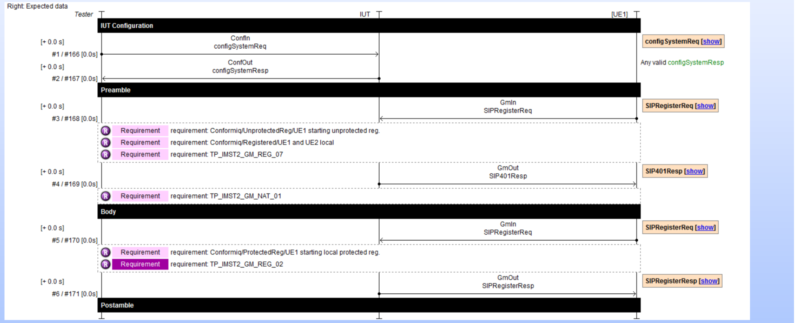
ETSI Case Study: Example

TPLan



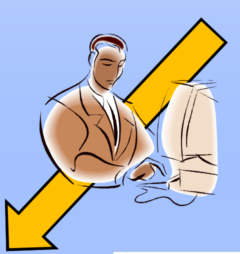
Test Purpose				
Identifier:	TP_IMST2_GM_REG_02			
Summary:	When a P-CSCF receives a protected REGISTER request from the UE and the Security-Verify header is not present, then the P-CSCF shall return a suitable SIP 4xx response.			
Clause:	5.2.2 first numbered list 6)			
References:	RQ_003_5011		Config Ref:	CF_1Gm
IUT Role:	IMS		Selection Expression:	PICS A.2/1
	Entities		Condition	
	UE1	IUT		
	✗	✗	UE1 not registered in IUT	
		✓	IUT configured for establishing security association	
	✓		UE1 has sent unprotected REGISTER and has received 401 response	
	✓		UE1 has initiated security association establishment	
	UE1	IUT		
Step	Direction		Message	IF
1	→	→	protected REGISTER ✗ Security-Verify header	
2	←	←	4xx response	

Qtronic
(State-Based)

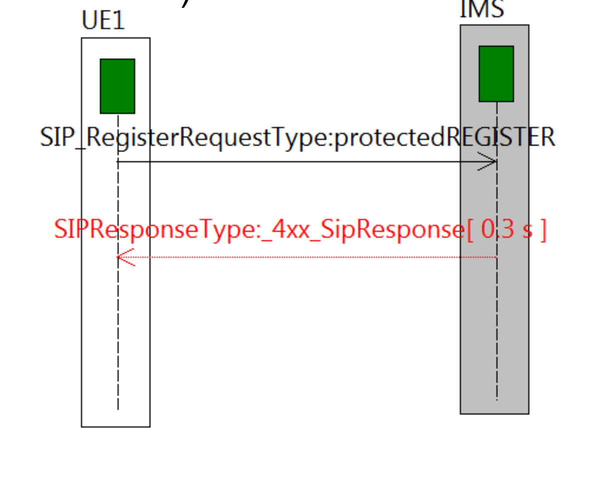


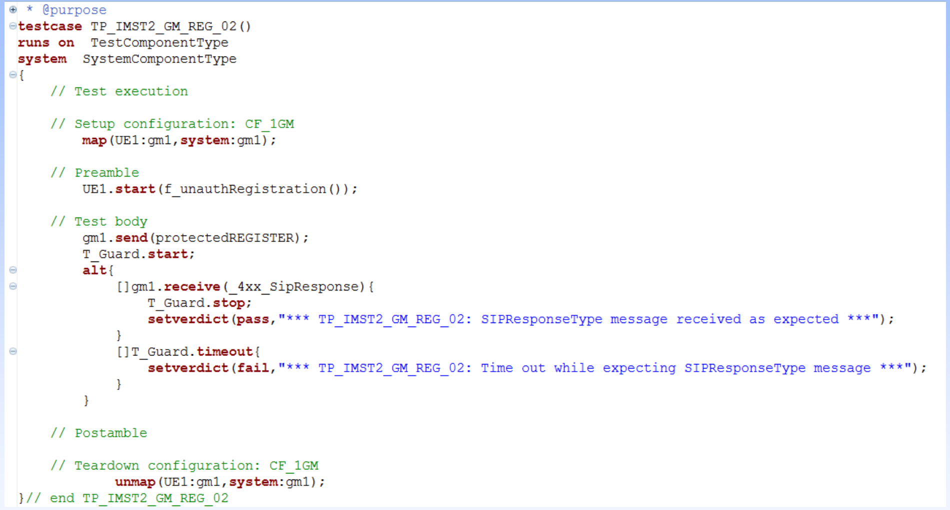
Right: Expected data

	Tester		IUT		[UE1]	
	IUT Configuration					
[+ 0.0 s] #1 / #166 [0.0s]		ConfIn configSystemReq →				configSystemReq [show]
[+ 0.0 s] #2 / #167 [0.0s]		ConfOut configSystemResp ←				Any valid configSystemResp
	Preamble					
[+ 0.0 s] #3 / #168 [0.0s]				GmIn SIPRegisterReq ←		SIPRegisterReq [show]
	Requirement	requirement: Conformiq/UnprotectedReg/UE1 starting unprotected reg.				
	Requirement	requirement: Conformiq/Registered/UE1 and UE2 local				
	Requirement	requirement: TP_IMST2_GM_REG_07				
[+ 0.0 s] #4 / #169 [0.0s]				GmOut SIP401Resp →		SIP401Resp [show]
	Requirement	requirement: TP_IMST2_GM_NAT_01				
	Body					
[+ 0.0 s] #5 / #170 [0.0s]				GmIn SIPRegisterReq ←		SIPRegisterReq [show]
	Requirement	requirement: Conformiq/ProtectedReg/UE1 starting local protected reg.				
	Requirement	requirement: TP_IMST2_GM_REG_02				
[+ 0.0 s] #6 / #171 [0.0s]				GmOut SIPRegisterResp →		SIPRegisterResp [show]
	Postamble					



UTML
Pattern-Oriented
(Sequence-Based)


UE1
IMS
SIP_RegisterRequestType:protectedREGISTER
SIPResponseType:_4xx_SipResponse[0.3 s]



```
 * @purpose
testcase TP_IMST2_GM_REG_02()
runs on  TestComponentType
system  SystemComponentType
{
    // Test execution

    // Setup configuration: CF_1GM
        map(UE1:gm1,system:gm1);

    // Preamble
        UE1.start(f_unauthRegistration());

    // Test body
        gm1.send(protectedREGISTER);
        T_Guard.start;
        alt{
            []gm1.receive(_4xx_SipResponse){
                T_Guard.stop;
                setverdict(pass,"*** TP_IMST2_GM_REG_02: SIPResponseType message received as expected ***");
            }
            []T_Guard.timeout{
                setverdict(fail,"*** TP_IMST2_GM_REG_02: Time out while expecting SIPResponseType message ***");
            }
        }

    // Postamble

    // Teardown configuration: CF_1GM
            unmap(UE1:gm1,system:gm1);
}// end TP_IMST2_GM_REG_02
```

Qtronic@ETSI Case Study: Status

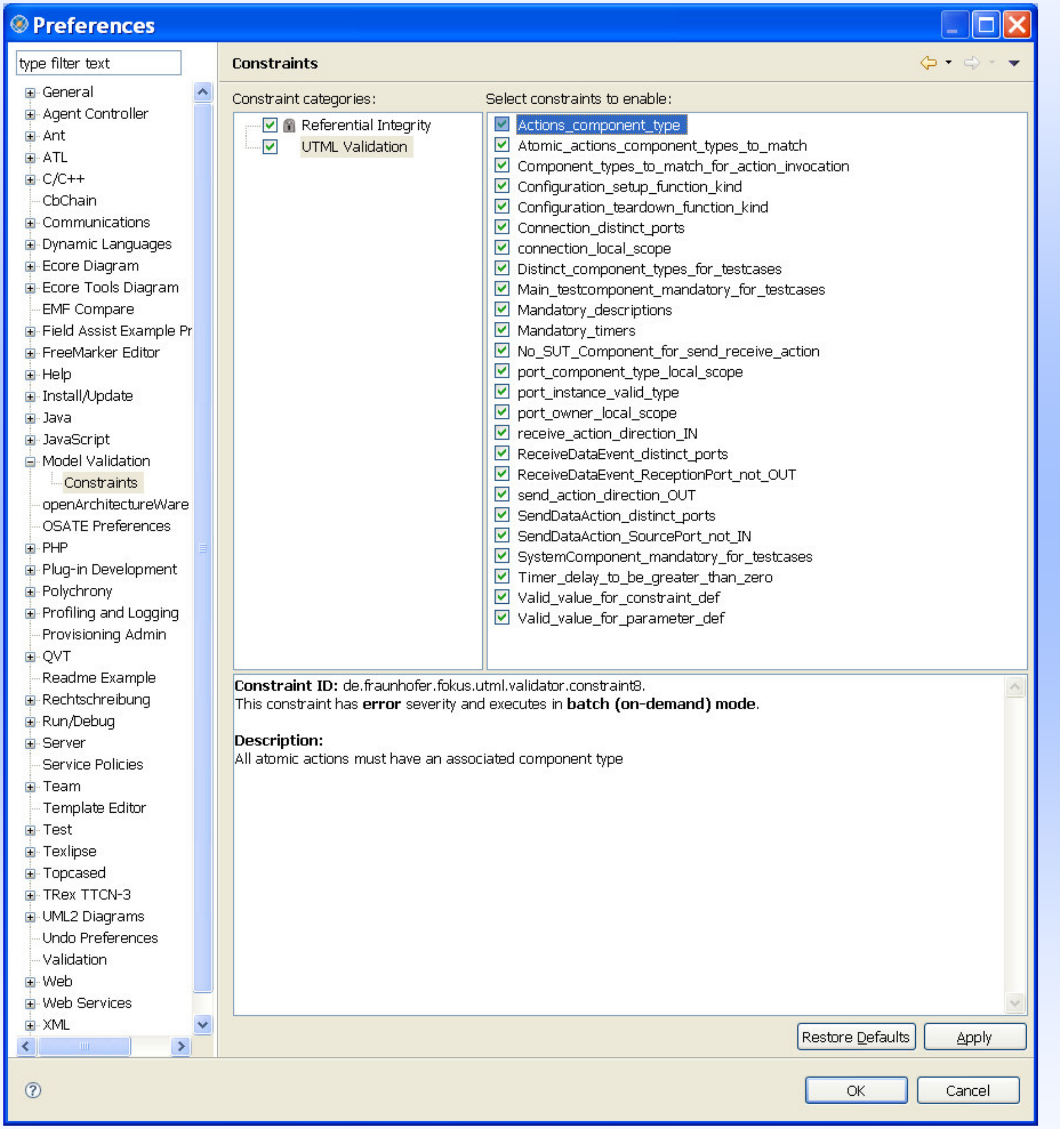
- Automatic generation of ETSI defined Test Purposes (TP) for the 3GPP IP Multimedia Subsystem (IMS)
- Conformiq was to create a model that would cover the TPs from the existing ETSI TP documentation (DTS/TISPAN-06035-2 V002F)
- The TPs described in the documentation are written for IMS core network functionality that is accessible through SIP based interfaces
- All generated test purposes were presented in HTML format for manual inspection and comparison against existing TPs

UTML Implementation status

- UTML Metamodel: Done, open for improvements
- Prototype Tool Chain Architecture: Done
- Prototype Tool Chain Implementation: Version 1.4.0
- New Features
  - Test Model Quality
    - New OCL-Constraints
    - HTML-Reporting for statistics and documentation
  - Comparison of test models
    - To trace changes
    - For version checking
    - Allows parallel processing of test model in teams

MDTester: OCL-Based Test Model Quality Check

- 25 Built-in OCL-Constraints
- API allows for further OCL queries/check to be added for validation or statistics



Constraints

Constraint categories:

- ☑ Referential Integrity
- ☑ UTML Validation

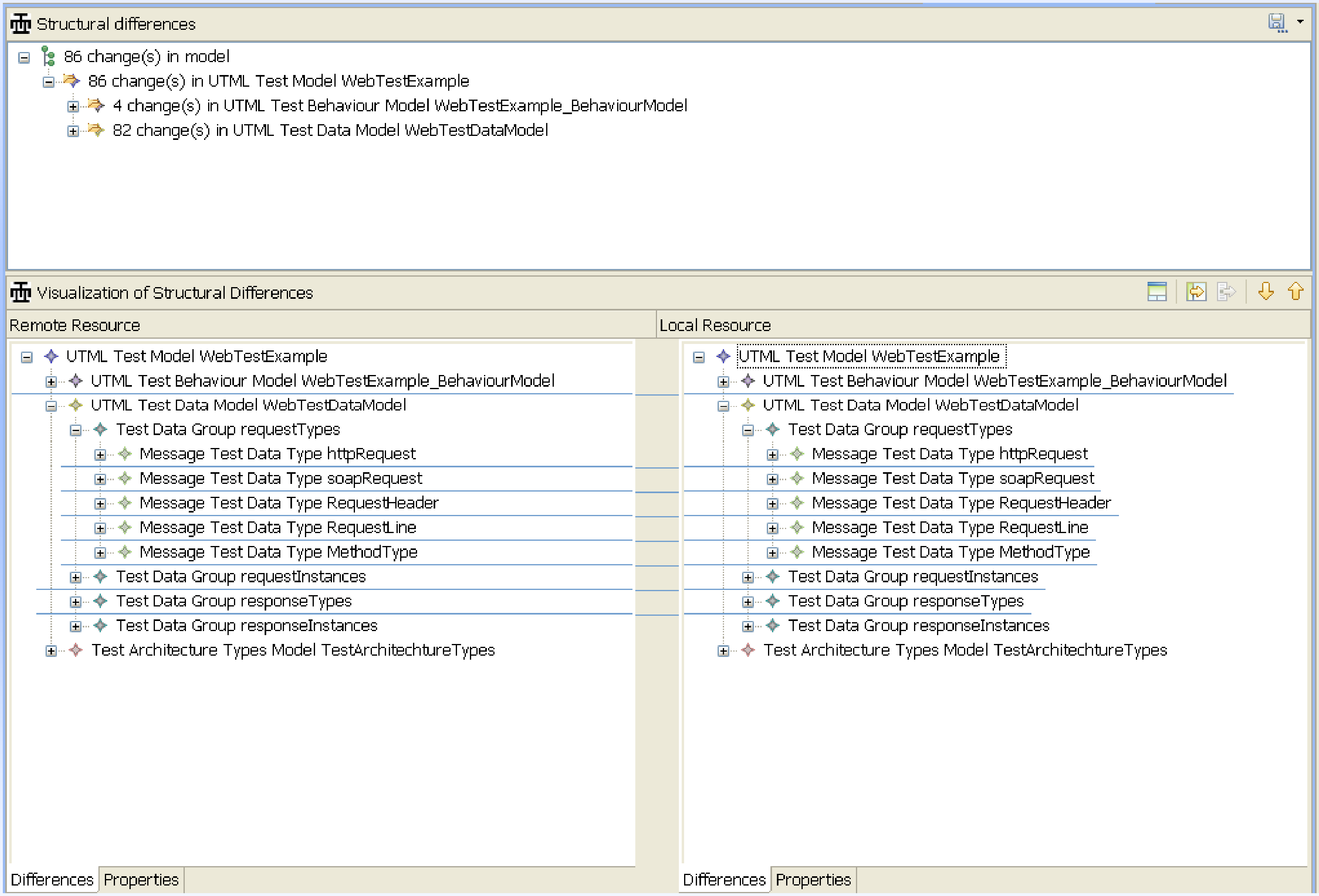
Select constraints to enable:

- ☑ Actions_component_type
- ☑ Atomic_actions_component_types_to_match
- ☑ Component_types_to_match_for_action_invocation
- ☑ Configuration_setup_function_kind
- ☑ Configuration_teardown_function_kind
- ☑ Connection_distinct_ports
- ☑ connection_local_scope
- ☑ Distinct_component_types_for_testcases
- ☑ Main_testcomponent_mandatory_for_testcases
- ☑ Mandatory_descriptions
- ☑ Mandatory_timers
- ☑ No_SUT_Component_for_send_receive_action
- ☑ port_component_type_local_scope
- ☑ port_instance_valid_type
- ☑ port_owner_local_scope
- ☑ receive_action_direction_IN
- ☑ ReceiveDataEvent_distinct_ports
- ☑ ReceiveDataEvent_ReceptionPort_not_OUT
- ☑ send_action_direction_OUT
- ☑ SendDataAction_distinct_ports
- ☑ SendDataAction_SourcePort_not_IN
- ☑ SystemComponent_mandatory_for_testcases
- ☑ Timer_delay_to_be_greater_than_zero
- ☑ Valid_value_for_constraint_def
- ☑ Valid_value_for_parameter_def

Constraint ID: de.fraunhofer.fokus.utml.validator.constraint8.
This constraint has **error** severity and executes in **batch (on-demand) mode**.

Description:
All atomic actions must have an associated component type

MDTester: Structural Comparison of Test Models



Structural differences

- 86 change(s) in model
  - 86 change(s) in UTML Test Model WebTestExample
    - 4 change(s) in UTML Test Behaviour Model WebTestExample_BehaviourModel
    - 82 change(s) in UTML Test Data Model WebTestDataModel

Visualization of Structural Differences

Remote Resource

- UTML Test Model WebTestExample
  - UTML Test Behaviour Model WebTestExample_BehaviourModel
  - UTML Test Data Model WebTestDataModel
    - Test Data Group requestTypes
      - Message Test Data Type httpRequest
      - Message Test Data Type soapRequest
      - Message Test Data Type RequestHeader
      - Message Test Data Type RequestLine
      - Message Test Data Type MethodType
    - Test Data Group requestInstances
    - Test Data Group responseTypes
    - Test Data Group responseInstances
  - Test Architecture Types Model TestArchitechtureTypes

Local Resource

- UTML Test Model WebTestExample
  - UTML Test Behaviour Model WebTestExample_BehaviourModel
  - UTML Test Data Model WebTestDataModel
    - Test Data Group requestTypes
      - Message Test Data Type httpRequest
      - Message Test Data Type soapRequest
      - Message Test Data Type RequestHeader
      - Message Test Data Type RequestLine
      - Message Test Data Type MethodType
    - Test Data Group requestInstances
    - Test Data Group responseTypes
    - Test Data Group responseInstances
  - Test Architecture Types Model TestArchitechtureTypes

Activities

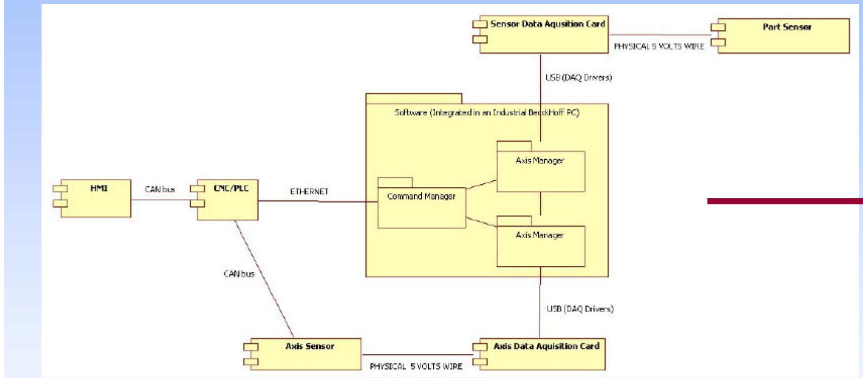
- UTML Web Site
  - English version online, but not yet published
  - Deutsch: Work in progress
- MDTester 1.0.0 Release to share IMS Test Model
  - Implementation & Bug fixes
  - Update site for Installation: Done
  - User Guide and Installation Manual
- Outlook
  - Front-End Plugin for Automated Transformation From TTCN-3 (e.g. Test Data, Test Behaviour model)
  - Further Back-End Plugins to export into other notations

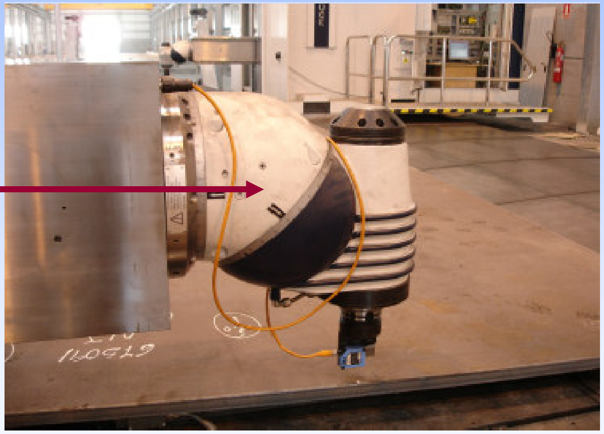
Soraluce/Ideko production engineering case study

- SUT: DIGITMILL mechatronic solution as part of a *milling machine*
- Focus in this case study is to get a more systematic test process based on MBT
- Models in use: *several UML diagrams* (component, architectural, sequence, state diagrams)

➢ Test case derivation from UML diagrams

SUT: DIGITMILL



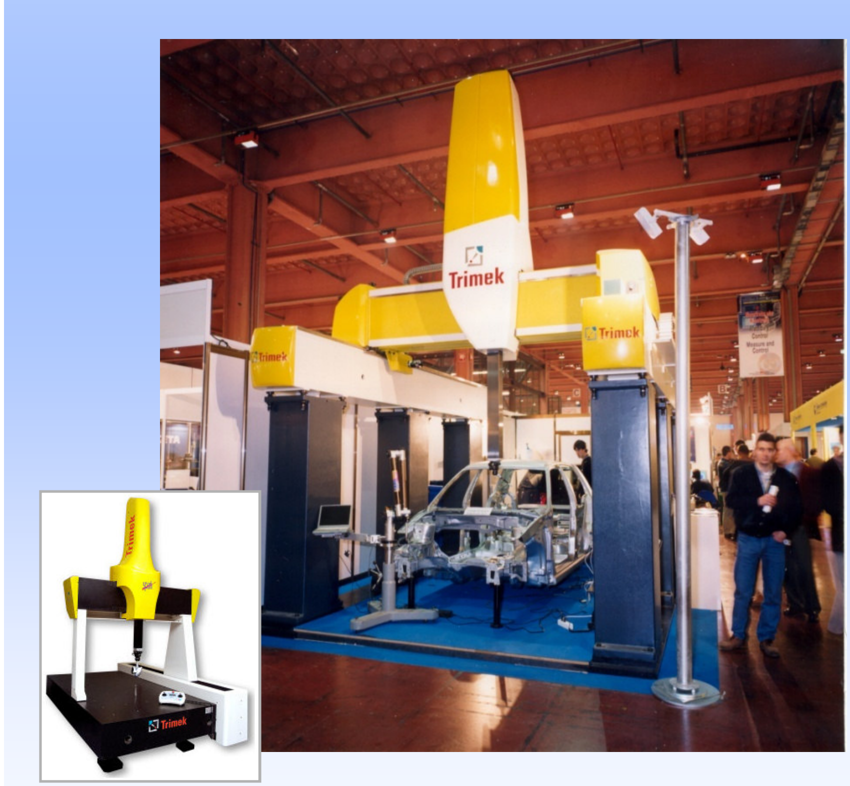


Trimek/Datapixel production engineering case study

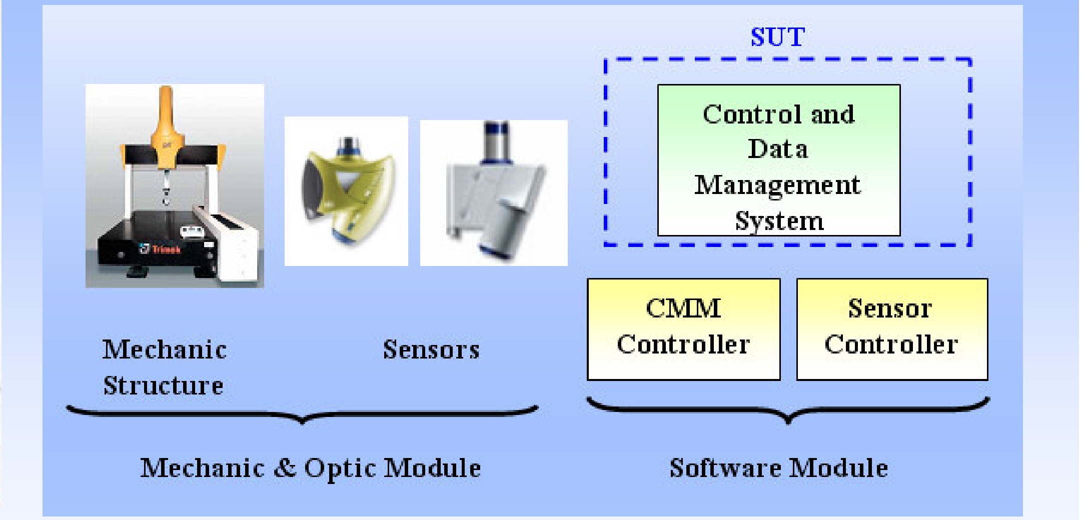
- SUT: Coordinates Measuring Machines (CMM) control software (CDMS) for controlling a measuring system

➢ Focus: test case derivation from UML models

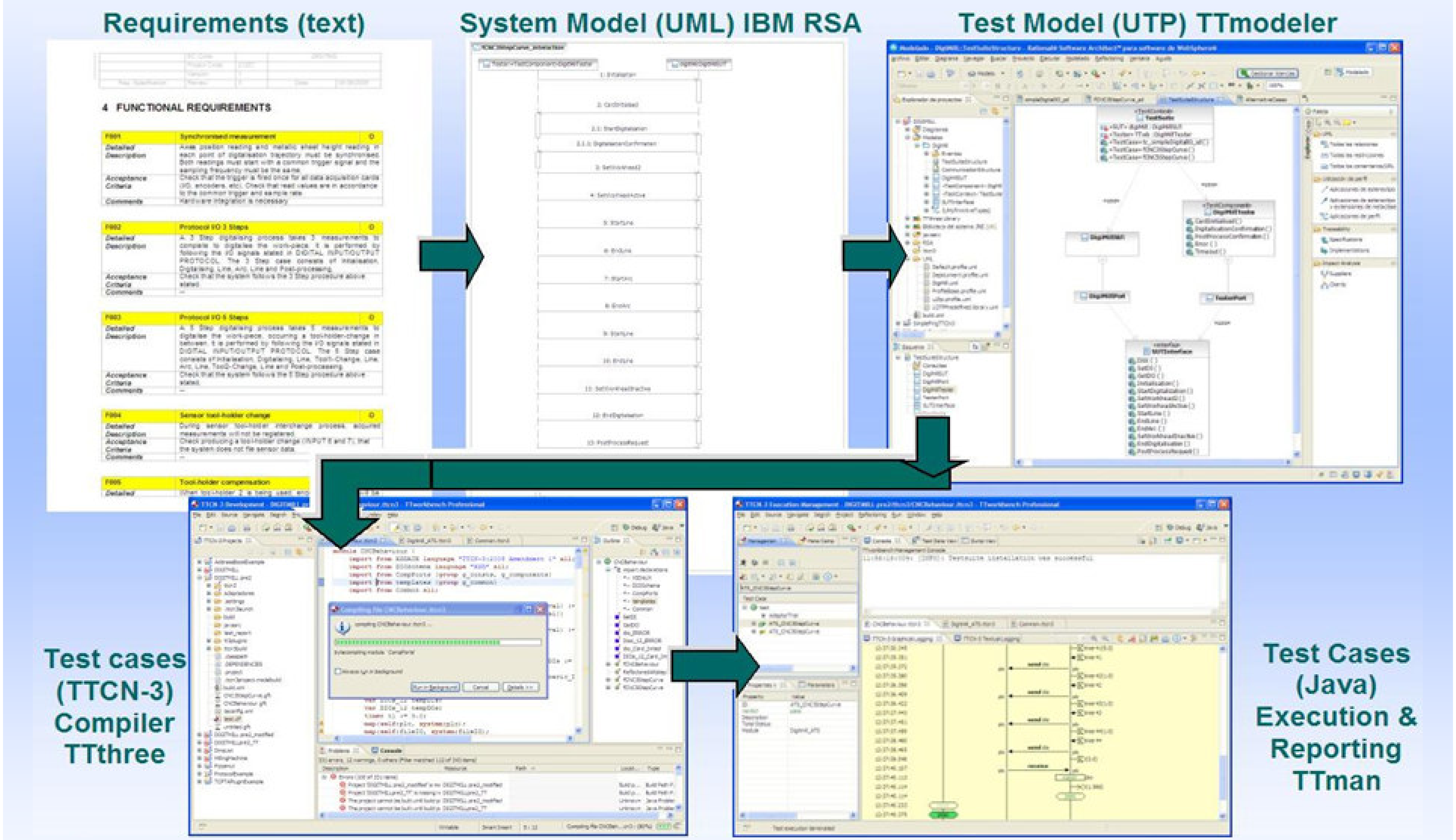
- Models in use: *UML class, sequence, state diagrams*



SUT: measuring system



TTCN-3 @ Trimek/Datapixel



Requirements (text)

4 FUNCTIONAL REQUIREMENTS

F001	Synchronized measurement	0
Detailed Description	Axes position reading and metallic sheet height reading in each point of digitalisation trajectory must be synchronised. Both readings must start with a common trigger signal and the sampling frequency must be the same.	
Acceptance Criteria	Check that the trigger is fired once for all data acquisition cards (I/O, encoders, etc). Check that read values are in accordance to the common trigger and sample rate.	
Comments	Hardware integration is necessary	

F002	Protocol I/O 3 Steps	0
Detailed Description	A 3 Step digitalising process takes 3 measurements to complete to digitalise the work-piece. It is performed by following the I/O signals stated in DIGITAL INPUT/OUTPUT PROTOCOL. The 3 Step case consists of Initialisation, Digitalising, Line, Arc, Line and Post-processing.	
Acceptance Criteria	Check that the system follows the 3 Step procedure above stated.	
Comments	--	

F003	Protocol I/O 5 Steps	0
Detailed Description	A 5 Step digitalising process takes 5 measurements to digitalise the work-piece, occurring a tool-holder-change in between. It is performed by following the I/O signals stated in DIGITAL INPUT/OUTPUT PROTOCOL. The 5 Step case consists of Initialisation, Digitalising, Line, Tool1-Change, Line, Arc, Line, Tool2-Change, Line and Post-processing.	
Acceptance Criteria	Check that the system follows the 5 Step procedure above stated.	
Comments	--	

F004	Sensor tool-holder change	0
Detailed Description	During sensor tool-holder interchange process, acquired measurements will not be registered.	
Acceptance Criteria	Check producing a tool-holder change (INPUT 6 and 7), that the system does not file sensor data.	
Comments	--	

F005	Tool-holder compensation
Detailed	When tool-holder 2 is being used, and

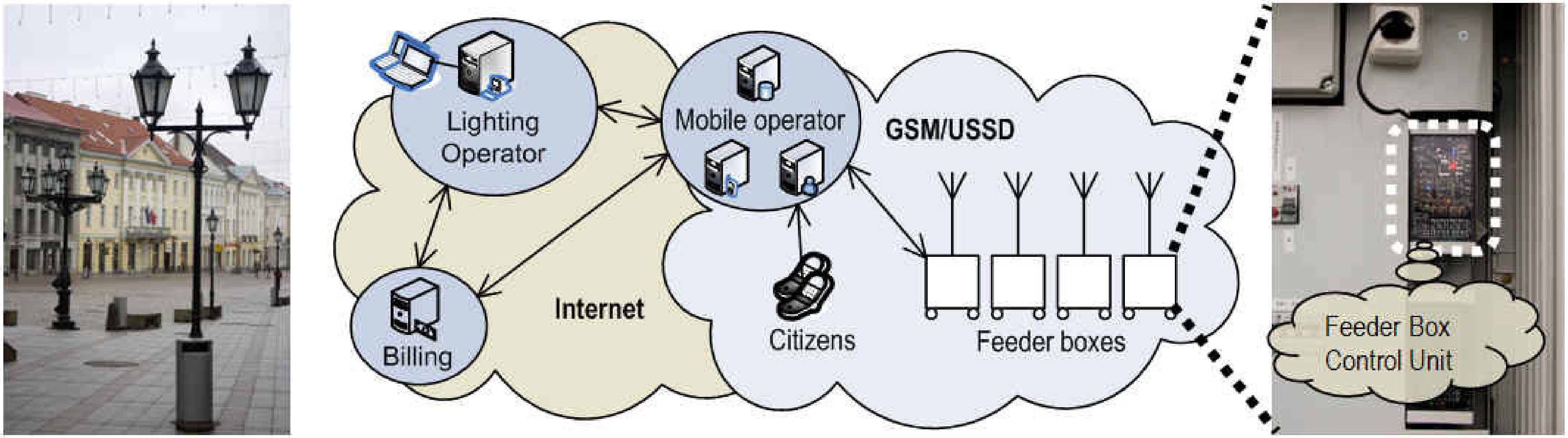
System Model (UML) IBM RSA

Test Model (UTP) TTmodeler

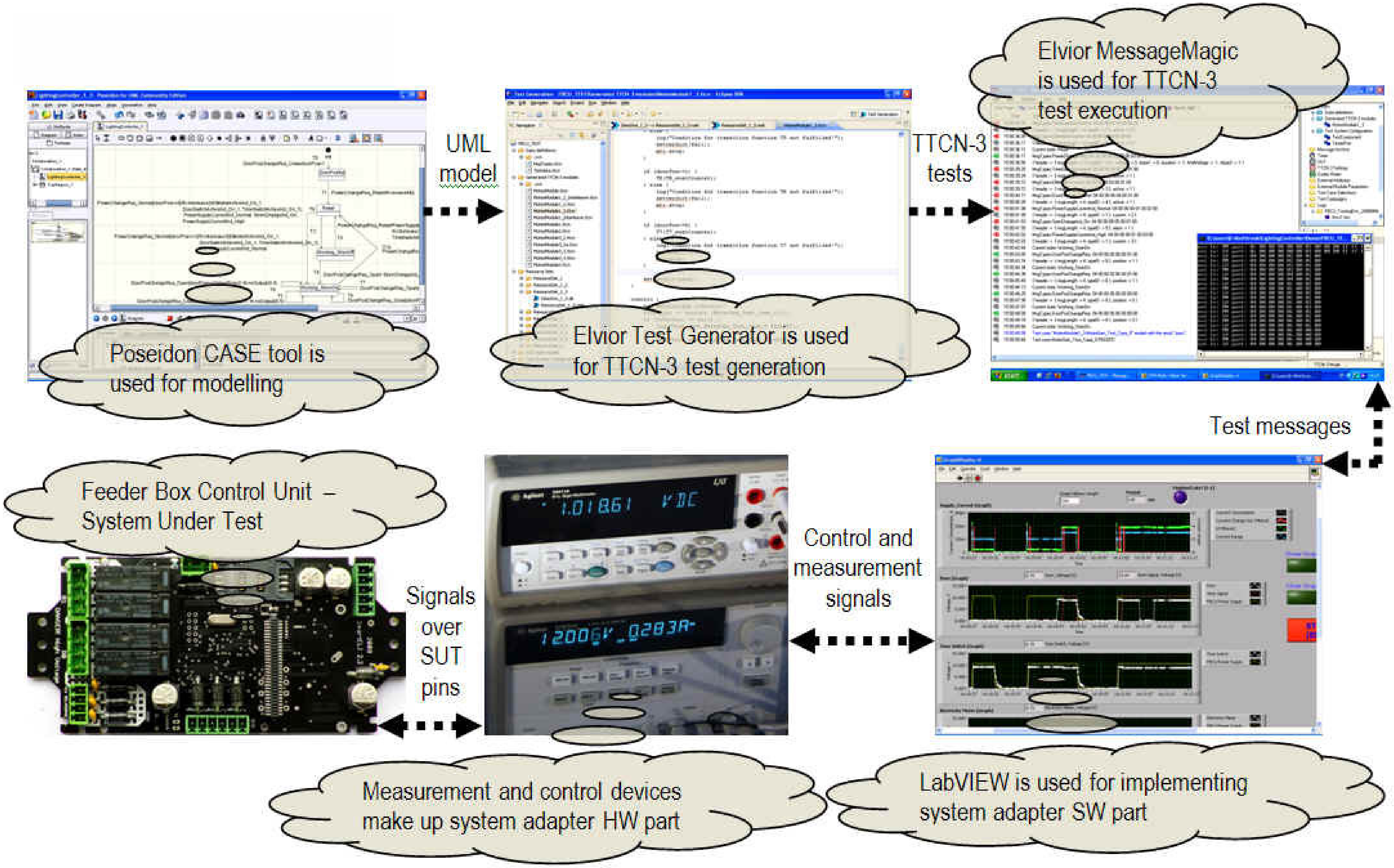
Test cases (TTCN-3) Compiler TTthree

Test Cases (Java) Execution & Reporting TTman

Eliko case study: Street Lighting System



Demonstrator



Poseidon CASE tool is used for modelling

UML model

Elvior Test Generator is used for TTCN-3 test generation

TTCN-3 tests

Elvior MessageMagic is used for TTCN-3 test execution

Test messages

Feeder Box Control Unit – System Under Test

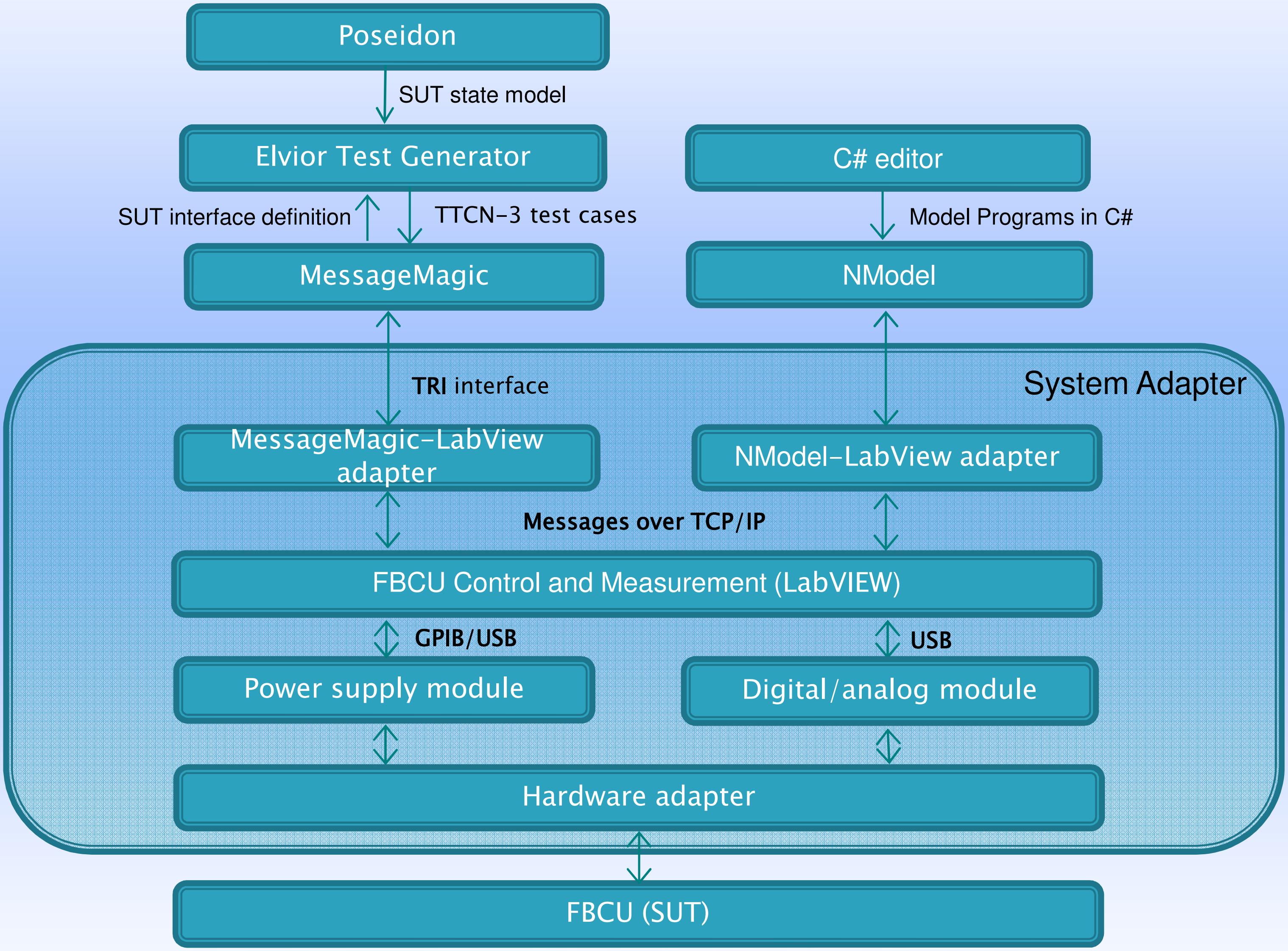
Signals over SUT pins

Control and measurement signals

Measurement and control devices make up system adapter HW part

LabVIEW is used for implementing system adapter SW part

Demonstrator general architecture

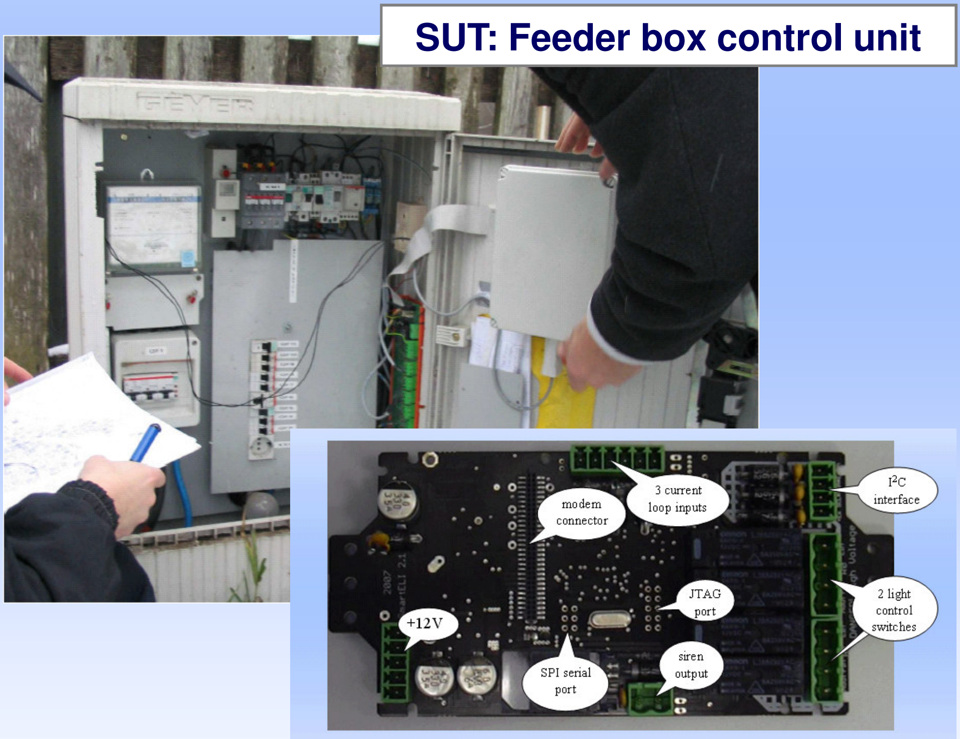


Eliko city street lights case study

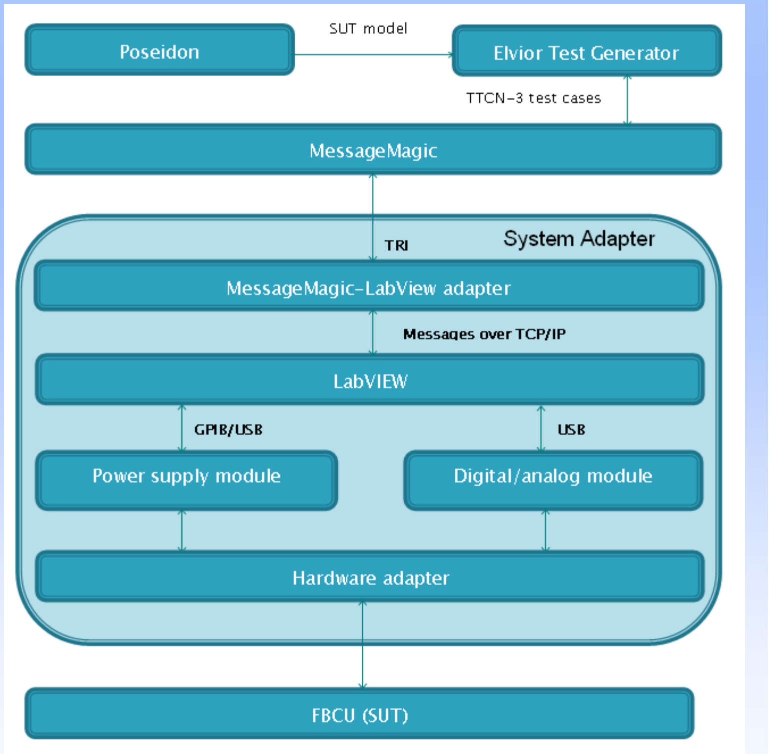
- SUT: Eliko *street lighting control system* feeder box control unit (FBCU)
- Models for the SUT: *UML state charts*, produced with tool Poseidon

➢ Elvior test generator derives *TTCN-3 test cases* from state charts

SUT: Feeder box control unit



Test system architecture


Poseidon

SUT model

Elvior Test Generator

TTCN-3 test cases

MessageMagic

TRI

System Adapter

MessageMagic-LabView adapter

Messages over TCP/IP

LabVIEW

GPIB/USB

USB

Power supply module

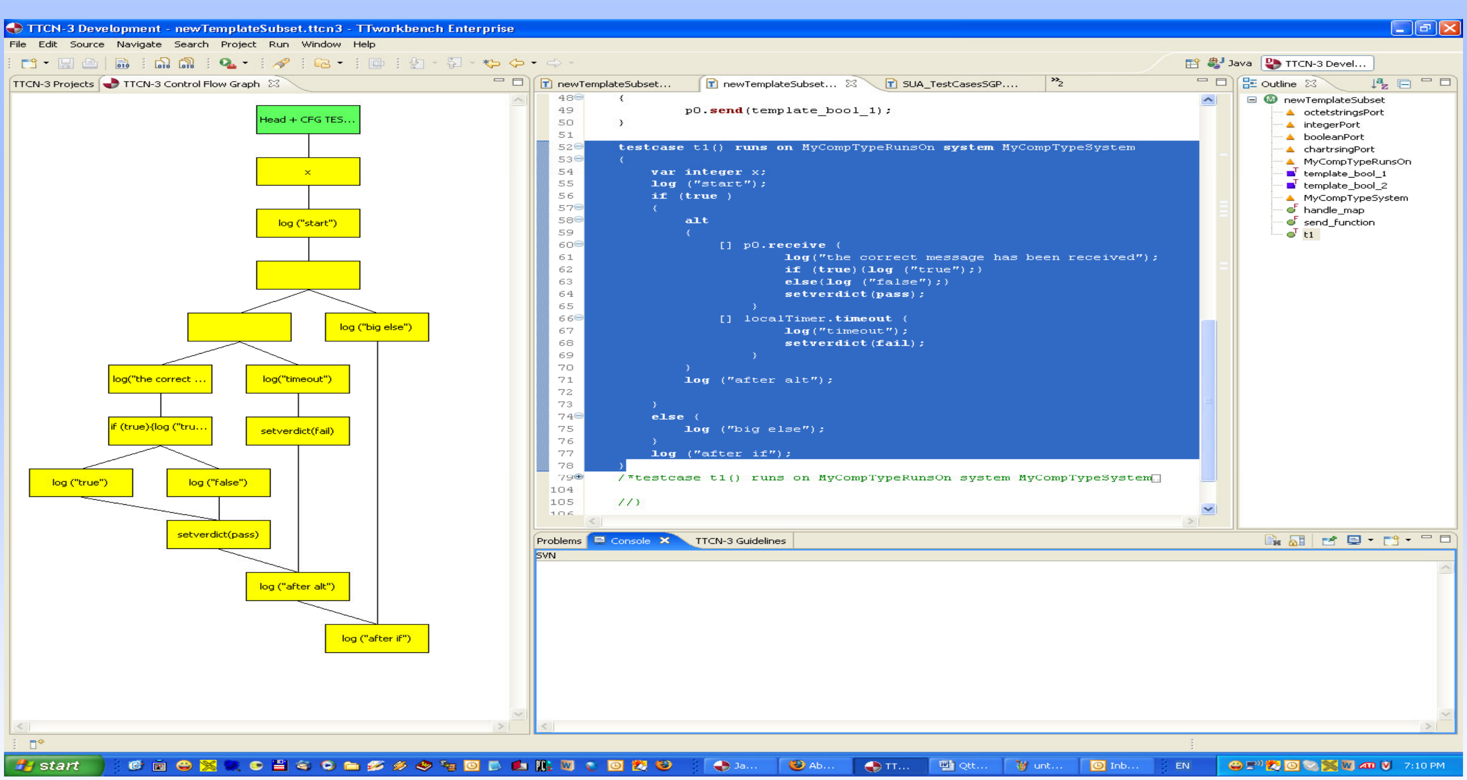
Digital/analog module

Hardware adapter

FBCU (SUT)

AddOn: Test quality

- TTCN-3 testcase behaviour -> Control flow graphs (CFG) -> Contraints finder
- CFG complexity (analysis) indicator -> recommend simplification
- Guideline checker (naming conventions)



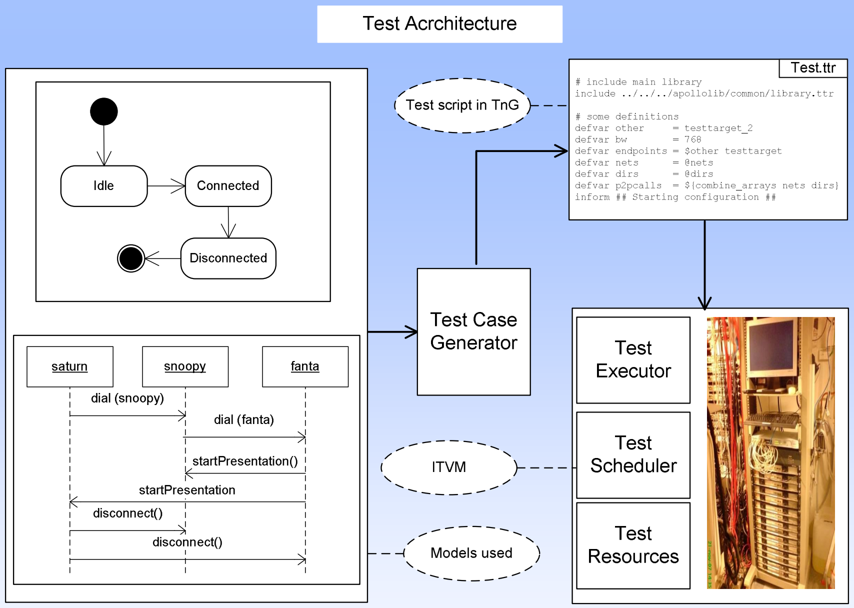
```
    {
            p0.send(template_bool_1);
    }

    testcase t1() runs on MyCompTypeRunsOn system MyCompTypeSystem
    {
        var integer x;
        log ("start");
        if (true )
        {
            alt
            {
                [] p0.receive {
                        log("the correct message has been received");
                        if (true){log ("true");}
                        else{log ("false");}
                        setverdict(pass);
                    }
                [] localTimer.timeout {
                        log("timeout");
                        setverdict(fail);
                    }
            }
            log ("after alt");

        }
        else {
            log ("big else");
        }
        log ("after if");
    }
    /*testcase t1() runs on MyCompTypeRunsOn system MyCompTypeSystem

    //}
```

Tandberg case study

- SUT: *Video conferencing systems* with support for
  multiple simultaneous calls and presentations

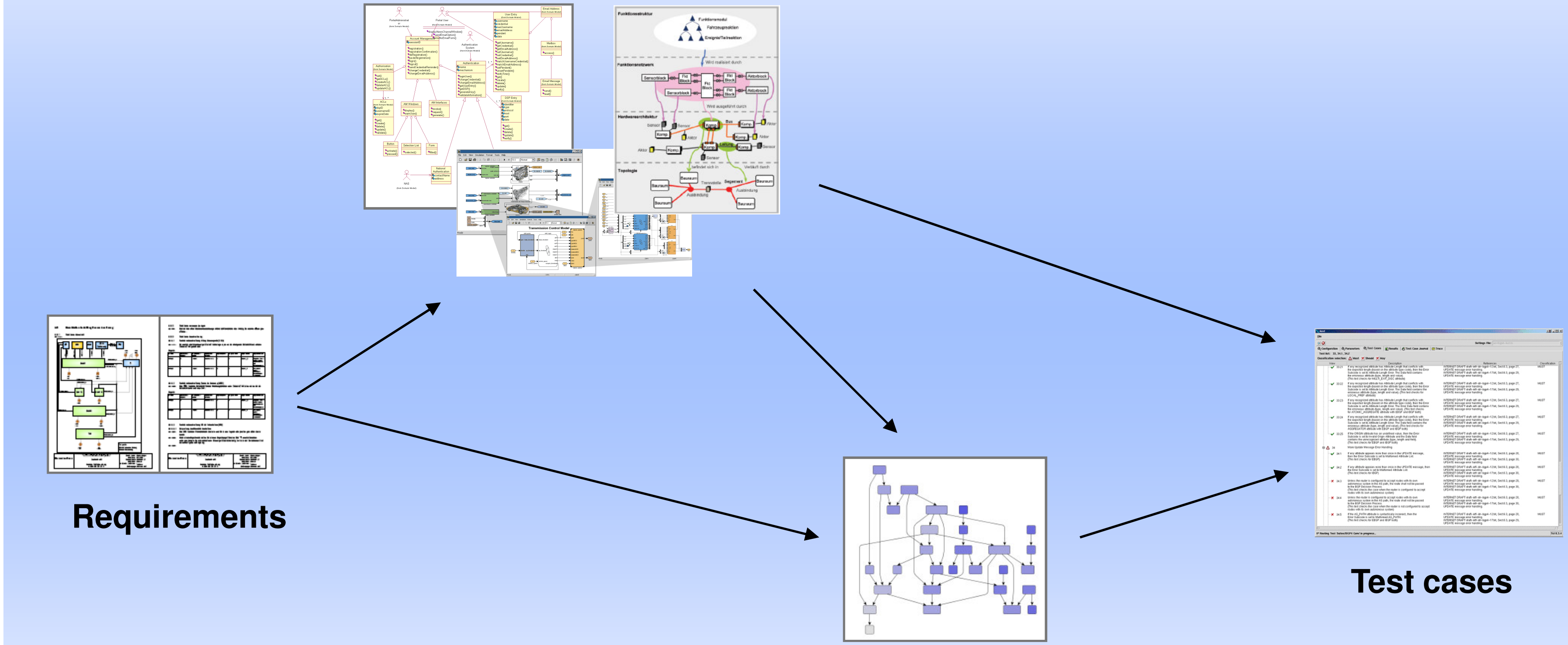
➢ Focus: Model-based functional, stress, and robustness testing

- Models in use: *UML state machines, sequence diagrams*
  (and profiles such as MARTE and UML Testing Profile)



```
# include main library
include ../../../apollolib/common/library.ttr

# some definitions
defvar other     = testtarget_2
defvar bw        = 768
defvar endpoints = $other testtarget
defvar nets      = @nets
defvar dirs      = @dirs
defvar p2pcalls  = ${combine_arrays nets dirs}
inform ## Starting configuration ##
```

Case studies approach



Design/development models

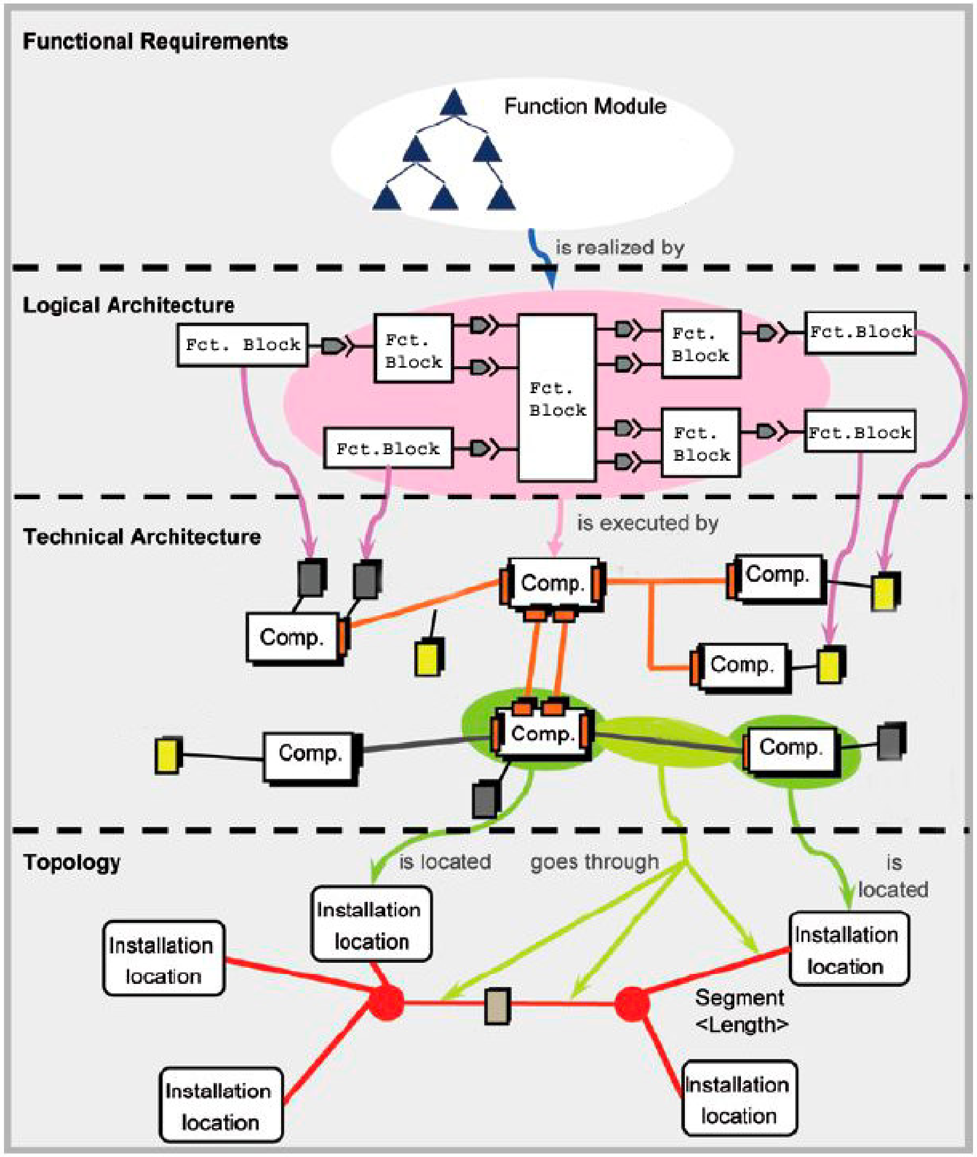
Requirements

Test model

Test cases

D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Common System Architecture Framework



Functions offered
from the user's point of view
(functionality the user can see)

Functional blocks and their **interconnections**
realizing the above functions
– without any specific technical (e.g. hardware) aspects

Assignment of the functional blocks,
communication channels to **real hardware**
adding and respecting technical
requirements

Taking into account the **locations** /
geometry of the hardware and its wiring

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Commonality

D-MINT COMMON APPROACH

PROCESS	Requirement Documentation	Modeling for Test Derivation	Test Derivation	Test Implementation	Test Execution	Test Reporting
ABSTRACTION	Abstraction Level: System Architecture Viewpoints: Requirements (all), Logical (all), Technical (most) , Topological (possibly of interest, but not realized)					

METHODS
(P: Priority,
Q: Quality,
M: Methodology)
	Structured Requirements (Sometimes up to "Precondition, Event, Reaction" formal level)	Architecture based; Behavior modeled by state chart, sequence charts or signal flows; Priority by annotations of usage, risk, safety,…	Test cases derived from architecture models and behavior with respect to annotated priorities and coverage criteria	Abstract test cases in a test model or test specific language "compiled" to some byte code – some times only test descriptions for manual execution	Online,
					offline,
					HIL
						Pass/Fail;
						Statistical analysis;
						Test execution traces;
						Back-tracing of Req's

NOTATION	Textual format	UML;
		SYSML;
		Domain specific languages;
		Model Annotations for priority
			QML
			TTCN-3
			Toolspecific
				QML
				TTCN-3
				Toolspecific
					Machine code
					java byte code
					EAST scripts
						HTML

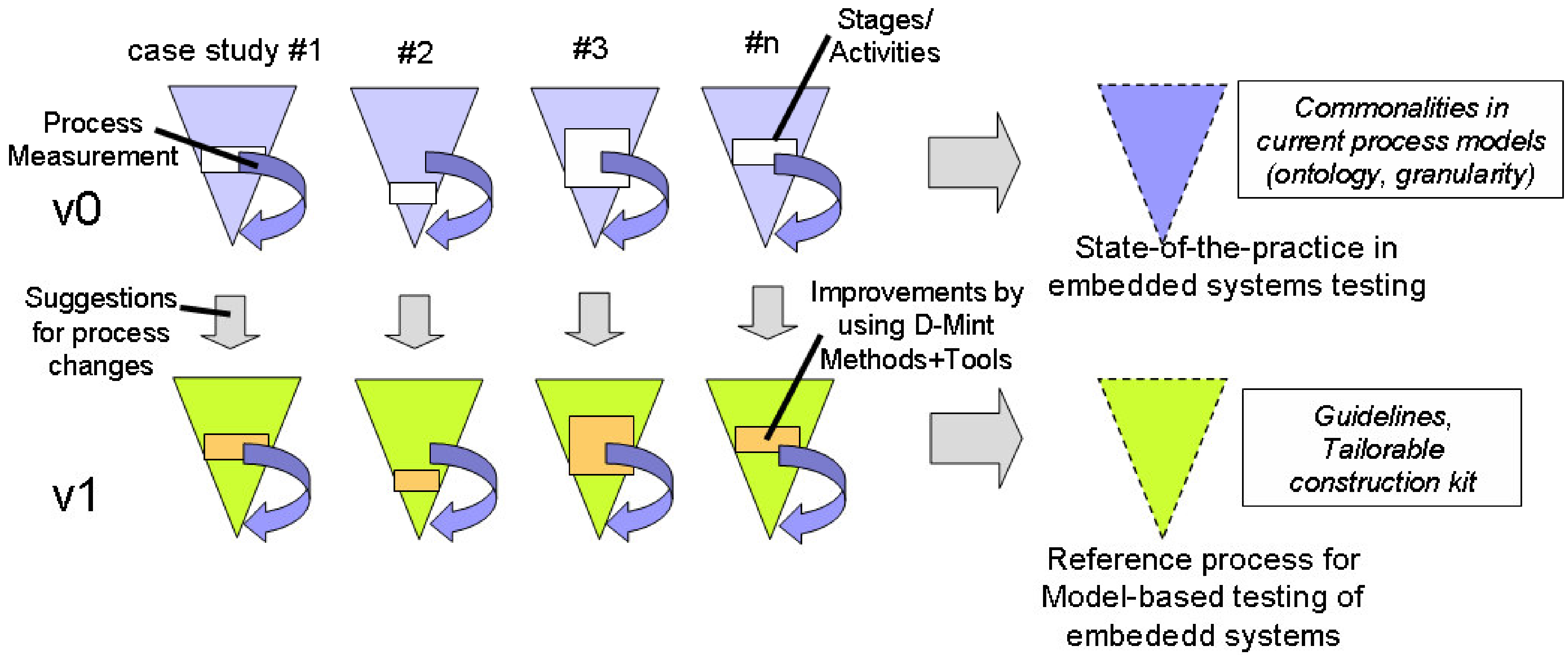
TOOLS	Text based tools	MagicDraw, EA, StarUML TTModeler, MySQL, Jumbl, TPT, PreeVision	Qtronic, TTModeler, PreeVision	Qtronic TTWorkbench Jumbl TPT (PreeVision, OpenOffice)	Qtronic EAST TTWorkbench iXtronics Testrig	Qtronic EAST TTworkbench

Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT
  - Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Introduction

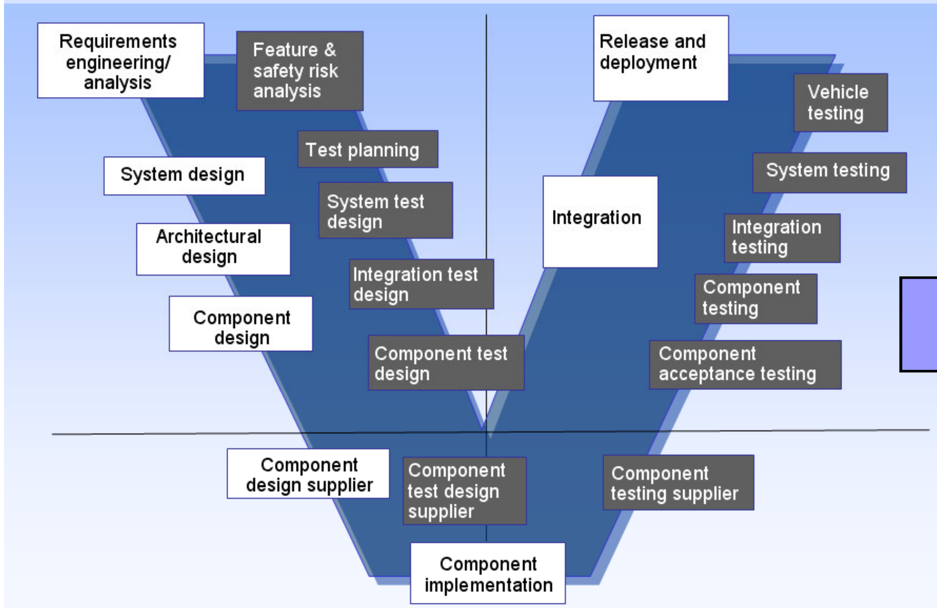
- Goal of the evaluation task: Measurable improvement through MBT technologies
  - Evaluate the **effects** of technologies and processes for performing model-based testing in order to understand them, improve them and accelerate their introduction into industrial practice



D-MINT
Deployment of Model-Based Technologies to Industrial Testing

Evaluation Example

Process now



Requirements engineering/ analysis

Feature & safety risk analysis

Test planning

System design

System test design

Architectural design

Integration test design

Component design

Component test design

Release and deployment

Vehicle testing

System testing

Integration

Integration testing

Component testing

Component acceptance testing

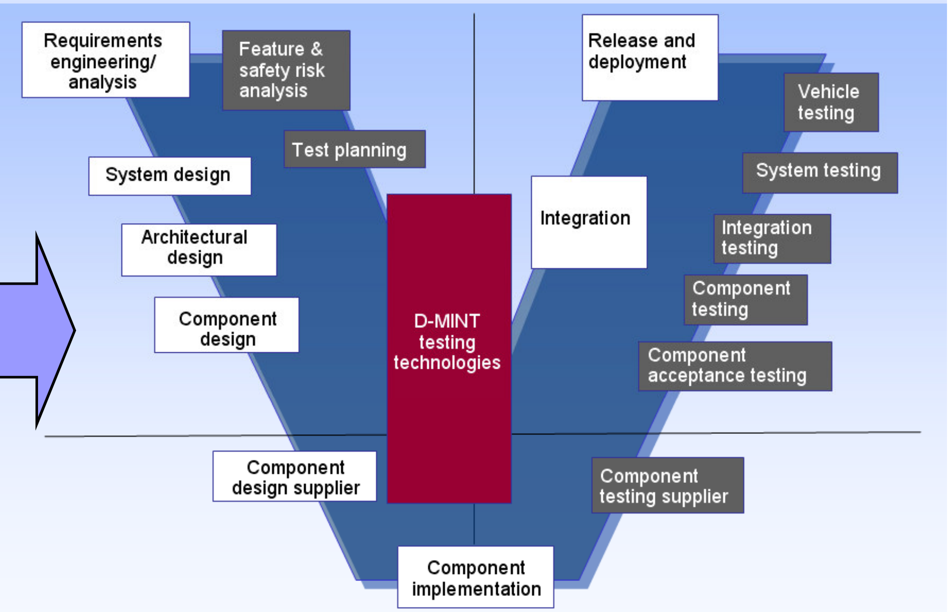
Component design supplier

Component test design supplier

Component testing supplier

Component implementation

Process after integration of D-MINT MBT



Requirements engineering/ analysis

Feature & safety risk analysis

Test planning

System design

Architectural design

Component design

D-MINT testing technologies

Release and deployment

Vehicle testing

System testing

Integration

Integration testing

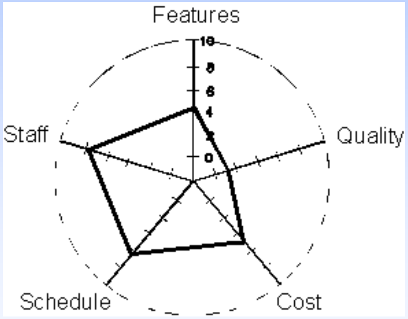
Component testing

Component acceptance testing

Component design supplier

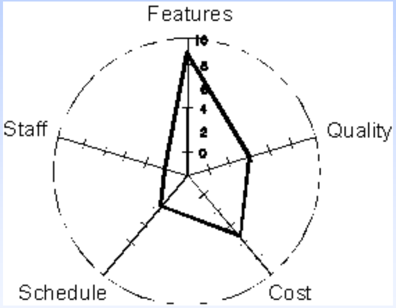
Component testing supplier

Component implementation



Technology Assessment

Non-MBT vs MBT



ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

Defining Metrics

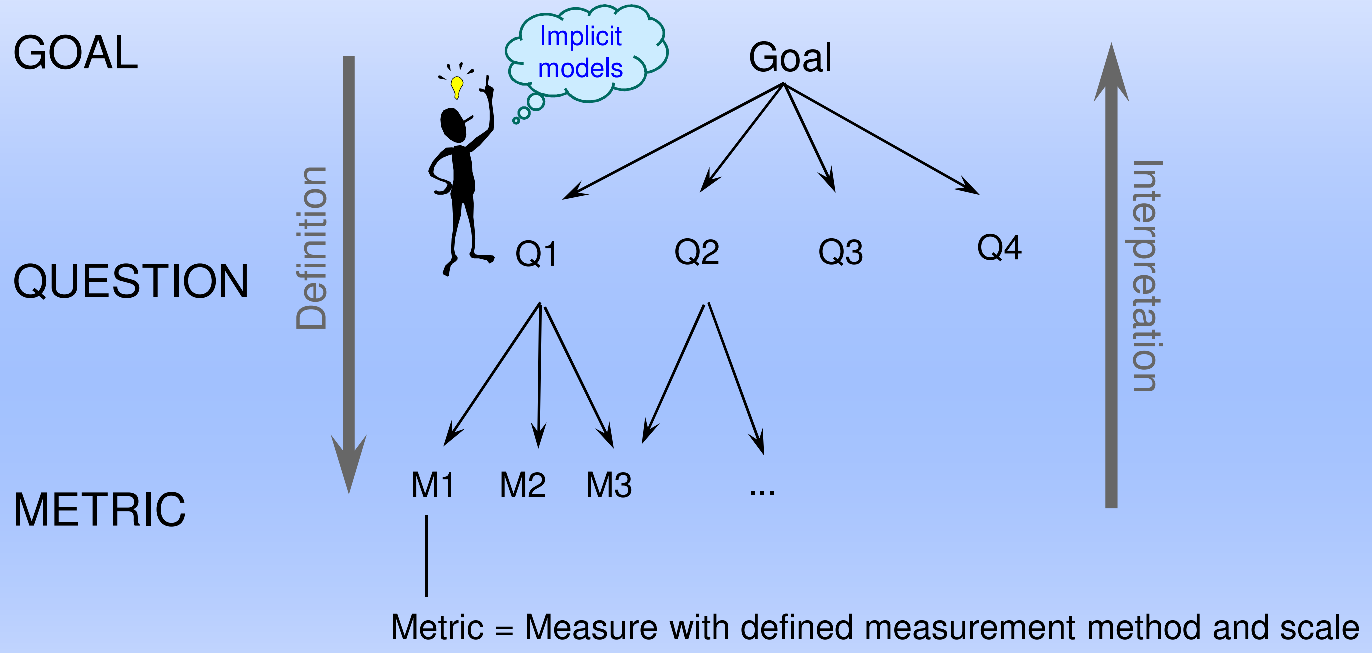
Entities	*Attributes*	*Rules*	*Numbers/Symbols*
 Process	effort	PD from start to end	10,53 h
 Product	size	Number of Lines of Code	700 LoC
 Resource	experience	>10 projects	“high”

Potential Measurement Problems

- Too many unnecessary data is collected
  - Unnecessary effort
  - Low data quality
  - Hard to make conclusions
  - Discouraging for people collecting/analyzing data
- Data is not analyzed in the right environment
  - Context and influencing factors are not considered
  - Wrong conclusions are drawn
- Standard measures are postulated that would be valid in every possible environment (without adaptation)
- Important aspects cannot be analyzed because important data is missing

- Solution: Goal-oriented Measurement using the GQM Method

The Goal Question Metric Method

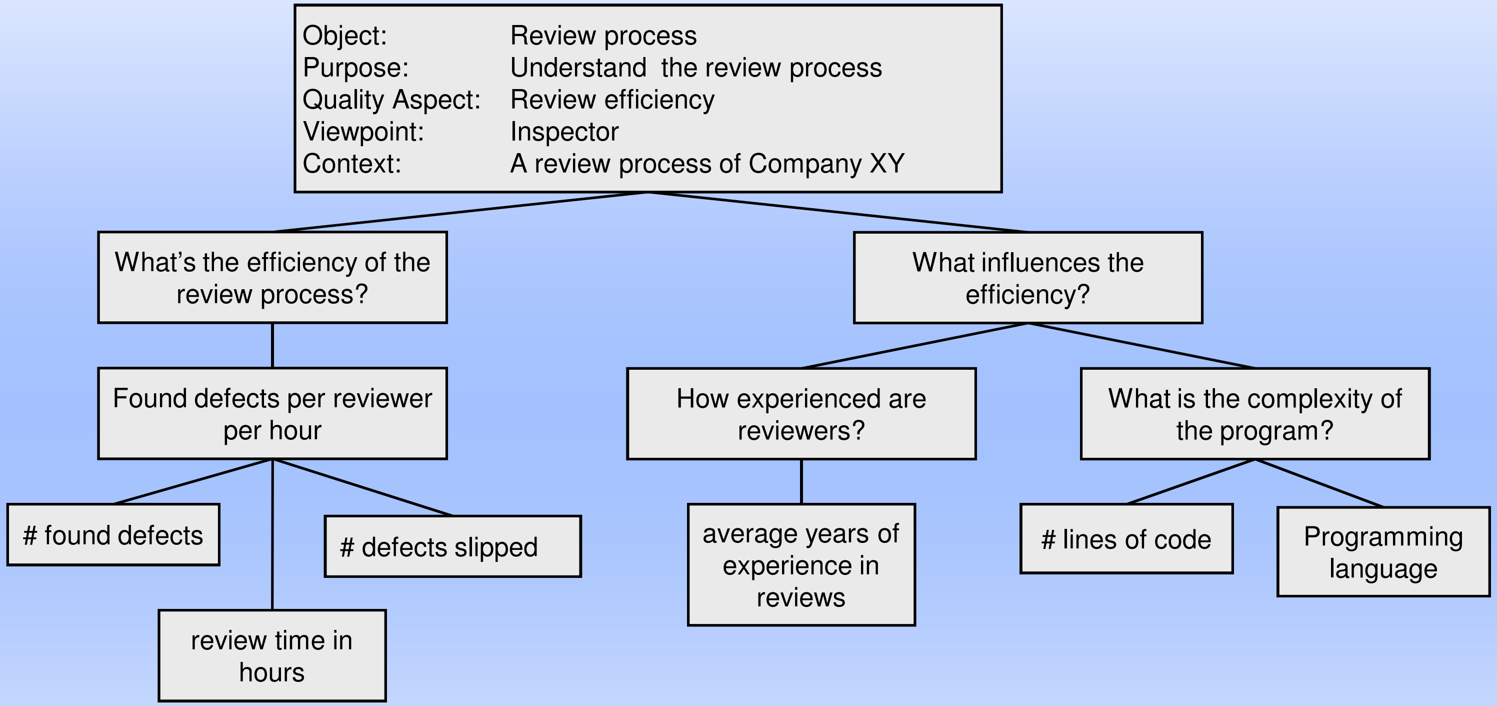


Metric = Measure with defined measurement method and scale

Template to Define GQM Goals

Dimension	Description	Examples
Object	What is analyzed?	Process, Product, Resource, …
Purpose	Why is the object analyzed?	Characterize, Evaluate, Compare, Improve, ...
Quality Aspect	Which property of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality aspect analyzed?	Developer, Manager, Tester, Project Manager, …
Context	In which context is the analysis conducted?	Organization, Project, Application, ...

Example: GQM Goal, Questions and Metrics



Object: Review process
Purpose: Understand the review process
Quality Aspect: Review efficiency
Viewpoint: Inspector
Context: A review process of Company XY

What's the efficiency of the review process?

Found defects per reviewer per hour

# found defects

# defects slipped

review time in hours

What influences the efficiency?

How experienced are reviewers?

average years of experience in reviews

What is the complexity of the program?

# lines of code

Programming language

Example: GQM Abstraction Sheets

Object	Purpose	Quality Aspect	Viewpoint	Context
Inspection	Understand	Effectiveness	Inspector	X

Quality Focus

- M1: # defects detected
- M2: # defects slipped
- M3: M1 / (M1 + M2) %
- M4: # hours per detection

Variation Factors

- M5: Experience of personnel (- , **0** , +)
- M6: Size of program (- , **0** , +)
- M7: Language (**L1**, L2 , L3)

Baseline Hypotheses

- M3: 75%
- M4: 3 h

Impact of Variation Factors

- if (M5=‘+’) then (M3=‘90%’)&(M4=‘2.5 h’)
- if (M7=‘L2’)&(M6=‘+’) then (M3=‘60%’)&(M4=‘4 h’)

Development of Measurement Plans
for Case Studies

Presentation Overview

- Basic terminology
- Techniques
  - TTCN-3, UTP, MiLEST, TPT, Statistical testing
- D-MINT
  - Introduction + scope
  - Industrial domains + case studies
  - Evaluation processes
- Summary + outlook

Summary + outlook

- MBT is evolving
- Several techniques and tools are available in multiple domains
- Selected tools applied in industrial case studies
- TTCN-3 is used in several domains as binding link between modelling and execution

- Demonstrator and experience package in preparation for end of 2009
- 2nd MoTiP workshop at ECMDA, June 2009

www.d-mint.org

Thank you!