ALL4TEC

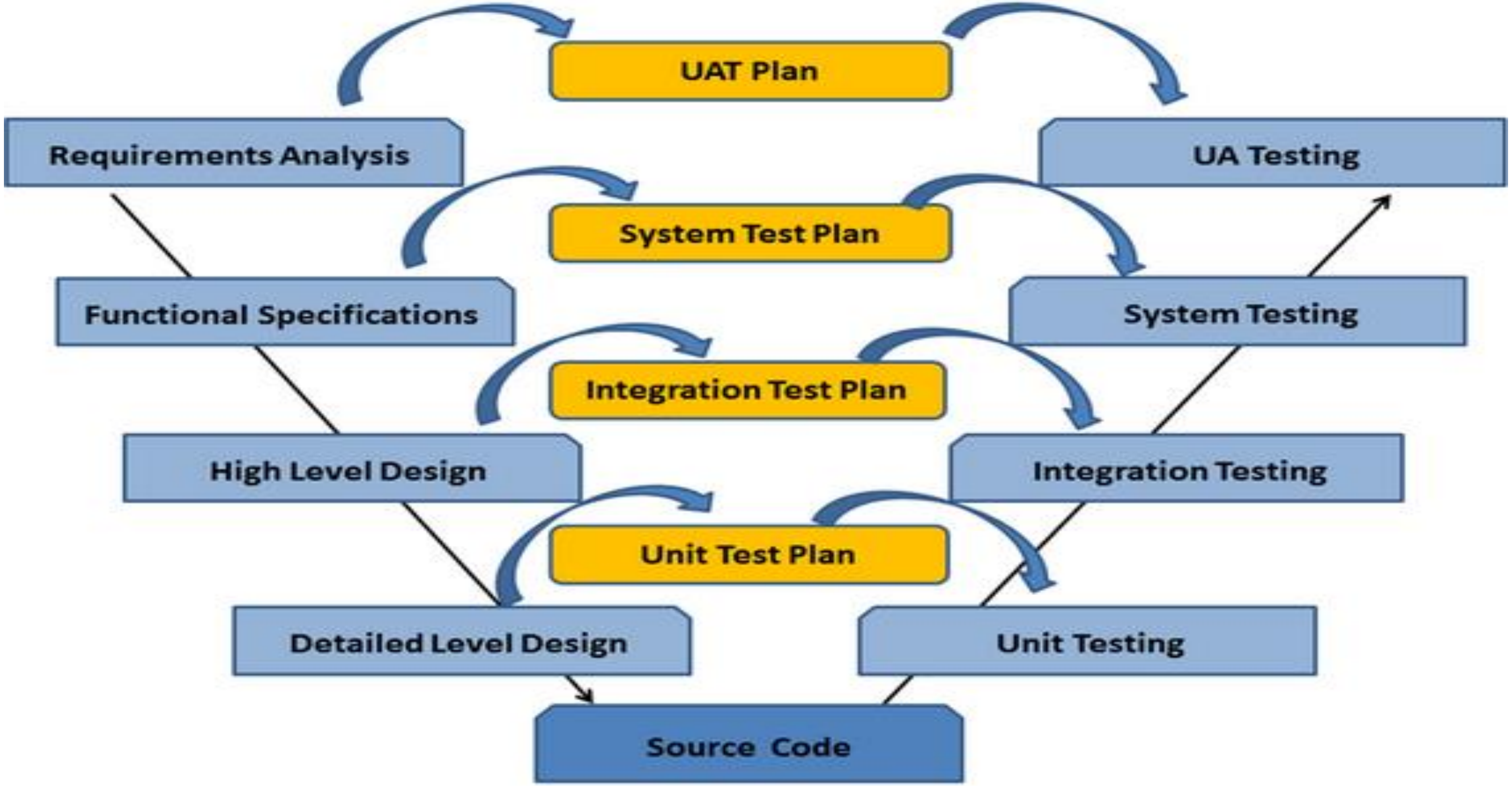
Using Model-Based Testing during the life cycle of your product

Alexis DESPEYROUX & René-Christian TUYISHIME
Support And Pre-Sales

SUMMARY

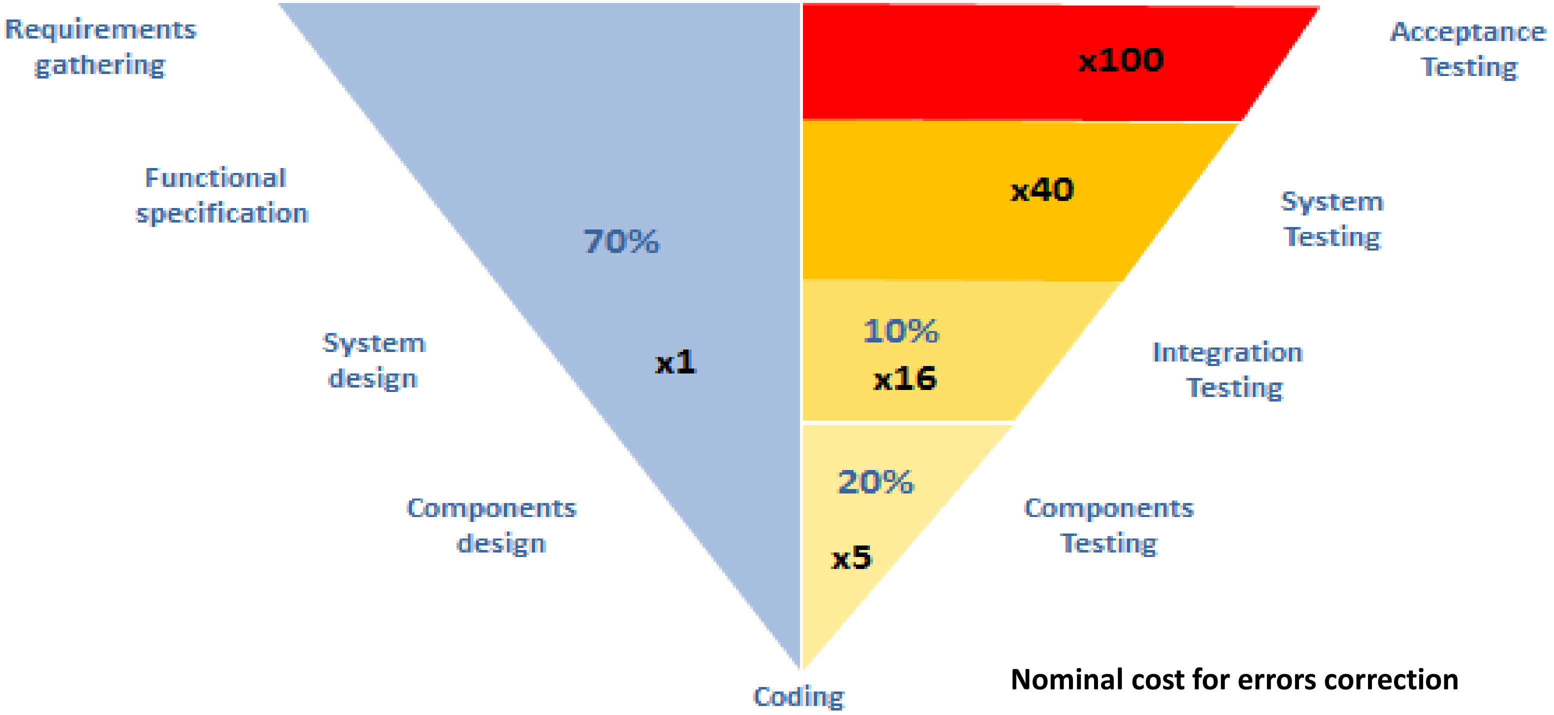
- ❑ **Introduction**
- ❑ Tutorial
- ❑ Conclusion

Test in V cycle



Test challenges

Error correction cost



Nominal cost for errors correction

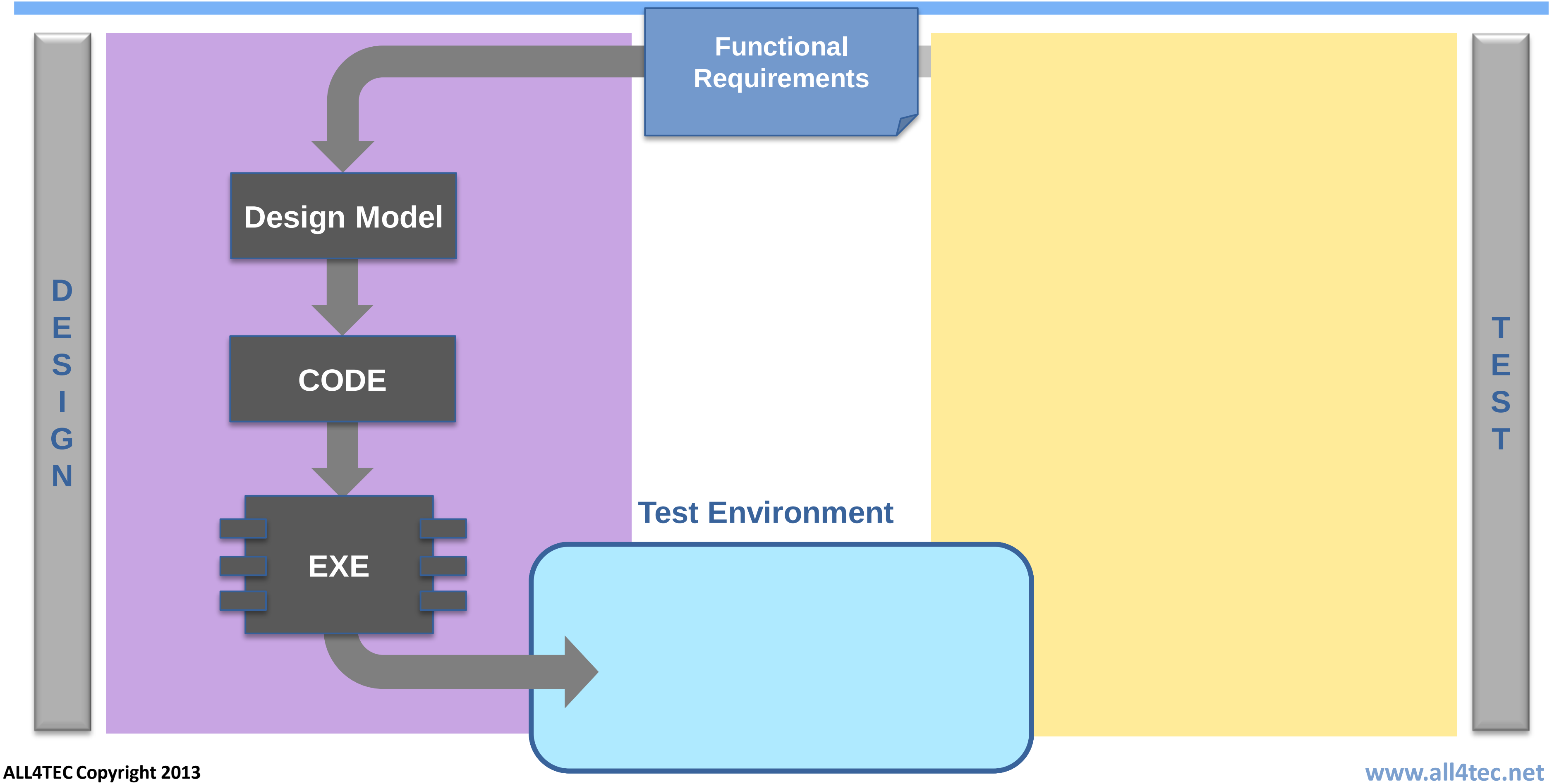
Where errors are introduced

- IBM Systems Sciences Institute
- Crosstalk, the Journal of Defense Software Engineering

Some Testing issues

- ❑ Test plans are written very later
  - ⇨ Often after the system implementation
  - ⇨ Errors detected later -> expensive correction
- ❑ Maintain test cases and test scripts
- ❑ Information on the requirements coverage rate
- ❑ Maintain resources
- ❑ Lack of communications between designers, developers and testers
  - ⇨ Lack of processes
  - ⇨ Projects are abandoned
- ❑ Etc.

Model-Driven Engineering

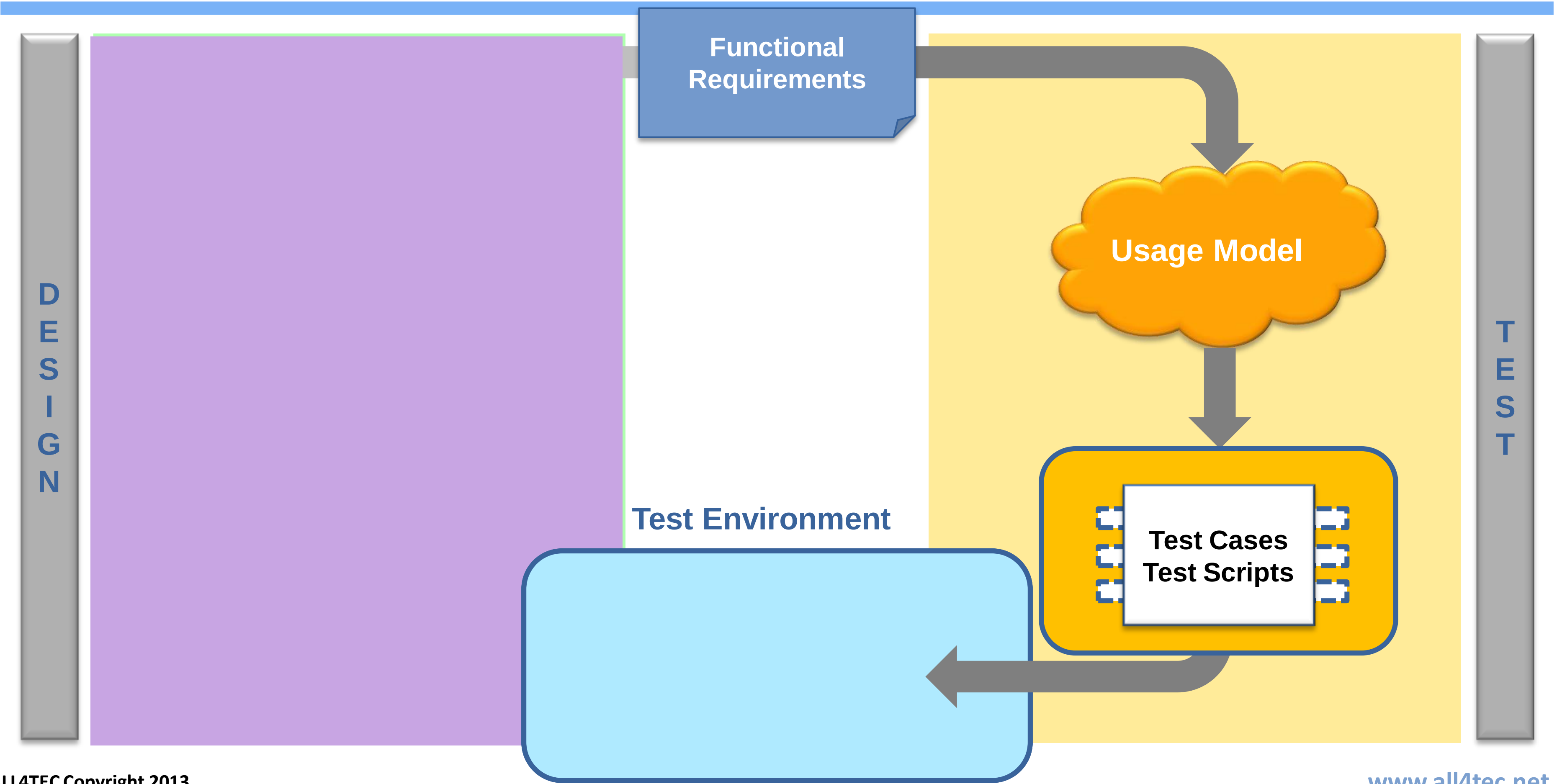


Design Model

- ❑ Embedded system development
- ❑ Abstract representation of the system based on specifications
  - ⇨ Verification
  - ⇨ Automatic code generation
  - ⇨ Executable in the target environment

MATLAB SIMULINK
SCADE SUITE™
NATIONAL INSTRUMENTS LabVIEW
Scilab
MapleSim™
High-Performance Multi-Domain Modeling and Simulation
SIMULATION X
Powered by ITI

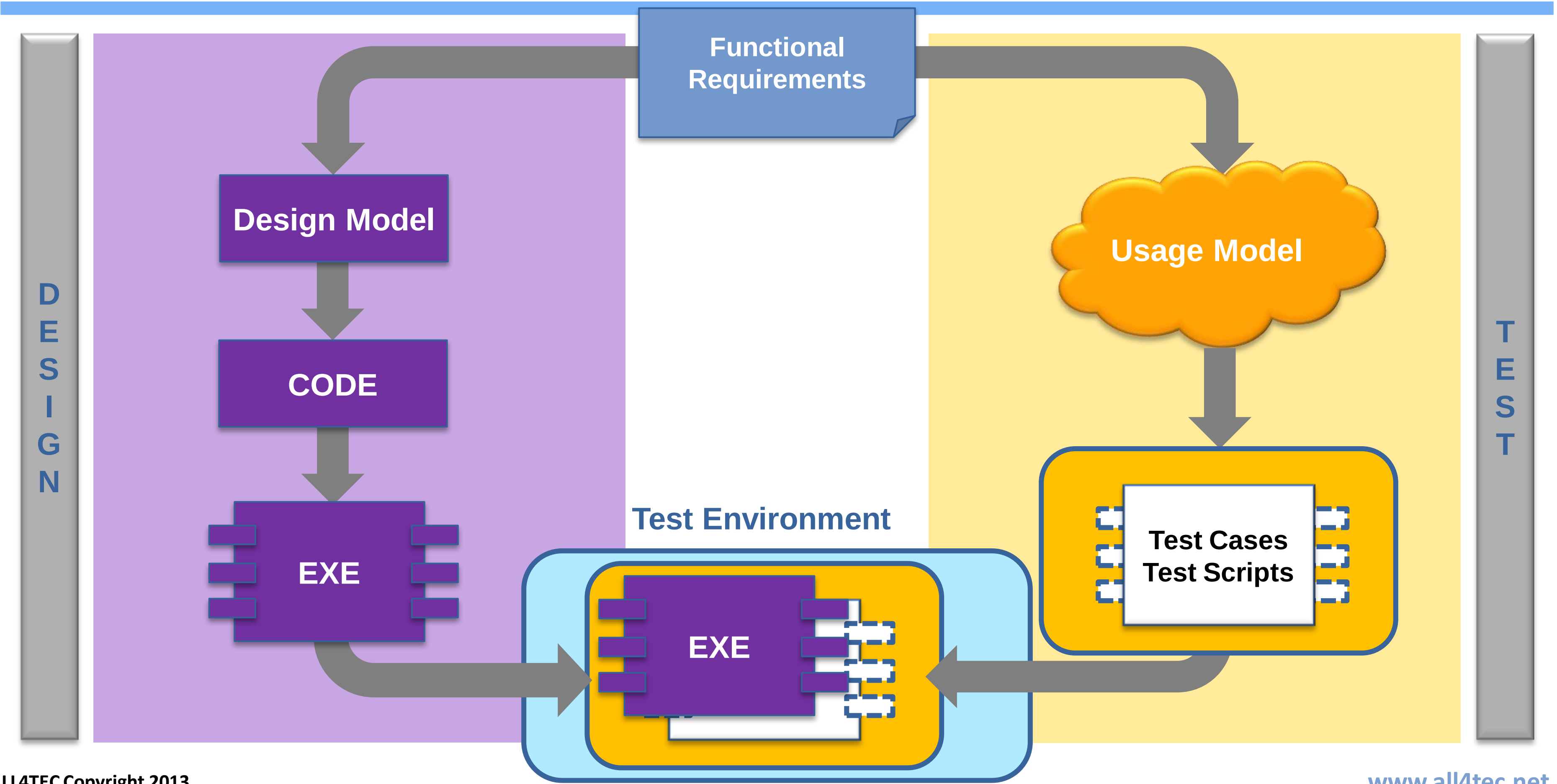
Model-Driven Engineering



Usage Model

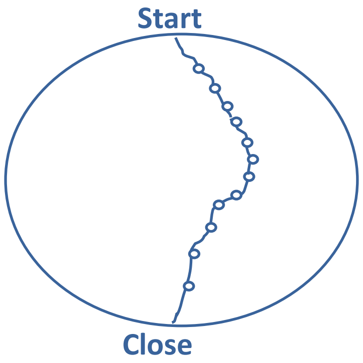
- ❑ Behavior of the SUT (System Under Test)
  - ⇨ Stimulations or actions
  - ⇨ Verifications
  - ⇨ Constraints (time,…)
- ❑ Formal languages are used
  - ⇨ UML, Markov chains, states charts…
- ❑ Test cases generation
  - ⇨ Test strategies
- ❑ The usage model construction has to begin as soon as we have a big part of requirements
  - ⇨ Detect and remove ambiguities early

Model-Driven Engineering



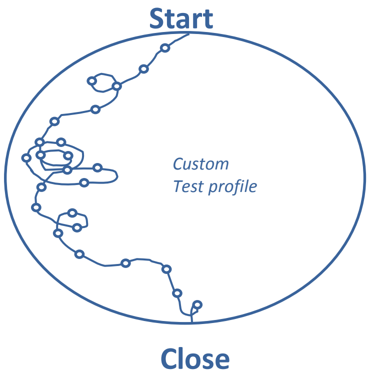
Test Strategies

Most probable approach



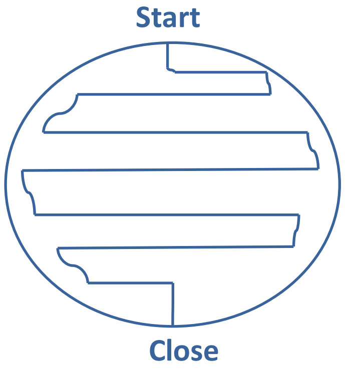
FREQUENCY
FOCUS

Risk based Approach User Oriented - Limit



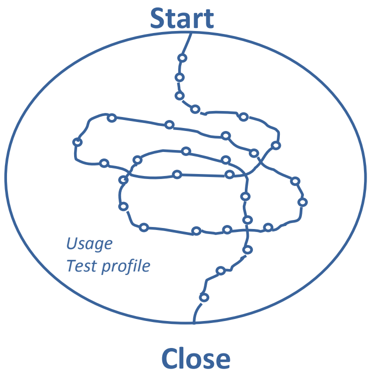
CRITICALITY, COMPLEXITY
UPDATE FOCUS

Arcs coverage approach



REQUIREMENTS
COVERAGE

Usage approach Random

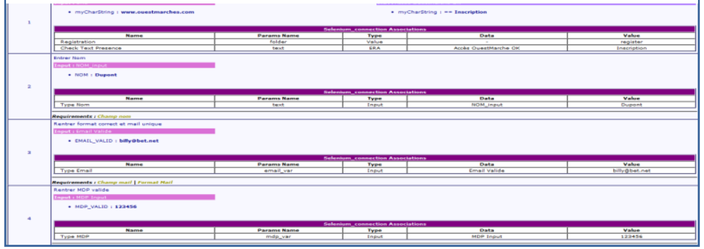


OPERATIONAL
COVERAGE

Test Cases

- ❑ **Manual test cases**
  - ⇨ Test suite
  - ⇨ Test generation report
  - ⇨ Requirement coverage reports
- ❑ **Translation in test scripts**
  - ⇨ According to test automation tool
- ❑ **Executed in a Test environment**

Test generation report



1

- myCharString : **www.ouestmarches.com**
- myCharString : == **Inscription**

Selenium_connection Associations				
Name	**Params Name**	**Type**	**Data**	**Value**
Registration	folder	Value	-	register
Check Text Presence	text	ERA	Accès OuestMarche OK	Inscription

2

Entrer Nom

Input : NOM_input

- NOM : **Dupont**

Selenium_connection Associations				
Name	**Params Name**	**Type**	**Data**	**Value**
Type Nom	text	Input	NOM_input	Dupont

Requirements : Champ nom

3

Rentrer format correct et mail unique

Input : Email Valide

- EMAIL_VALID : **billy@bet.net**

Selenium_connection Associations				
Name	**Params Name**	**Type**	**Data**	**Value**
Type Email	email_var	Input	Email Valide	billy@bet.net

Requirements : Champ mail | Format Mail

4

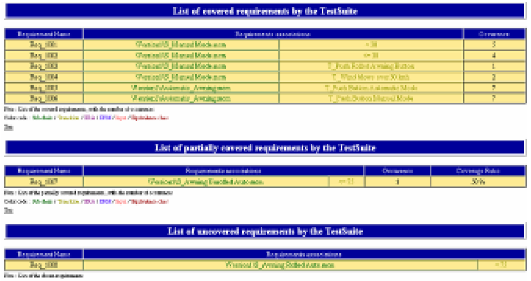
Rentrer MDP valide

Input : MDP Input

- MDP_VALID : **123456**

Selenium_connection Associations				
Name	**Params Name**	**Type**	**Data**	**Value**
Type MDP	mdp_var	Input	MDP Input	123456

Requirements Coverage



List of covered requirements by the TestSuite

Requirement Name	Requirements associations		Occurrence
Req_1001	[illegible]	[illegible]	3
Req_1002	[illegible]	[illegible]	4
Req_1003	[illegible]	[illegible]	1
Req_1004	[illegible]	[illegible]	2
Req_1005	[illegible]	[illegible]	[illegible]
Req_1006	[illegible]	[illegible]	[illegible]

[illegible]

List of partially covered requirements by the TestSuite

Requirement Name	Requirements associations		Occurrences	Coverage Ratio
Req_1007	[illegible]	[illegible]	1	50%

[illegible]

List of uncovered requirements by the TestSuite

Requirement Name	Requirements associations	
Req_1008	[illegible]	[illegible]

[illegible]

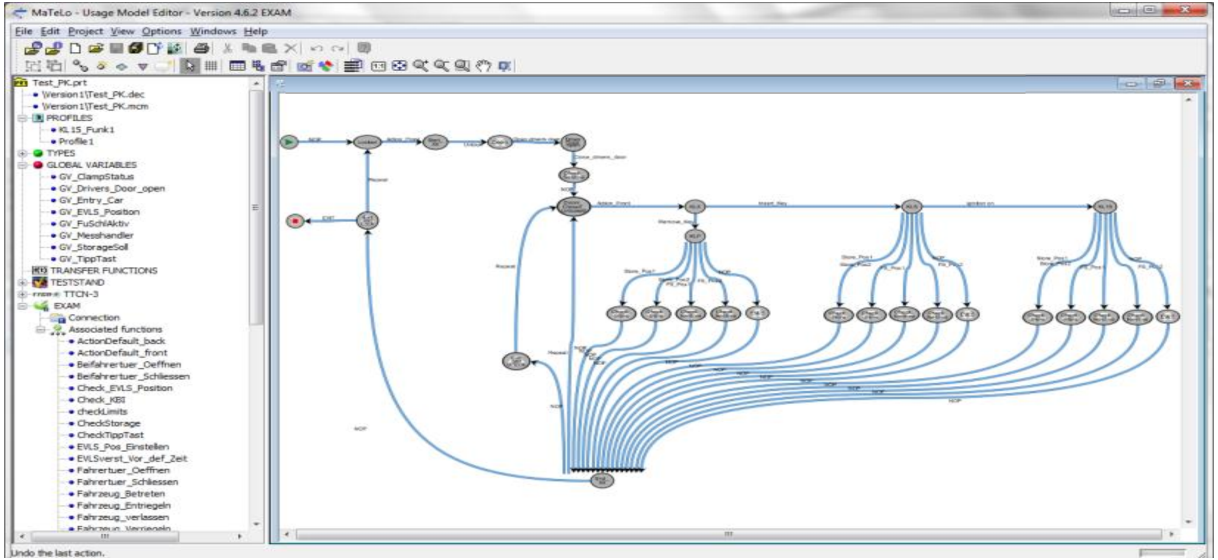
SUMMARY

- ❑ Introduction
- ❑ **Tutorial**
- ❑ Conclusion

Used tools

❑ **MaTeLo (Markov Test Logic)**

- ⇨ **Model-Based Testing approach**
- ⇨ **Markov chains logic**

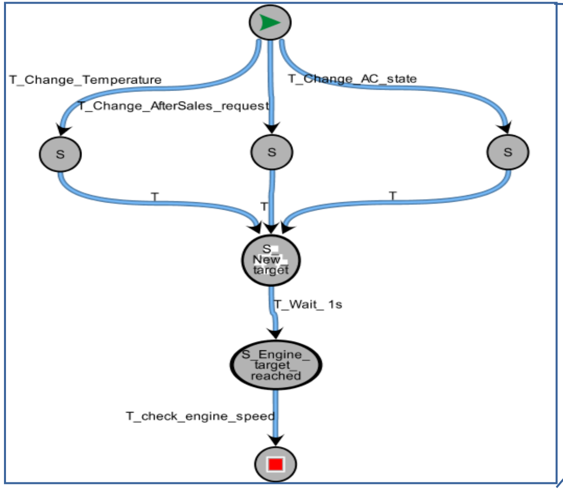


Undo the last action.

MaTeLo Usage Models

- ❑ Setup “Transition” as “Test Step”
- ❑ Mapping of Requirements
- ❑ Configuration for Test Automation

MACRO CHAIN

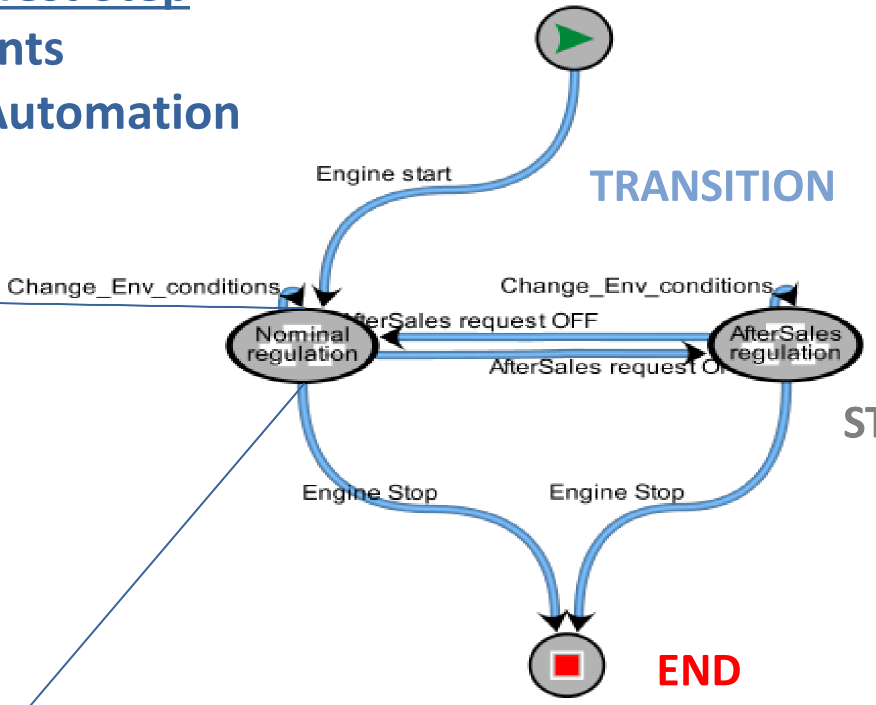


BEGINNING

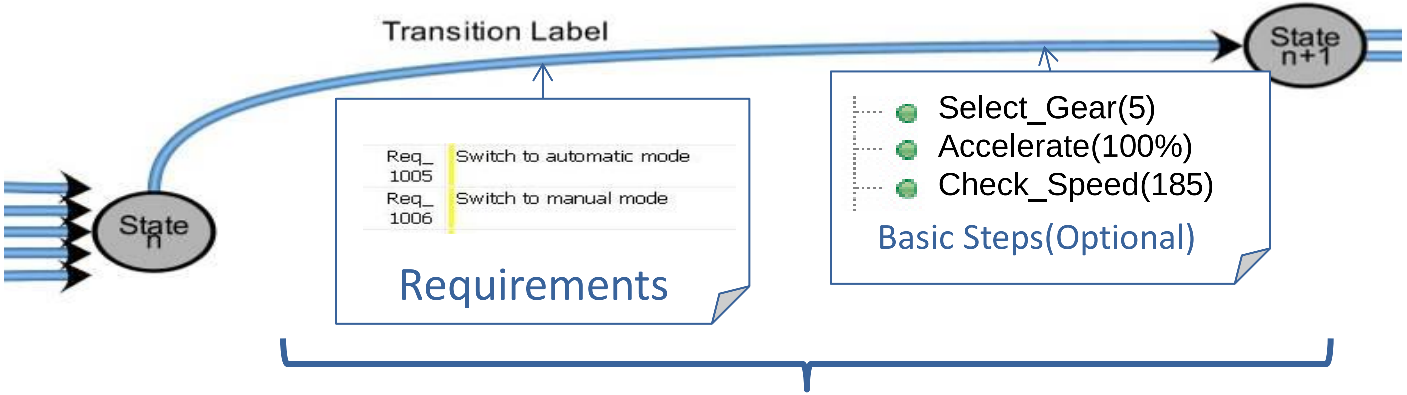
TRANSITION

STATE

END



Model Transition = Test Step



Req_ 1005	Switch to automatic mode
Req_ 1006	Switch to manual mode

Requirements

- Select_Gear(5)
- Accelerate(100%)
- Check_Speed(185)

Basic Steps(Optional)

Stimulations

Inputs Stimulation

- Equivalence Classes
- Timing

Transfer Function

python™

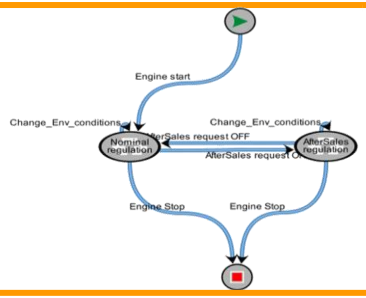
Scilab

MATLAB SIMULINK

Verifications Points

- Expected Results
- Timing

MaTeLo Ecosystem



MaTeLo

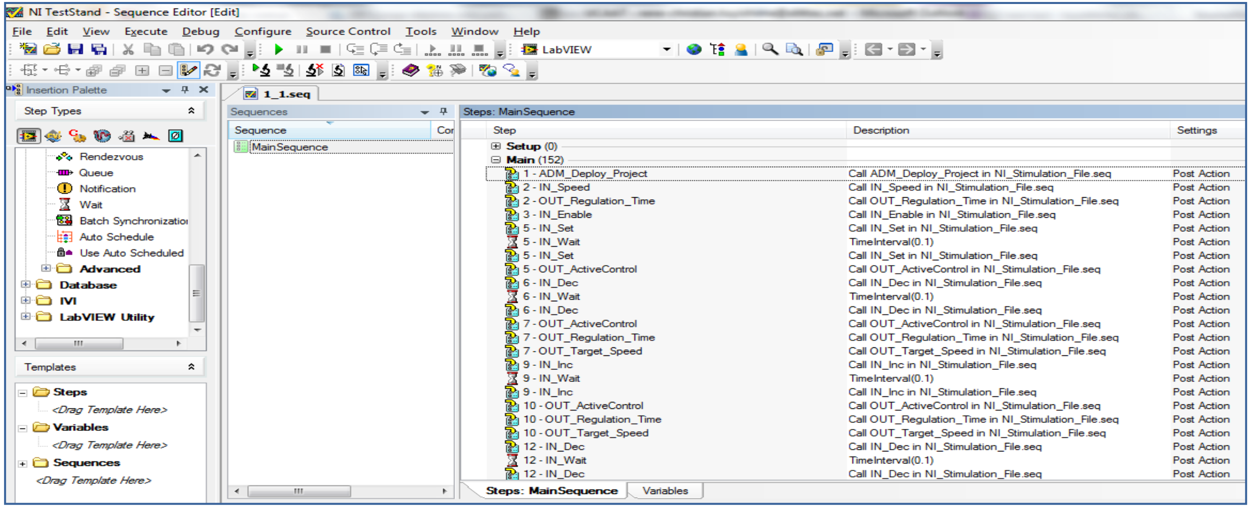
Test Cases

Native		Plugin
Bench language	Scripting	
TESTSTAND	Python	VeriStand
EXAM	Selenium IDE	JUNIT
TTCN-3	Custom Script	Selenium
…	Open to ALL languages	…

Test Scripts

Used tools

- ❑ **NI TestStand** NATIONAL INSTRUMENTS™ TestStand
  - ⇨ **Test sequencer**
    - Execute test sequences generated by MaTeLo



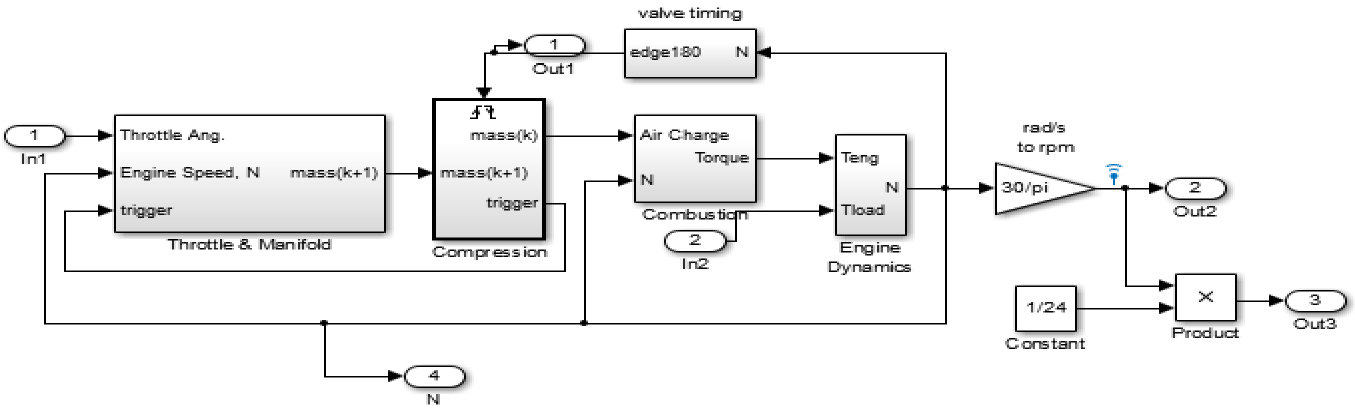
Steps: MainSequence

Step	Description	Settings
⊞ **Setup** (0)		
⊟ **Main** (152)		
1 - ADM_Deploy_Project	Call ADM_Deploy_Project in NI_Stimulation_File.seq	Post Action
2 - IN_Speed	Call IN_Speed in NI_Stimulation_File.seq	Post Action
2 - OUT_Regulation_Time	Call OUT_Regulation_Time in NI_Stimulation_File.seq	Post Action
3 - IN_Enable	Call IN_Enable in NI_Stimulation_File.seq	Post Action
5 - IN_Set	Call IN_Set in NI_Stimulation_File.seq	Post Action
5 - IN_Wait	TimeInterval(0.1)	Post Action
5 - IN_Set	Call IN_Set in NI_Stimulation_File.seq	Post Action
5 - OUT_ActiveControl	Call OUT_ActiveControl in NI_Stimulation_File.seq	Post Action
6 - IN_Dec	Call IN_Dec in NI_Stimulation_File.seq	Post Action
6 - IN_Wait	TimeInterval(0.1)	Post Action
6 - IN_Dec	Call IN_Dec in NI_Stimulation_File.seq	Post Action
7 - OUT_ActiveControl	Call OUT_ActiveControl in NI_Stimulation_File.seq	Post Action
7 - OUT_Regulation_Time	Call OUT_Regulation_Time in NI_Stimulation_File.seq	Post Action
7 - OUT_Target_Speed	Call OUT_Target_Speed in NI_Stimulation_File.seq	Post Action
9 - IN_Inc	Call IN_Inc in NI_Stimulation_File.seq	Post Action
9 - IN_Wait	TimeInterval(0.1)	Post Action
9 - IN_Inc	Call IN_Inc in NI_Stimulation_File.seq	Post Action
10 - OUT_ActiveControl	Call OUT_ActiveControl in NI_Stimulation_File.seq	Post Action
10 - OUT_Regulation_Time	Call OUT_Regulation_Time in NI_Stimulation_File.seq	Post Action
10 - OUT_Target_Speed	Call OUT_Target_Speed in NI_Stimulation_File.seq	Post Action
12 - IN_Dec	Call IN_Dec in NI_Stimulation_File.seq	Post Action
12 - IN_Wait	TimeInterval(0.1)	Post Action
12 - IN_Dec	Call IN_Dec in NI_Stimulation_File.seq	Post Action

Used tools

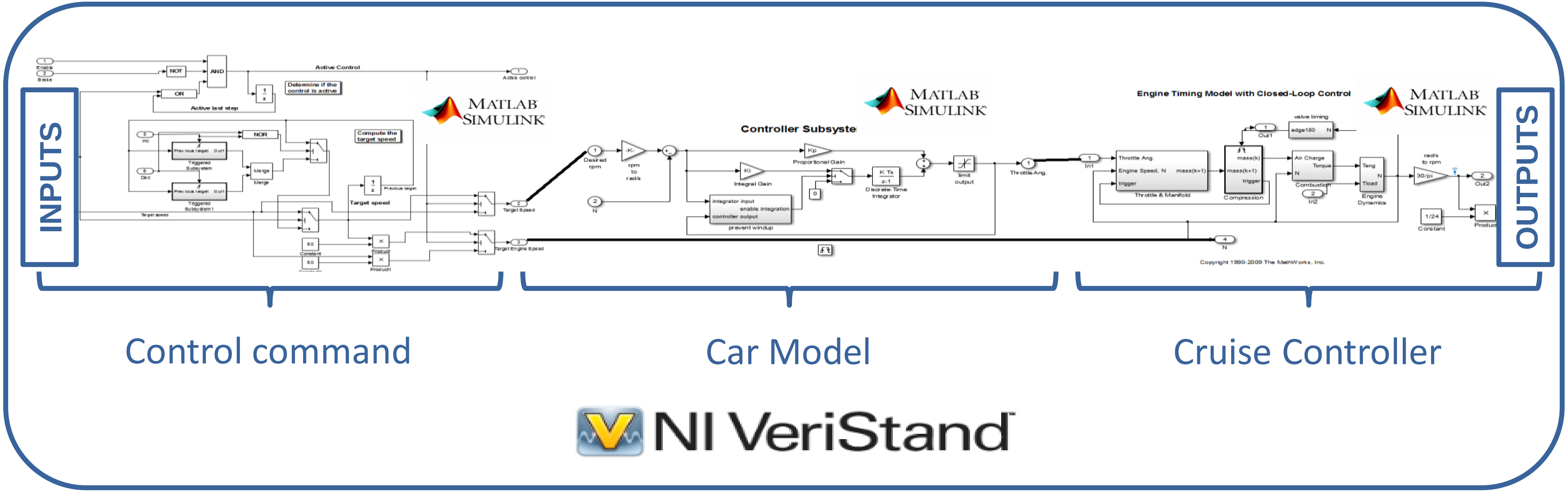
- ❑ Matlab Simulink for design models

Engine Timing Model with Closed-Loop Control



Copyright 1990-2009 The MathWorks, Inc.

SUT: Cruise Control Simulation



Control command

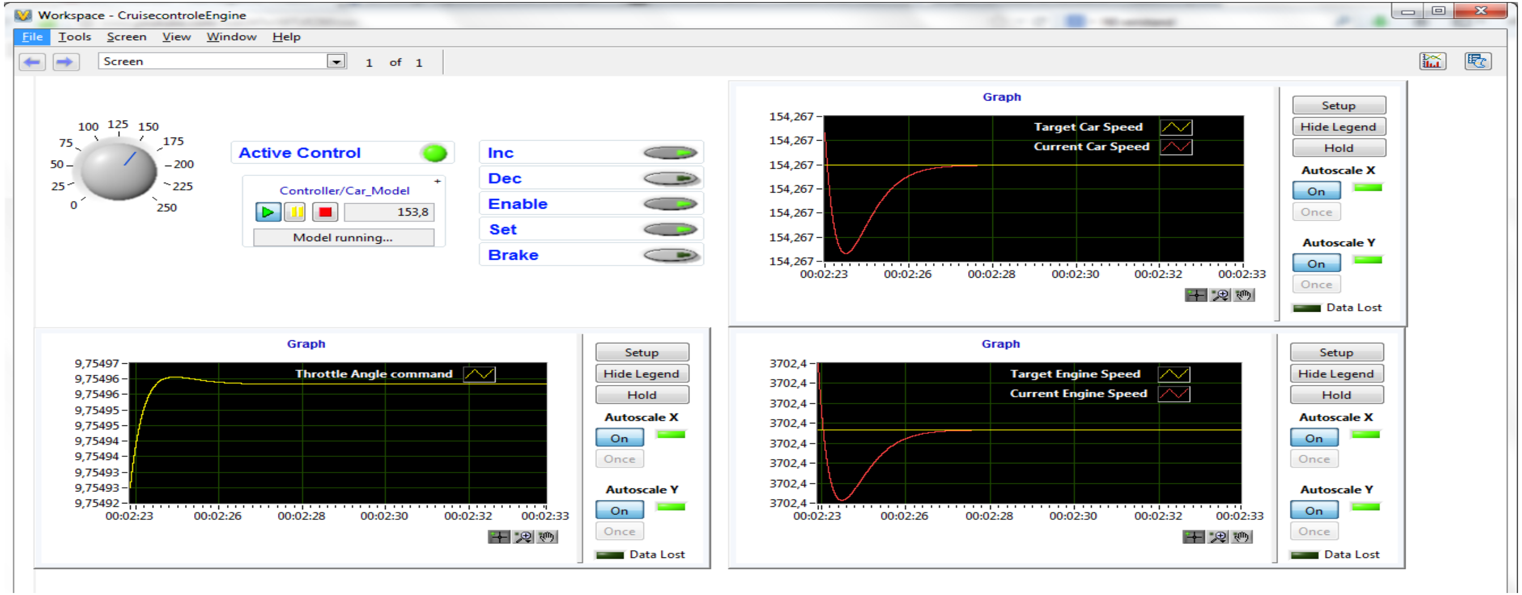
Car Model

Cruise Controller

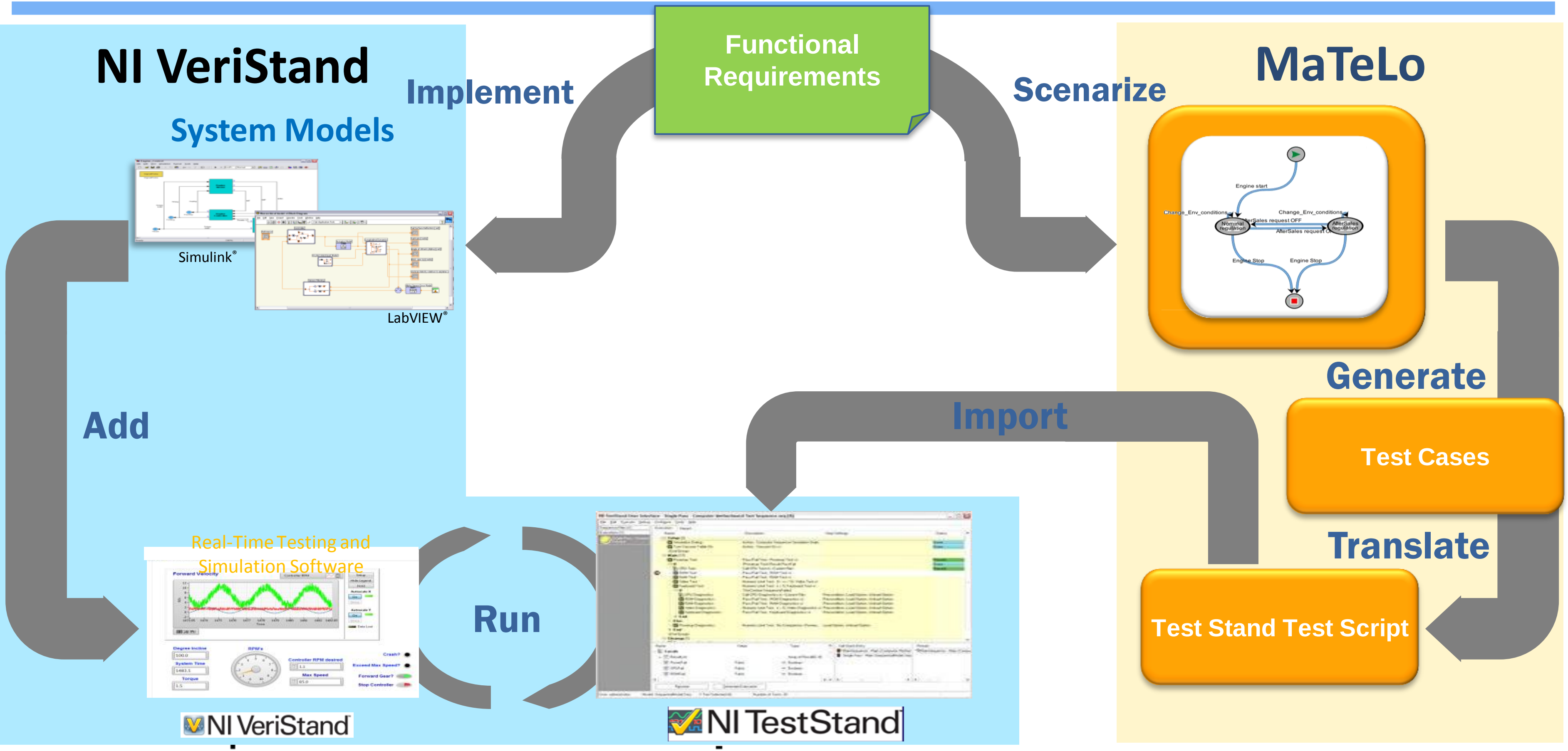
NI VeriStand

Used tools

- ❑ **NI VeriStand**
  - ⇨ Real time environment



Testing methodology



Functional Requirements

NI VeriStand

Implement

System Models

Scenarize

MaTeLo

Generate

Test Cases

Translate

Test Stand Test Script

Import

Add

Real-Time Testing and Simulation Software

Run

SUT requirements

Req_01:

- A push on the ON button activates the cruise control and the led is switched on

Req_02:

- A push on the OFF button deactivates the cruise control and the led is switched off

Req_03:

- Pressing the brake pedal deactivates the cruise control and the led is switched off

Req_04:

- When the cruise control is activated:
- A push on the button SET imposes the current speed as the target speed
- One push on the button “Inc” increases the cruise control target speed by 1km
- One push on the button “Dec” decreases the cruise control target speed by 1 km
- The increase or decrease of 1km must last at maximum 50 ms

Req_05:

- The cruise control is effective between [30,150] Km/h

SUMMARY

Live DEMO

SUMMARY

- □Introduction
- □Tutorial
- □**Conclusion**

Conclusion

- ❑ Quantified and optimized requirements coverage
  - ⇨ Model-Based Testing tools give requirements coverage indicators
- ❑ Consolidation of functional requirements
  - ⇨ Ambiguities in specifications are removed early
- ❑ Pertinent test cases (usage profiles, risks, …)
  - ⇨ Possibility to define usage profiles
  - ⇨ Risks are taken into account in the generation strategies
- ❑ Easy test cases maintenance
  - ⇨ It is more easier to maintain an usage model than manual test cases
- ❑ Easy test cases automation
  - ⇨ Test effort lowered

ALL4TEC

ALL4TEC Laval (HQ)	FRANCE
ALL4TEC Paris	FRANCE
ALL4TEC Munich	GERMANY
ALL4TEC Stockholm	SWEDEN

Sales	sales@all4tec.net
Support	http://all4tec-support.net
Marketing	marketing@all4tec.net

To discover our company and download a **full trial MaTeLo evaluation (30days)**, visit our web site: **www.all4tec.net**