TTCN-3
Testing & Test Control Notation

TTCN-3 in end-to-end model based testing explained on a case study

TTCN-3 User Conference 2010

8-10 June, Beijing, China

Presenter: Andrus Lehtmets (andrus.lehtmets@elvior.com)

Elvior

www.elvior.com

elvior
software test automation

About Elvior

Founded in 1992

Location: Tallinn, Estonia

Test tools

- **TestCast TTCN-3 test tool**
- **TestCast Generator**
- **XML–Simulator**

Testing services

- **TTCN-3 testing**
- **Model based testing**
- **Embedded systems testing**
- **Building automated test environments**

www.elvior.com

elvior
software test automation

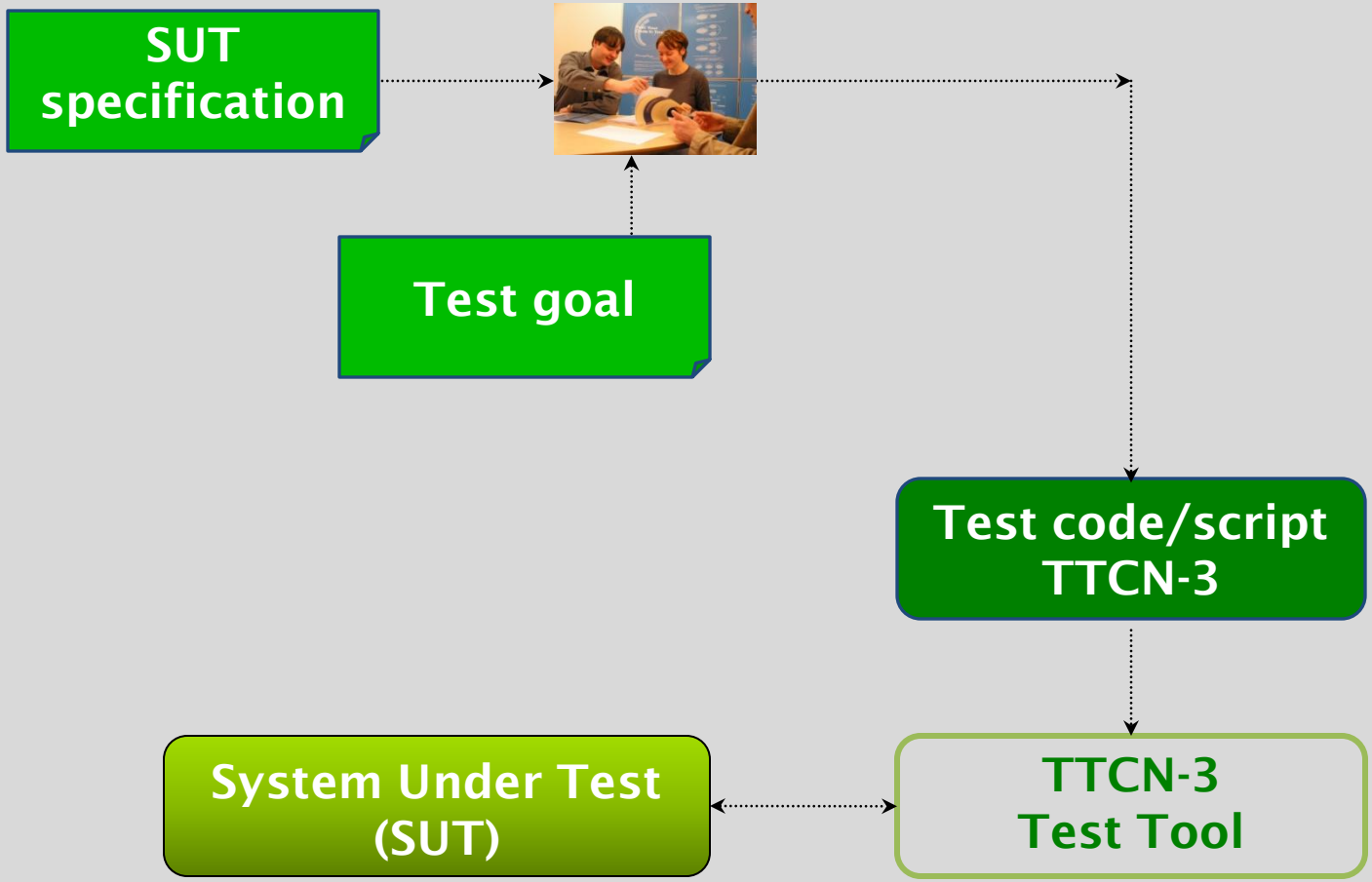
Agenda

1. Overview of model-based testing (MBT)
2. Tools used in practical exercise
3. Example SUT (Light switch)
4. Workflow of MBT
5. System Adapter (SA) used in example
6. Light switch state model
7. Test cases generation
8. Execution of generated TTCN-3 test cases
9. Example of real industrial case study
10. Questions

Model Based Testing (MBT) – what is it?

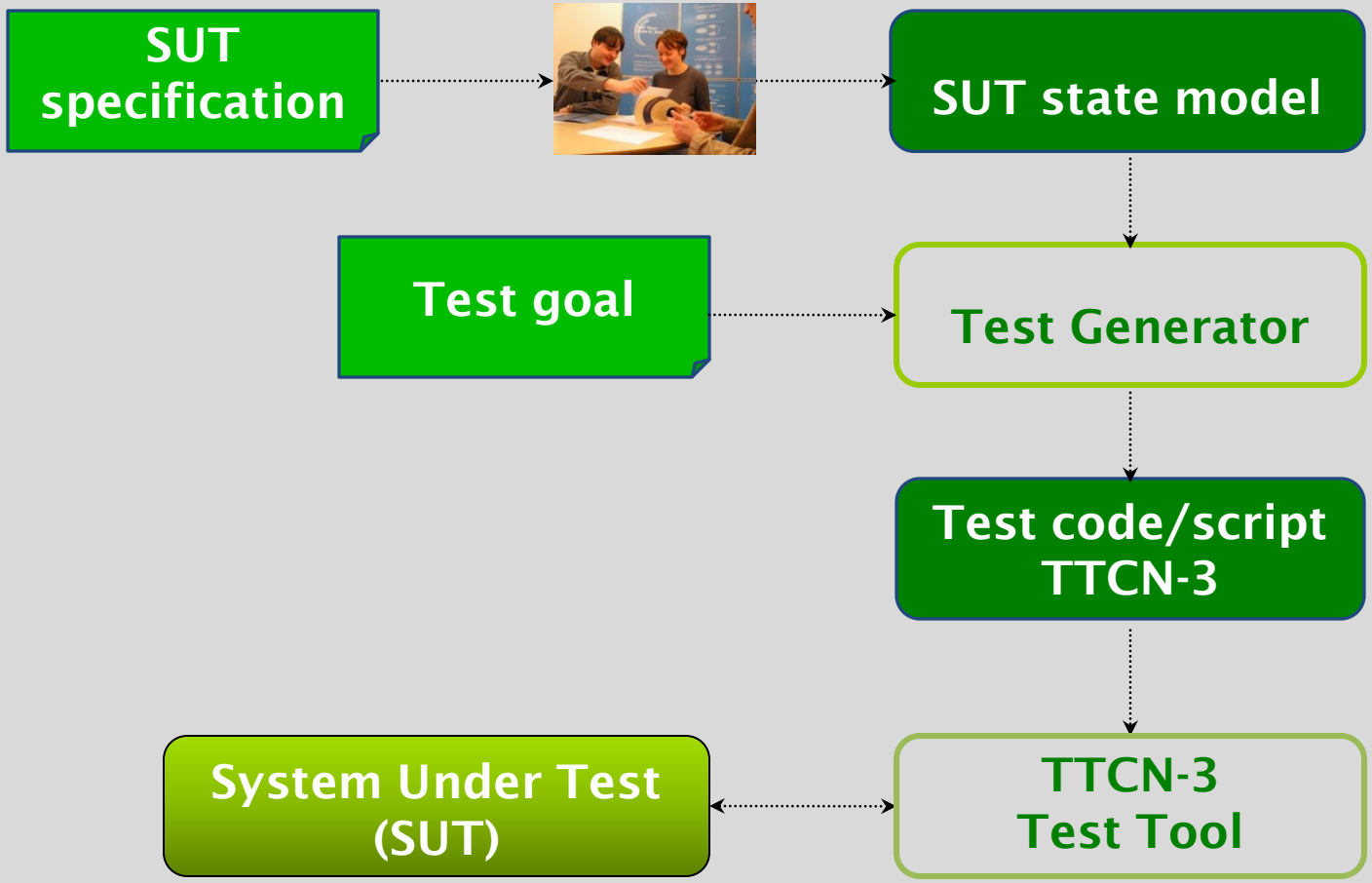
- is software testing where
- from a **model** that describes some (usually functional) aspects of the system under test
- + **model coverage criterion**
- **test cases** (**scripts**) are derived automatically by some tool

TTCN-3 testing



elvior
software test automation

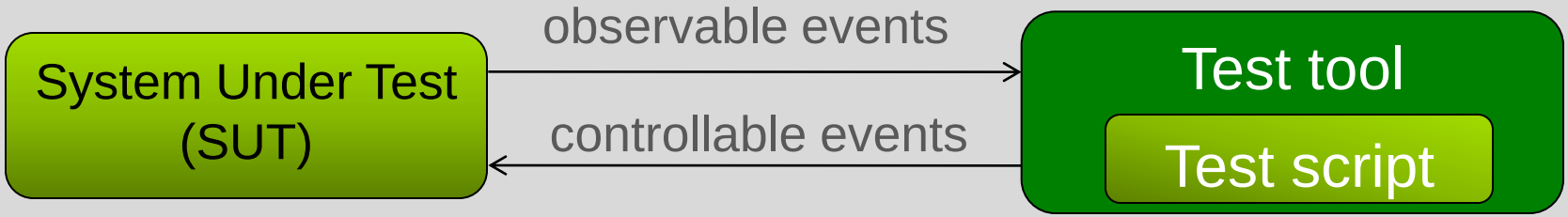
Model Based Testing and TTCN-3



elvior
software test automation

MBT – when to use?

- it is possible to formalize system behavior **OK** / not **NOK**
- functional testing **OK** / GUI testing **NOK**
- automated testing **OK** / manual testing **NOK**
- it must be possible to **control testing** by test script and SUT behavior must be **observable**



elvior
software test automation

Benefits of MBT

- Writing and maintenance of test scripts is a time and effort consuming task.

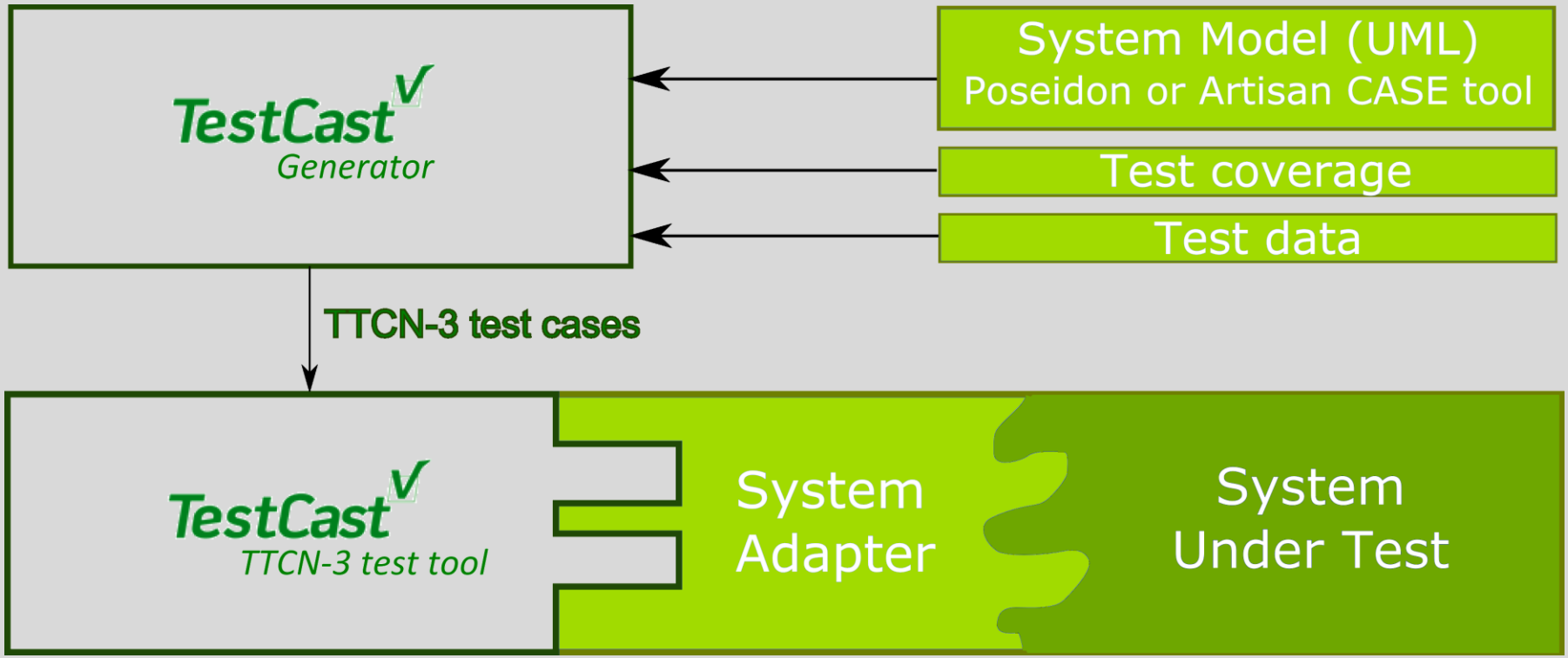
- **Better tests**. Easier and cheaper to generate sufficient amount of test scripts to achieve a good enough test coverage.

- **Lower costs**. Work effort for test suite maintenance will reduce significantly.
  - Instead of maintaining huge amount of test scripts the test engineer should maintain a SUT model only.
  - If there are changes in the behaviour of the SUT then it is rather easy to update the model correspondingly and re-generate all test scripts once again.

Classical expectations to MBT

1. Through formalization discloses ambiguity in specifications and helps validation of specifications.
2. Better test coverage.
3. Cost effective in maintenance phase.

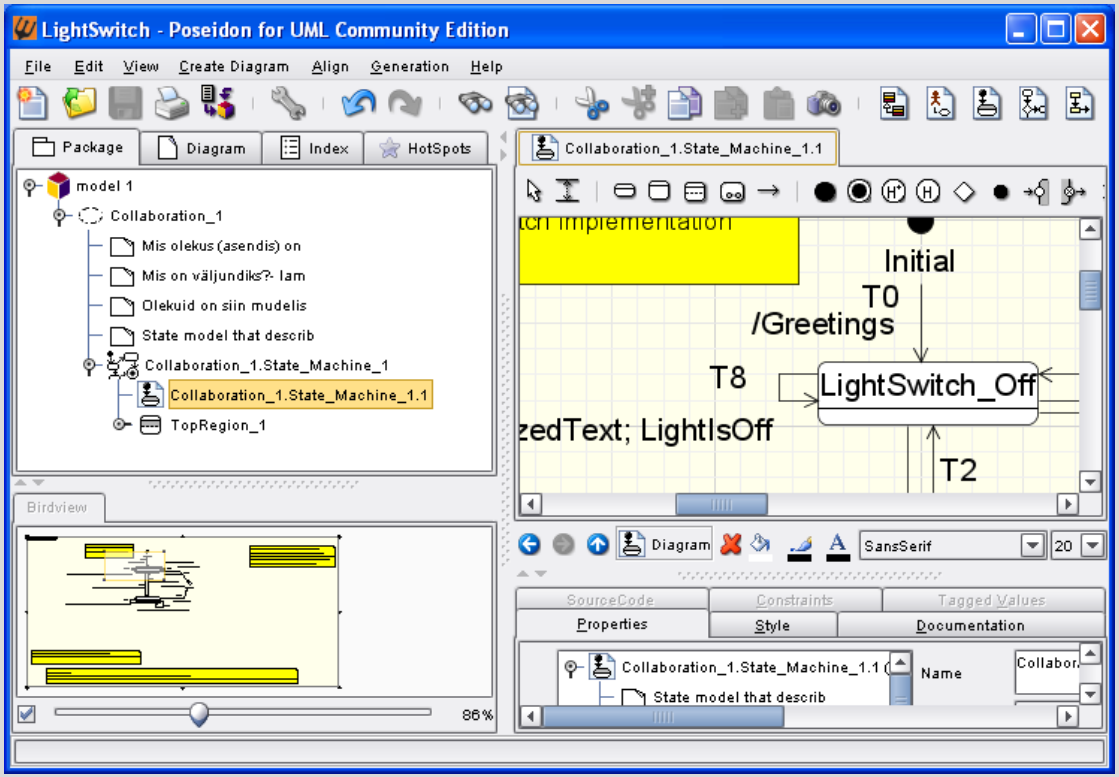
Test environment in MBT (Elvior approach)



elvior
software test automation

Tools – Poseidon for UML (3rd party tool)

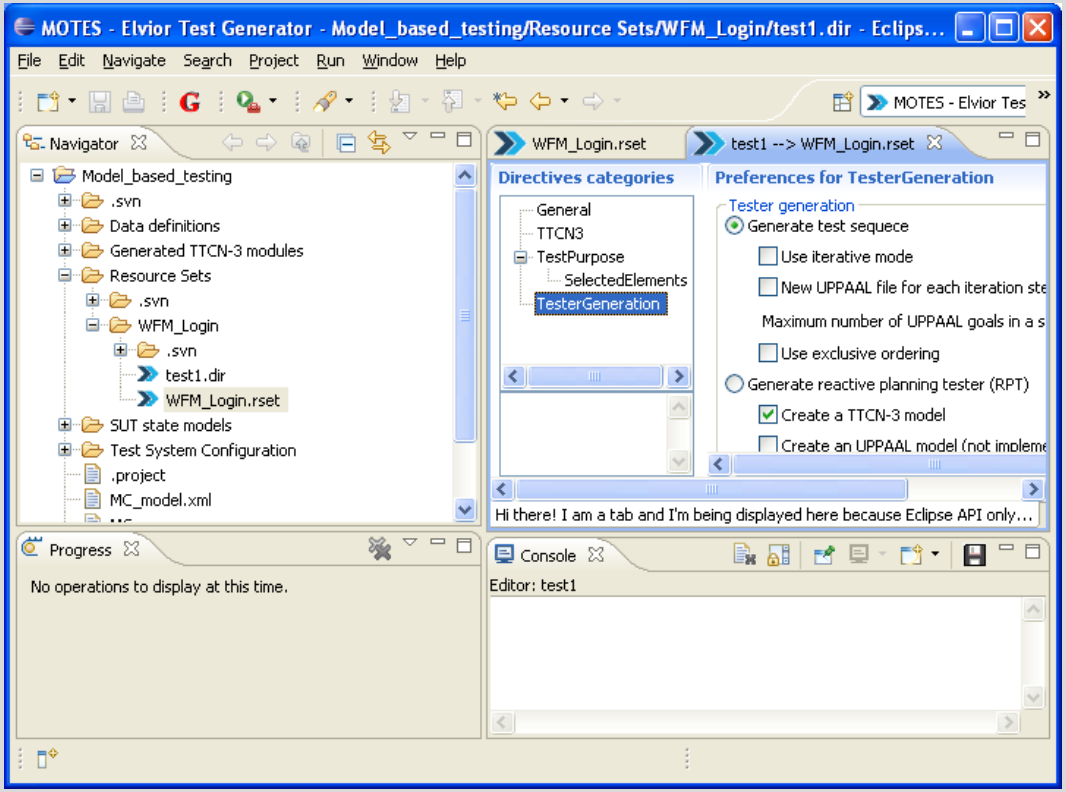
- Used for creating SUT model
  - transition language
  - subset of TTCN-3



elvior software test automation

Tools – TestCast Generator (Elvior test generator)

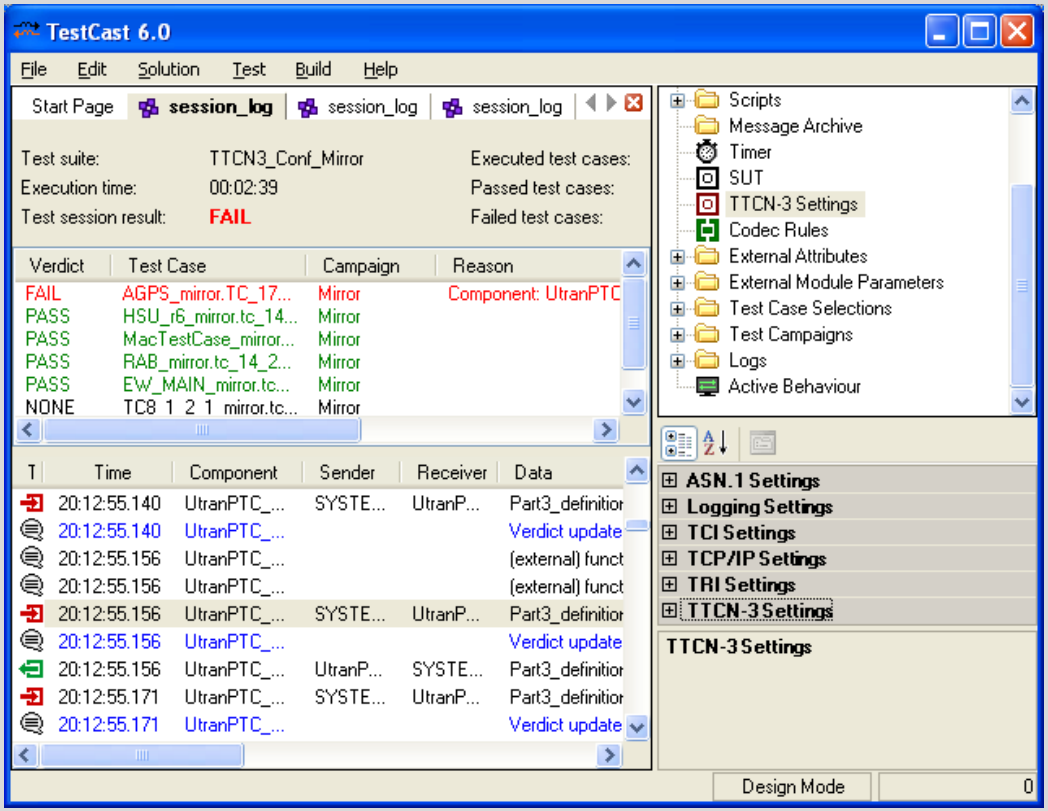
- Used for generating tests (TTCN-3 scripts)
  - uses SUT model in XMI format (created by Poseidon)
  - Runs on Eclipse platform



elvior
software test automation

Tools – TestCast (Elvior TTCN-3 test tool)

- Used for executing TTCN-3 tests
  - uses test scripts generated by TestCast Generator
  - Runs on .NET framework

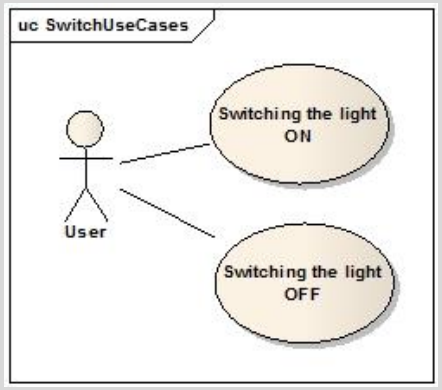


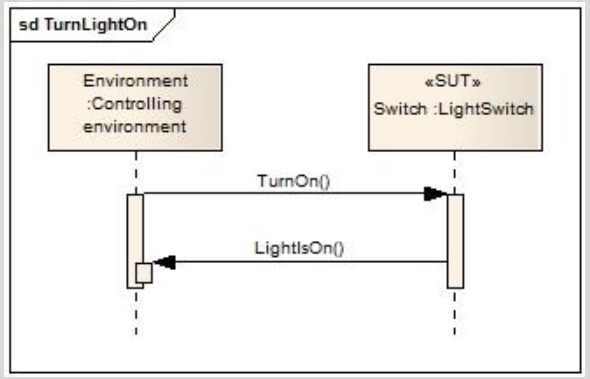
www.elvior.com

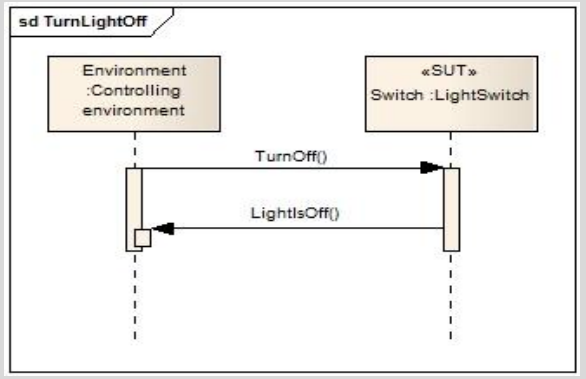
elvior software test automation

SUT–LightSwitch–the example SUT (description)

The system under test (SUT) is a lighting system that consist of a switch that turns lights on or off at the user's request







elvior software test automation

SUT–LightSwitch–the example SUT (requirements)

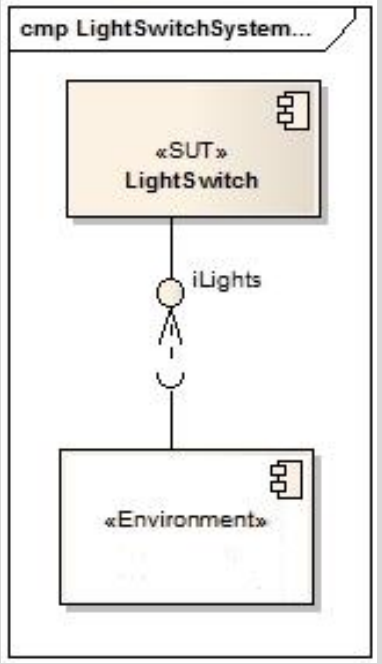
- The light shall be **switched on** by the request from the controlling environment,
- The light shall be **switched off** by the request from the controlling environment.
- If the light is already on/off, requesting the same operation (turning light on/off respectively) shall not change the system state.
- if SUT receives not supported command, then it notifies the controlling environment.
- https://d-mint.cc.ioc.ee/moodle/

SUT–LightSwitch–the example SUT (use cases)

#	Precondition	Input (to the SUT)	Expected result (from the SUT)
1	Light is off	Command turnOn	lightIsOn
2	Light is on	Command turnOff	lightIsOff
3	Light is off	Command turnOff	lightIsOff
4	Light is on	Command turnOn	lightIsOn
5	Light is on or off	Unknown command	Unrecognised command

SUT–LightSwitch–the example SUT (interface)

1. SUT interacts with outside world using console interface (standard input/output)
2. iLights interface defines commands and SUT responses



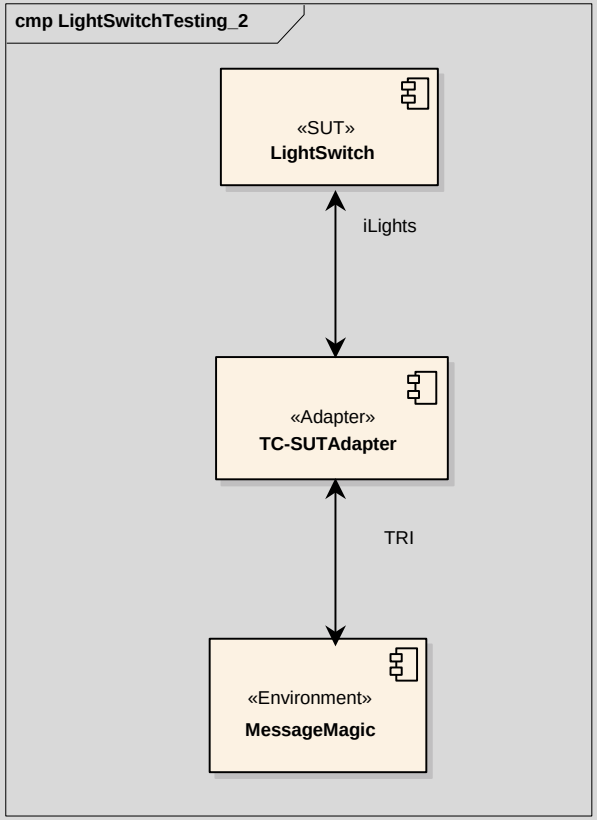
#	Input (to the SUT)	Output (from the SUT)
1	string command	string currentLampState

Text constants for input and respective output

#	Input (to the SUT)	Output (from the SUT)
1		ready
2	turnOn	lightIsOn
3	turnOff	lightIsOff
4	xyz	Unrecognized command
5	exit	

SUT–LightSwitch–the example SUT (test environment)

Adapter between test tool and SUT is needed.



System Adapter used in example (general)

- System Adapter (SA) connects testing tool (TestCast (TC)) with System Under Test (SUT).
- TRI - TTCN-3 standardizes interface between testing tool and SA, this interface is called TTCN-3 Runtime Interface.
- Interface between SA and SUT is always proprietary and therefore needs to be implemented within SA.

- TRI interface is mapped for different languages (C, C++, C#, Java) (Part 5: TTCN-3 Runtime Interface)
- Implementation is tool dependent.
- Most important is what to implement in the methods of the interfaces (i.e. triSend, triEnqueueMsg, triMap)

System Adapter used in example (implementation)

1. Implemented in C#, separate executable
2. SUT specific implementations for ITriCommunicationSA:
   - TriMap, TriUnmap
   - TriSend
   - TriExecuteTestCase, TriEndTestCase

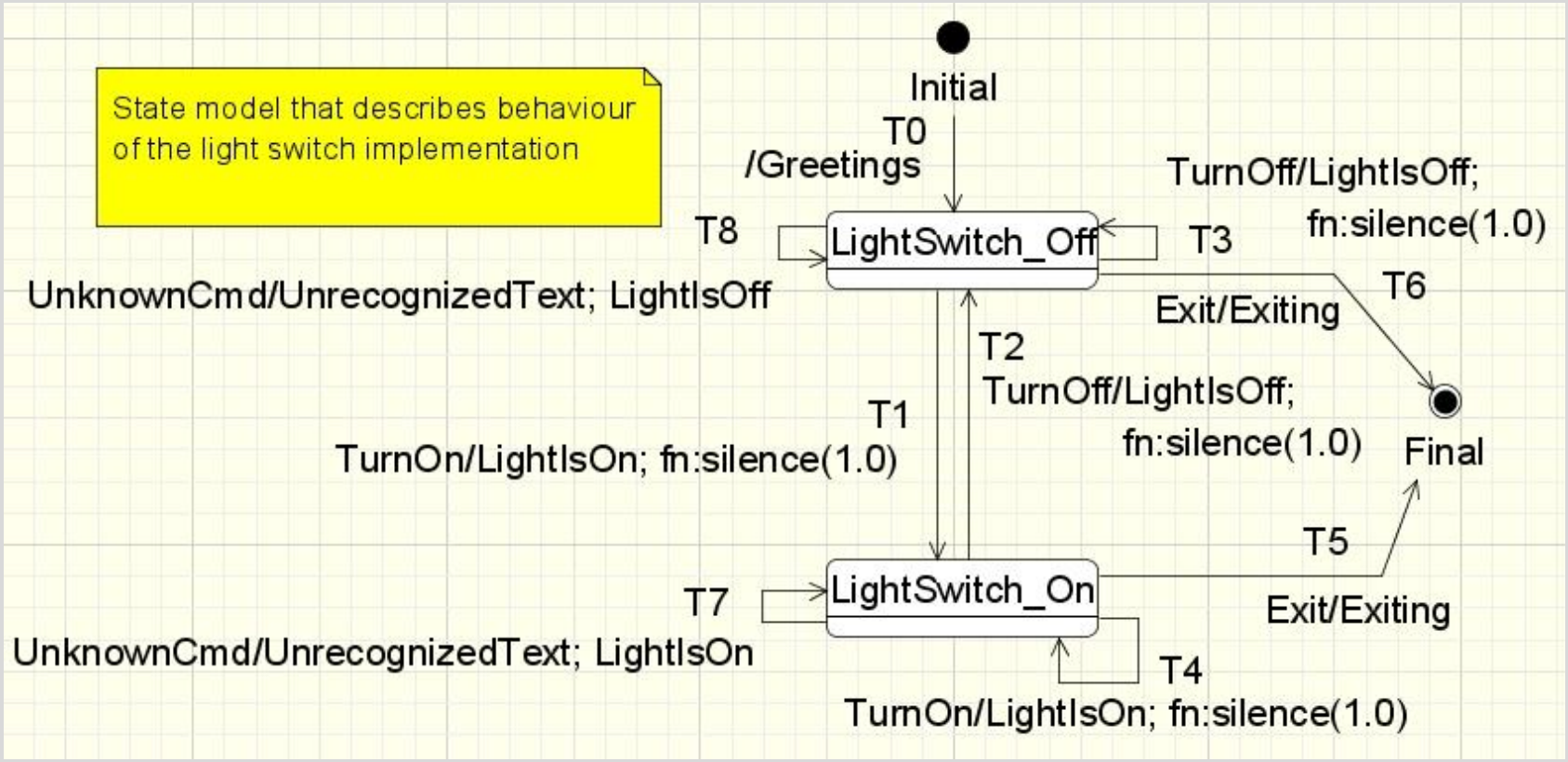
3. SUT specific implementations for ITriCommunicationTE:
   - EnqueueMessage

4. Additional functionality:
   - SUT start, stop
   - Msg traffic logging in SA window

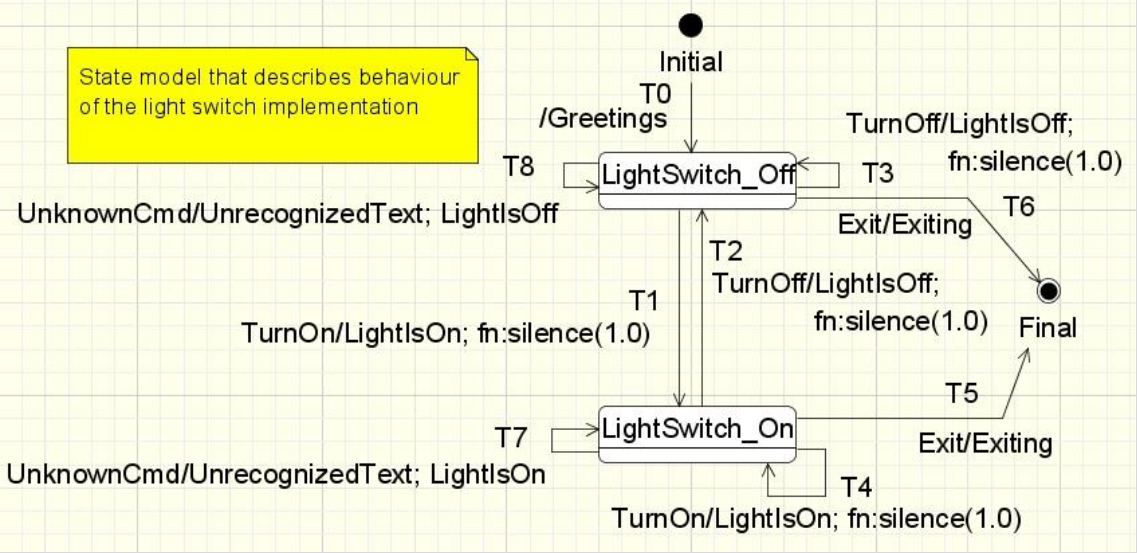
Workflow of MBT (Elvior approach)

- Create SUT model.
- Prepare test data, messages, configuration, functions in TTCN-3.
- Create system adapter according to TTCN-3 TRI.
- Create codecs.
- Generate tests for specified test goal.
- Execute tests.
- Evaluate results and continue with next increment.

State Model of SUT



Inputs from external tools (Poseidon case tool)



Transition	Starting state	Trigger (cmd)	Effect (output)	Next state
T1	LightSwitch_Off	TurnOn	LightIsOn, Silence	LightSwitch_On
T2	LightSwitch_On	TurnOff	LightIsOff, silence	LightSwitch_Off
T3	LightSwitch_Off	TurnOff	LightIsOff, silence	LightSwitch_Off
T4	LightSwitch_On	TurnOn	LightIsOn, Silence	LightSwitch_On
T7	LightSwitch_On	UnknownCmd	LightIsOn, Silence	LightSwitch_On
T8	LightSwitch_Off	UnknownCmd	LightIsOff, Silence	LightSwitch_Off

Inputs from external tools (static scripts used in test cases generation)

1. TestData.ttcn describes the possible messages (commands) sent to SUT, such as turnOn and turnOff (for turning the switch on/off), and possible response types from SUT (such as lightIsOn, lightIsOff).

2. TestConfiguration.ttcn describes test component (Tester), its port for message exchange (iLights) and the message types (Command and Output). In addition, it defines the function silence(float duration_sec) for better visualization of the LightSwitch SUT.

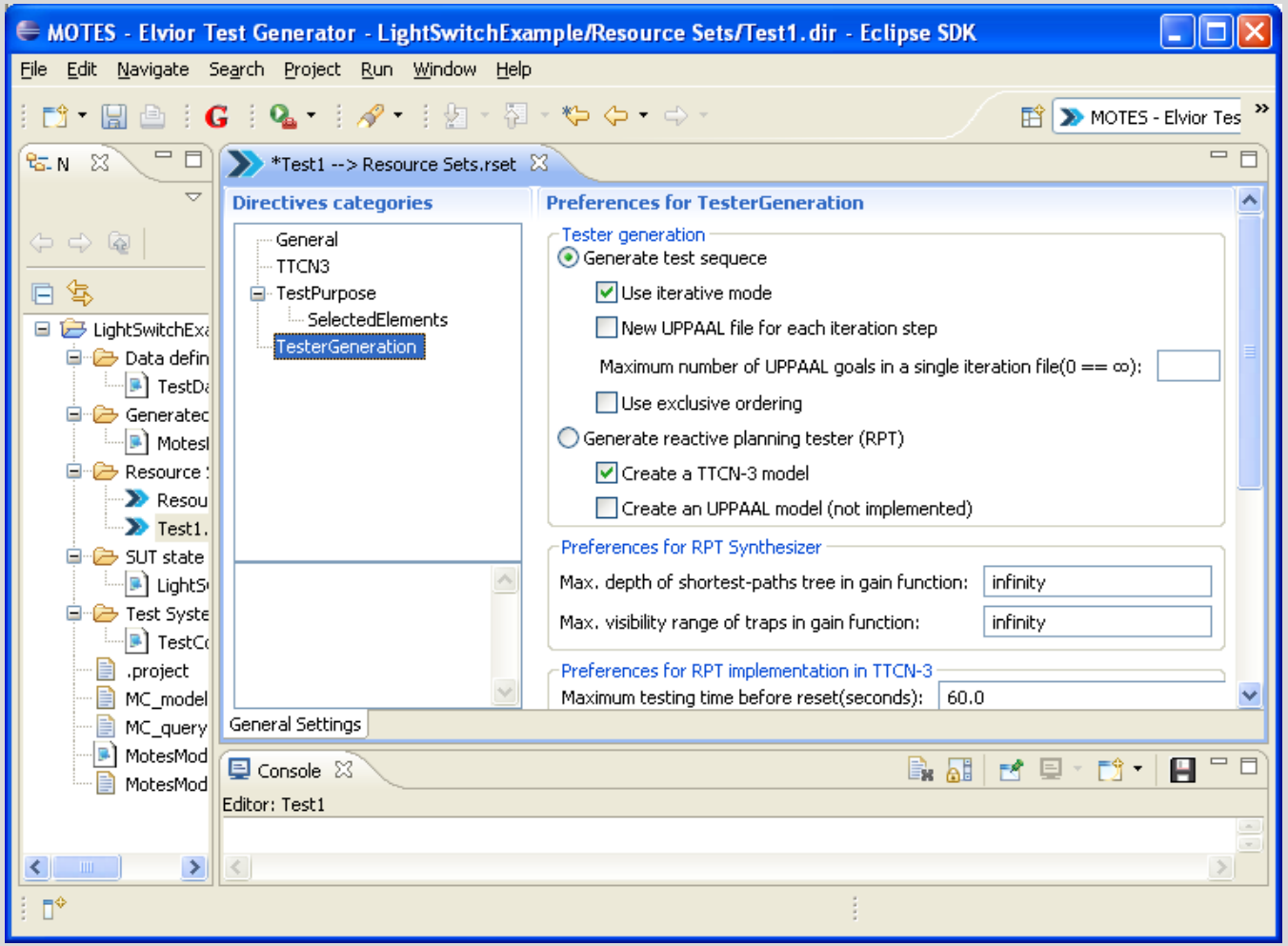
Test cases generation

Precondition: Eclipse framework and TestCast Generator are installed

Steps for test scripts generation:

- Creating a new TestCast Generator project
- Handling test generation inputs (from external tools)
- Linking external test inputs to a test generation task resource set
- Defining guidelines for a test generation task
- Generating TTCN-3 test scripts

Test cases generation – TestCast Generator preferences

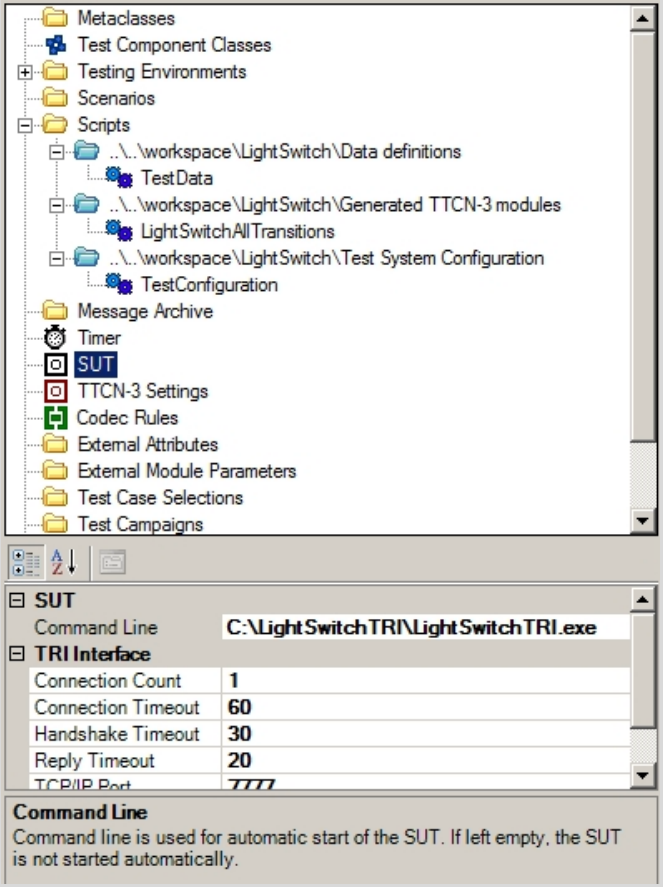


www.elvior.com

elvior
software test automation

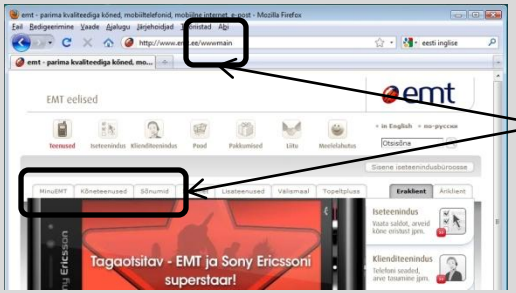
Execution of generated test cases

Precondition: TestCast TTCN3 tool installed, system adapter exists (TRI), SUT is reachable.



elvior
software test automation

WEB page testing – industrial case study 1



Test purpose

SUT model

Selenium Core

Remote Control Server

System Adapter

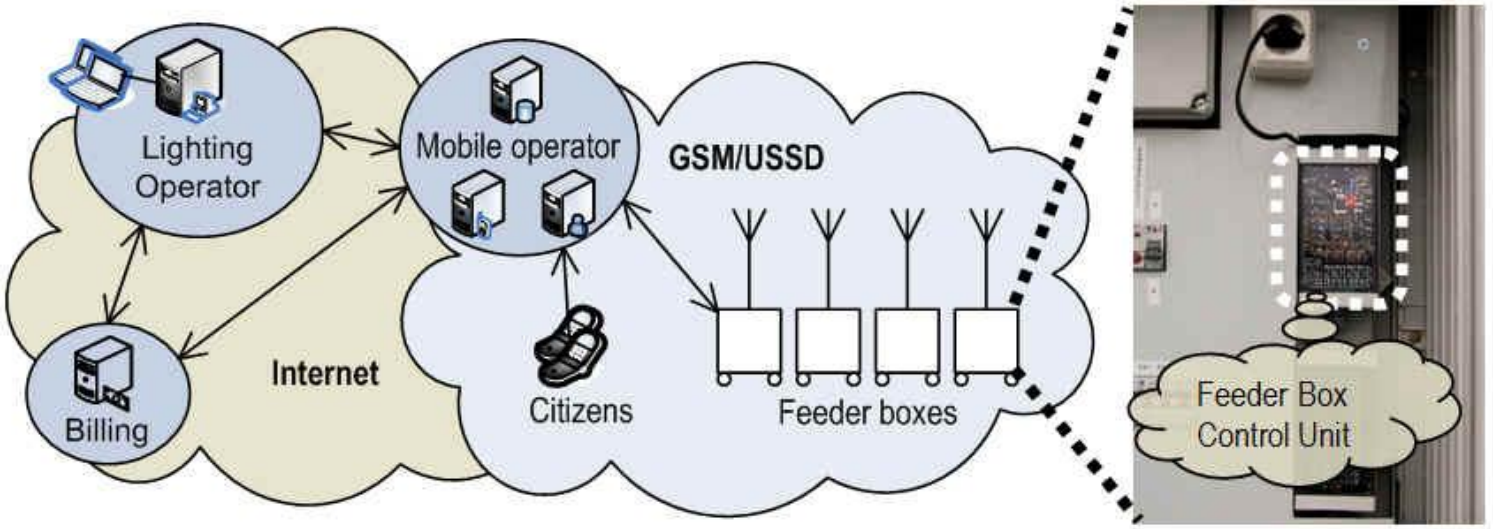
TestCast
TTCN-3 test tool

TestCast
Generator

elvior
software test automation

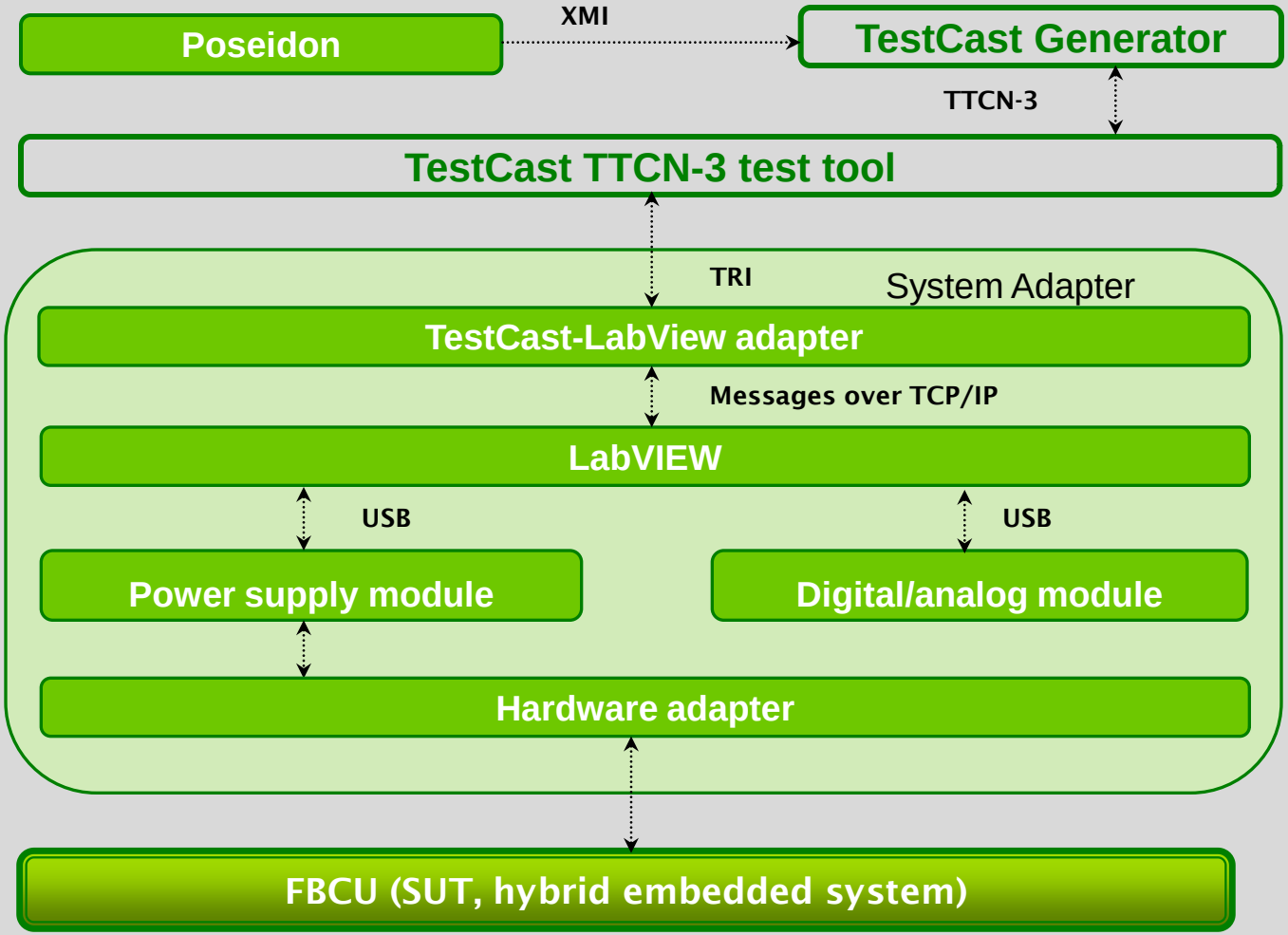
Industrial case study 2 - Feeder Box Control Unit

Feeder Box Control Unit (FBCU). It is a subsystem of the street lighting control system functioning today in Tartu, the second biggest city of Estonia.



elvior
software test automation

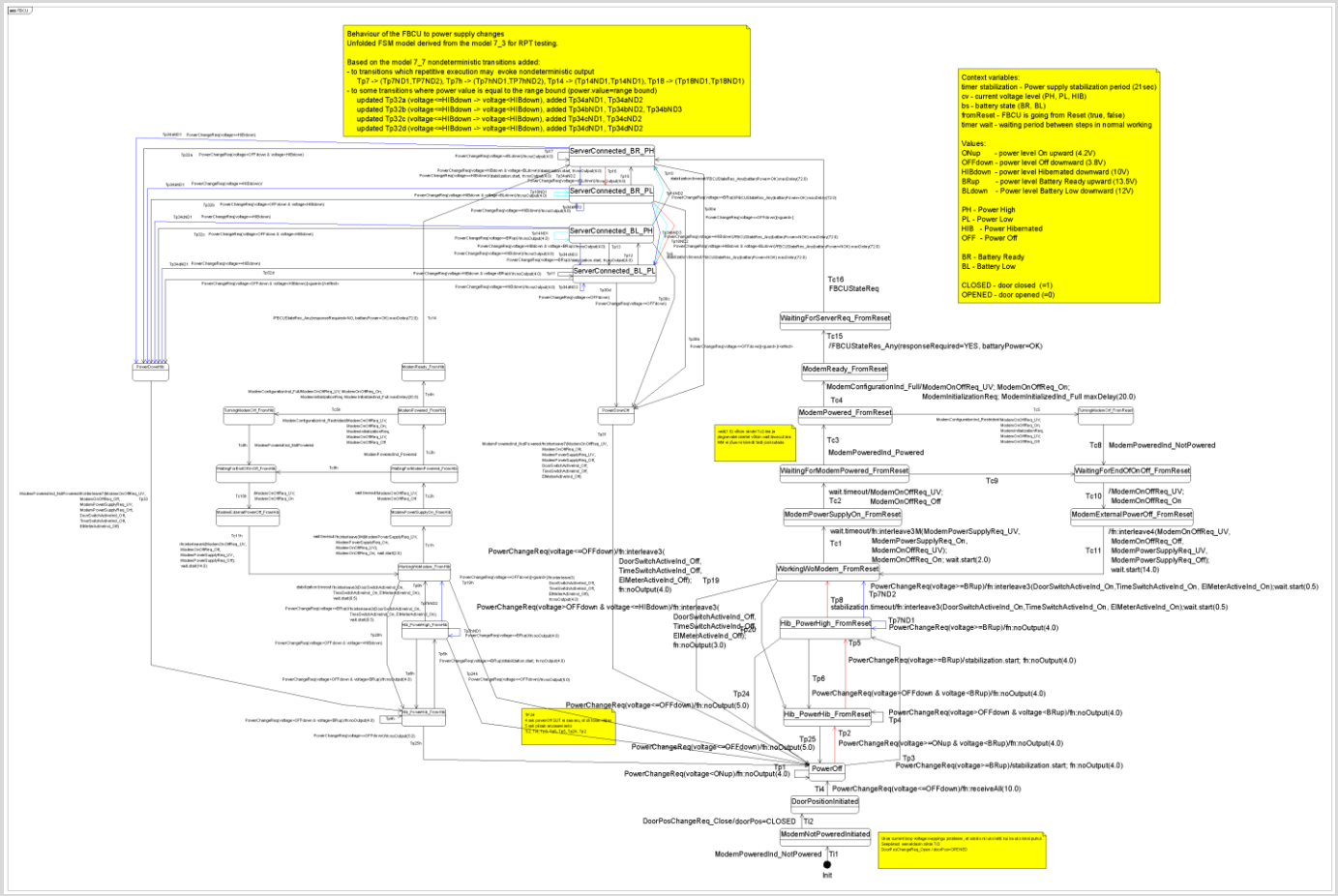
Industrial case study 2 – test environment



elvior
software test automation

Industrial case study 2 – SUT state model

Model of FBCU power management (31 states, 73 transitions)



www.elvior.com

elvior
software test automation

Industrial case study 2 – results, increment 1

Using MBT in this case study is very efficient, because FBCU behavior is complex and it is easier to change model than rewrite test code – proved in practice.

Numbers (first increment):

		Time	Code lines
1	TTCN-3 code (messages, test data, configuration)	~ 15 days	~ 1100
2	System adapter	150 days	~ 15 000
3	Model building	~ 45 days	NA
4	Generated tests	NA	~ 20 000

Industrial case study 2 – results, increment 2

FBCU changed significantly, new model was built from scratch.

Numbers (second increment):

		Time	Code lines
3	Model building	~ 10 days	NA
4	Generated tests	NA	~ 20 000

3 fatal bugs found.

Conclusion

- There are common tasks to be solved in both cases (manual and model based TTCN-3 testing).
- Using MBT with TTCN-3 gives extra advantage (TTCN-3 is dedicated for tests, it is natural to generate TTCN-3).
- Building the model formalizes SUT behavior and therefore discloses ambiguity in SUT specifications.
- Model building is resources consuming work, it pays back in maintenance phase – it is easier to alter model and generate tests again.
- MBT advantages are more visible with complex SUT models.
- MBT gives very handy approach for exploratory testing.

Thank you !

Questions ?

References:
testcast.elvior.com
www.d-mint.org

Supported by:

elvior
software test automation