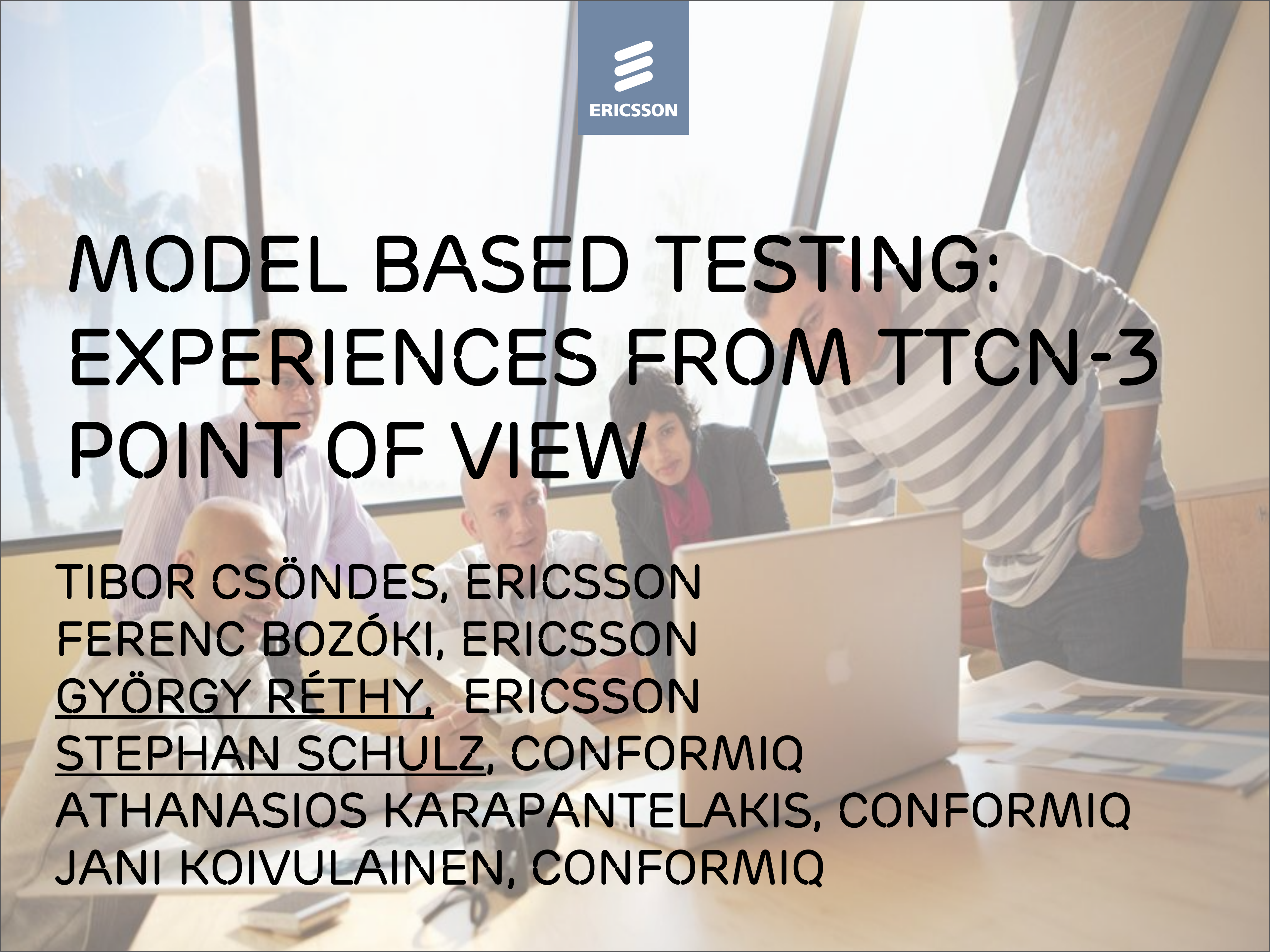
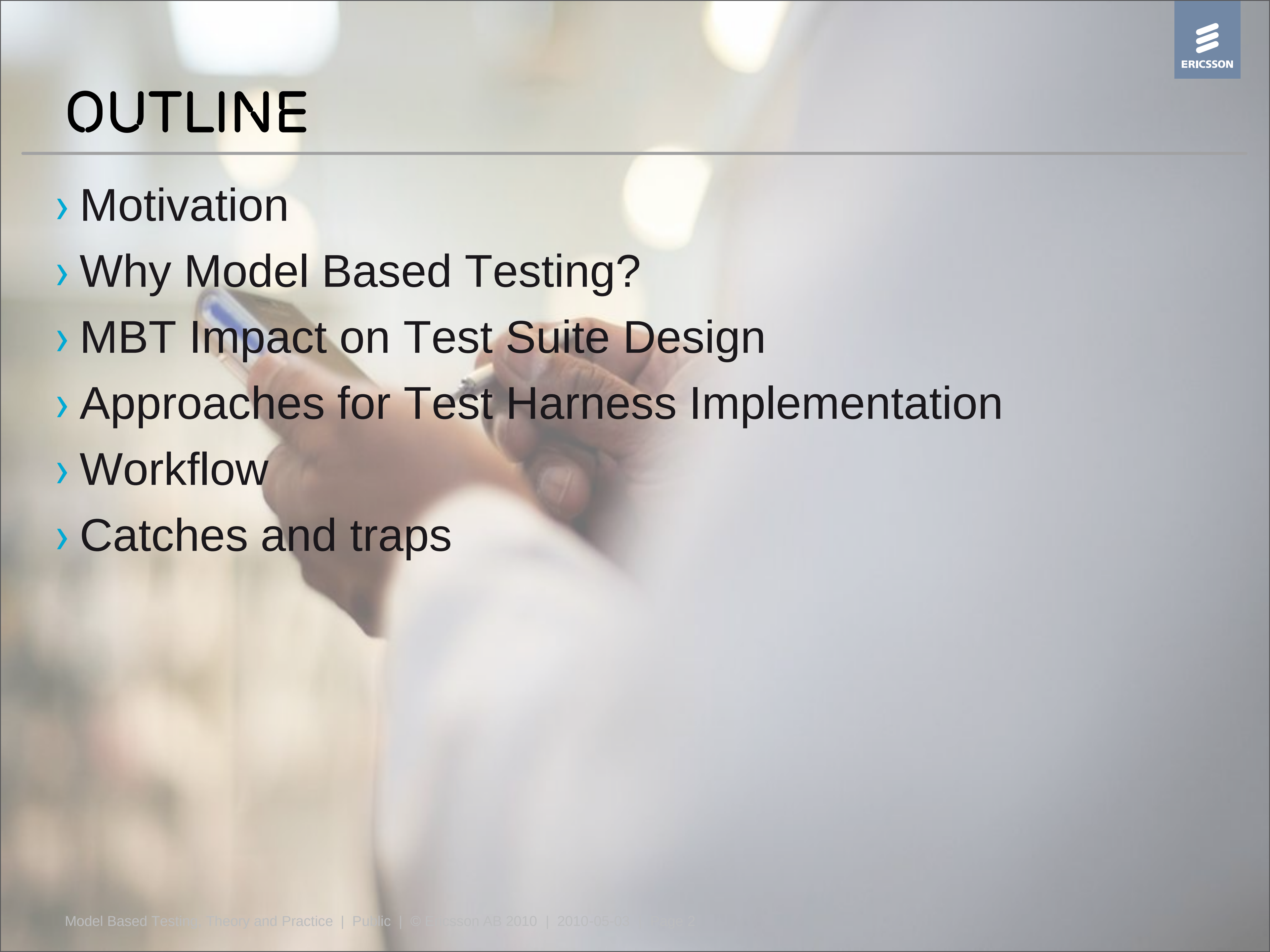


MODEL BASED TESTING: EXPERIENCES FROM TTCN-3 POINT OF VIEW

TIBOR CSÖNDES, ERICSSON
FERENC BOZÓKI, ERICSSON
GYÖRGY RÉTHY, ERICSSON
STEPHAN SCHULZ, CONFORMIQ
ATHANASIOS KARAPANTELAKIS, CONFORMIQ
JANI KOIVULAINEN, CONFORMIQ

OUTLINE

- Motivation
- Why Model Based Testing?
- MBT Impact on Test Suite Design
- Approaches for Test Harness Implementation
- Workflow
- Catches and traps

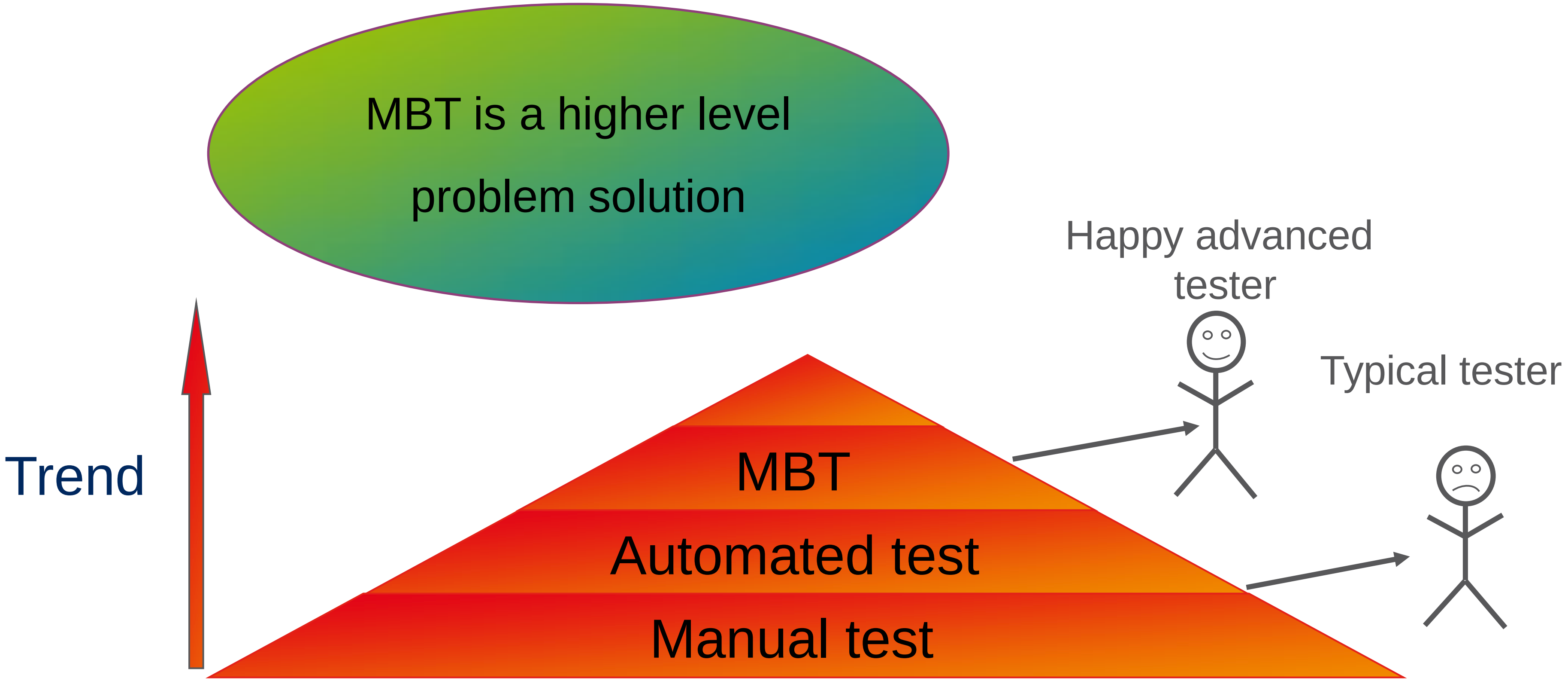


MOTIVATION

- Introduction of Model Based Testing in context of TTCN-3
- Give a summary about the differences of manually designed and model based test suites
- Investigate the different approaches of test harness implementation
- Share our experiences with model generated TTCN-3 test suites

TEST AUTOMATION

- “Classical” test automation: automation of test execution
- MBT: automation of test design (automatic test generation from a model)



MBT is a higher level problem solution

Happy advanced tester

Typical tester

Trend

MBT

Automated test

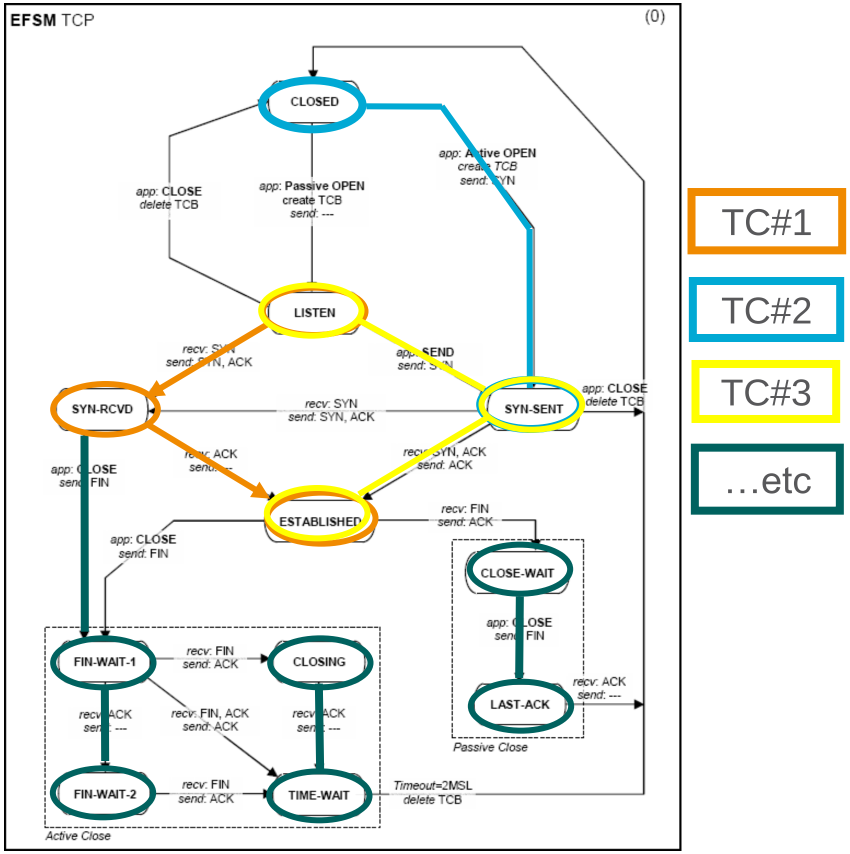
Manual test

TEST AUTOMATION (CONTD.)

› "Classical" automated testing

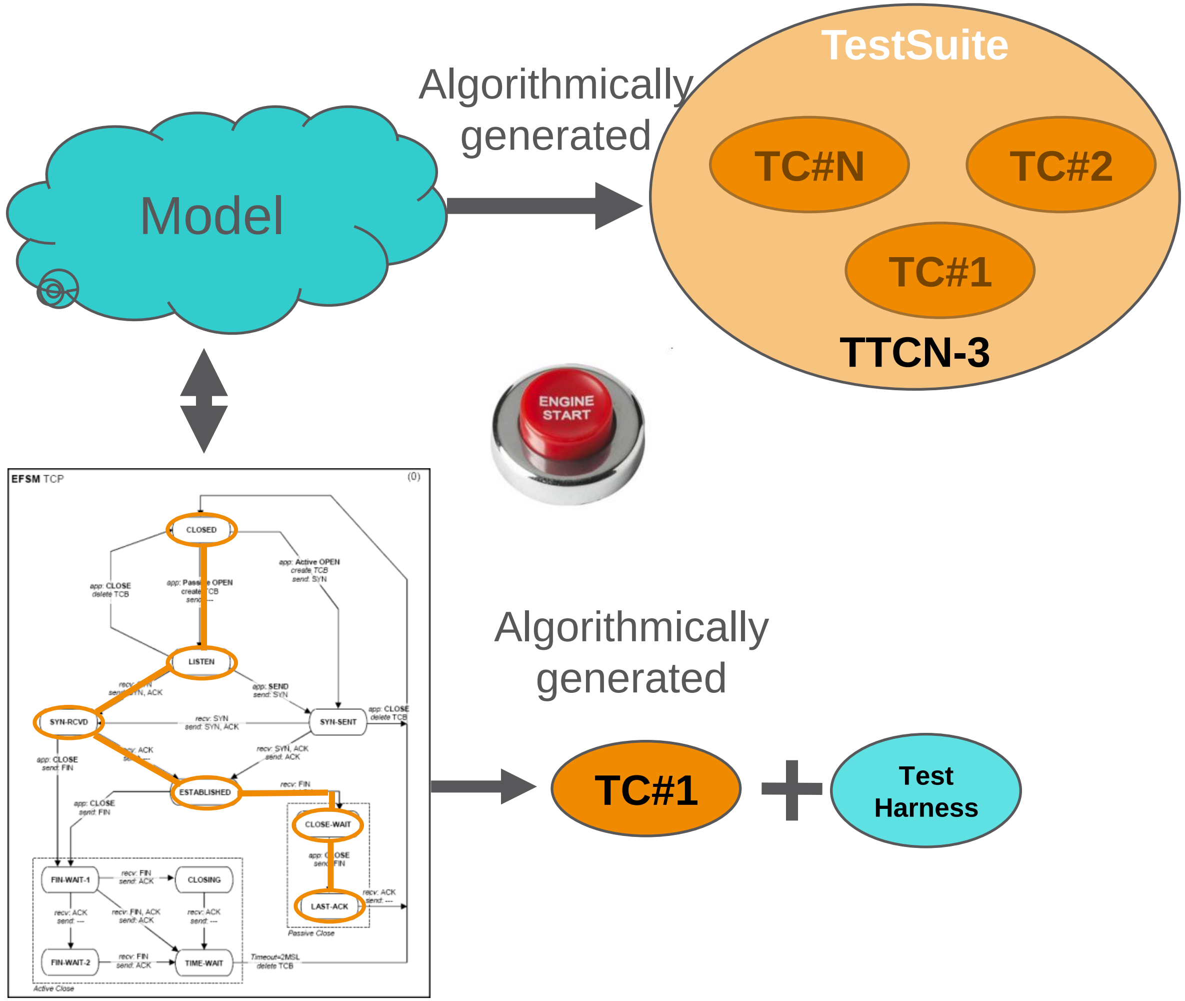
- each test case checks one or a few transitions
- each test case is developed separately
- each test case is maintained separately
- each test engineer is exposed to details of SUT interfaces

```
testcase tc_TP#1 ()
 runs on MTCType_CT system MTCType_CT
 {
  map(mtc:My_PCO, system:My_PCO);
  My_PCO.send ( t_SYN(A));
  alt {..
     [] My_PCO.receive(t_SYN_ACK(A+1,B))
       {My_PCO.send(t_ACK(A+1,B+1))
        setverdict(pass)};
  }
```



TEST AUTOMATION (CONTD.)

› Model Based Testing
  - tests are generated from an SUT model
  - at SUT change the model is updated and test cases are re-generated
  - models only include interface aspects & data related to the functionality to be tested
  - tests are generated based on coverage criteria



Model

Algorithmically generated

TestSuite

TC#N

TC#2

TC#1

TTCN-3

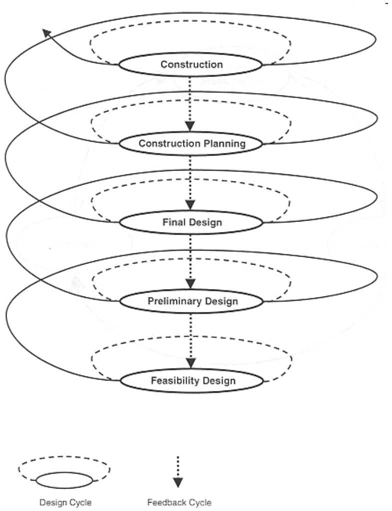
Algorithmically generated

TC#1 + Test Harness

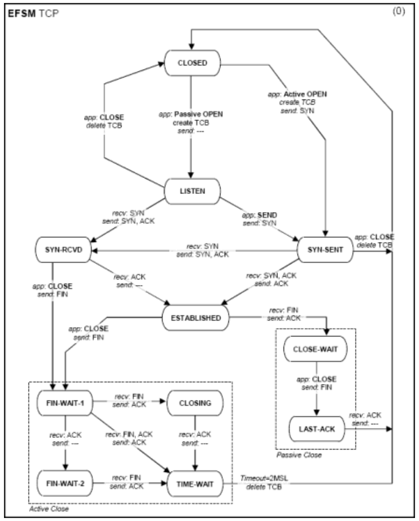
MODEL BASED TESTING ON FIELD

- Pros and Cons of Model Based Testing
  - Reduces fault slip through

Design phase



Testing phase



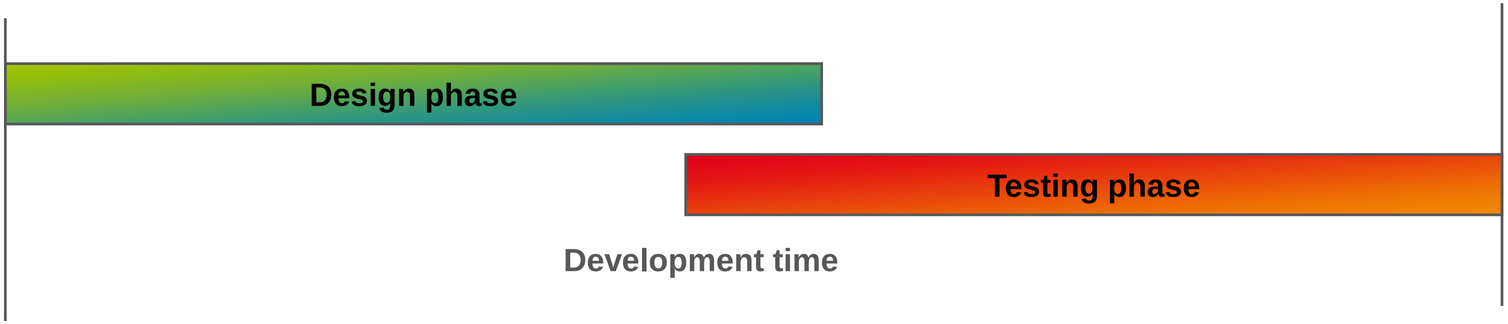
Model development of the Design and model development of the Testing could take place parallel

➔ model development for testing verifies the model of the design

➔some faults could be found in the "development phase"

➔Reduces development time

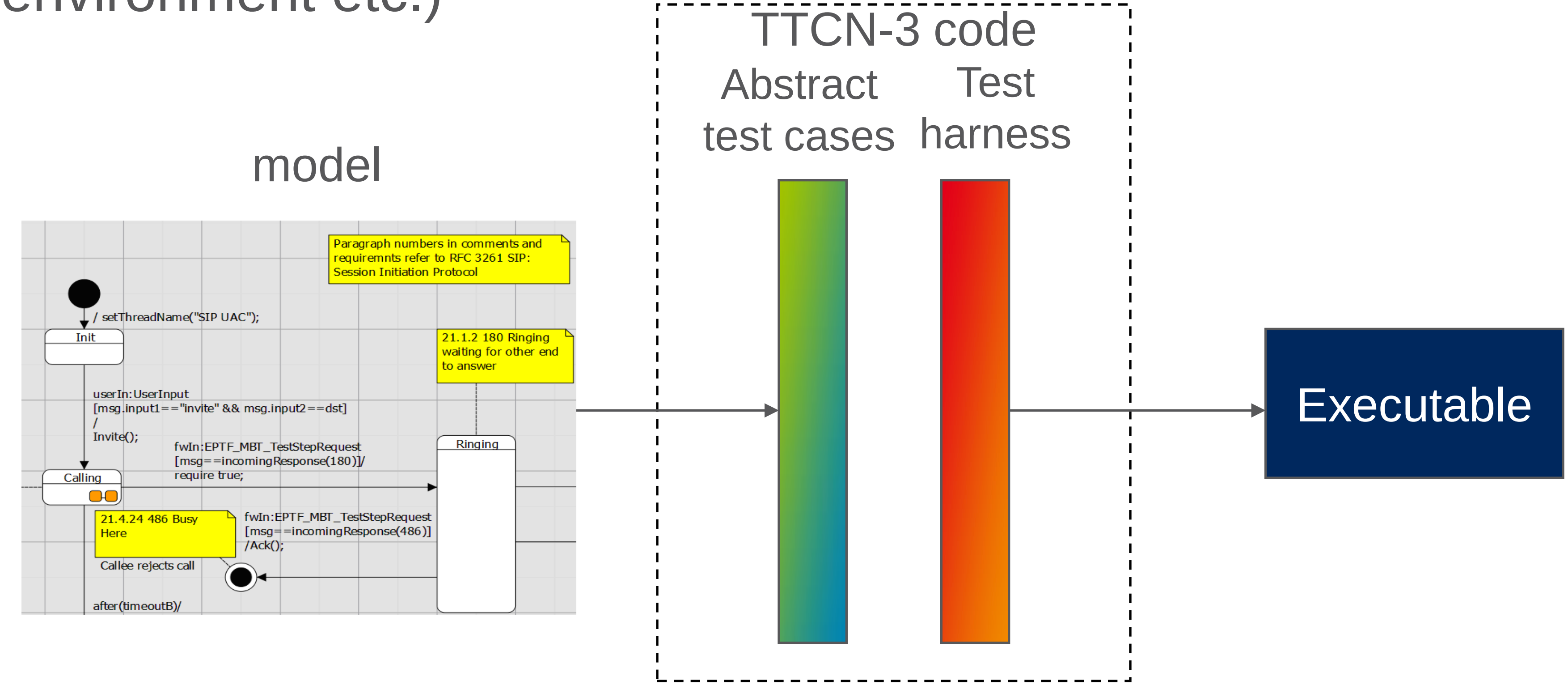
➔Model Driven Engineering



Development time

CASE STUDIES: TEST ARRANGEMENT

- Generated tests: abstract TTCN-3 test cases (not directly executable)
- Test harness: all the the extras that makes the abstract test cases executable (TTCN-3 code, adapters, TTCN-3 tool environment etc.)



model

TTCN-3 code

Abstract test cases

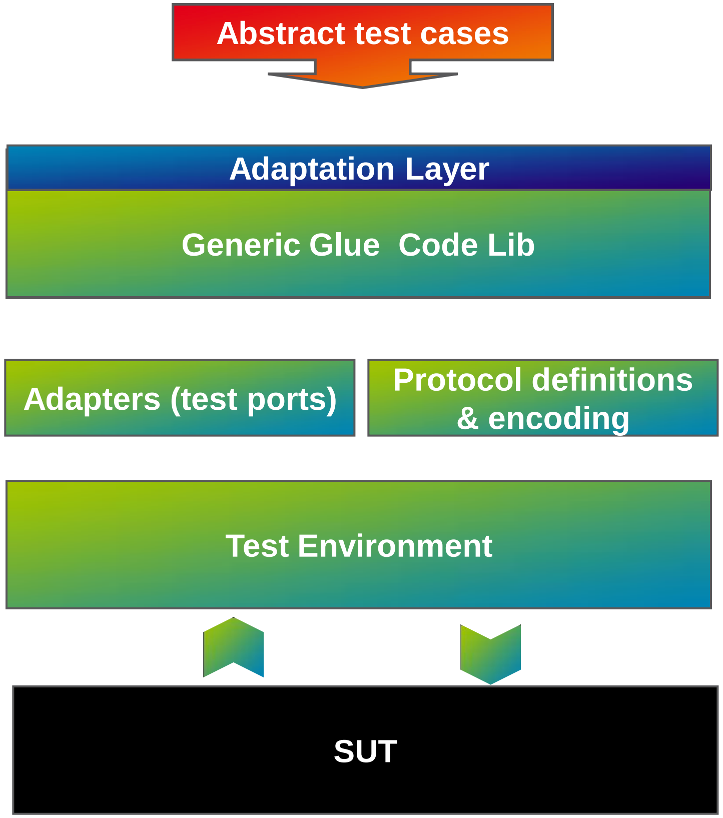
Test harness

Executable

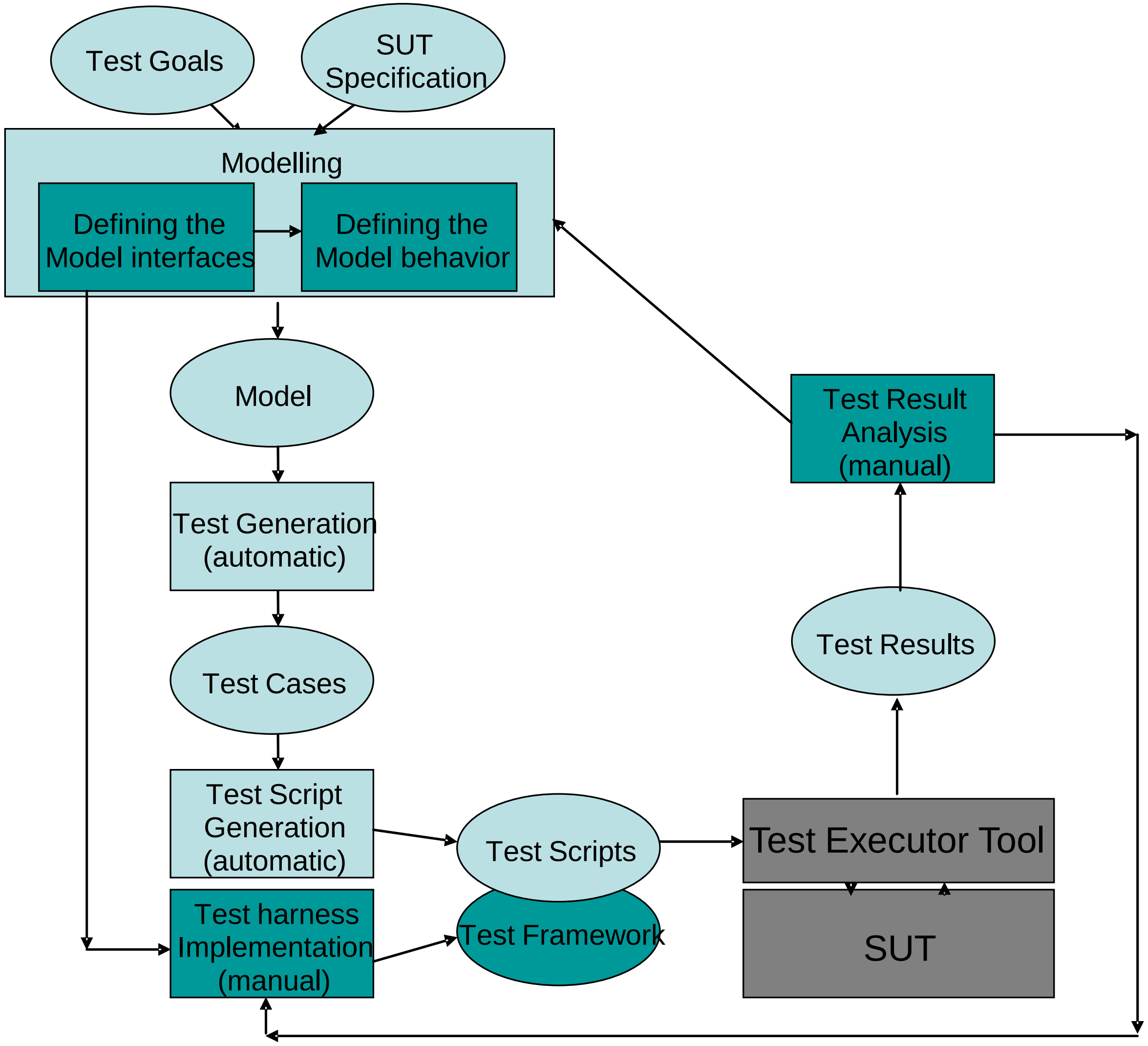
Paragraph numbers in comments and requiremnts refer to RFC 3261 SIP: Session Initiation Protocol
/ setThreadName("SIP UAC");
Init
21.1.2 180 Ringing waiting for other end to answer
userIn:UserInput [msg.input1=="invite" && msg.input2==dst] / Invite();
fwIn:EPTF_MBT_TestStepRequest [msg==incomingResponse(180)]/ require true;
Ringing
Calling
21.4.24 486 Busy Here
fwIn:EPTF_MBT_TestStepRequest [msg==incomingResponse(486)] /Ack();
Callee rejects call
after(timeoutB)/

APPROACHES FOR TEST HARNESS

- Hand written glue code
  - Demands advanced knowledge of TTCN-3 and the TTCN-3 tool
  - Demands advanced knowledge of the underlying test harness
  - Repeated development if the tested scenario changes
  - Test harness is project-specific
- Using generic glue code
  - Built on top of already existing generic SW libraries (TitanSim)
  - Requires only minor project-specific adaptation
  - Generic part: write once, use several times: additional gain to test case generation



WORKFLOW


Test Goals
SUT Specification
Modelling
Defining the Model interfaces
Defining the Model behavior
Model
Test Generation (automatic)
Test Cases
Test Script Generation (automatic)
Test harness Implementation (manual)
Test Scripts
Test Framework
Test Executor Tool
SUT
Test Results
Test Result Analysis (manual)

EXPERIENCES, RECOMMENDATIONS

- MBT is a paradigm shift
- “Right” competence is required, training is needed
- New roles should be established within the test organisation, especially the model designer/”test architect”
- When designing the **good** model, the tester shall not think in terms of test cases – the tester should, ultimately, only think of the system behaviour
- The generated test cases cover several events (Model/Test requirements), while the traditional test cases normally only cover one event/situation
- Start with a smaller, well defined, well encapsulated, area/functionality
- Save time and money! On average: ~20-30%

ERICSSON