Model Driven Design of a Test Automation Software using OpenModelica

Lutz Berger
Bernhard Thiele

LINKÖPING UNIVERSITY

BIT COSMOS

Abstract

- Model Driven Design (MDD): Architecture design as model which is used for the verification of requirements
  - Design of test automation software
  - Simulation of design in simulation environment
  - Find faults in early stage => time to market, costs

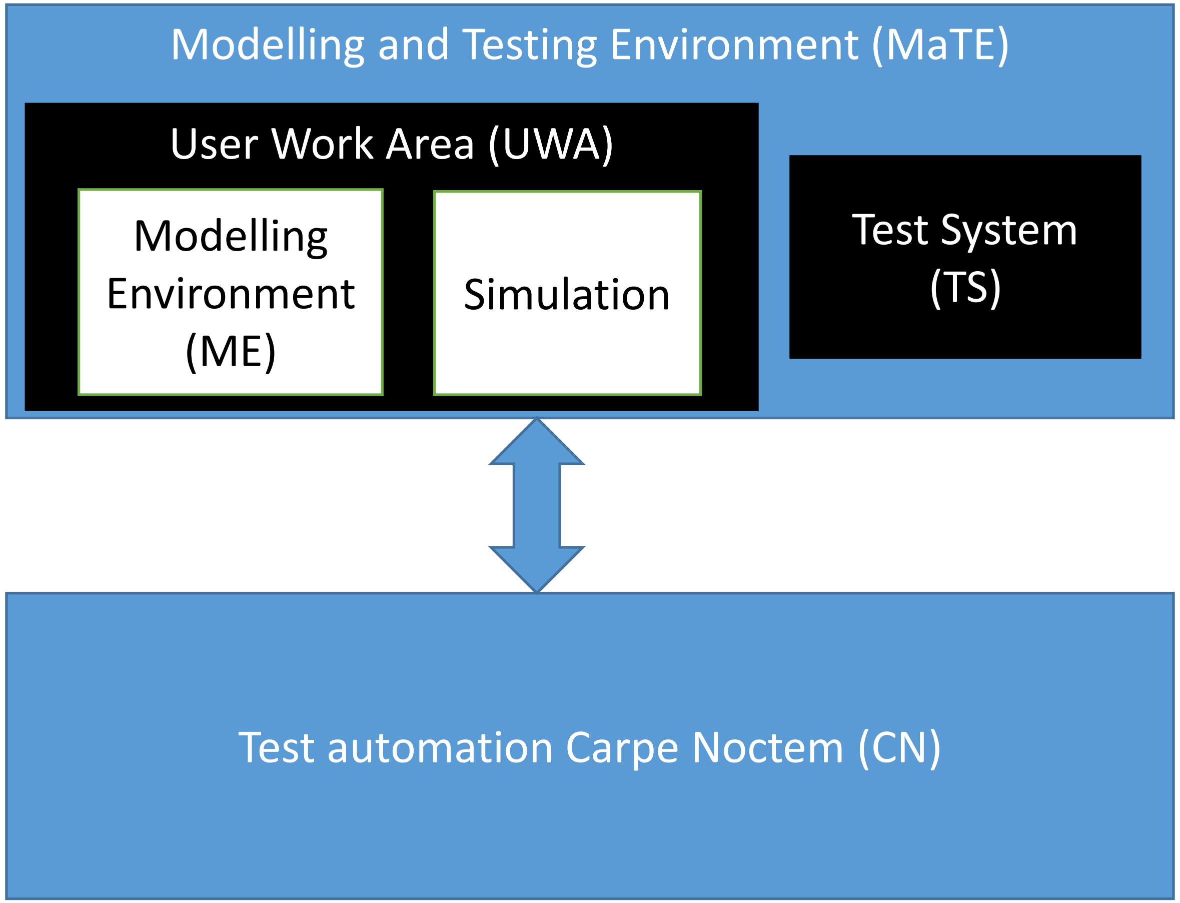
Introduction

- Case study on test automation software “Carpe Noctem” (CN)
  - Some Requirements
    - Test-sets shall be queued when starting on same Machine
    - Test-sets shall start not before a configurable start time
    - Test-sets shall stop when exceeding a configurable stop time
    - Test-sets shall restart if a system error is detected if configured
    - Each test shall stop with a configurable time-out
    - Each test shall repeat until successful (configurable)
    - Each test shall repeat not more times than a configurable parameter
  - Realization with Hierarchical State Machine

Introduction

- Modelica states
- Embedded in environment model
- Contribution to state-machine implementation in OpenModelica
- Final successful running on v1.13.0-dev-122-gfba8150

Test Automation of Flight Simulator



Modelling and Testing Environment (MaTE)

User Work Area (UWA)

Modelling Environment (ME)

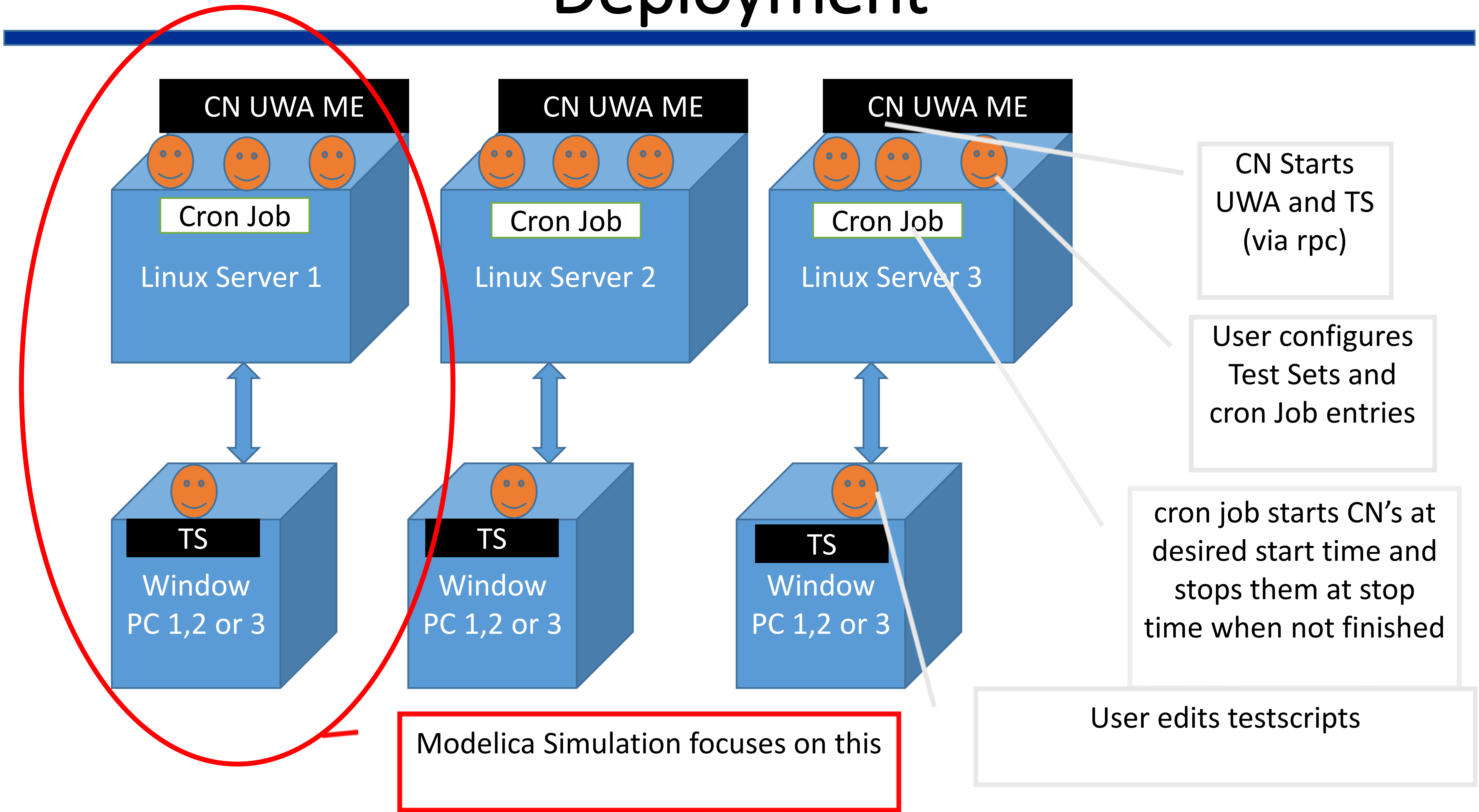
Simulation

Test System (TS)

Test automation Carpe Noctem (CN)

- UWA’s run on
  - Different Machines
  - Each Machine is dedicated for special tests of software parts
  - The TS communicates to the UWA via sockets. It executes test scripts written in a domain specific language
  - Each UWA contains several programs spawned by the ME
- Two instances of UWA’s should not run on the same machine at the same time
- ME schedules the programs of the UWA, provides API for programs, manages transactions on the simulated avionics bus.

Deployment



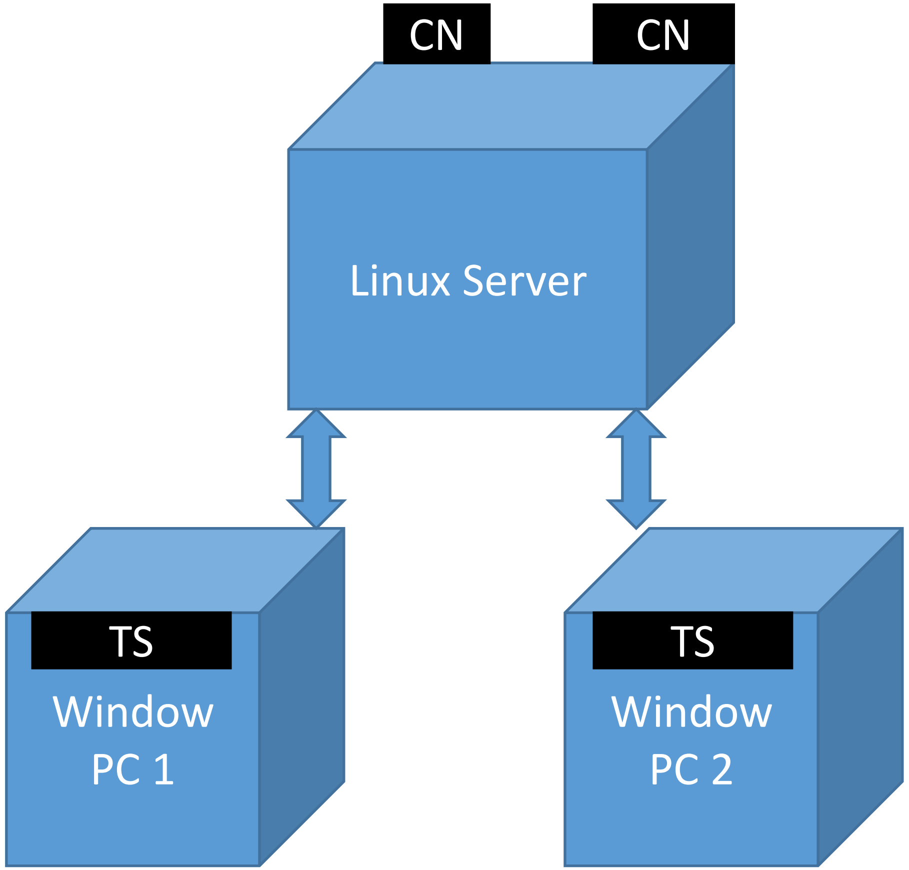
CN Starts UWA and TS (via rpc)

User configures Test Sets and cron Job entries

cron job starts CN's at desired start time and stops them at stop time when not finished

User edits testscripts

Modelica Simulation focuses on this



No parallel execution of CN
i.e. MaTE , only sequential
is allowed

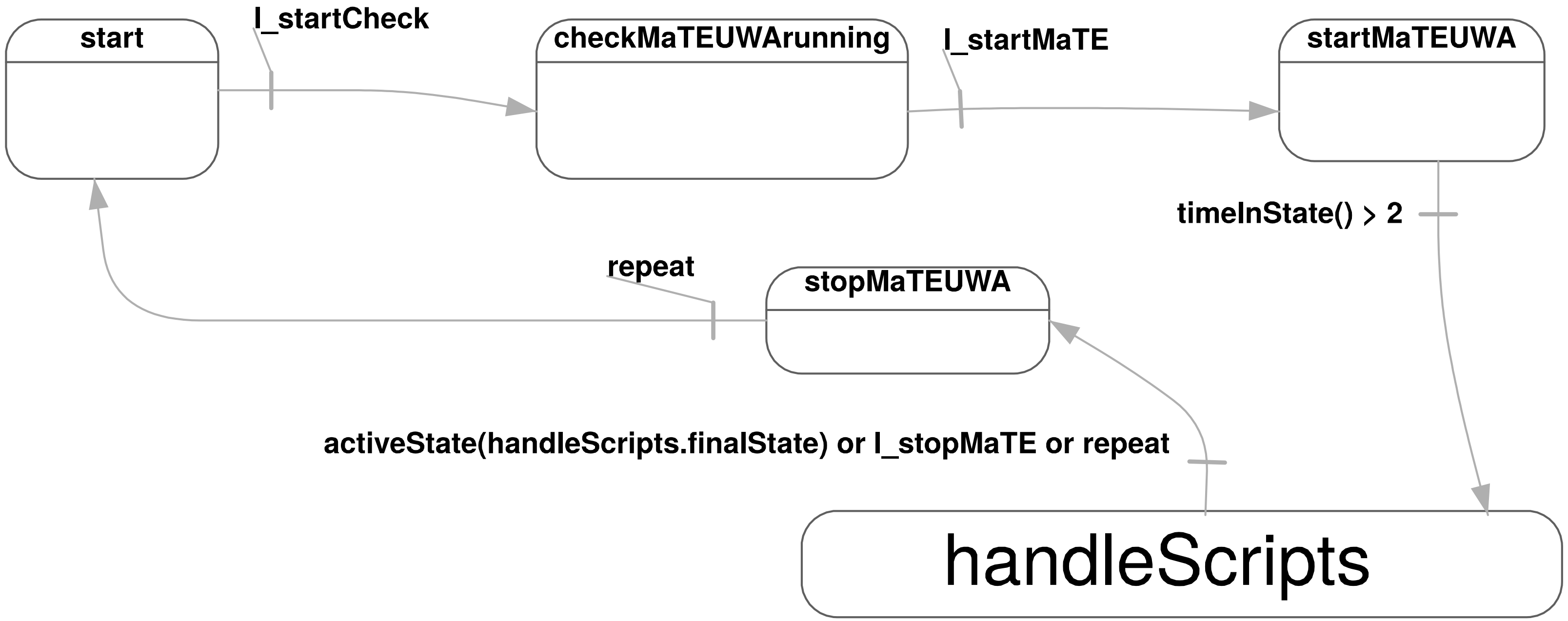
- CN (Linux server) starts UWA(Linux server) and TS(Windows PC)
- UWA start ME and Simulation with rehosted[1] SW from aircraft and simulation software
- CN connects via remote procedure call (rpc) the UWA one TS
- CN starts all tests in a test set via rpc on that TS and collects the results

[1] rehosted means: code from aircraft transferred to and adopted for the simulator

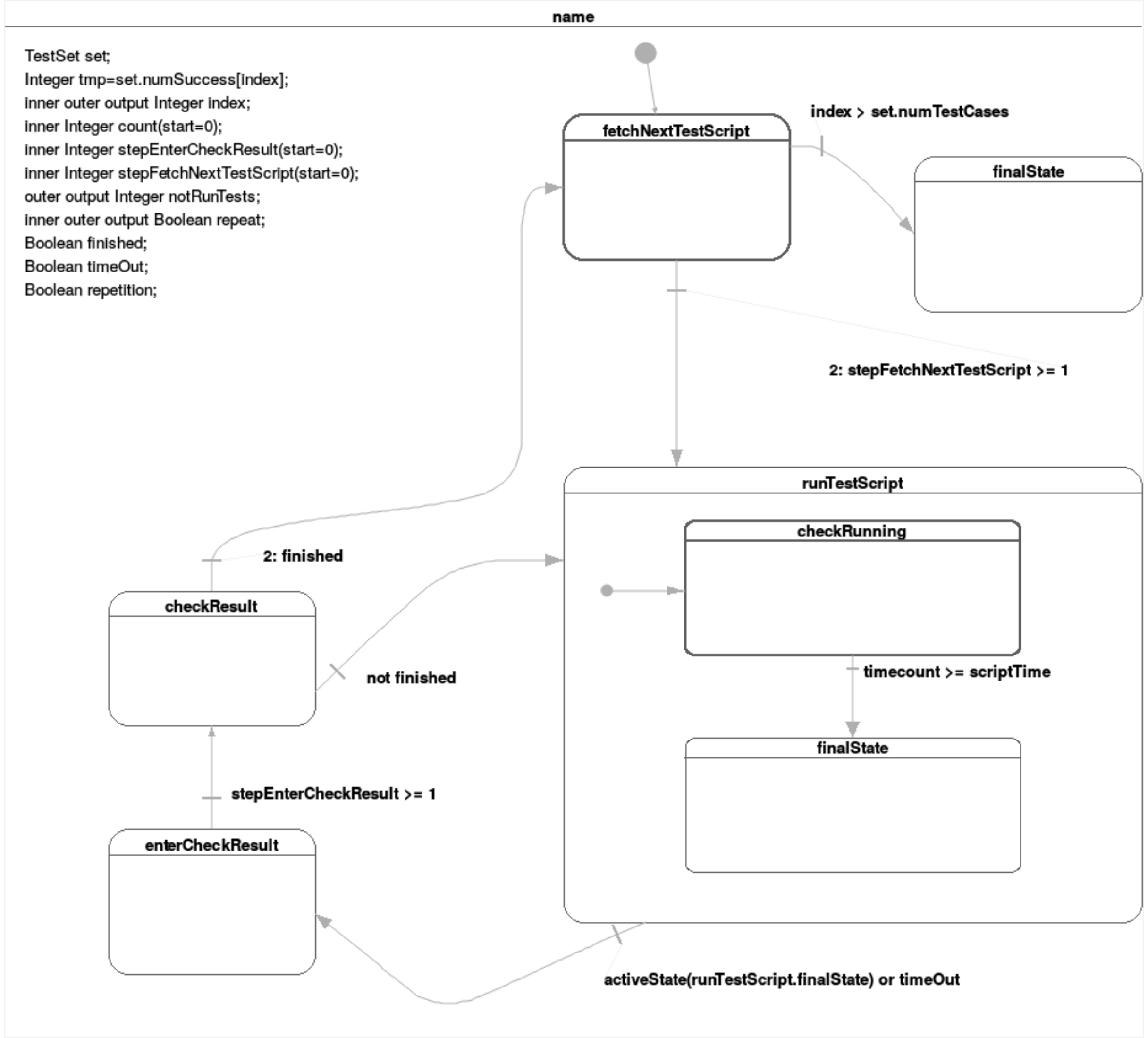
Model Design

- CN designed with Modelica’s state machines
- UWA and TestSystem start simulated with fixed delays
- Test runs simulated with fixed delay of 10 s
- Several instances of CN modelled with connectors

Software Design



Handle Scripts



```
TestSet set;
Integer tmp=set.numSuccess[index];
inner outer output Integer index;
inner Integer count(start=0);
inner Integer stepEnterCheckResult(start=0);
inner Integer stepFetchNextTestScript(start=0);
outer output Integer notRunTests;
inner outer output Boolean repeat;
Boolean finished;
Boolean timeOut;
Boolean repetition;
```

Handle Scripts

- Initial State: FetchNextTestScript
- Tick/TimeInState not available in sub states => use counters
  - stepEnterCheckResult, stepFetchNextTestScript, timecount
- Entry/Exit Action not available in Modelica => extra Enter State e.g. stepEnterCheckResult with one iteration.
- Rough state flow description: FetchNextTestScript=>RunTestScript=>CheckResult=>RunTestScript or FetchNextTestScript

Remark

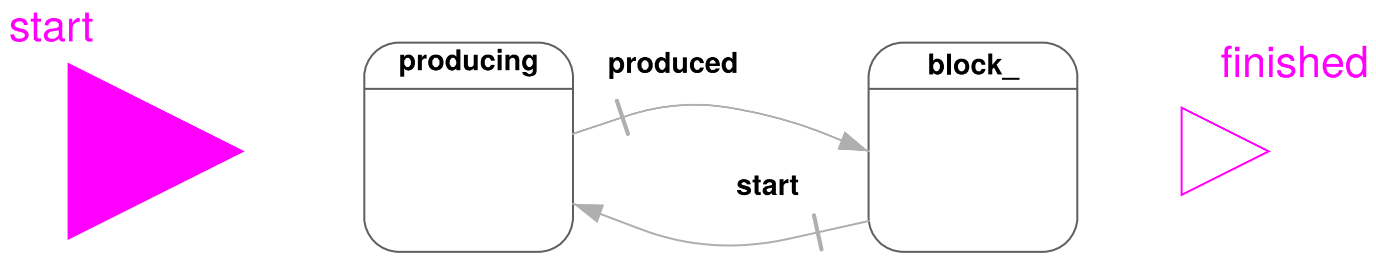
- scriptTime: time of simulated script execution
- finished: in Modelica code `finished = count>= set.numSuccess[index] and set.returnOnSuccess[index] or timeOut or repetition or set.scriptError[index];`

Environment Simulation

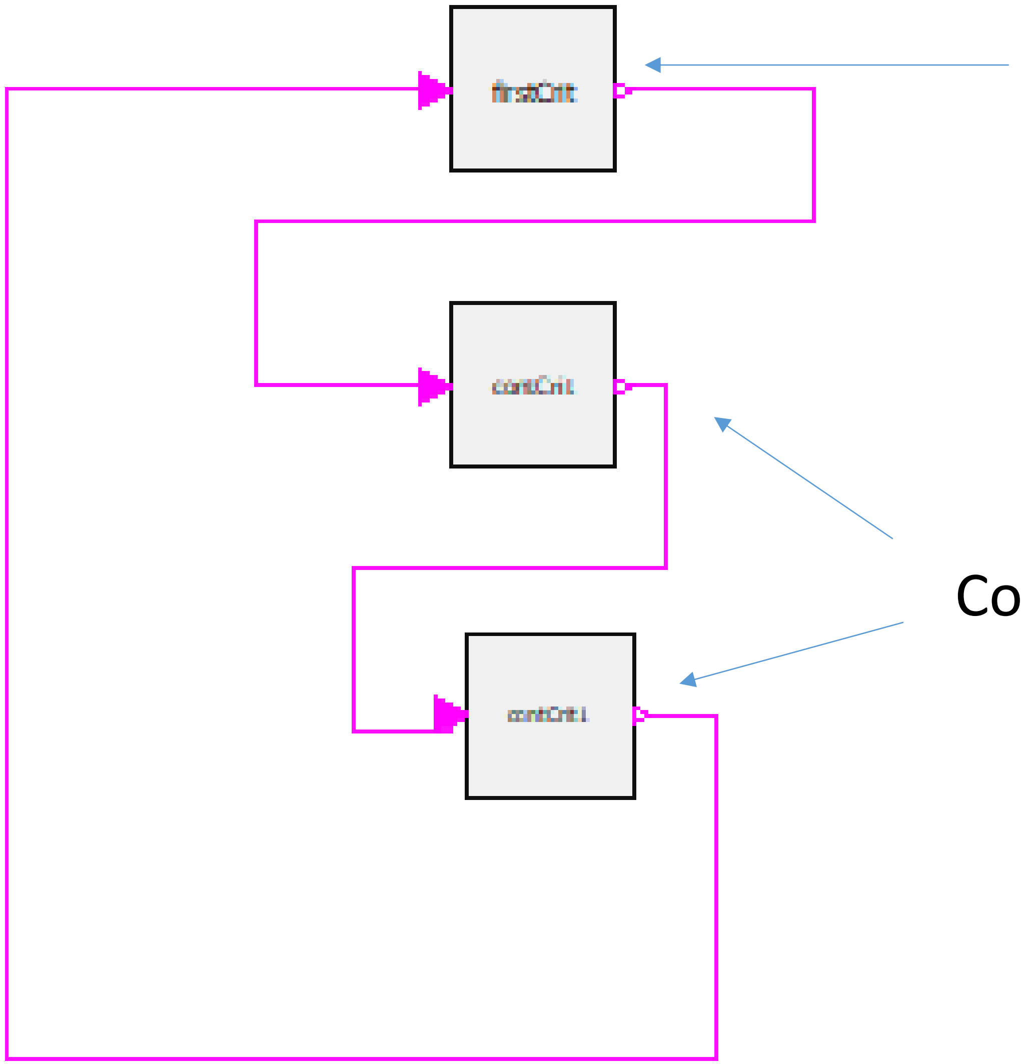
- Challenge: only one instance can run at the same time on one machine => Mechanism has to be implemented
- Solution: Producer-Consumer Problem

Producer Consumer Problem

Partial Class without initial state



Derived class with initial state producing => Producer
Derived class with initial state block_ => Consumer

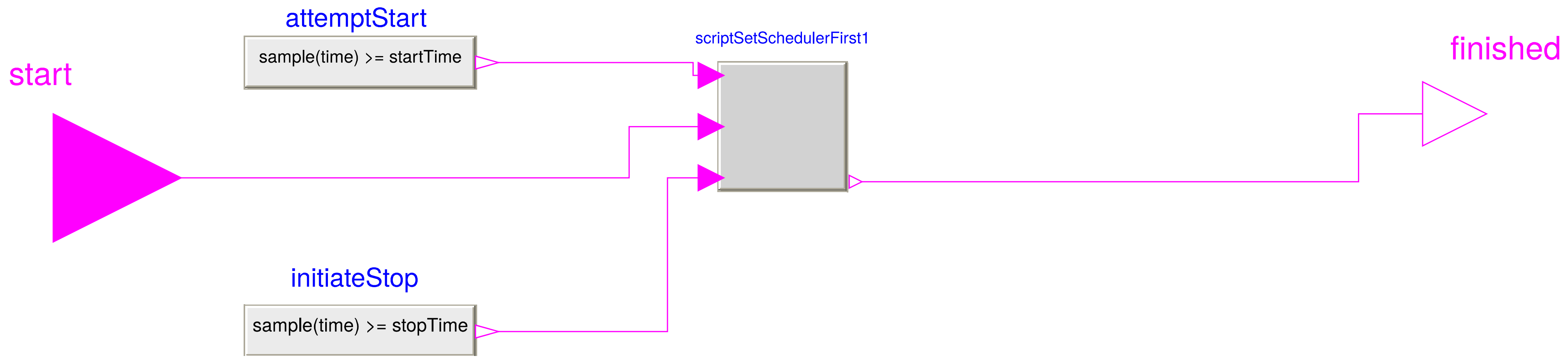


Producer

Consumer

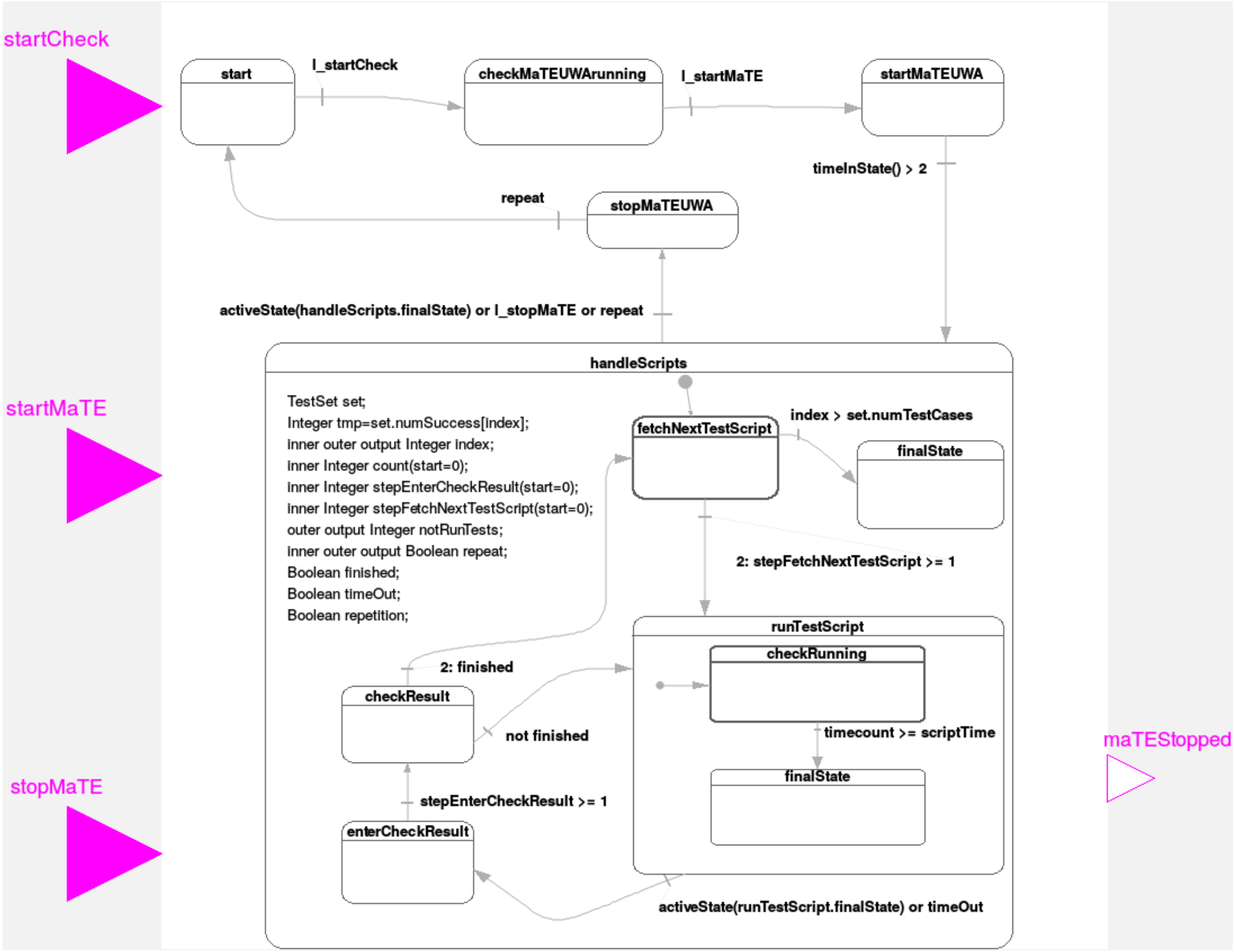
Only one instance can produce
at the same time => pattern realized with semaphores

Modeling Start and Stop Parameter



Realised with Boolean expressions
“attemptStart” and “InitiateStop”

Partial Class ScriptScheduler



```
TestSet set;
Integer tmp=set.numSuccess[index];
inner outer output Integer index;
inner Integer count(start=0);
inner Integer stepEnterCheckResult(start=0);
inner Integer stepFetchNextTestScript(start=0);
outer output Integer notRunTests;
inner outer output Boolean repeat;
Boolean finished;
Boolean timeOut;
Boolean repetition;
```

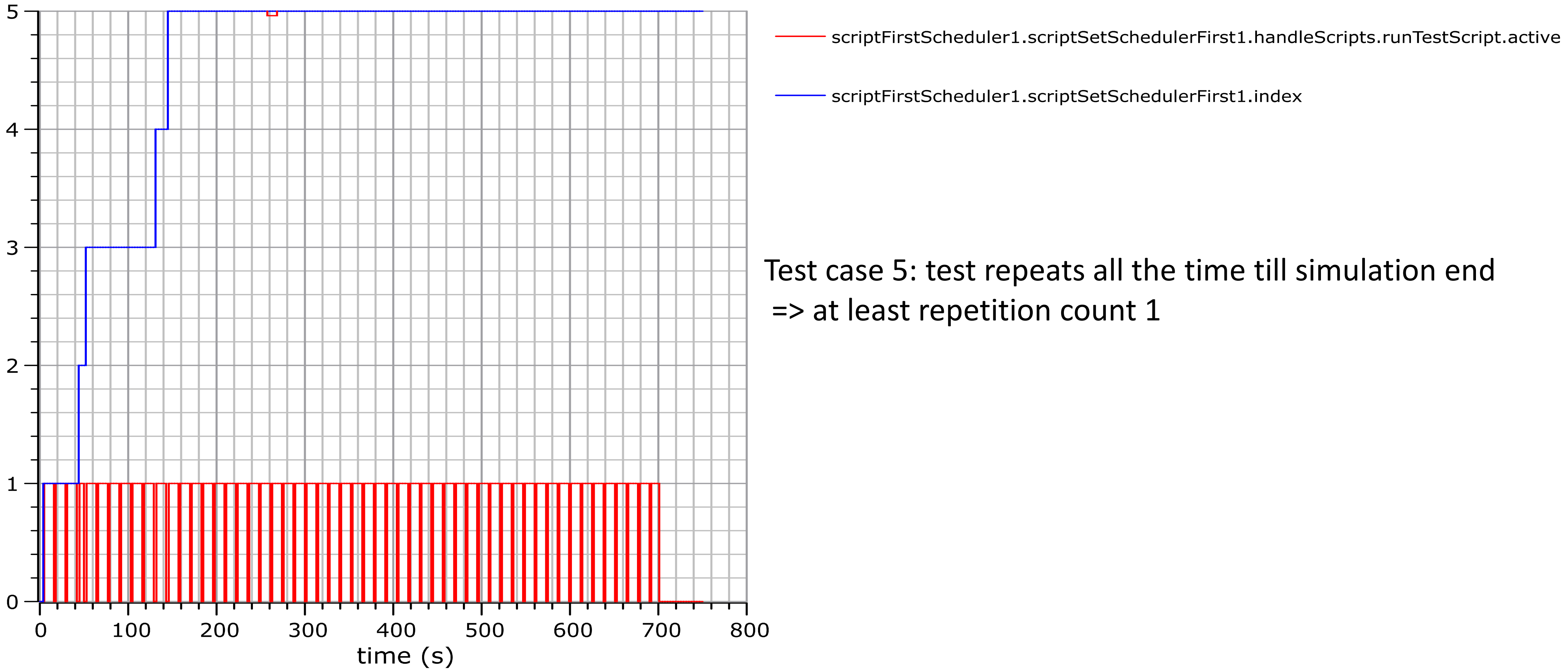
Simulation Constants

- Simulation constants for test purpose in “TestSet.mo” parsed in SM
  - Integer numSuccess[numTestCases]: After which repetition the test returns SUCCESS
  - Integer numMaTEError[numTestCases]: After which repetition a MaTE Error is detected
  - Boolean scriptError[numTestCases]: Which test has a script error

Test Cases

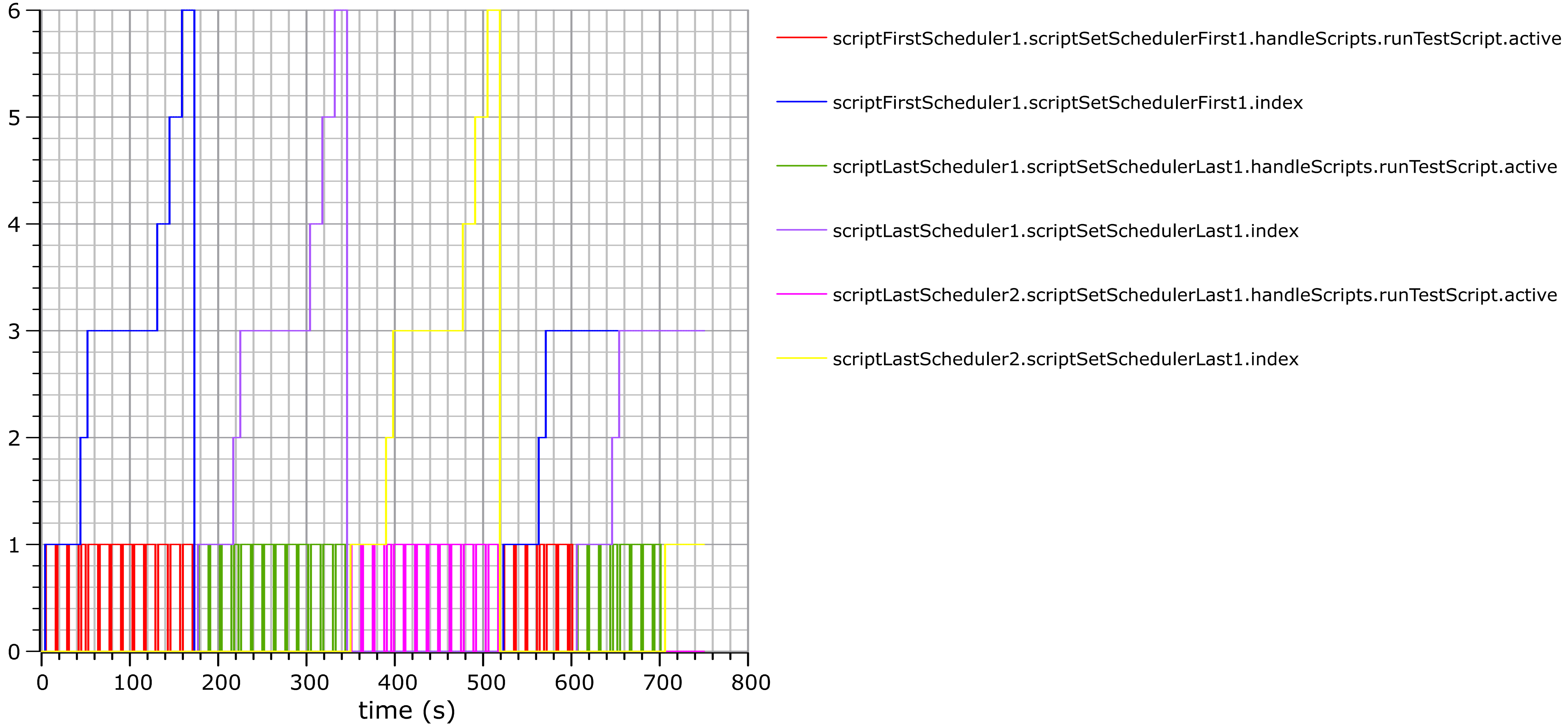
1. success after 3rd repetition of test, returnOnSuccess enabled
2. success after 1st repetition of test and timeout after 5 s
3. success after 1st repetition but run 6 times, returnOnSuccess disabled
4. Script error
5. what happens if none is configured? – infinite loop, ensure repetitionCount min 1!
6. application error after first run

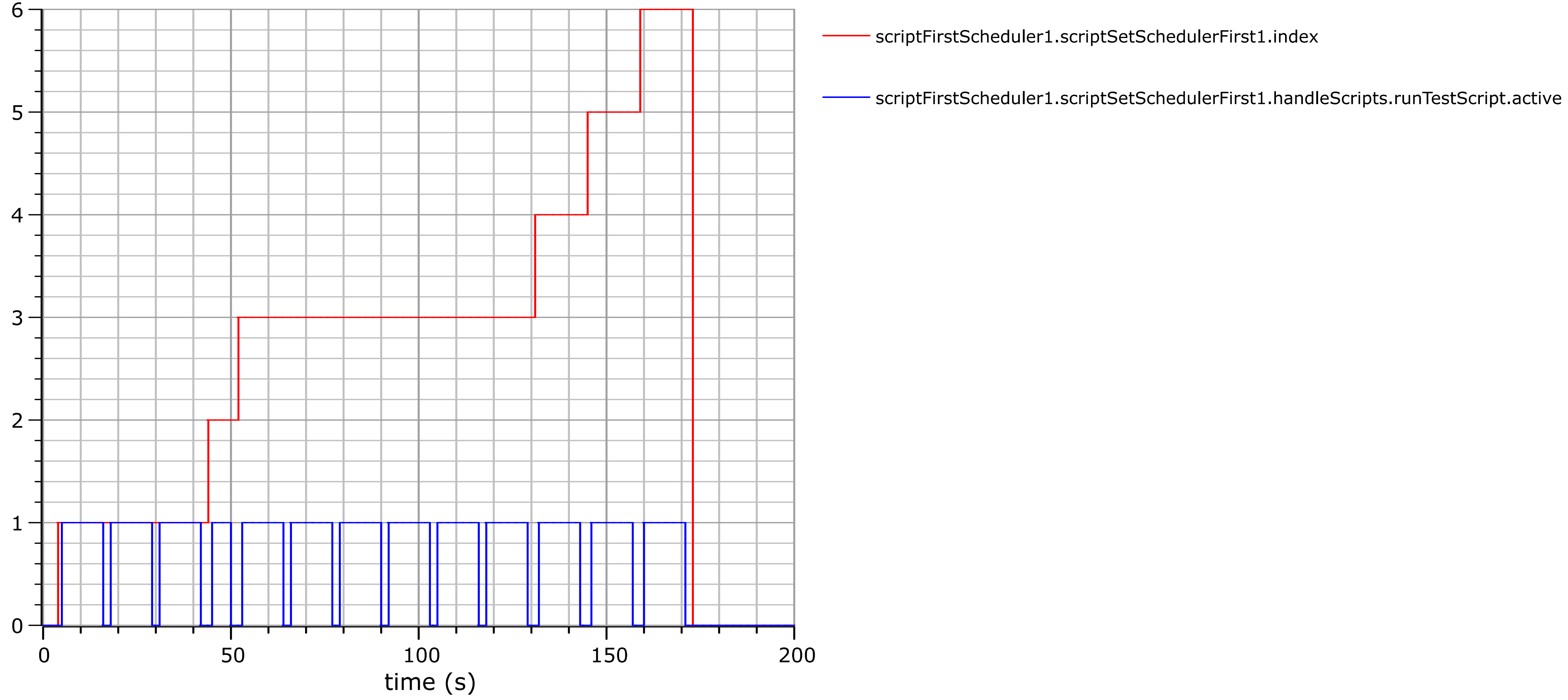
Simulation Results



Test case 5: test repeats all the time till simulation end

=> at least repetition count 1


0
1
2
3
4
5
6
0
100
200
300
400
500
600
700
800
time (s)
scriptFirstScheduler1.scriptSetSchedulerFirst1.handleScripts.runTestScript.active
scriptFirstScheduler1.scriptSetSchedulerFirst1.index
scriptLastScheduler1.scriptSetSchedulerLast1.handleScripts.runTestScript.active
scriptLastScheduler1.scriptSetSchedulerLast1.index
scriptLastScheduler2.scriptSetSchedulerLast1.handleScripts.runTestScript.active
scriptLastScheduler2.scriptSetSchedulerLast1.index


6
5
4
3
2
1
0
0
50
100
150
200
time (s)
scriptFirstScheduler1.scriptSetSchedulerFirst1.index
scriptFirstScheduler1.scriptSetSchedulerFirst1.handleScripts.runTestScript.active

Simulation time	Index (Test case)	Description
2 - 42 s	1	First test is repeated for 3 times
43 - 50 s	2	Time out after 5 s, 2nd test is aborted
51 - 130 s	3	Repeat 3rd test 6 times
130 - 144 s	4	script error, test runs only one time (remark: real test can’t run, simplification in simulation)
145 - 159 s	5	repetitionCount must be 1, otherwise endless repetition
160 - 171 s	6	MaTE error simulated, all test will rerun as soon as resource is available
175 - 520 s	1 till 6	repetition with same test set in scriptLastScheduler1/2
520 - 600 s	1 till 3	stopTime is 600, scheduler stops
600 - 700 s	1 till 3	stopTime is 700, scheduler stops

Test Results Successful

- 3 Instances of CN with start time of 2 s:
  - scriptSchedulerFirst1, scriptSchedulerLast1 and scriptSchedulerLast2. Although they start all at the same time, they are queued. Since the time of script execution is not relevant, it is configured constant as 10 s for each test.
  - At ca. 520 s the scriptSchedulerFirst1 starts again because of a detected system error, but stops at the configured stop time of 600s.
  - scriptSchedulerLast1 starts and stops at its configured stop time of 700 s.

Conclusions

- Problems found at an early stage
  - Mechanism for queueing applications with semaphore needed
  - repetitionCount must be at least 1.
  - => Detecting errors at an early stage saves cost and time in development cycle
- **Vision**: Modelica based *Model Driven Development*
  - The Modelica design model becomes the actual “source-code” of the application
  - E.g., Real-time synchronization and non-Modelica based application code realized using external objects and C-function code like in the Modelica Device Drivers library
  - No manual coding of state machines
  - Simplification of maintenance and development cycle