Model Driven Design of a Test Automation Software using OpenModelica

Lutz Berger
Bernhard Thiele
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

- **Linux Server 1**
  - Cron Job
  - CN UWA ME
  - TS
  - Window PC 1,2 or 3

- **Linux Server 2**
  - Cron Job
  - CN UWA ME
  - TS
  - Window PC 1,2 or 3

- **Linux Server 3**
  - Cron Job
  - CN UWA ME
  - TS
  - Window PC 1,2 or 3

**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

start

I_startCheck

checkMaTEUWA

running

I_startMaTE

startMaTEUWA

timeInState() > 2

repeat

stopMaTEUWA

activeState(handleScripts.finalState) or I_stopMaTE or repeat

handleScripts
Handle Scripts

TestSet set;
Integer tmp = set.numSuccess[index];
inner outer output Integer index;
inner Integer count[index=0];
inner Integer step2EnterCheckResult[start=0];
inner Integer step2FetchNextTestScript[start=0];
outer output Integer notRunTests;
outer outer output Boolean repeat;
Boolean finished;
Boolean timeOut;
Boolean repetition;

name

index > set.numTestCases

finalState

2: step2FetchNextTestScript >= 1

runTestScript

checkRunning

timecount >= scriptTime

finalState

activeState(runTestScript.finalState) or timeOut

enterCheckResult

not finished

step2EnterCheckResult >= 1

2: finished
Handle Scripts

• Initial State: FetchNextTestScript
• Tick/TimeInState not available in sub states => use counters
  • stepEnterCheckResult, stepFetchNextTestScript, timeout
• 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

\[
\text{finished} = \text{count} \geq \text{set}\text{.numSuccess}[\text{index}] \quad \text{and} \quad \\
\text{set}\text{.returnOnSuccess}[\text{index}] \quad \text{or} \quad \text{timeOut} \quad \text{or} \quad \\
\text{repetition} \quad \text{or} \quad \text{set}\text{.scriptError}[\text{index}];
\]
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
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

```java
Partial Class ScriptScheduler
```

```
startCheck

start... checkMaTEUWArunning... l_startMaTE

repeat

stopMaTEUWA

timeInState() > 2

activeState(handleScripts.finalState) or l_stopMaTE or repeat

startMaTE

TestSet set;
Integer tmp=set.numSuccess(index);
Inner outer output Integer index;
Inner Integer count(start=0);
Inner Integer step(EnterCheckResult(start=0);
Inner Integer step(FetchNextTestScript(start=0);
outer output Integer notRunTests;
inner outer output Boolean repeat;
Boolean finished;
Boolean timeOut;
Boolean repetition;

handleScripts

index > set.numTestCases

2: step(FetchNextTestScript >= 1

2: finished

checkResult

runTestScript

checkRunning

timecount == scriptTime

finalState

activeState(runTestScript.finalState) or timeOut

stopMaTE

stepEnterCheckResult >= 1

enterCheckResult
```

maTEStopped

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