On the Performance of UML State Machine Interpretation at Runtime

Edzard Höfig, Peter H. Deussen, Ina Schieferdecker
Motivation

Autonomic Systems Engineering

Self-Awareness*
The ability of a system to perceive its own operational state
- Requires representation of system behaviour

Homeostasis**
The ability of a system to adapt to environmental changes
- Requires support for functional scalability at runtime

➠ Use of behaviour models + runtime interpretation

* See M. Smirnow et al., Demystifying Self-awareness of Autonomic Systems, 2009

** See Chapter 5 in W. R. Ashby, Design for a Brain, Chapman & Hall, 1960
Execution of Behaviour Models

**Interpretation** is different than **compilation**

- This approach makes runtime adaptation easy
- With this approach runtime adaptation is difficult or impossible
Splitting Behaviour Representation

Use

C1

C2

C3

Application logic as model (UML State Machines) — interpreted at runtime

Platform code as static components — dynamic deployment & linkage
Conceptual Interpreter Architecture

Event Interface

Manager
Selects & Deploys
Controls
Creates, Adapts & Removes

Interpreter Engine
- Run-To-Completion Processing
- Expression Evaluation
- Timers & Parallelism

Evaluates & Updates State

Invocation

Event Queue

Sending

Behaviour Model Instance

History
Active State Config
Context

State

Functional Capabilities

BM

Conceptual Interpreter Architecture
Some Challenges...

- Processing compound transitions
- Determination of entered/exited states
- Implicit Fork / Join

A similar issue can be observed when finally joining the control flows: once the BM is executed, it can be seen that the initial fork pseudostate only specifies entry in states B and D; state G is entered implicitly by means of an initial pseudostate.

The shown BM can be executed from start to end by processing the two events e01 and e02. It can be seen that the initial fork pseudostate only specifies entry in states B and D; state G is entered implicitly by means of an initial pseudostate.

A similar issue can be observed when finally joining the control flows: once the BM is executed, it can be seen that the initial fork pseudostate only specifies entry in states B and D; state G is entered implicitly by means of an initial pseudostate.

Which does not use join or fork and solely uses implicit handling of parallel regions.
Performance Benchmark

Technology independent with mappings

The benchmark consists of 10 scenarios

 Defines as 10 measurement procedures

Used to compare behaviour model execution platforms

Consists of 13 reference models

Generated C++

UML Interpreter

100 performance indicators

Generated C++

UML Interpreter

Technology independent with mappings

UML + MVEL

SCXML + JEXL

Rational Rhapsody

13 reference models

Comparison

Generated C++

UML Interpreter

100 performance indicators

Technologies:

- SIZE
- ALTERNATIVE
- EPSILON
- EVENT
- GUARD
- COMPOUND
- EXPRESSION
- CONCURRENT
- CONFIG
- LIFECYCLE
Performance Benchmark

The benchmark

1 scenario

1 reference model

1 measurement procedure

3 performance indicators

Compare behaviour model execution platforms

Generated C++

UML Interpreter

Technology independent with mappings

UML + MVEL

SCXML + JEXL

Rational Rhapsody

EVENT

GENERATION

SIZE

ALTERNATIVE

EPSILON

GUARD

COMPOUND

EXPRESSION

CONCURRENT

CONFIG

LIFECYCLE
Behaviour Model

**EVENT Scenario**

```
<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>e01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

`/ timestamp()`
Behaviour Model

**EVENT Scenario**

\[5^3 = 125 \text{ different paths}\]
Measurement Procedure

**EVENT Scenario**

- Execute all 125 paths (a ,,trial“)
  - Uses all event combinations
- Repeat trials 125 times, starting with a different path each time
  - Averages measurements
  - Distributes build-up effects
- Process 15625 timestamp pairs to calculate three performance indicators

<table>
<thead>
<tr>
<th></th>
<th>Compiled</th>
<th>Interpreted</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT.MIN</td>
<td>1 μs</td>
<td>165 μs</td>
</tr>
<tr>
<td>EVENT.AVG</td>
<td>23.38 μs</td>
<td>186.45 μs</td>
</tr>
<tr>
<td>EVENT.MAX</td>
<td>128 μs</td>
<td>492 μs</td>
</tr>
</tbody>
</table>
Results

• Proposed approach is feasible

• Adequate performance
  • Exception: delay-sensitive systems with timing constraints << 1 ms

• Impact of model structure & model storage format
Benefits for **System Adaptation** and **Self-Management**

- **Consistency** has to be ensured for only a small number of artefacts
  - Artefacts *readily available* during runtime, with *well-defined formats*
- **State-transition systems are** reactive
  - *Quiescent* between processing steps
- **Application state is explicitly** managed
- **Models are an** abstraction of system behaviour
  - *Intuitive* & *formalised*
- **Model checking, simulation, testing, even proofs** are possible
Thanks for Listening!

Questions?

Edzard Höfig

email: edzard.hoefig@fokus.fraunhofer.de
web: www.fokus.fraunhofer.de/go/motion
UML Interpreter + MVEL

- TIMESTAMP
- SIZE
- ALTERNATIVE
- EPSILON
- EVENT
- GUARD
- COMPOUND
- EXPRESSION
- CONCURRENT
- CONFIG
- LIFECYCLE
Executable size is always size of JVM
ALTERNATIVE
EPSILON

![Graph showing data points at various time intervals (us) ranging from 0 to 60. The x-axis represents time in units of 10, ranging from 0 to 100. The y-axis represents time in units of 8us to 60us. Data points are scattered across the graph, indicating varying responses at different time intervals.]
COMPOUND

Only MIN, AVG

With MAX values
CONFIG