EVENTSIM – An Event-driven Palladio Software Architecture Simulator

Palladio Days ‘11 | 17th of November

Philipp Merkle, Jörg Henss
Motivation

- Performance Analyses of Palladio Models
  - analytical approaches $\rightarrow$ state space explosion
  - therefore, **simulative approaches for complex systems**

- Palladio’s simulator SimuCom is mainly process-oriented

- But process-oriented simulations known for inferior performance and scalability (e.g. [Banks 2010])

- **EventSim**: tackle performance and scalability issues using event-driven simulation
Agenda

- Foundations
- Related Work
- EventSim
- Validation
- Evaluation
- Conclusion
- Short Demo
Process- vs. Event-orientation in a Nutshell

**Event-Oriented**
- Behaviour modelled by **sequences of events**
  - Represent time instant
  - Do not overlap each other
  - Execute one after another using a **single thread**

**Process-Oriented**
- Behaviour modelled by **processes**
  - Represent periods of time
  - May overlap → concurrent control flow
  - **Java**: execute using multiple threads

User 1:  
User 2:  

Simulated Time

Motivation | Foundations | Related Work | EventSim | Validation | Evaluation | Conclusion
Related Work – PCM Simulators

- **SimuCom** [Becker 2008]
  - Palladio’s default simulator
  - process-oriented behaviour simulation
  - event-oriented resource simulation

- **SLAStic.SIM** [von Massow 2010]
  - focus on runtime reconfiguration
  - fully event-oriented, but no stochastic expressions

- **SimQPN** [Kounev and Buchmann 2006]
  - simulates queueing Petri nets (QPN)
  - Meier [Meier et al. 2011] developed a PCM to QPN transformation
  - fully event-oriented, but lower prediction accuracy
EventSim Overview

User \(\xrightarrow{\text{invokes}}\) Request \(\xrightarrow{\text{demands}}\) Resource

- EntryLevelSystemCall

**Workload Generator**

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion

Philipp Merkle, Jörg Henss: EventSim – An Event-driven Palladio Software Architecture Simulator
EventSim Overview

User invokes Request

Request demands Resource

generates Workload

<<EntryLevelSystemCall>>
callOperationA

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion
EventSim Overview

User invokes Request, which demands Resource. The Request is an EntryLevelSystemCall event. A StartTraversalStrategy event is generated, which invokes callOperationA. The diagram includes the following sections: Motivation, Foundations, Related Work, EventSim, Validation, Evaluation, and Conclusion.
EventSim Overview

User invokes Request demands Resource

EntryLevelSystemCall

EntryLevelSystemCall-TraversalStrategy

<<EntryLevelSystemCall>> callOperationA

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion
EventSim Overview

User invokes Request demands Resource

EntryLevelSystemCall

generates

Workload Generator

<<EntryLevelSystemCall>>
callOperationA

<<ExternalCallAction>>
callOperationB

<<InternalAction>>
internalAction

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion
EventSim Overview

User invokes Request demands Resource

generates Workload Generator

EntryLevelSystemCall
callOperationA
<<EntryLevelSystemCall>>
callOperationB
<<ExternalCallAction>>
internalAction
<<InternalAction>>

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion
EventSim Overview

- Limited in capacity
- **Contention** due to concurrent Requests
- Requests might have to wait
Usage of Events

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion
Behaviour Interpreter

Motivation
Fo Foundations
Related Work
EventSim
Validation
Evaluation
Conclusion

14 2011-08-26 Philipp Merkle, Jörg Henss: EventSim – An Event-driven Palladio Software Architecture Simulator

Software Design and Quality Group
Institute for Program Structures and Data Organization
Extensibility

- Traversal strategies allow for flexible simulator extension

Examples

- additional control flow element → new strategy
- adjust semantics of control flow element → change existing strategy
- additional performance metric (e.g. reliability) → decorate existing strategies
Validation Approach

Results „consistent“ if

- no difference at all (sum of differences = 0)
- no significant difference (fail to reject $H_0$ of Kolmogorov-Smirnov test)
### Validation Results

- 3 different experiments using „MediaStore“ Example [Koziolek et al. 2007]

<table>
<thead>
<tr>
<th>Concurrent Users</th>
<th>Scheduling Policies</th>
<th>Difference?</th>
<th>Difference Significant? ($\alpha = 0.05$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FCFS, PS</td>
<td>no</td>
<td>n/a</td>
</tr>
<tr>
<td>10</td>
<td>FCFS, PS</td>
<td>yes</td>
<td>(no)</td>
</tr>
<tr>
<td>10</td>
<td>FCFS</td>
<td>no</td>
<td>n/a</td>
</tr>
</tbody>
</table>

- Difference due to indeterministic scheduler implementation
- **EventSim is correct** (w.r.t the experiments)
Validation Results (cont’d)

- Significant difference reported in a single case (k-s test rejected $H_0$ "simulation results follow the same cdf")
Preliminary Work for Evaluation

Potential performance factors? (affecting simulator)

Comparison based on major performance factors

PCM Modelling Degrees of Freedom

Rank factors and select top ranked factors

EventSim

SimuCom

Motivation  Foundations  Related Work  EventSim  Validation  Evaluation  Conclusion

Philipp Merkle, Jörg Henss: EventSim – An Event-driven Palladio Software Architecture Simulator
Evaluation Approach

Repeat each simulation 30 times
Stop simulation at 1000 measurements

Base Model
Comprises one modelling element for each performance factor

Simulate
Simulation duration? Memory? Threads?

Add further elements of factor i

{1, 100, 200, ..., 1000} elements
Evaluation Results – ForkedBehaviours

- Simulation Duration
  (1000 repetitions)

  ~ 240\(i\) ms

- Thread Count

  ~ 5.28\(i\) ms

- EventSim around 50 times faster

Motivation  ➔  Foundations  ➔  Related Work  ➔  EventSim  ➔  Validation  ➔  Evaluation  ➔  Conclusion

Philipp Merkle, Jörg Henss: EventSim – An Event-driven Palladio Software Architecture Simulator
Evaluation Results Overview

- **Runtime** of both simulators scales linearly but at a different rate:
  - ForkedBehaviour – EventSim 50 times faster
  - Delay – EventSim 22 times faster
  - InternalAction – EventSim 4 times faster
  - Branch – SimuCom twice as fast

- **Memory consumption** independent of model complexity
Scalability Boundaries

- 64-bit JVM, 4 GB Memory, 1 GB Heap

<table>
<thead>
<tr>
<th>Factor</th>
<th>SimuCom</th>
<th>EventSim</th>
</tr>
</thead>
<tbody>
<tr>
<td># ForkedBehaviours</td>
<td>&lt; 820 a)</td>
<td>&gt; 100.000</td>
</tr>
<tr>
<td># InternalActions</td>
<td>&lt; 940 a)</td>
<td>&gt; 100.000</td>
</tr>
<tr>
<td># Delays</td>
<td>&lt; 1.560 a)</td>
<td>&gt; 100.000</td>
</tr>
<tr>
<td># Branches</td>
<td>&lt; 1.250 b)</td>
<td>&gt; 100.000</td>
</tr>
<tr>
<td>Workload Population</td>
<td>&lt; 90.000 c)</td>
<td>&gt; 100.000</td>
</tr>
</tbody>
</table>

Indications of limited scalability

- a) – StackOverflowError
- b) – exceeded 64 KB method size limit
- c) – OutOfMemoryError
## Conclusion and Future Work

- Each simulator has its strengths and weaknesses

### Scalability:
- Performance (simulation duration)
  - **Simulated Concurrency:**
    - (Forks, intensive workloads)
  - „**Time-advancing“ Actions:**
    - (Delays, InternalActions)
  - „**Control-flow-modifying“ Actions:**
    - (Loops, Branches)
  - Random Number Generator(*):
  - Evaluate Stochastic Expressions(*):

### Feature-Completeness:
- SimuCom
- EventSim

(*) due to identical implementation

## Approach the feature-completeness of SimuCom
EventSim Demonstration

- Input Model: MediaStore Example (for PCM 3.3)
  https://svnserver.informatik.kit.edu/i43/svn/code/Palladio/Core/trunk/Examples/PCM3.3_MediaStore
- PS scheduling policies replaced with FCFS
- Increased workload population: 100 (closed workload)
- No further modifications

Simulation Settings (default)
- Stop at 15,000 simulated time units...
  ...or at 10,000 usage scenario measurements (whatever comes first)
- Fixed Seed (0, 1, ..., 5)

Modifications of SimuCom
- Removed simulation of network latency (in calls.xpt)
  ...since not yet implemented in EventSim
- No further modifications
Bibliography


## EventSim Features (1)

### Usage Modelling

<table>
<thead>
<tr>
<th>Workload Types</th>
<th>Dynamic Behaviour</th>
<th>SEFF Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ OpenWorkload</td>
<td>Actions</td>
<td>✓ StartAction</td>
</tr>
<tr>
<td>✓ ClosedWorkload</td>
<td>✓ StopAction</td>
<td>✓ InternalAction</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td></td>
<td>✓ ExternalCallAction</td>
</tr>
<tr>
<td>✓ Start</td>
<td>✓ SetVariableAction</td>
<td></td>
</tr>
<tr>
<td>✓ Stop</td>
<td>✓ AcquireAction</td>
<td>✓ ReleaseAction</td>
</tr>
<tr>
<td>✓ Delay</td>
<td>✓ LoopAction</td>
<td>✓ CollectionIteratorAction</td>
</tr>
<tr>
<td>✓ EntryLevelSystemCall</td>
<td>✓ BranchAction</td>
<td>✓ ForkAction</td>
</tr>
<tr>
<td>✓ Loop</td>
<td>✓ ForkAction</td>
<td></td>
</tr>
<tr>
<td>✓ Branch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. both, probabilistic and guarded branch transitions are available
2. no support of synchronous forks, only asynchronous forks are available
**EventSim Features (2)**

<table>
<thead>
<tr>
<th>Static Structure</th>
<th>Resource Environment</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Composition</strong></td>
<td><strong>Scheduling Policies</strong></td>
<td><strong>Miscellaneous</strong></td>
</tr>
<tr>
<td>✓ BasicComponent</td>
<td>✓ DELAY</td>
<td>✓ Override Component Parameters</td>
</tr>
<tr>
<td>X CompositeComponent</td>
<td>✓ FCFS</td>
<td></td>
</tr>
<tr>
<td>X SubSystem</td>
<td>✓ PS</td>
<td></td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td>X¹ EXACT</td>
<td></td>
</tr>
<tr>
<td>✓ Passive Resources</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ Component Parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X² LinkingResource</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X² Connection</td>
<td></td>
</tr>
</tbody>
</table>

1. the EXACT scheduling policy emulates real schedulers of some operating systems
2. an ideal network connection is assumed to exist between resource containers with an infinite throughput and zero latency.
# Soft- and Hardware Configuration

## Hardware
- **Processor**: Intel® Core™2 Quad Q8300 @ 2.50 GHz (4 cores)
- **Main Memory**: 4.00 GB @ 800 MHz, single channel mode
- **Solid-State Drive**: OCZ Vertex 2

## Operating System
- **Version**: Windows 7 (Version 6.1)

## Java Virtual Machine (JVM)
- **Name**: Java HotSpot™64-Bit Server VM
- **Vendor**: Sun Microsystems Inc.
- **Version**: 20.1-b02
- **VM Arguments**: `-Xms512m, -Xmx1024m, -XX:PermSize=256M, -XX:MaxPermSize=512M`

## Simulators
- **Eclipse**: Galileo (Version 3.5)
- **Simulation Library**: SSJ (Version 2.1.3)
Factor Ranking Results

- SimuCom (y-axis in log scale)
- EventSim

![Box plots showing simulation runtime for different factors in SimuCom and EventSim.](image-url)
PCM Base Model

<<UsageScenario>>
<<ScenarioBehaviour>>

<<TimeSpecification>>
Specification = 1

<<LoopIteration>>
Specification = 1

<<Loop>>

<<Branch>>
Branch Probability = 0.5
Branch Probability = 0.5

<<EntryLevelSystemCall>>
callDelay

calls

<<EntryLevelSystemCall>>
callDoNothing

calls

<<ResourceDemandingSEFF>>
delay

<<InternalAction>>

<<Parametric Resource Demand>>
Required Resource = DELAY
Specification = 1

<<ForkAction>>

<<ForkedBehaviour>>

<<ResourceDemandingSEFF>>
doNothing
Executing Simulation Processes in Java

<table>
<thead>
<tr>
<th>Threads</th>
<th>Coroutines [Stadler et al. 2010]</th>
<th>Interpretation [Jacobs and Verbraeck 2004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Switching</td>
<td>OS-controlled</td>
<td>manually, suspend() statement</td>
</tr>
<tr>
<td>Synchronisation</td>
<td>yes, preserves execution order</td>
<td>not required</td>
</tr>
<tr>
<td>Exclusion Criteria</td>
<td>JRE extension</td>
<td>slow</td>
</tr>
</tbody>
</table>

- Threads often unavoidable
  – important to know induced overhead!