The Slingshot Simulator – An Extensible Event-Driven PCM Simulator

Julijan Katić
Floriment Klinaku
Steffen Becker

12th Symposium on Software Performance 2021
Introduction
**Motivation**

A new PCM meta-model extension has arrived...

...Let’s simulate it!
Motivation

- Adding new features to the simulator requires change in the core and possibly other modules (Ripple Effect)

Figure: Adding a new PCM feature, i.e. measuring hybrid quantum software systems.
Motivation

- Either an interface or API description must be predefined, such as *explicit extension points* or Java interfaces.

**Figure:** SimuLizar provides many extension points, but adding new extension points might again cause ripple effect.
Goals

1. Improve Extensibility
Reduce complexity for adding new modules into the existing software system.

2. Validate simulation results
The simulation results from Slingshot should be realistic to the investigated software system, as well as comparable to other simulators’ results.

3. Keep performance
Performance shouldn’t be (much) worse compared to the existing simulators.
Goals

**ACHIEVED - 1. Improve Extensibility**
Reduce complexity for adding new modules into the existing software system.

**Under Investigation - 2. Validate simulation results**
The simulation results from Slingshot should be realistic to the investigated software system, as well as comparable to other simulators’ results.

**Next - 3. Keep performance**
Performance shouldn’t be (much) worse compared to the existing simulators.
Slingshot Simulator
# Differences

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation Paradigm</td>
<td>Process-Interaction</td>
<td>Event-Oriented</td>
<td>Event-Oriented</td>
</tr>
<tr>
<td>Architecture</td>
<td>Monolithic</td>
<td>Monolithic</td>
<td>Event-Driven</td>
</tr>
<tr>
<td>Extension Type</td>
<td>Various Extension Points</td>
<td>Guice-Module/Traversal-Strategy</td>
<td>Event-Handler</td>
</tr>
</tbody>
</table>

**Table:** Differences between the (well-established) simulators
Components do not know where or by whom the published events are handled.

⇒ Enables extremely loosely coupled modularity
**Differences**

**EventSim**

```
EventSimExtension
+ id: String
+ name: String

SimulationModule
+ id: String
+ name: String
+ priority: Int

simulation_modules 0..*
```

**GuiceModule** 0..1 0..1 0..1

**LaunchContrib.**

**EntryPoint**

**SimulationStrategy** 0..*

```
«interface»
ITraversalStrategy
Entity
```

**SlingshotExtension**

```
«interface»
SimulationBehaviorExtension
```

**Figure: Extension Interface Definition in EventSim**

**Figure: Extension Interface Definition in Slingshot**
Figure: Core Architecture of Slingshot
Figure: Typical overview with all modules.
Event-Driven Architecture

Example

Figure: Example of the publisher-subscriber pattern in this extension
Event-Driven Architecture

Example

```java
@Subscribe
public ResultEvent<QuantumSimulationStarted> onSimulationStarted(final RDSeffElementVisited<QuantumAction> action) {
    // some handling
    return ResultEvent.of(new QuantumSimulationStarted());
}
```

Register the class and connect it to the event traffic.

```xml
<extension point="org.palladiosimulator.analyzer.slingshot.behavior">
    <behavior eventHandlers="...QuantumInterpreter.java" />
    <behavior eventHandlers="...ExtendedUsageInterpreter.java" />
</extension>
```
Kinds of Extensions

1. Interpretation of certain Model elements

2. Contribution of Launch Configurations

3. Introducing new model elements

4. New Monitors and Measuring Points
Metric and Modularity
What is Extensibility?
Extensibility is a subjective experience[2]

If no documentation is available, developers will still struggle extending the system.

Developers must understand the system (i.e. Eclipse system) and the build process.

Every extensible software is incomplete: New extension points either must be introduced or the current architecture refined to allow the new type of extension.
Modularity
Affects extensibility a lot!

![Diagram](image_url)

**Figure**: Modularity and its relationship to Extensibility

- **EXTCBO**: Coupling between *Components/Modules*.
- **CC**: Component Cohesion, Coupling between classes within the same module.
- **LOCS**: Complexity and changeability of components.
Results

Figure: (Efferent) EXTCBO results for the core

Figure: CC results for the core
Conclusion
A new simulator was introduced that bases on the **Event-Driven Architecture**. Aims to be **extensible**.

The complexity of extending the system was decreased by having only **one** extension point, which can be used for almost every extension type.

The results of component metrics suggest that modularity has been increased, hence extensibility can be increased as well.
Outlook

1. Further analysis
Conduct a case study to find “extensibility holes” and improve extensibility.

2. Self-Adaptation
Add self-adaptation behavior of the system.

3. Validation and Performance
Investigate both the performance as well as the validity of Slingshot simulation.
Julijan Katić
Software Quality and Architecture, Universität Stuttgart

eMail  st154933@stud.uni-stuttgart.de
