Universität Stuttgart

Julijan Katić
Floriment Klinaku
Steffen Becker

The Slingshot Simulator –

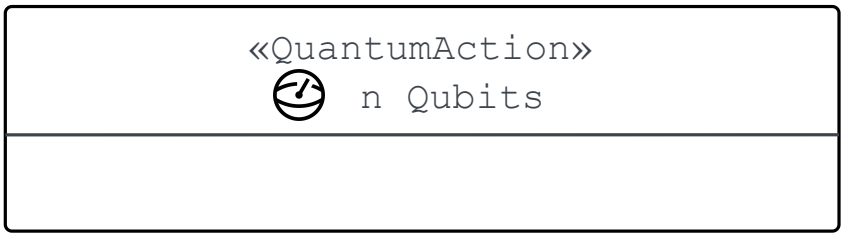
An Extensible Event-Driven PCM Simulator

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**)

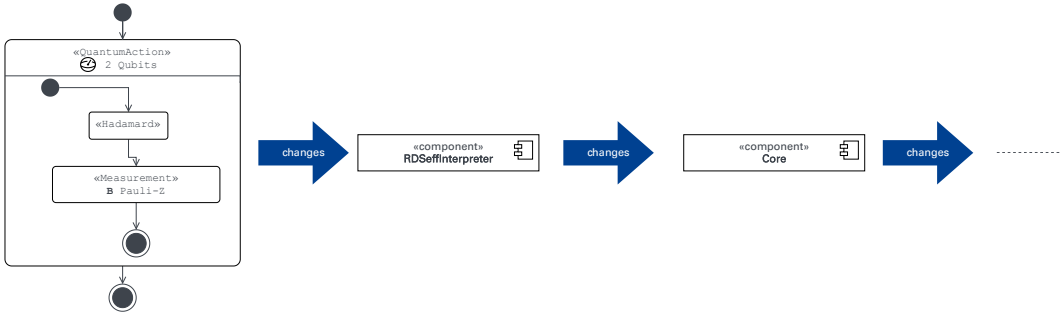


Figure: Adding a new PCM feature, i.e. measuring hybrid quantum software systems.

Motivation

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

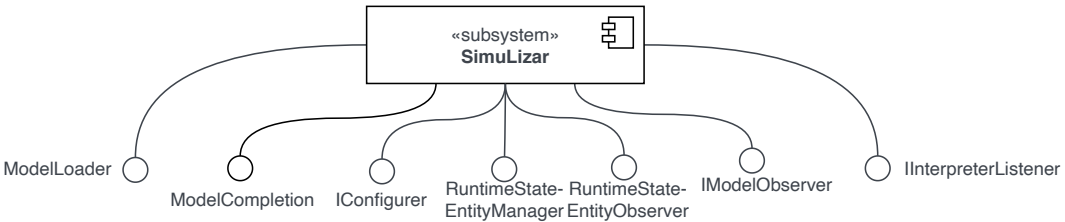


Figure: SimuLizar provides many extension points, but adding new extension points might again cause ripple effect.

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

Aspect	SimuLizar[1]	EventSim[3]	Slingshot
Simulation Paradigm	Process-Interaction	Event-Oriented	Event-Oriented
Architecture	Monolithic	Monolithic	**Event-Driven**
Extension Type	Various Extension Points	Guice-Module/ Traversal-Strategy	**Event-Handler**

Table: Differences between the (well-established) simulators

Event-Driven Architecture

Publisher-Subscriber-Pattern

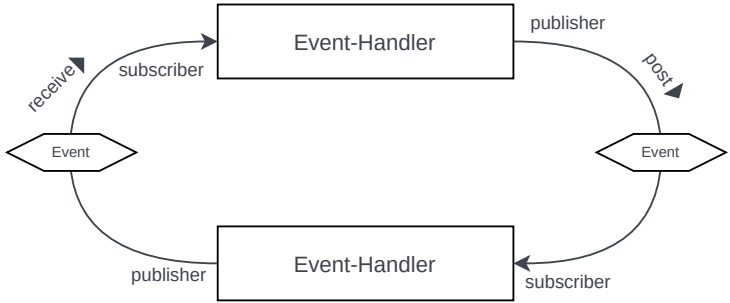


Figure: Bidirectional Publisher-Subscriber-Pattern in Slingshot

- Components do not know where or by whom the published events are handled.

$\Longrightarrow$ Enables **extremely loosely coupled** modularity

Differences

EventSim

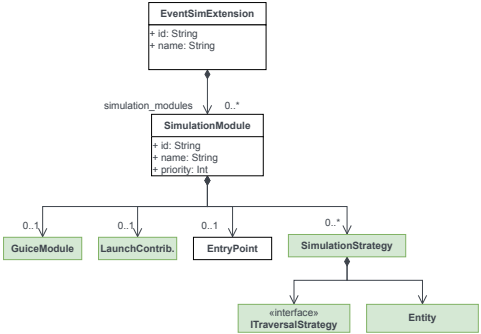


Figure: Extension Interface Definition in EventSim

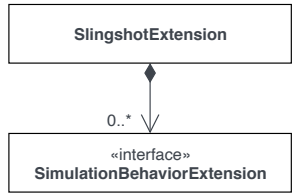


Figure: Extension Interface Definition in Slingshot

Slingshot

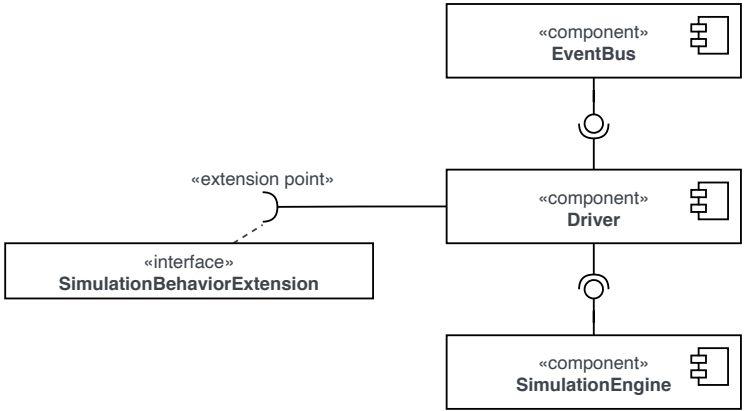


Figure: Core Architecture of Slingshot

Slingshot

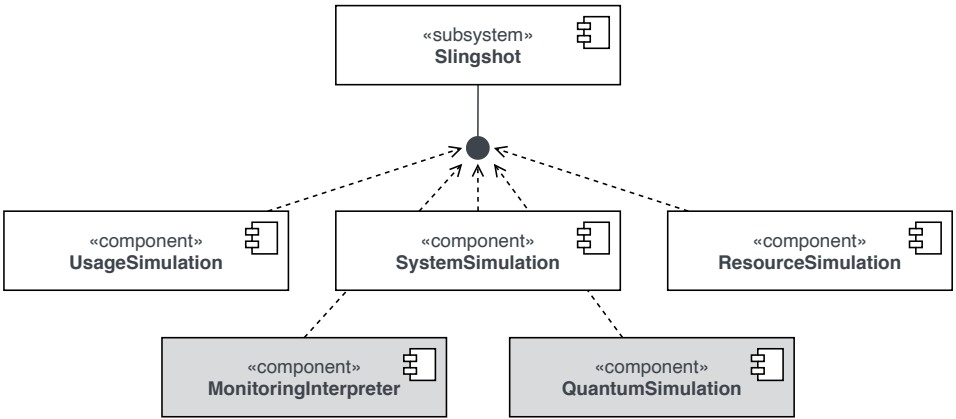


Figure: Typical overview with all modules.

Event-Driven Architecture

Example

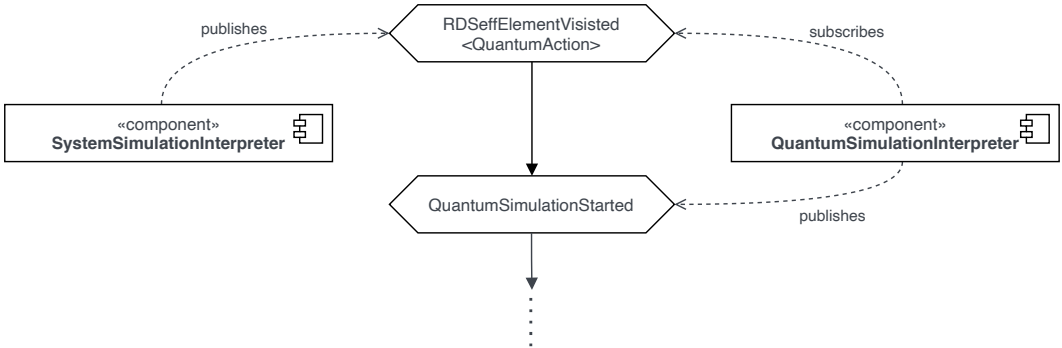


Figure: Example of the publisher-subscriber pattern in this extension

Event-Driven Architecture

Example

Add the event handler inside a class:

```
@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.

```
<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

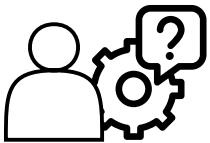
Metric and Modularity

3

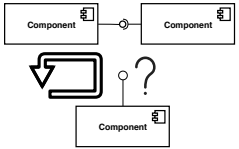
What is Extensibility?

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 extensionpoints either must be introduced or the current architecture refined to allow the new type of extension.

Modularity

Affects extensibility a lot!

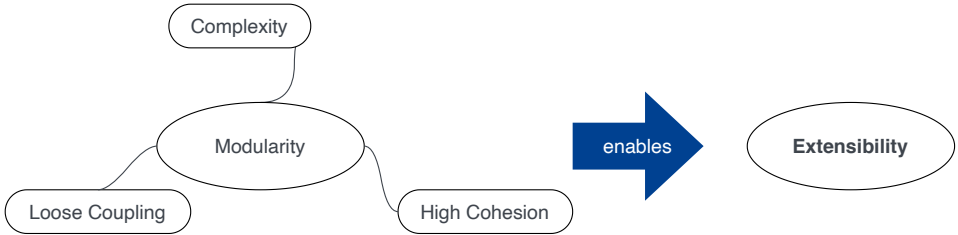


Figure: Modularity and its relationship to Extensibility

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

Results

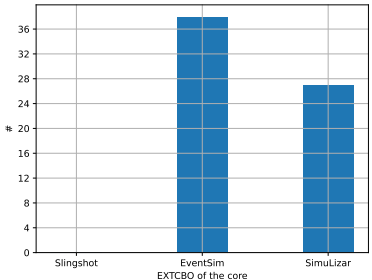


Figure: (Efferent) EXTCBO results for the core

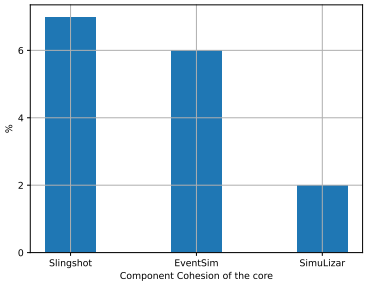
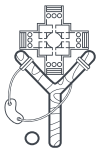


Figure: CC results for the core

Conclusion

4

Conclusion



Slingshot

A new simulator was introduced that bases on the **Event-Driven Architecture**.
Aims to be **extensible**.

Complexity

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

Measurement

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.

Universität Stuttgart

Julijan Katić
Software Quality and Architecture, Universität Stuttgart

eMail st154933@stud.uni-stuttgart.de

References

[1] Matthias Becker, Steffen Becker, and Joachim Meyer. "SimuLizar: Design-Time Modeling and Performance Analysis of Self-Adaptive Systems". In: *Software Engineering 2013*. Ed. by Stefan Kowalewski and Bernhard Rumpe. Bonn: Gesellschaft für Informatik e.V., 2013, pp. 71–84.

[2] Allan Kelly. "The philosophy of extensible software". In: *The Internet's Coming Silent Spring* (2002). URL: https://www.accu.org/journals/overload/10/50/overload50.pdf#page=25.

[3] Philipp Merkle and Jörg Henß. "EVENTSIM - An Event-driven Palladio Software Architecture Simulator". In: *Palladio Days 2011. Proceedings, 17-18 November 2011, FZI Forschungszentrum Informatik, Karlsruhe, Germany. Ed.: S. Becker*. Vol. 2011. Karlsruhe Reports in Informatics (früher: Interner Bericht. Fakultät für Informatik, Karlsruher Institut für Technologie) 32. Karlsruher Institut für Technologie (KIT), 2011, pp. 15–22.