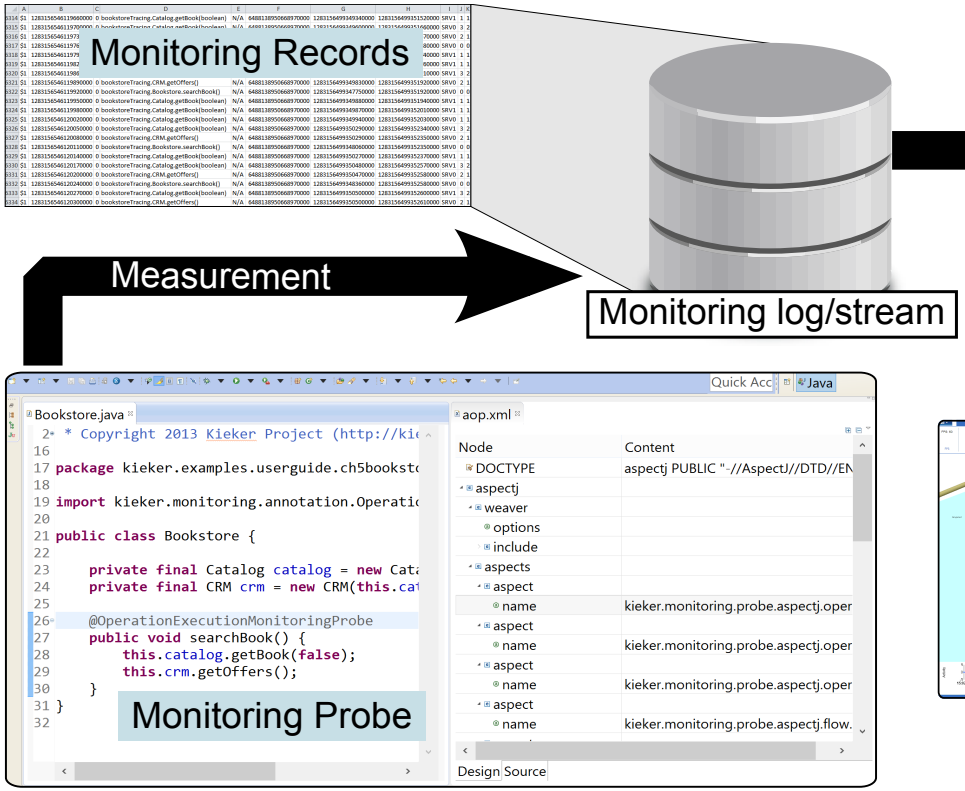
Architecture Extraction with Kieker and SLAstic

André van Hoorn
University of Stuttgart, Germany
Contact: van.hoorn@informatik.uni-stuttgart.de

(including contributions by many colleagues)

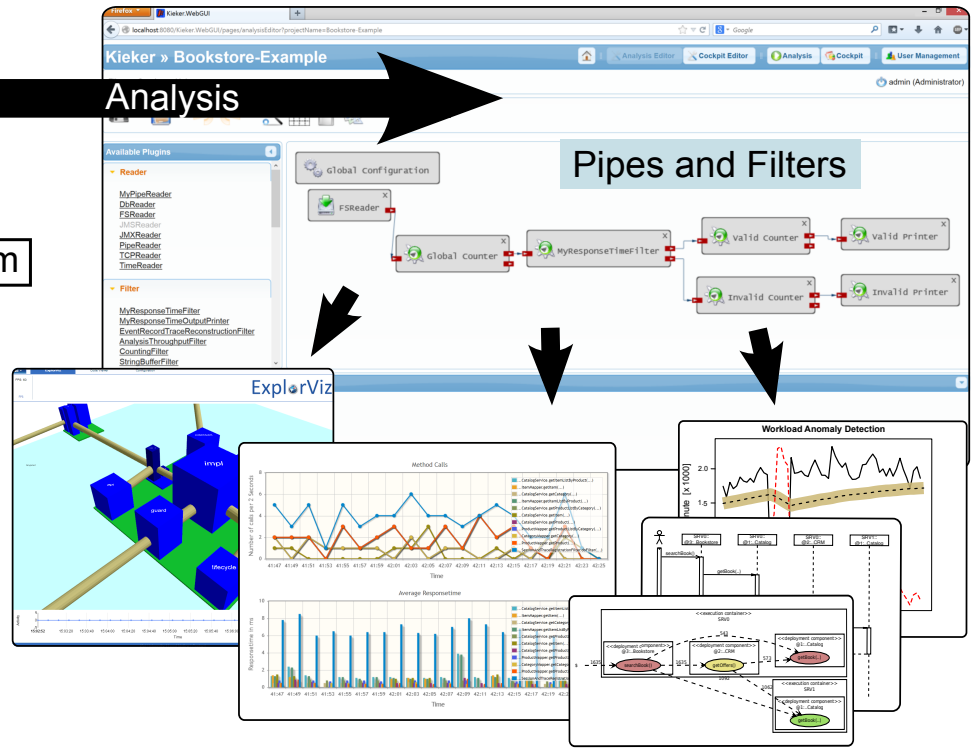
August 22, 2014 @ SPEC RG DevOps Performance WG

Kieker: Dynamic Analysis Workflow



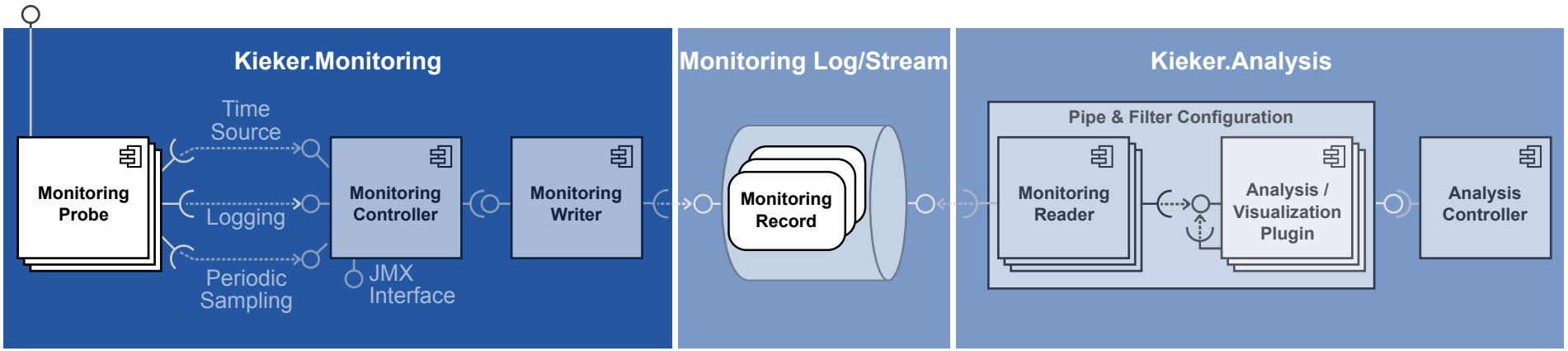
Software System with Monitoring Instrumentation

Analysis Configuration (via API and WebGUI)



Online and Offline Visualization

Core Kieker Framework Components



Java probes/samplers:

Monitoring Probes/Samplers

Control-flow tracing	Manual instrumentation		
	AspectJ	Spring	
	Servlet	CXF/SOAP	
	<your interception technology>		
Resource monitoring	Servlet / Sigar	CPU utilization	
		Memory usage	
	<your technology>		
<your monitoring probe>			

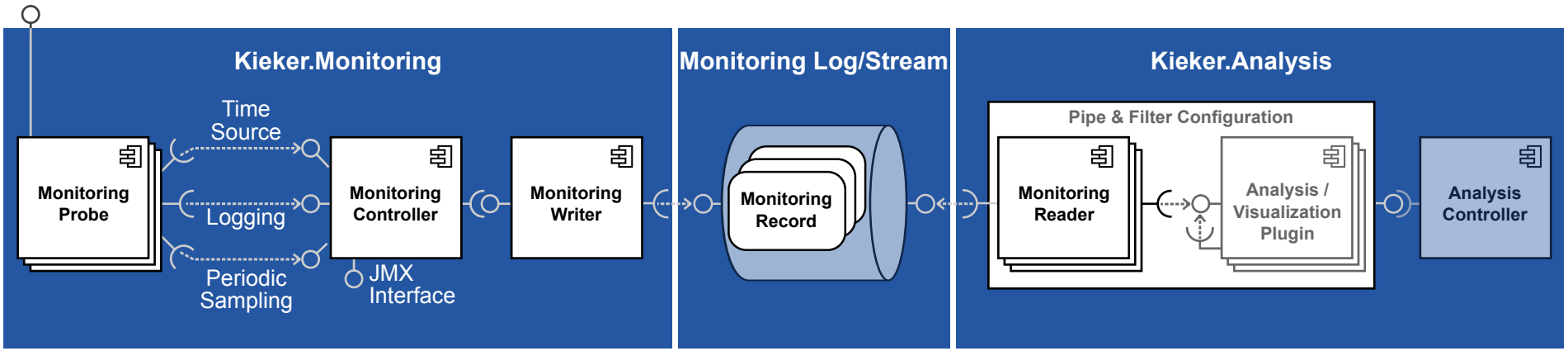
+ basic adapters for

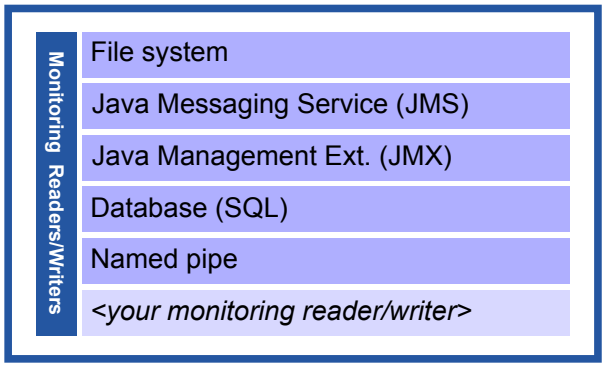
- C#/.NET
- Visual Basic 6/COM
- COBOL

Monitoring Records

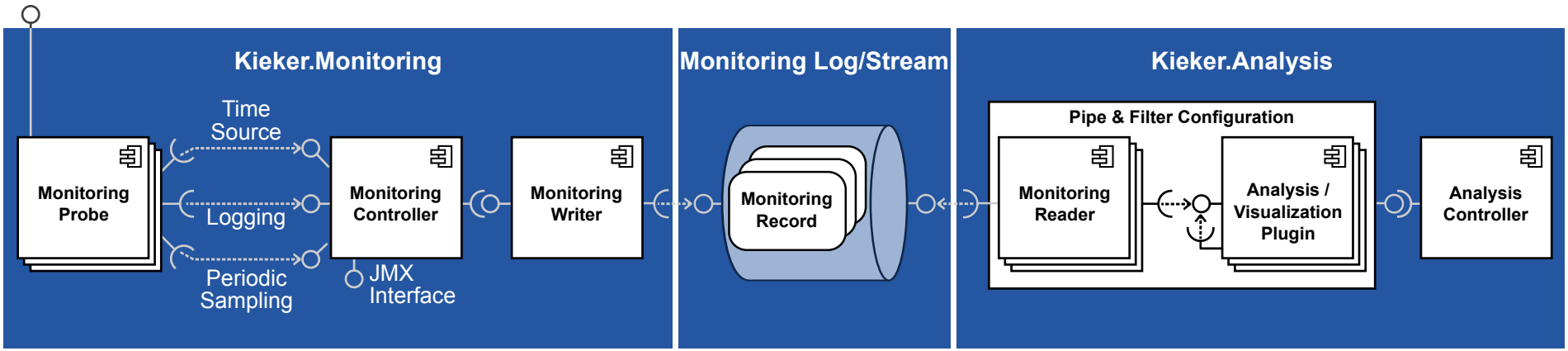
Operation execution	
Control-flow events	
CPU utilization	Memory/swap usage
Resource utilization	
Current time	
<your monitoring record type>	

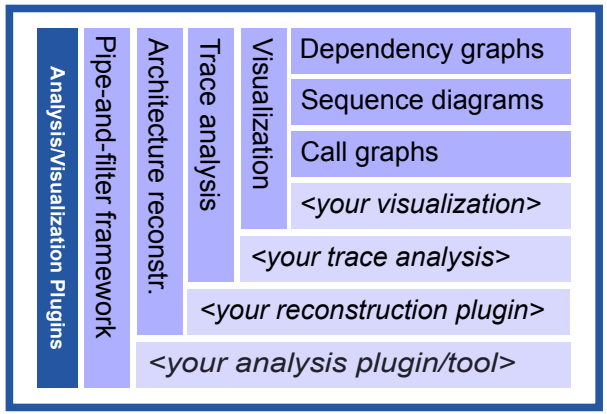
Core Kieker Framework Components





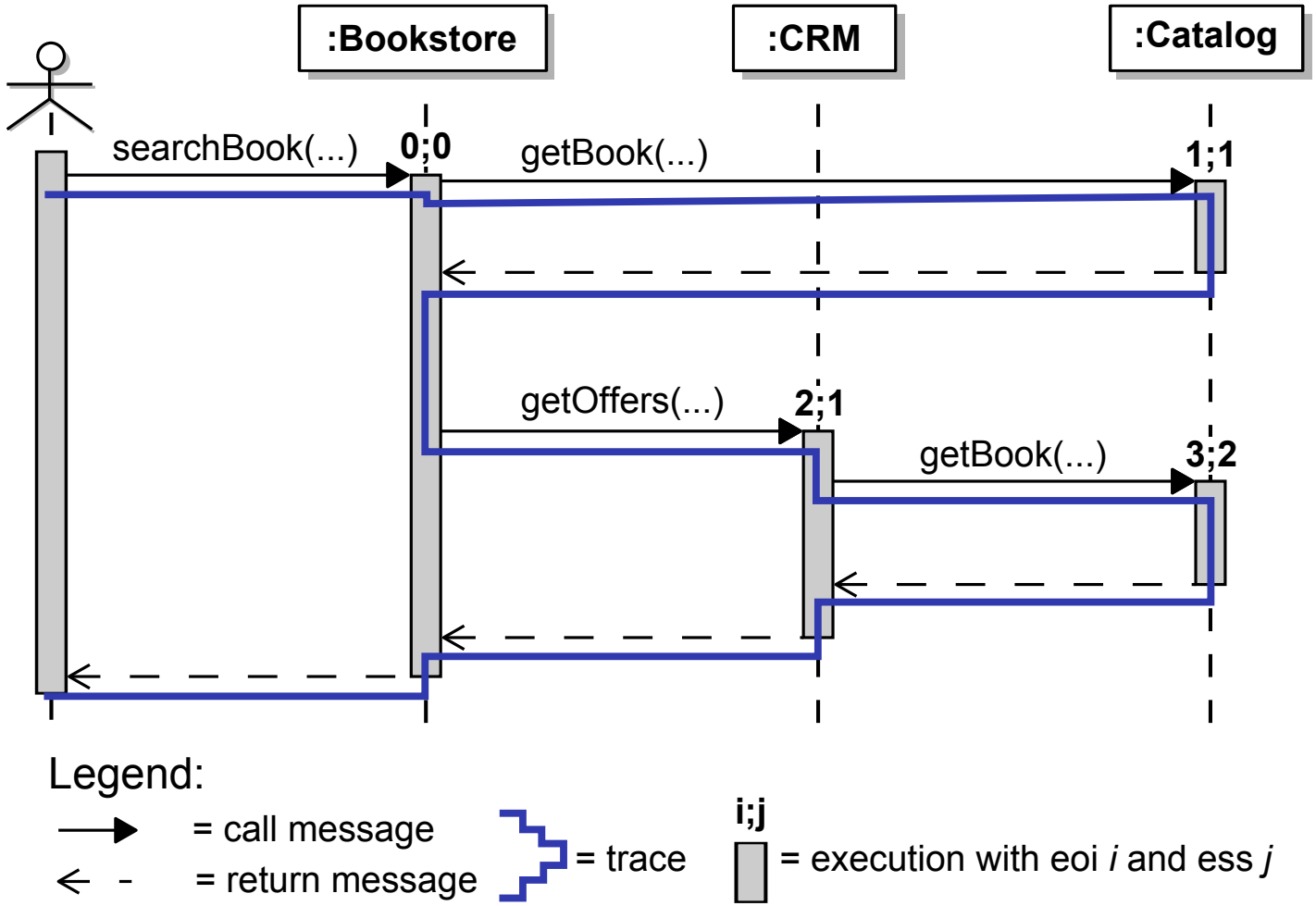
Core Kieker Framework Components





Architecture Discovery: Model Extraction + Visualization

Trace Terminology

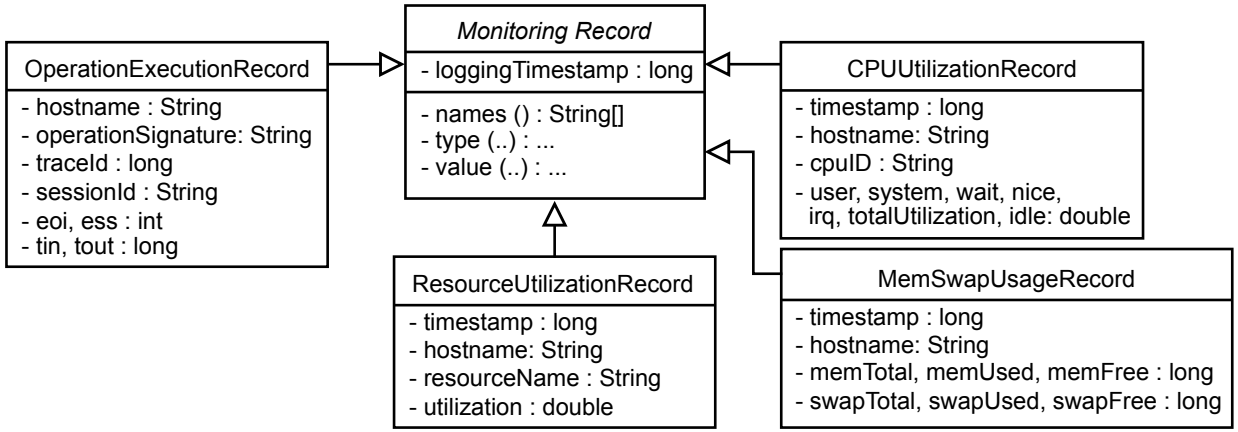


Execution order index (eoi) *i*: *i*-th started execution in a trace

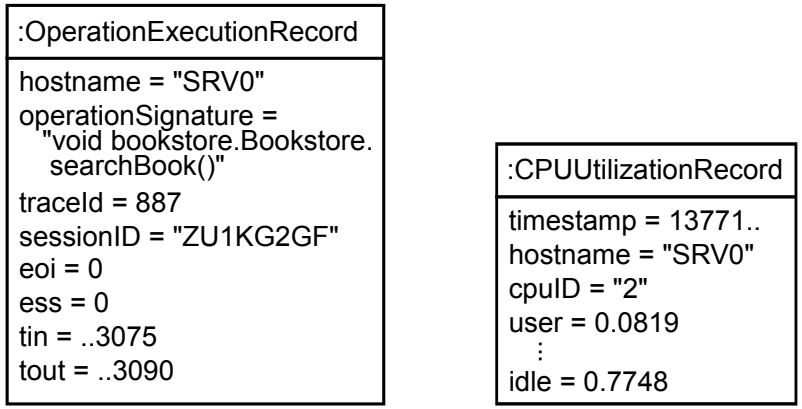
Execution stack size (ess) *j*: execution started at stack depth *j*

Abstract and Example Monitoring Records

- Extensible Monitoring Record (Meta-)Model



- Example Monitoring Record instances



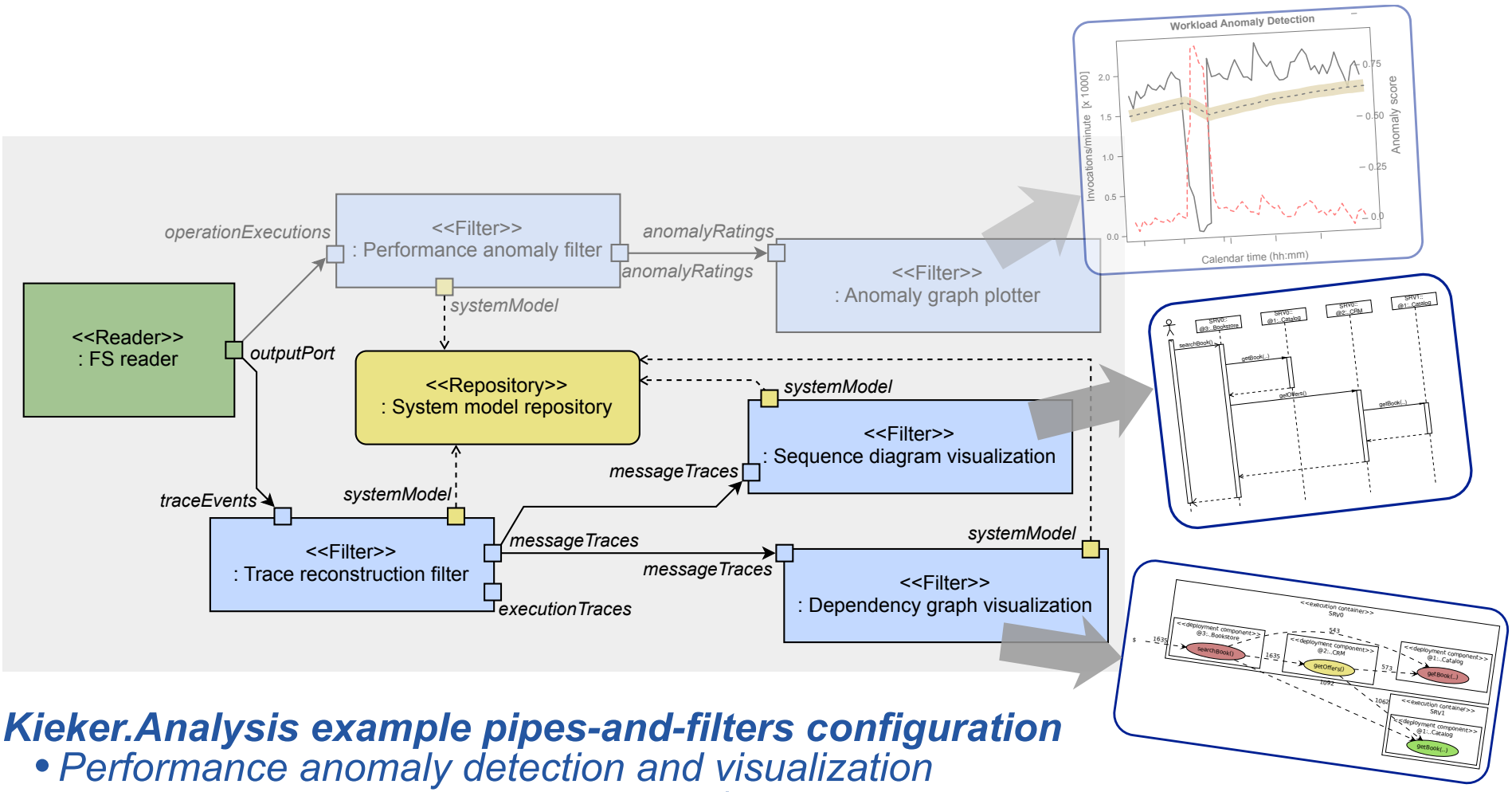
Listing 1: META-INF/aop.xml

```
<!DOCTYPE aspectj PUBLIC "-//AspectJ//DTD//EN" "http://www.aspectj.org↩
    /dtd/aspectj_1_5_0.dtd">
<aspectj>
    <weaver options="">
        <include within="*"/>
    </weaver>
    <aspects>
      <aspect name="kieker.monitoring.probe.aspectj.operationExecution.↩
          OperationExecutionAspectFull"/>
    </aspects>
</aspectj>
```

Start the monitored application:

```
$ java -javaagent:lib/kieker-1.9_aspectj.jar BookstoreStarter
```

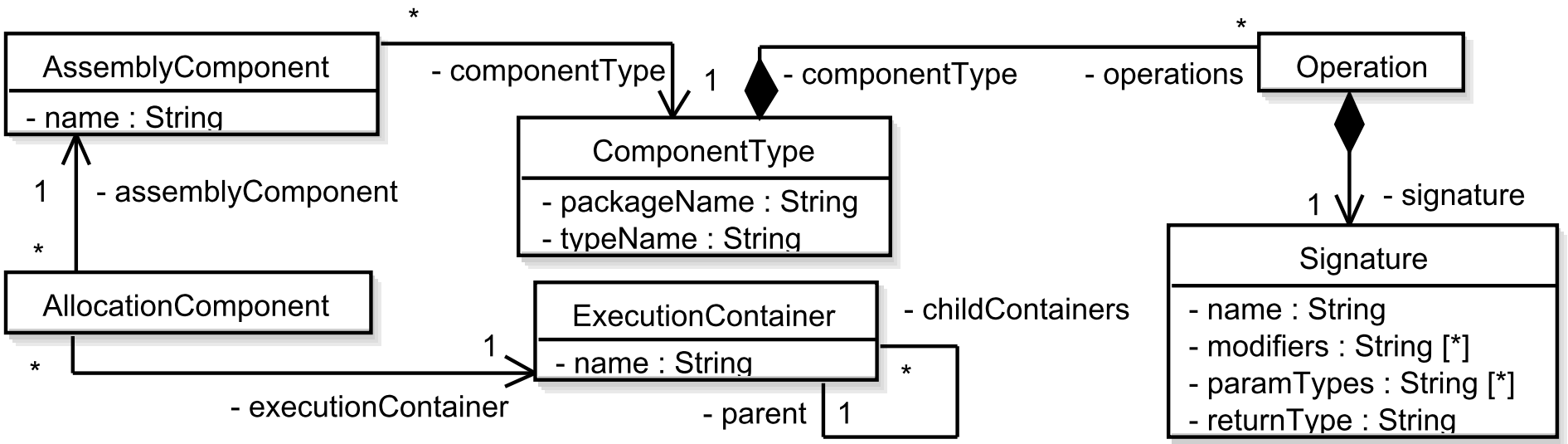
Example Pipe-and-Filter Configuration



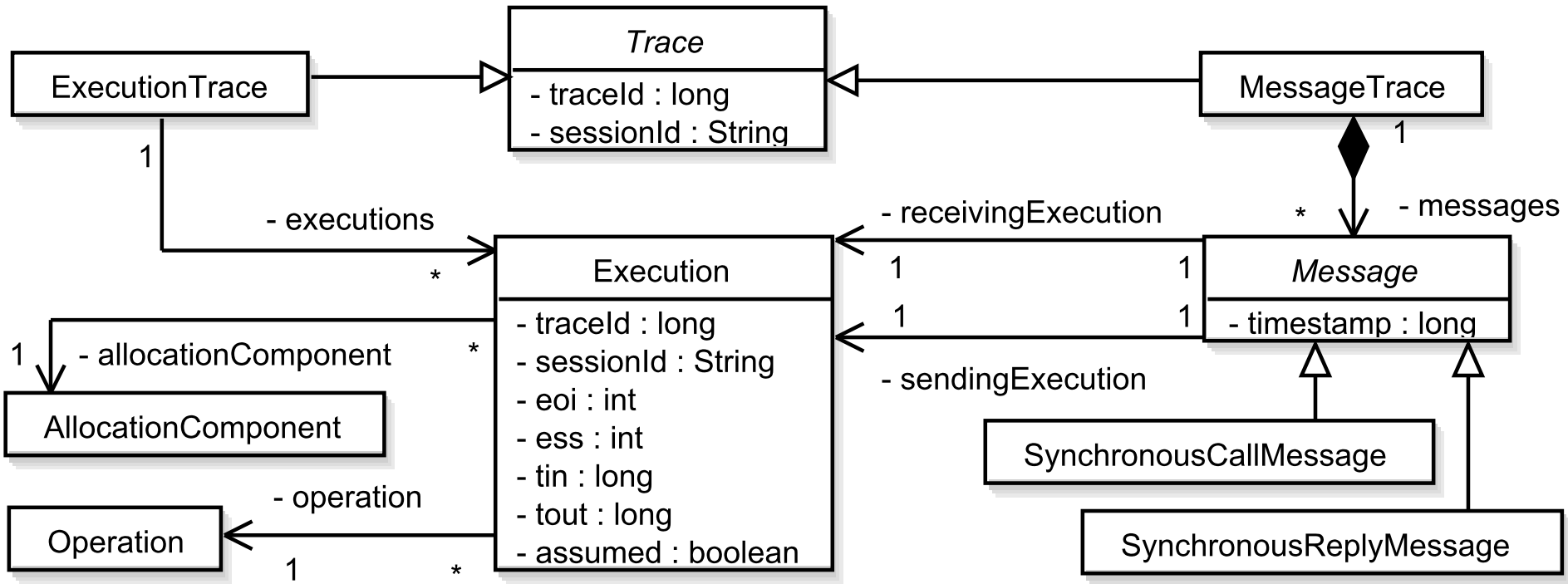
Kieker.Analysis example pipes-and-filters configuration

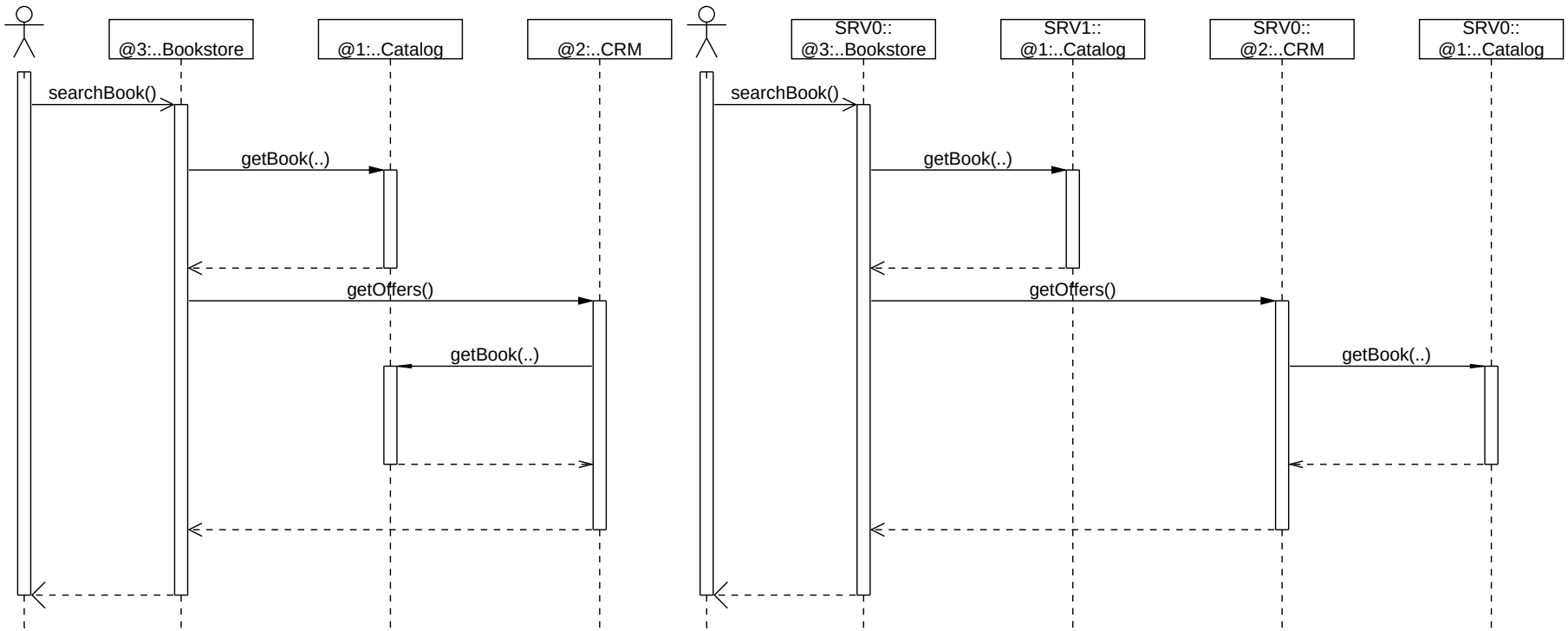
- *Performance anomaly detection and visualization*
- *Architecture and trace reconstruction/visualization*

Kieker's System Meta-Model



Kieker's Trace Meta-Model



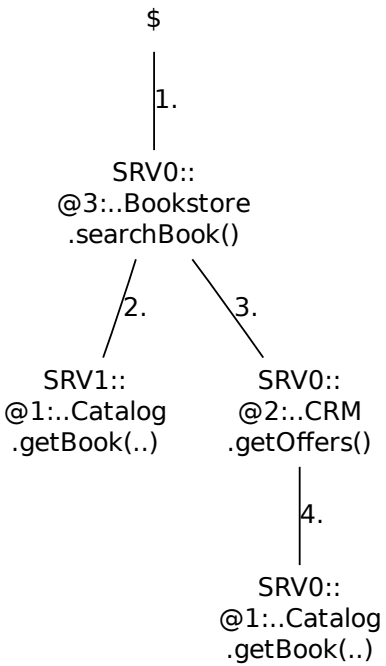


(a) Assembly-level view

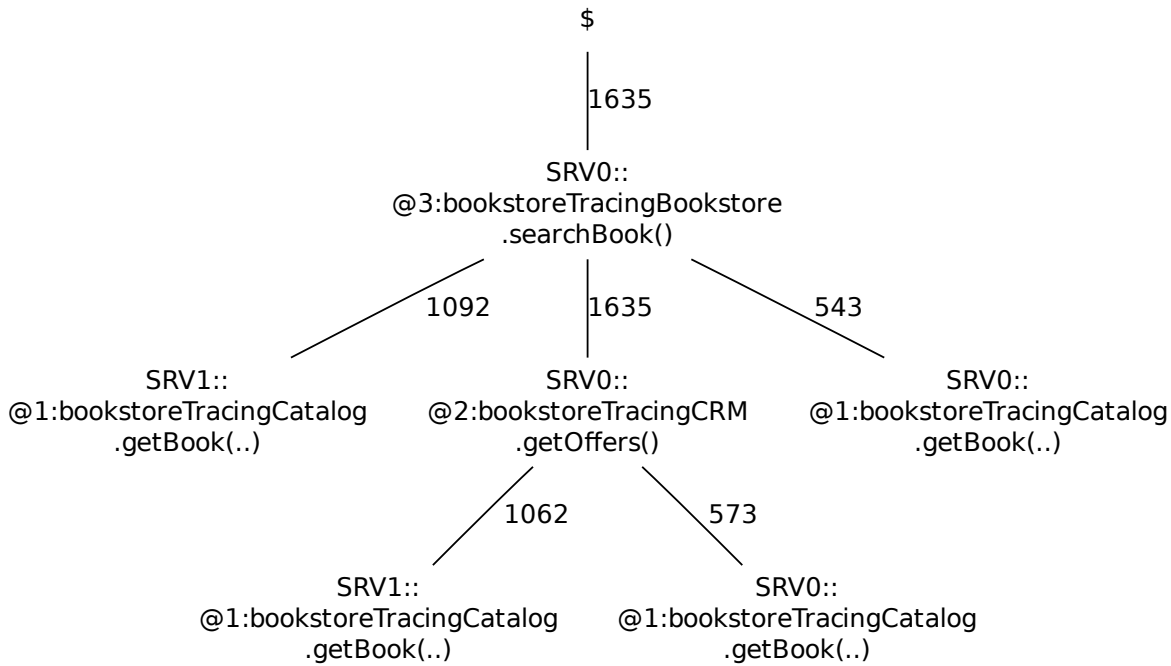
(b) Deployment-level view

Dynamic Call Trees

1. Sequence diagrams
2. **Dynamic call trees**
3. Hierarchical calling dependency graphs
4. System model



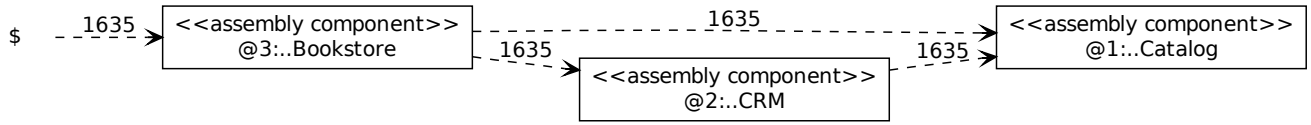
(a) Dynamic call tree **(single trace)**



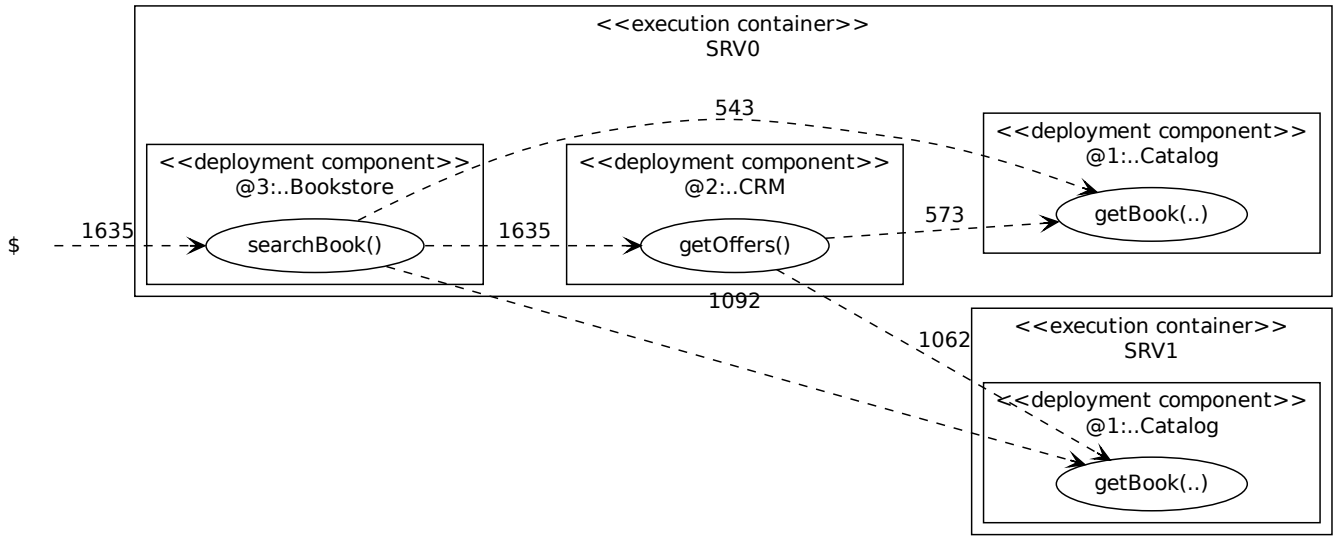
(b) **Aggregated** deployment-level call tree

Hierarchical Calling Dependency Graphs

1. Sequence diagrams
2. Dynamic call trees
3. **Hierarchical calling dependency graphs**
4. System model

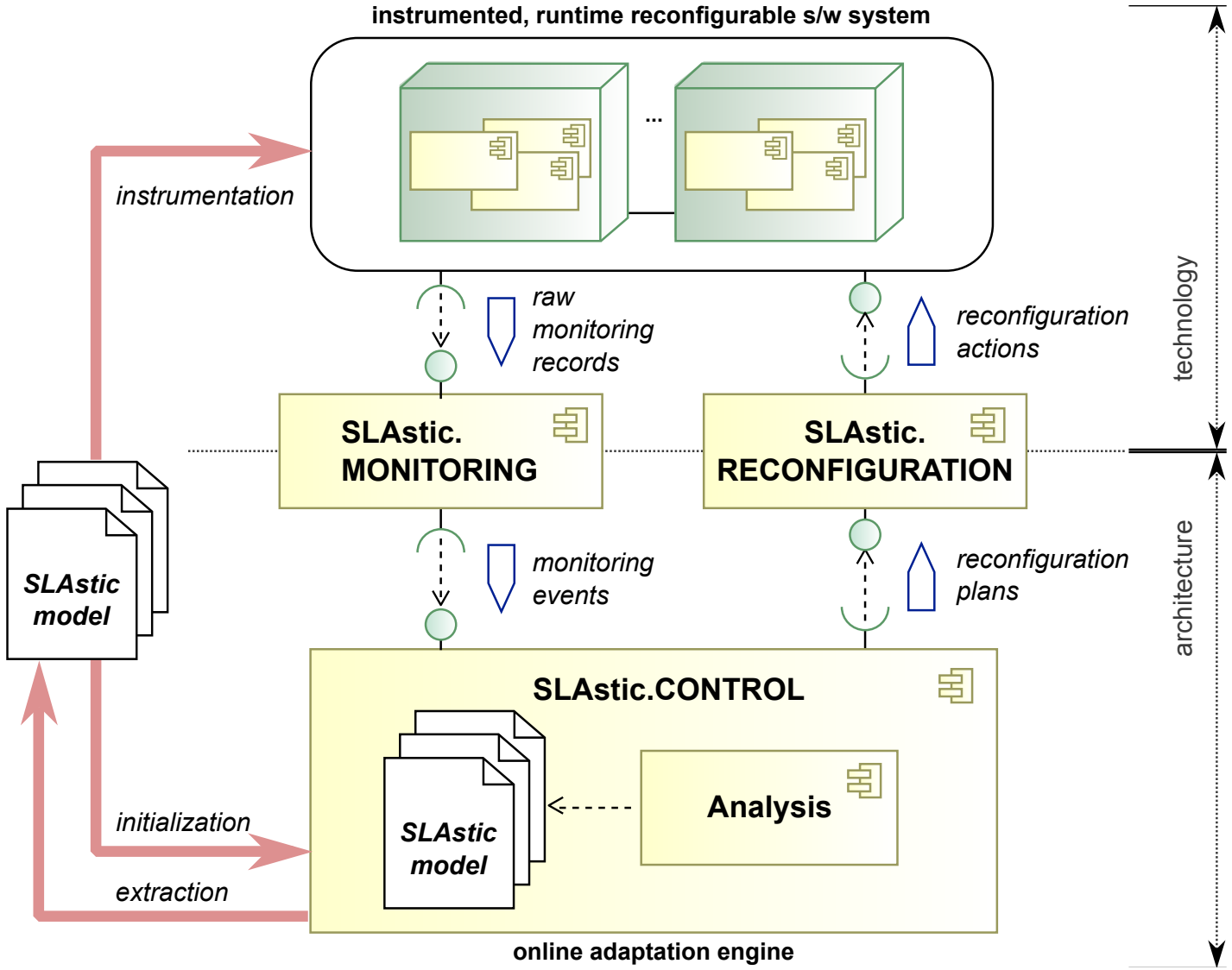


(a) Assembly-level **component** dependency graph



(b) Deployment-level **operation** dependency graph

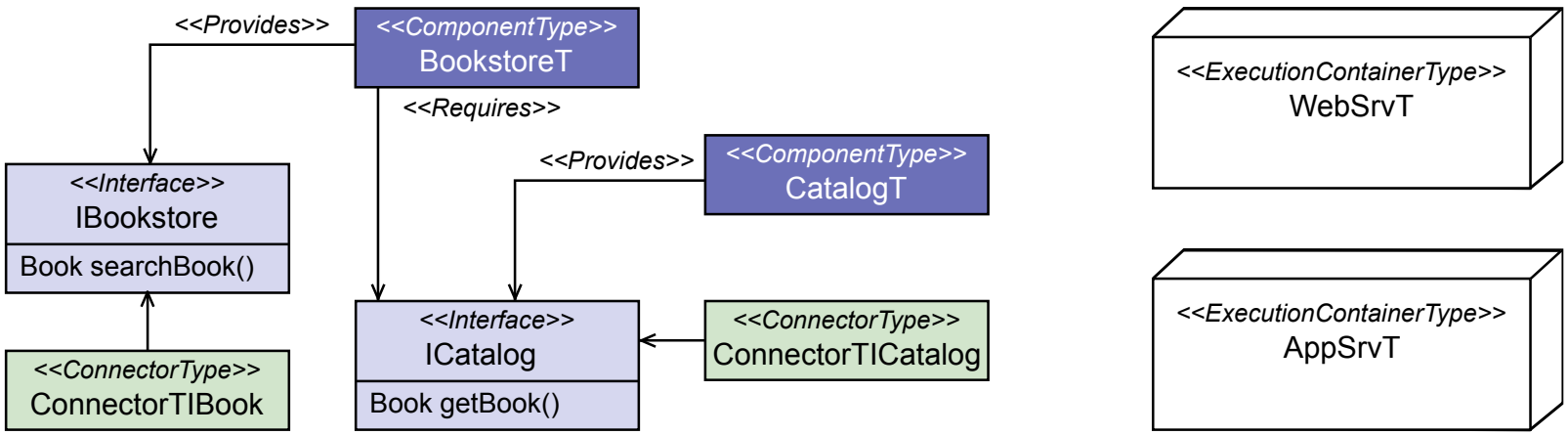
SLAstic Framework Overview



SLAstic Meta-Model

System Partition (also used as runtime model)

1. Type repository (e.g., component types, interfaces, connector types, execution container types)

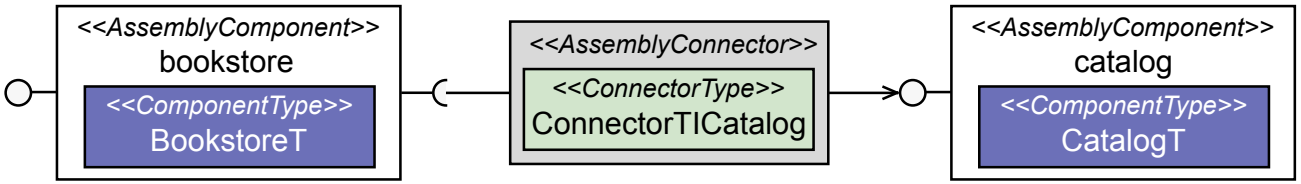


Example type repository

SLAstic Meta-Model

System Partition (also used as runtime model)

1. Type repository (e.g., component types, interfaces, connector types, execution container types)
2. Component assembly (e.g., assembly of components via connectors)

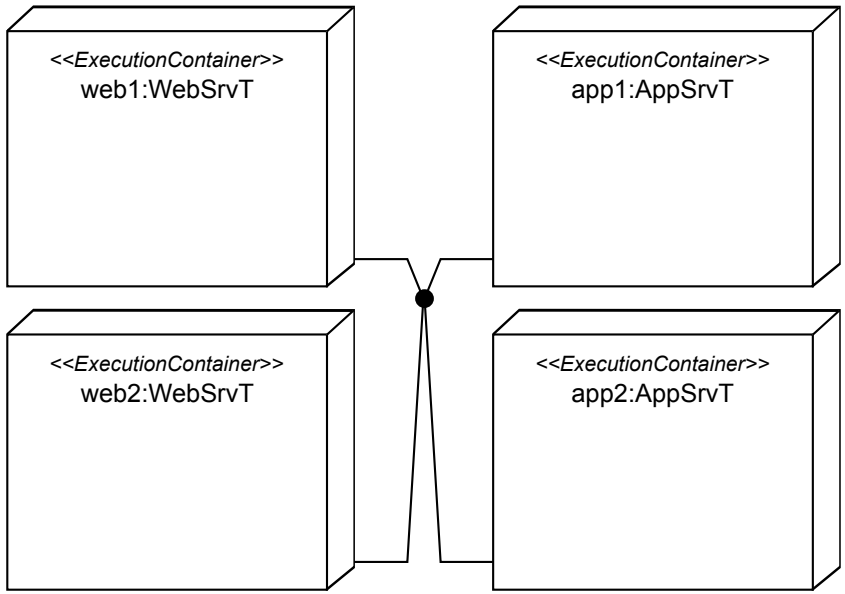


Example component assembly

SLAstic Meta-Model

System Partition (also used as runtime model)

1. Type repository (e.g., component types, interfaces, connector types, execution container types)
2. Component assembly (e.g., assembly of components via connectors)
3. Execution environment (e.g., execution containers and interconnection via links)

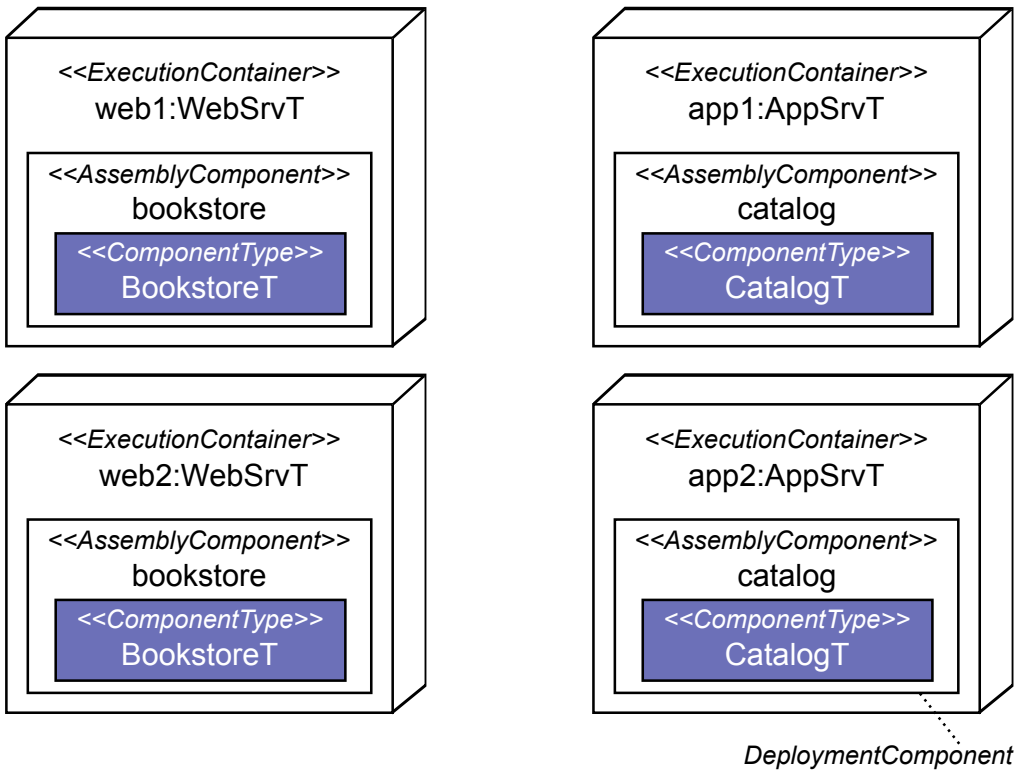


Example execution environment

SLAstic Meta-Model

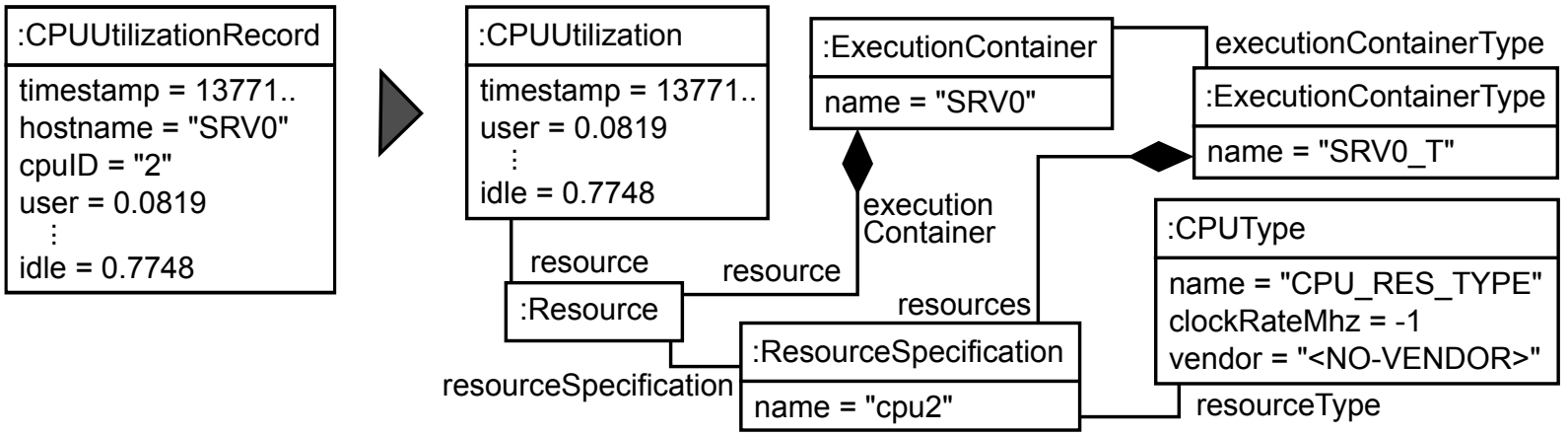
System Partition (also used as runtime model)

1. Type repository (e.g., component types, interfaces, connector types, execution container types)
2. Component assembly (e.g., assembly of components via connectors)
3. Execution environment (e.g., execution containers and interconnection via links)
4. Component deployment (mapping: assembly components → containers)

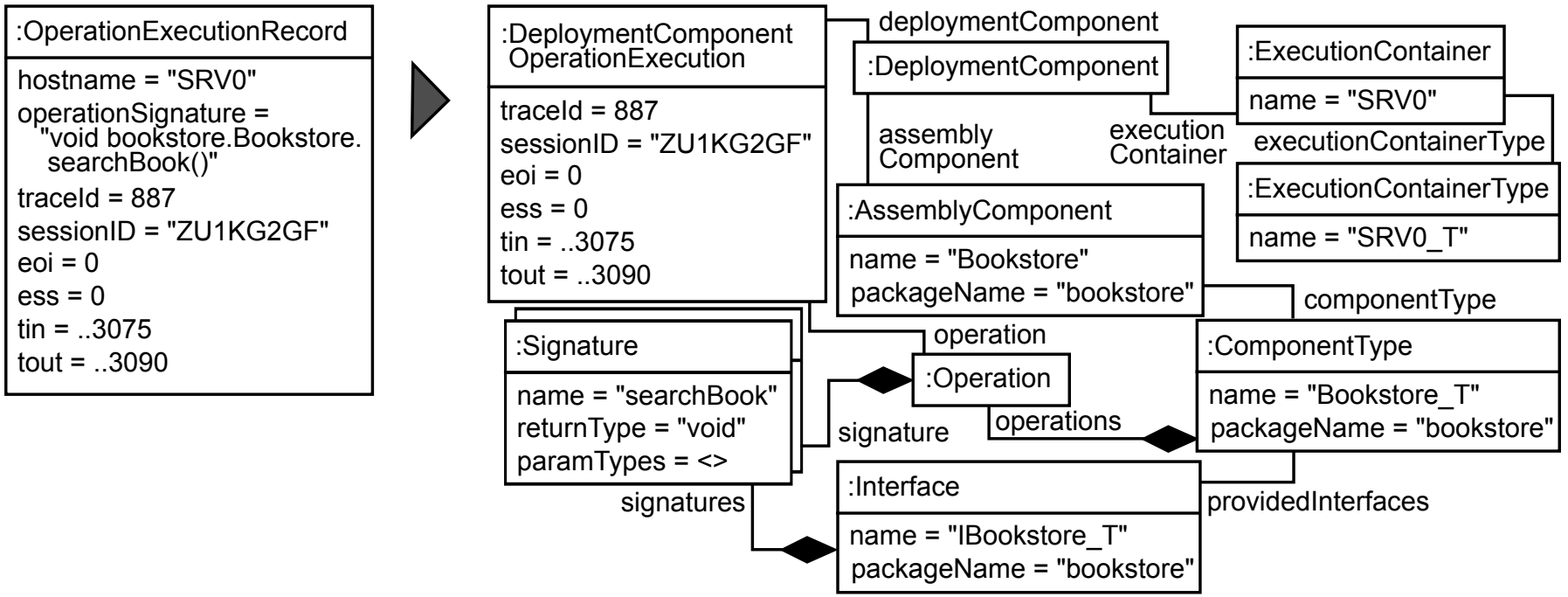


Example component deployment

On-Demand Architecture Extraction

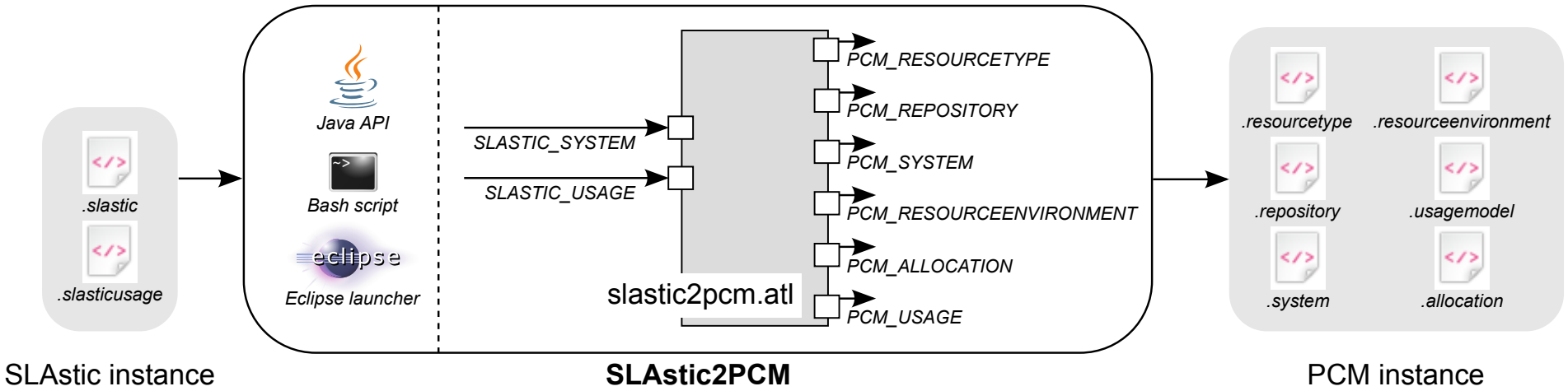


On-Demand Architecture Extraction (cont'd)



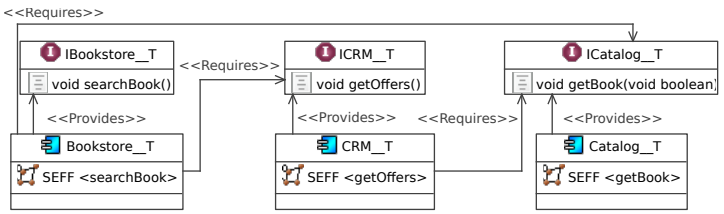
- Different component (name) abstraction modes supported
- Extraction of usage model not depicted in this talk

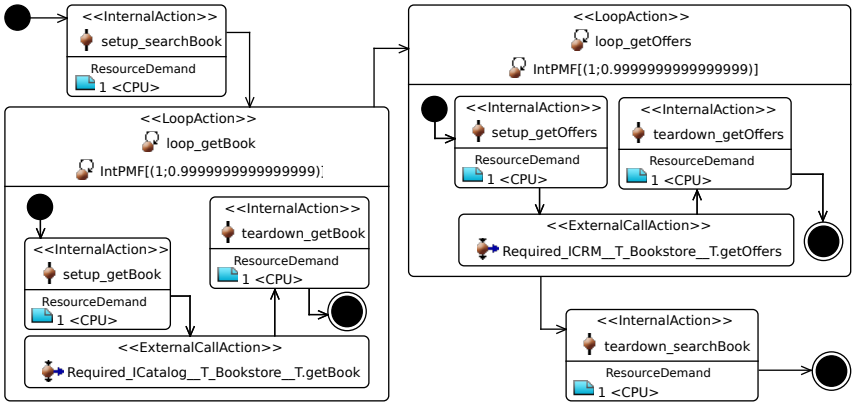
M2M Transformation from SLAstic to PCM

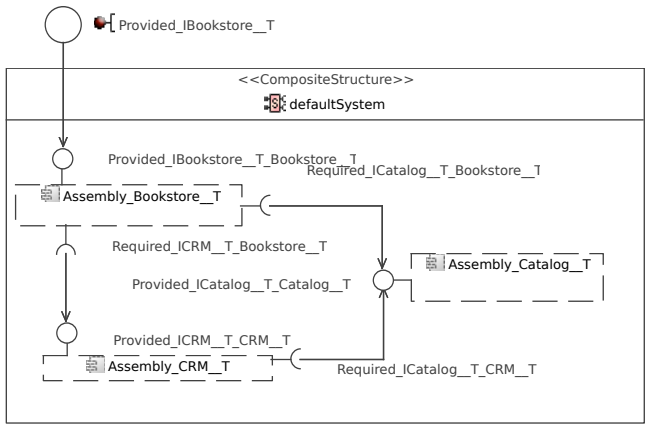


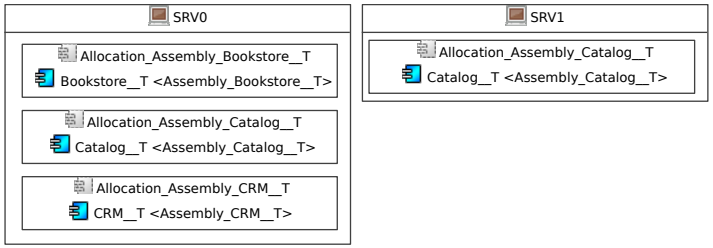
- Assumption that PCM models completed in subsequent step
- Decoration concept to connect SLAstic and PCM model

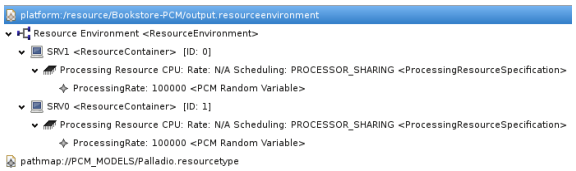
Transformation Results (Selected Examples)

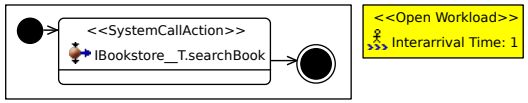












Resources

- Kieker:
  - `http://kieker-monitoring.net`

- SLAstic
  - van Hoorn [2014]
  - Download:
    `http://kieker-monitoring.net/research/projects/slastic/`

Thanks to All Contributors

Various people contributed to Kieker in the past years.

Tillmann (Till) Bielefeld, Peer Brauer, Philipp Döhring, Jens Ehlers, Nils Ehmke, Florian Fittkau, Thilo Focke, Sören Frey, Tom Frotscher, Henry Grow, Wilhelm (Willi) Hasselbring, André van Hoorn, Reiner Jung, Benjamin Kiel, Dennis Kieselhorst, Holger Knoche, Arnd Lange, Marius Löwe, Marco Lübcke, Felix Magedanz, Nina Marwede, Robert von Massow, Jasminka Matevska, Oliver Preikszas, Sönke Reimer, Bettual Richter, Matthias Rohr, Nils Sommer, Lena Stöver, Jan Waller, Robin Weiß, Björn Weißenfels, Matthias Westphal, Christian Wulf

—Alphabetic list of people who contributed in different form (source code, bug reports, promotion, etc) and intensity

Literature

Kieker Project. Kieker 1.9 user guide. `http://kieker-monitoring.net/documentation/`, Apr. 2014a.

Kieker Project. Kieker web site. `http://kieker-monitoring.net/`, 2014b.

A. van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Number 2014/<++NUMBER> in Kiel Computer Science Series. Department of Computer Science, Kiel University, Kiel, Germany, 2014. Dissertation (under review), Faculty of Engineering, Kiel University.

A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst. Continuous monitoring of software services: Design and application of the Kieker framework. Technical Report TR-0921, Department of Computer Science, University of Kiel, Germany, Nov. 2009.

A. van Hoorn, J. Waller, and W. Hasselbring. Kieker: A framework for application performance monitoring and dynamic software analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12)*, pages 247–248. ACM, Apr. 2012. ISBN 978-1-4503-1202-8. doi: 10.1145/2188286.2188326.

For a comprehensive list of publications, talks, and theses about Kieker, visit:
`http://kieker-monitoring.net/research/`