Architecture-based Online Capacity Management: Modeling, Monitoring, and Adaptation

D3S Seminar/Department Meeting
Department of Distributed and Dependable Systems
Charles University, Prague

André van Hoorn

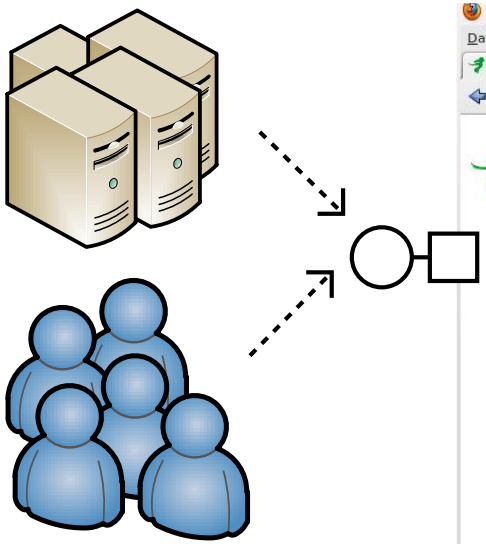
Software Engineering Group
University of Kiel, Germany

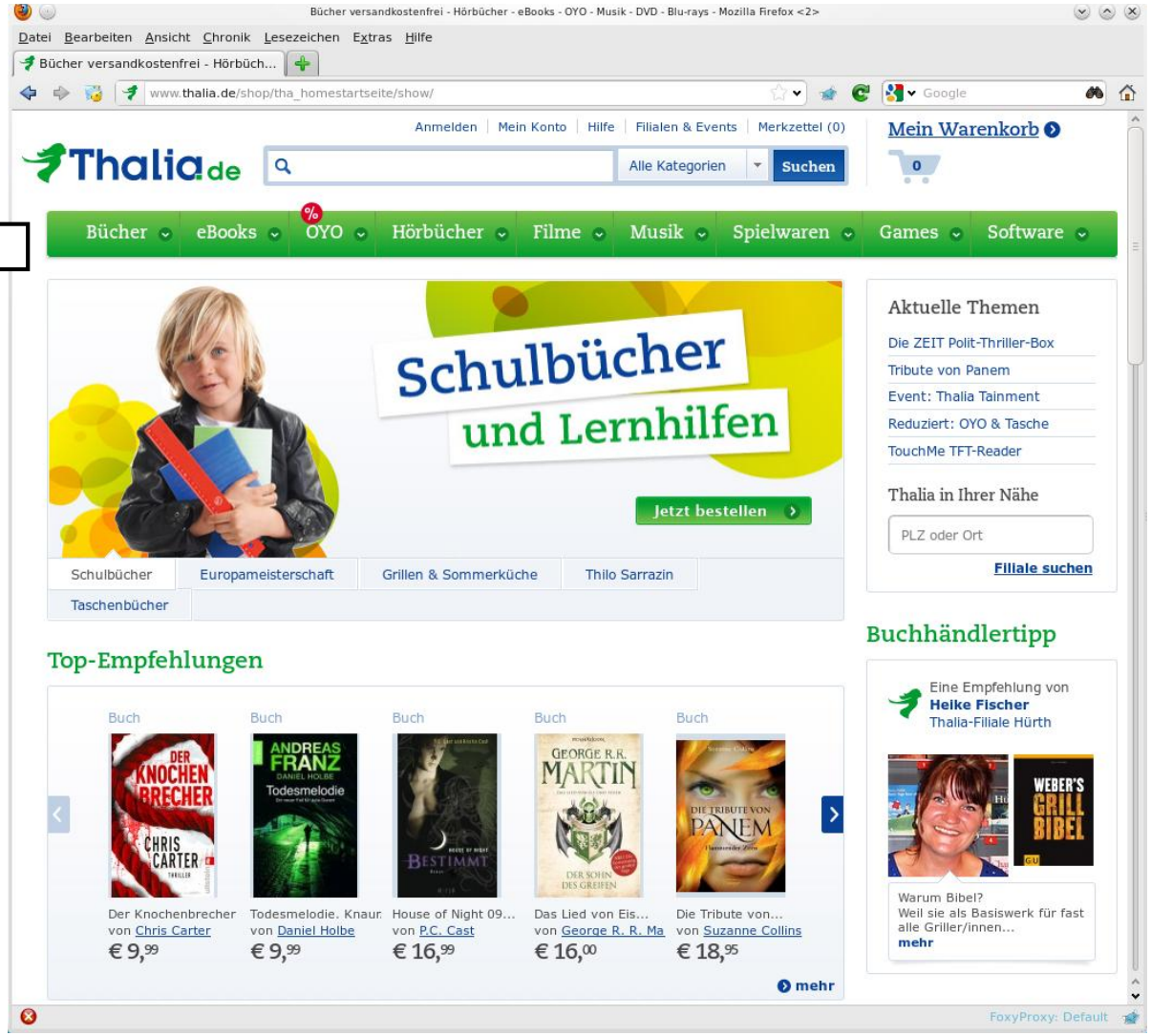
avh@informatik.uni-kiel.de

June 20, 2012 @ Prague

Motivation & Overall Goal

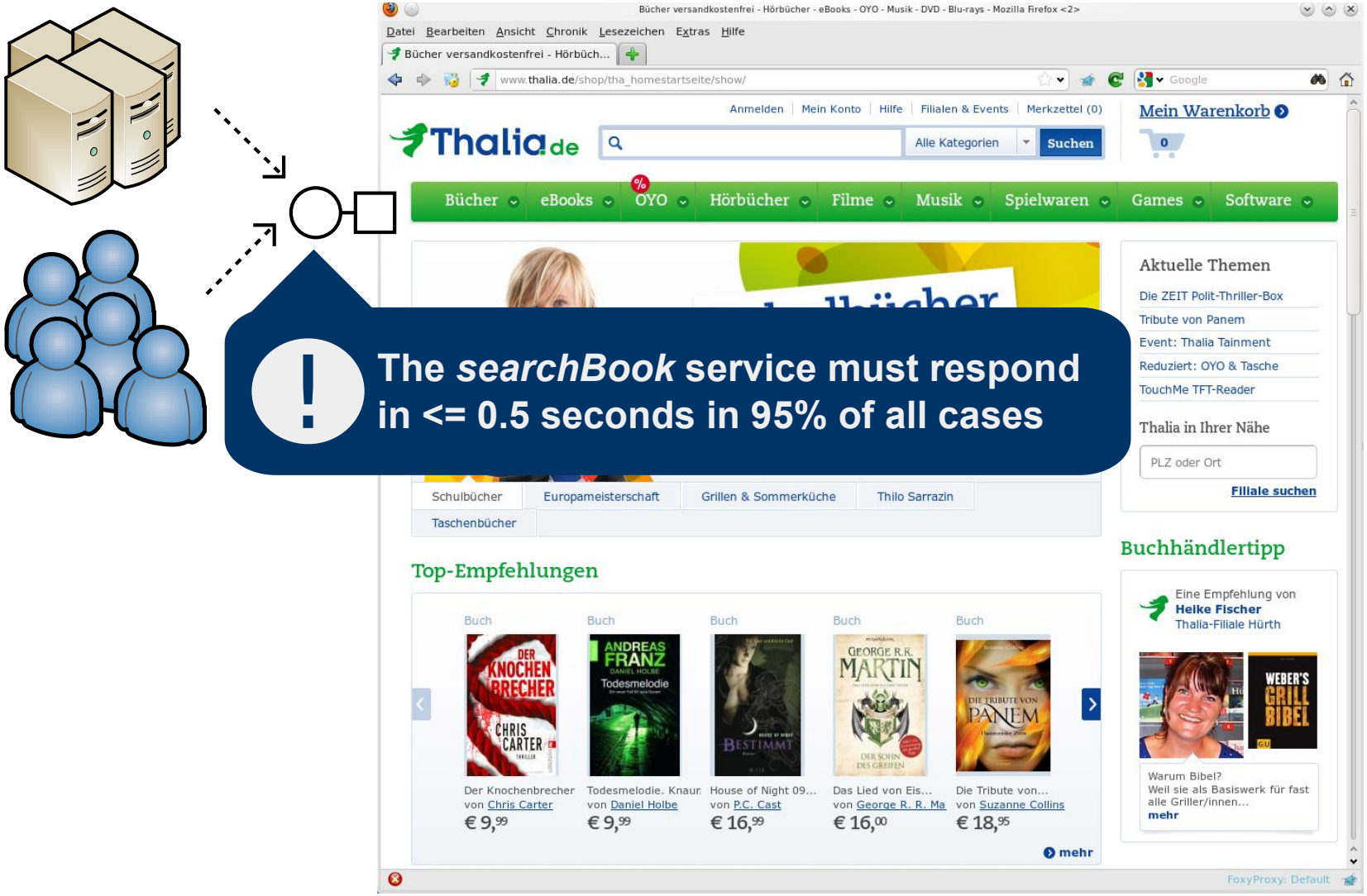
Christian-Albrechts-Universität zu Kiel





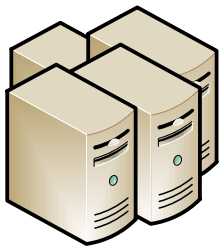
Motivation & Overall Goal

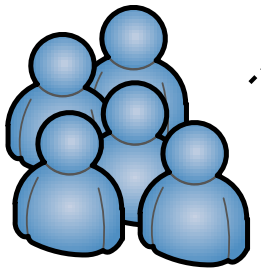
Christian-Albrechts-Universität zu Kiel

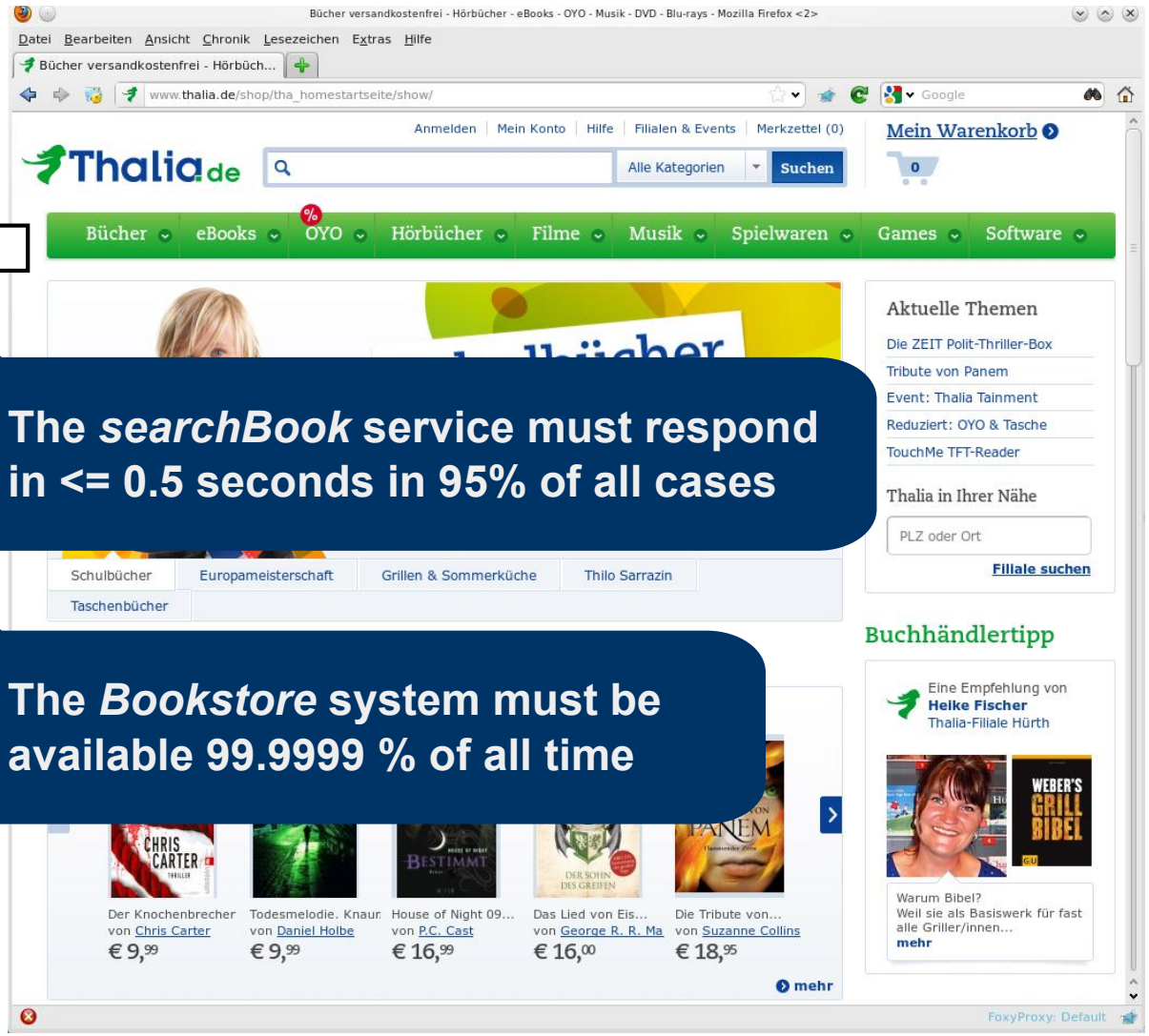


The *searchBook* service must respond in <= 0.5 seconds in 95% of all cases

Motivation & Overall Goal





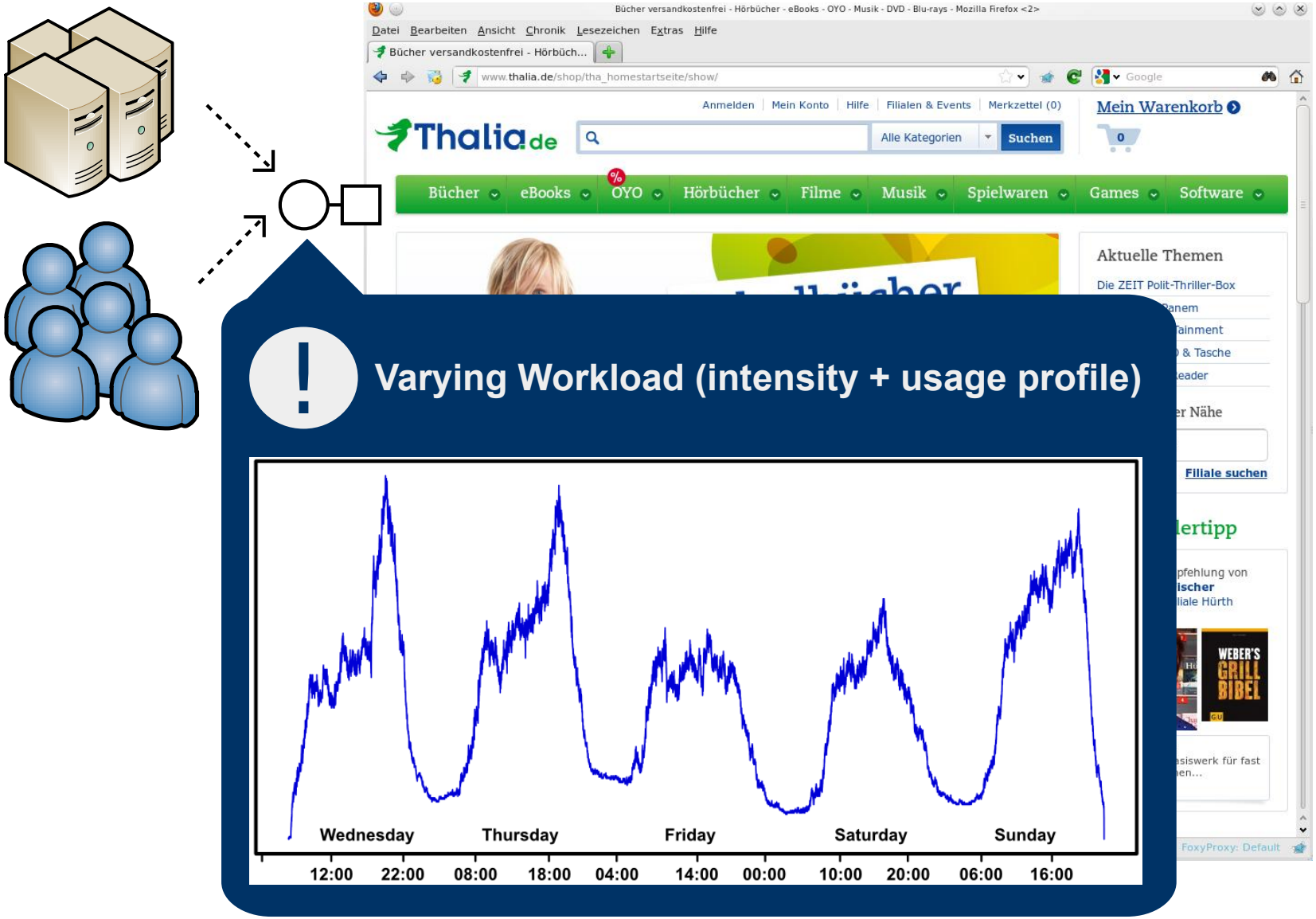


The *searchBook* service must respond in <= 0.5 seconds in 95% of all cases

The *Bookstore* system must be available 99.9999 % of all time

Motivation & Overall Goal

Christian-Albrechts-Universität zu Kiel

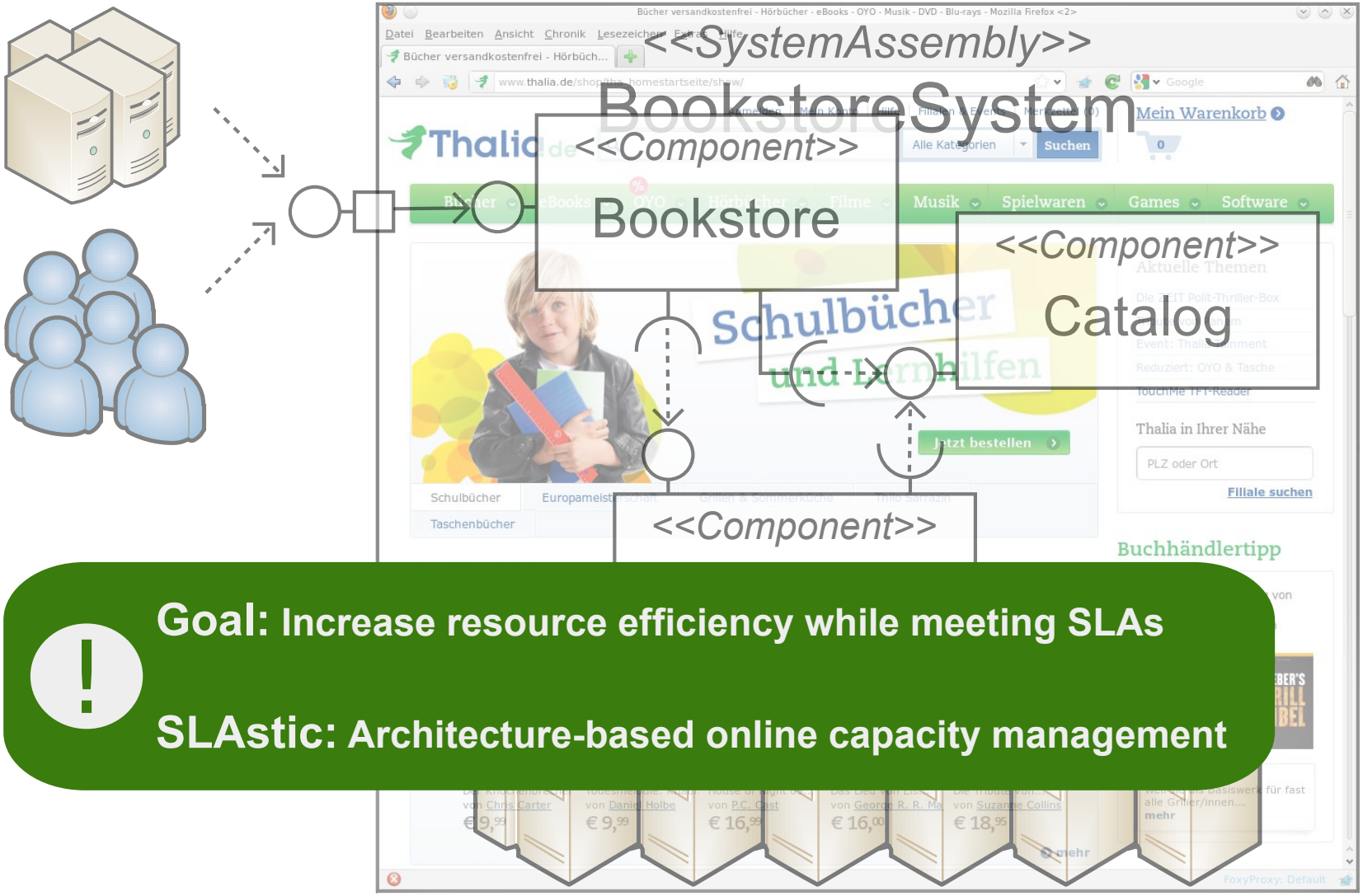


Motivation & Overall Goal

Christian-Albrechts-Universität zu Kiel

Motivation & Overall Goal

Christian-Albrechts-Universität zu Kiel

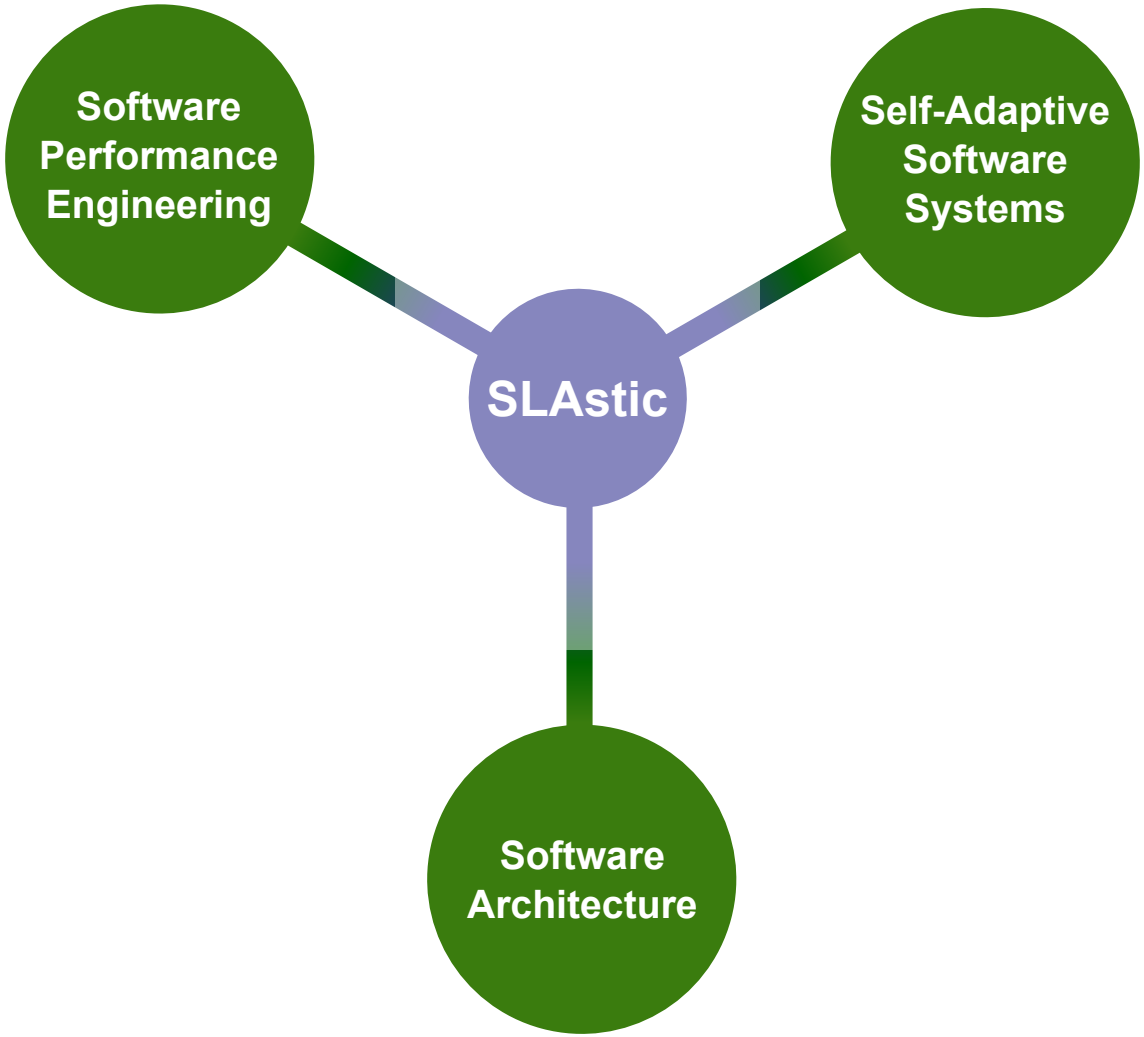


Goal: Increase resource efficiency while meeting SLAs

SLAstic: Architecture-based online capacity management

Scientific Context

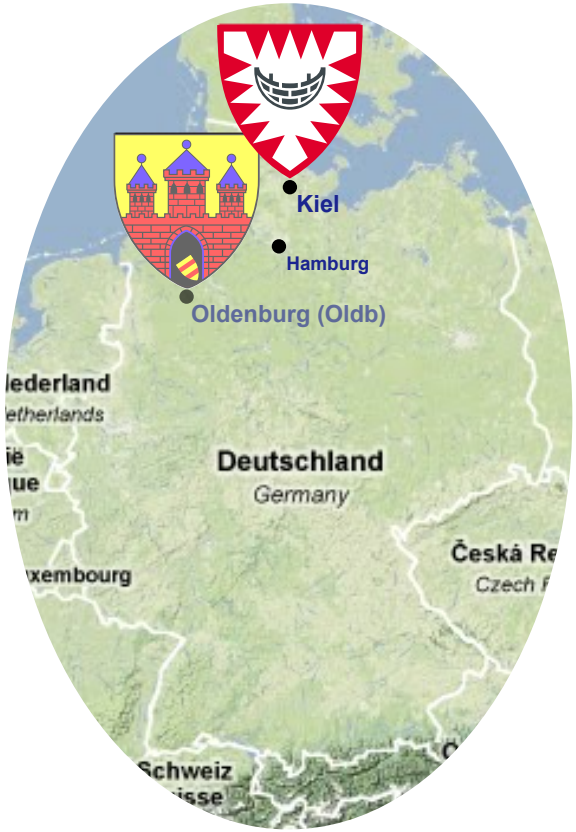
Christian-Albrechts-Universität zu Kiel



About Me

Affiliation

- **University of Kiel, Software Engineering Group**
- Before: University of Oldenburg (DFG RTG TrustSoft)



About Me

Christian-Albrechts-Universität zu Kiel

Affiliation

- **University of Kiel, Software Engineering Group**
- Before: University of Oldenburg (DFG RTG TrustSoft)

Research Interests

1. **Software performance engineering & self-***
   - {Model,Architecture}-{based,driven} SPE techniques (+ tools)
   - Online performance management (monitoring, analysis)
   - (Architectural) performance models @ runtime
2. **Software architecture**
   - Component- & service-based software systems, MDSD
   - QoS (particularly, performance and resource efficiency)
   - Runtime reconfiguration/adaptation, self-*

Current Projects

1. **SLAstic** — Model-Driven Online Capacity Management for C-B Software Systems
2. **Kieker** — Application Performance Management and Dynamic Software Analysis

About Me

Affiliation

- **University of Kiel, Software Engineering Group**
- Before: University of Oldenburg (DFG RTG TrustSoft)

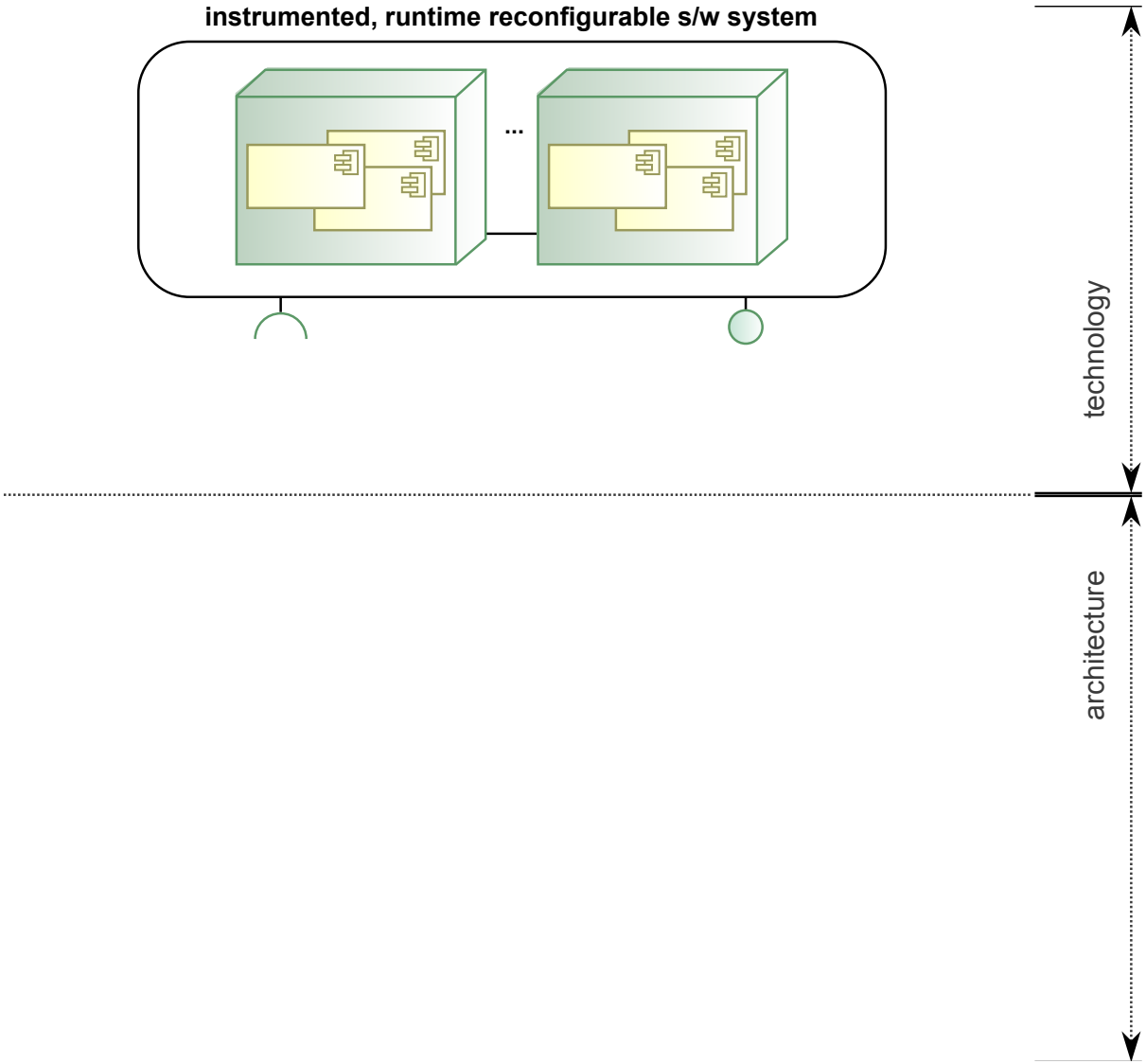
Research Interests

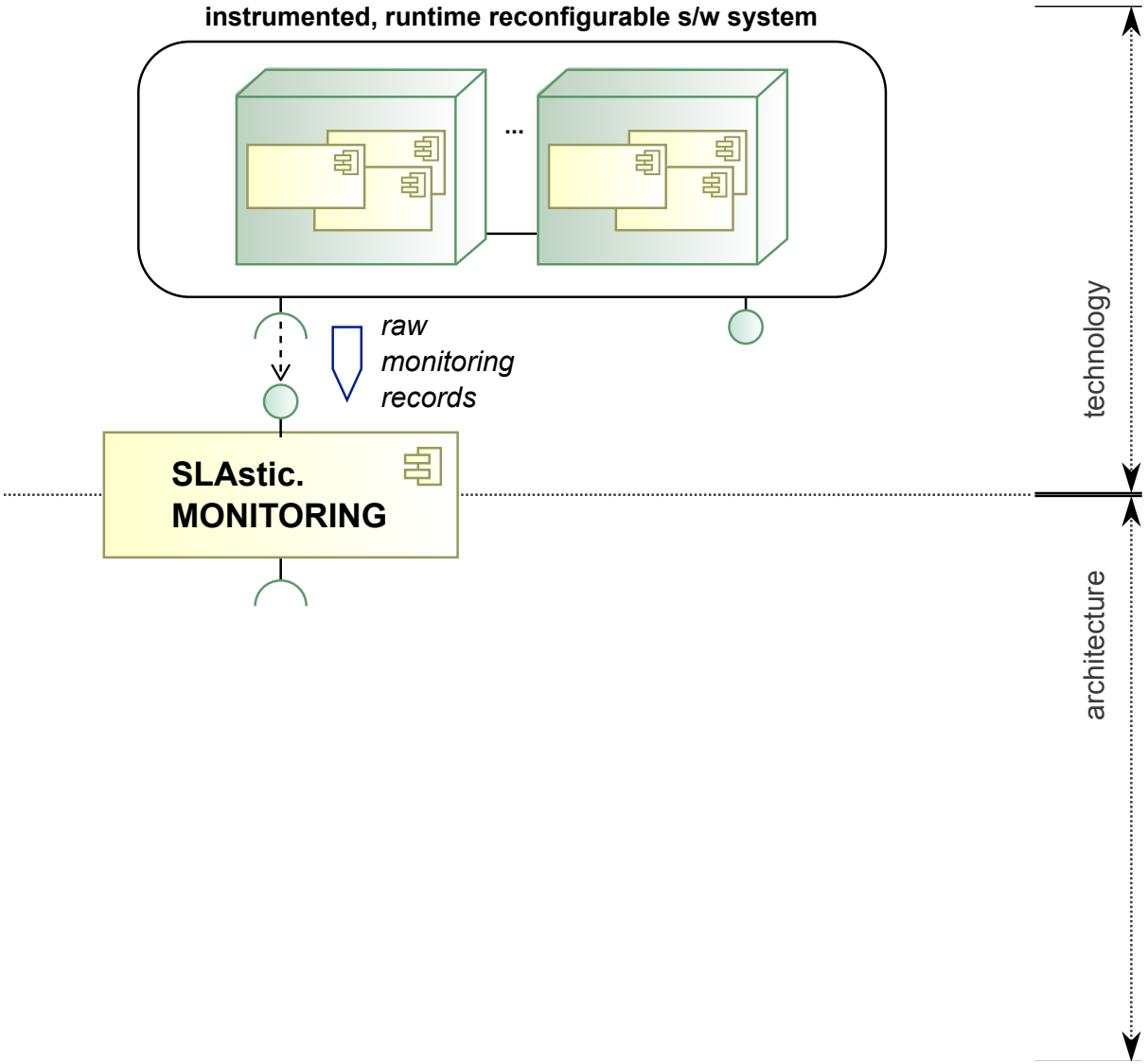
1. **Software performance engineering & self-***
   - {Model,Architecture}-{based,driven} SPE techniques (+ tools)
   - Online performance management (monitoring, analysis)
   - (Architectural) performance models @ runtime
2. **Software architecture**
   - Component- & service-based software systems, MDSD
   - QoS (particularly, performance and resource efficiency)
   - Runtime reconfiguration/adaptation, self-*
3. **Software re(verse)-engineering**
   - Dynamic and hybrid (legacy) software analysis
   - Extraction of architectural models and usage profiles
   - Architecture-based software modernization

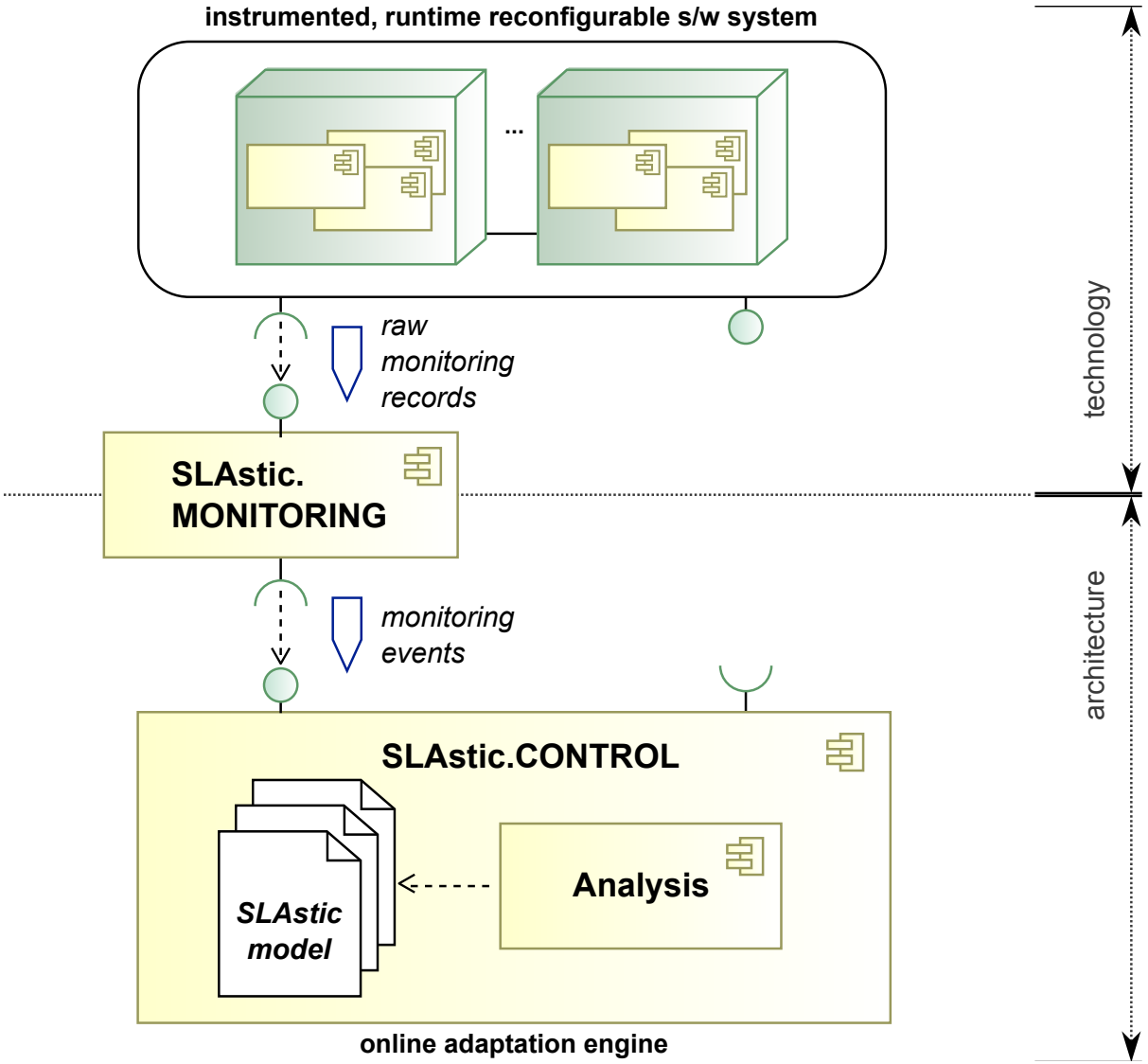
Current Projects

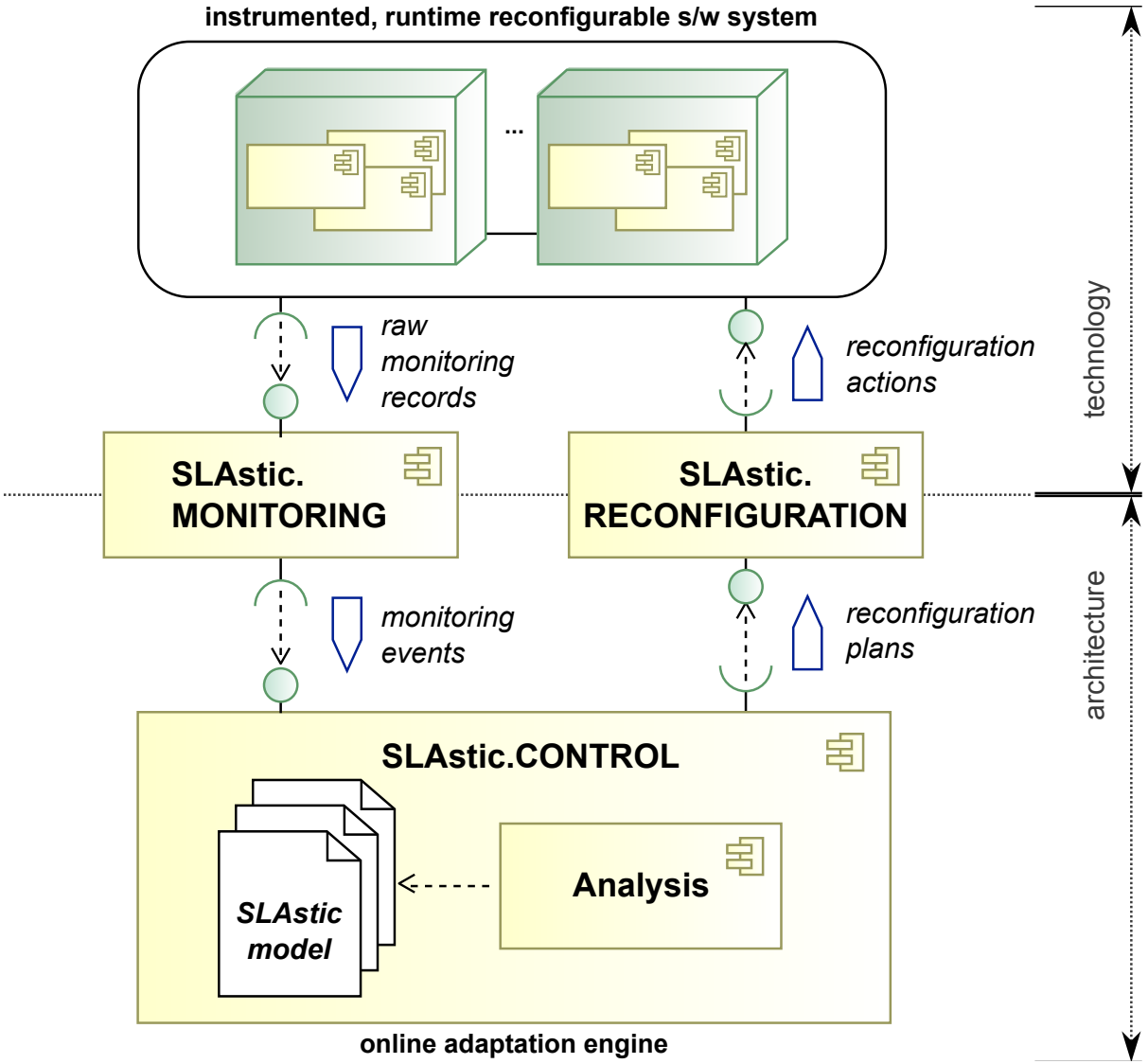
1. **SLAstic** — Model-Driven Online Capacity Management for C-B Software Systems
2. **Kieker** — Application Performance Management and Dynamic Software Analysis
3. **DynaMod** — Dynamic Analysis for Model-Driven Software Modernization

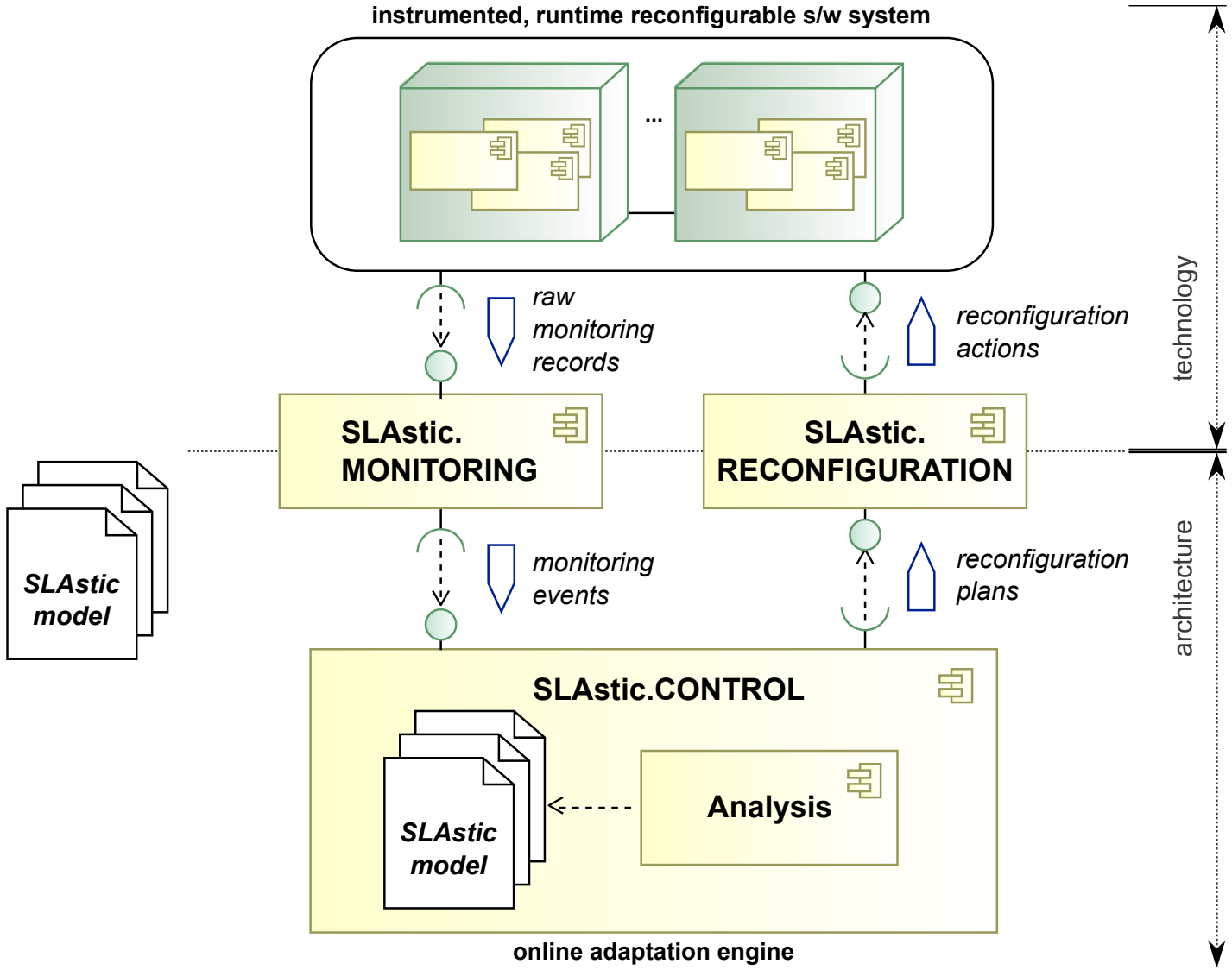
Architecture-based Online Capacity Management: Modeling, Monitoring, and Adaptation

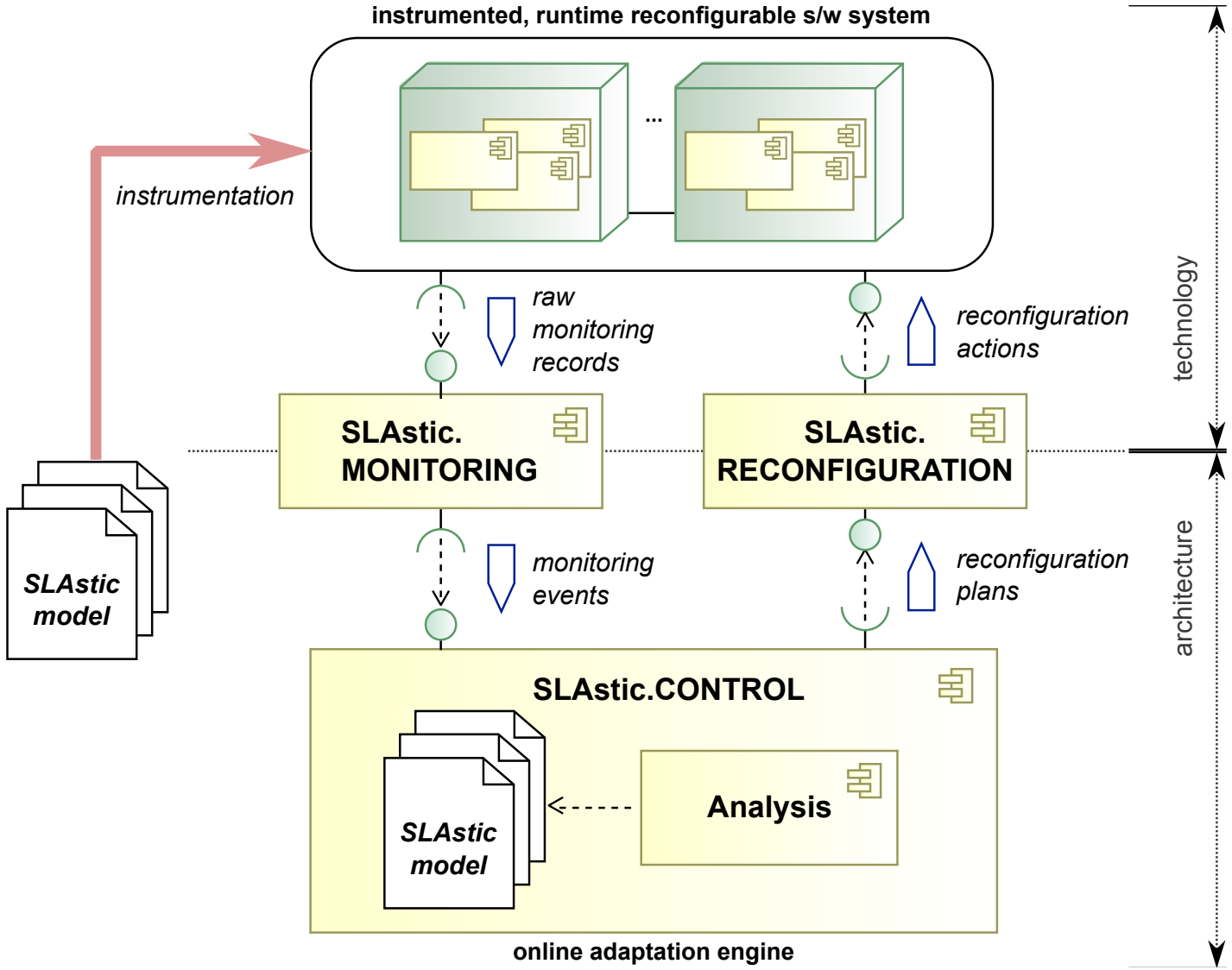

instrumented, runtime reconfigurable s/w system
...
technology
architecture


instrumented, runtime reconfigurable s/w system
...
raw
monitoring
records
SLAstic.
MONITORING
technology
architecture


instrumented, runtime reconfigurable s/w system
...
raw
monitoring
records
SLAstic.
MONITORING
monitoring
events
SLAstic.CONTROL
SLAstic
model
Analysis
online adaptation engine
technology
architecture

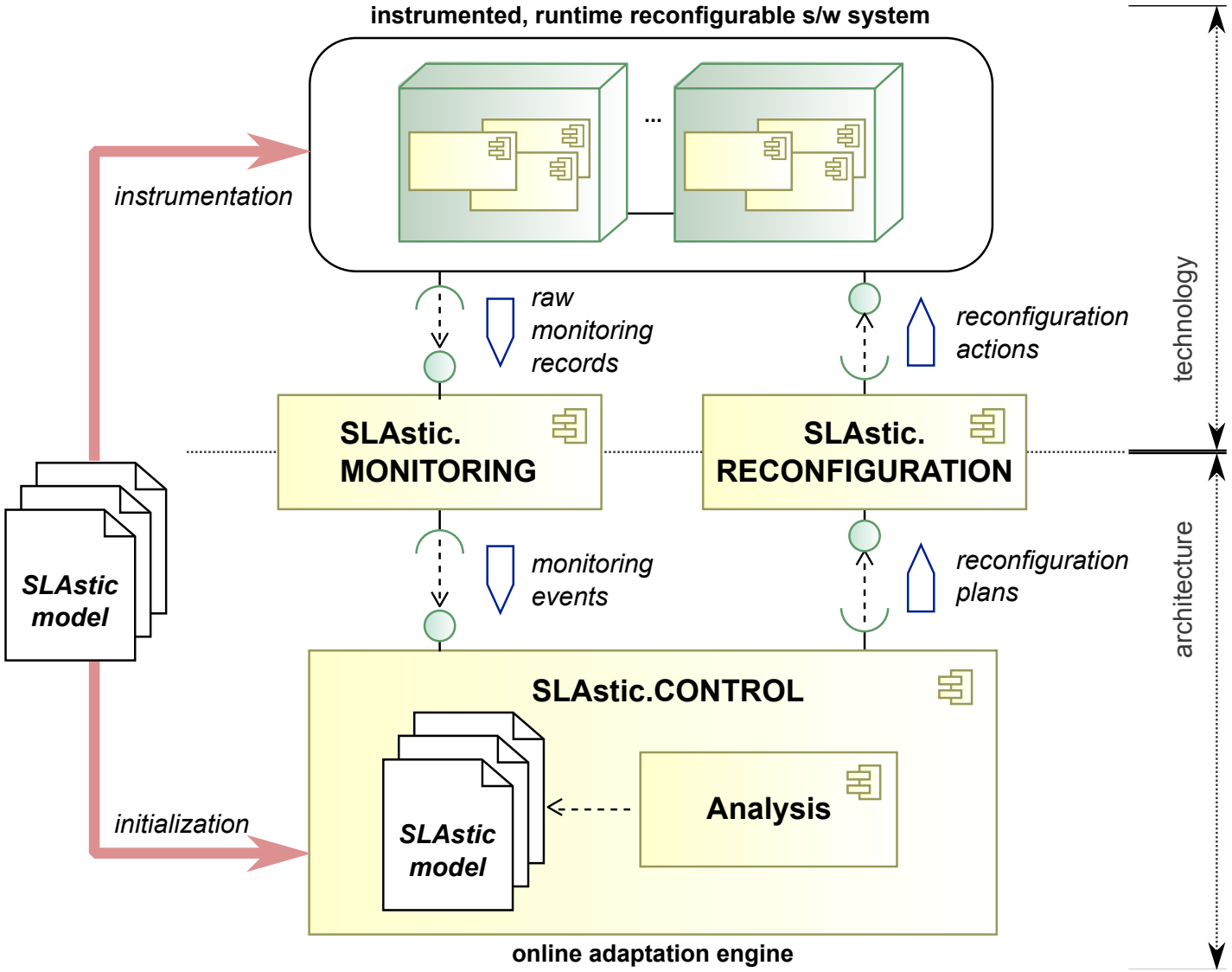

instrumented, runtime reconfigurable s/w system
...
raw monitoring records
reconfiguration actions
SLAstic. MONITORING
SLAstic. RECONFIGURATION
monitoring events
reconfiguration plans
SLAstic.CONTROL
SLAstic model
Analysis
online adaptation engine
technology
architecture


instrumented, runtime reconfigurable s/w system
...
raw monitoring records
reconfiguration actions
technology
SLAstic. MONITORING
SLAstic. RECONFIGURATION
SLAstic model
monitoring events
reconfiguration plans
architecture
SLAstic.CONTROL
SLAstic model
Analysis
online adaptation engine

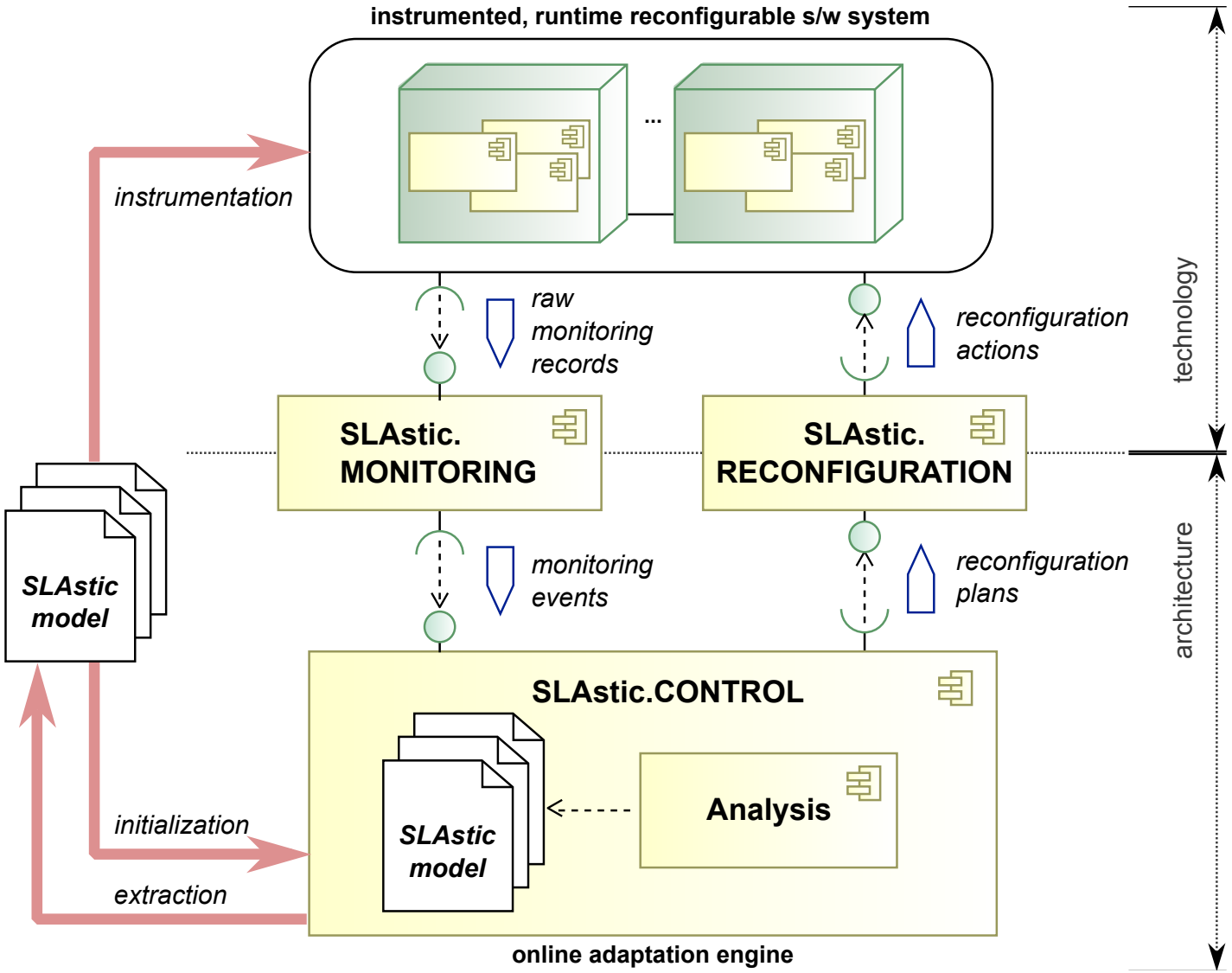

instrumented, runtime reconfigurable s/w system
instrumentation
raw monitoring records
reconfiguration actions
SLAstic. MONITORING
SLAstic. RECONFIGURATION
SLAstic model
monitoring events
reconfiguration plans
SLAstic.CONTROL
SLAstic model
Analysis
online adaptation engine
technology
architecture

Initialization of Framework & Runtime Models

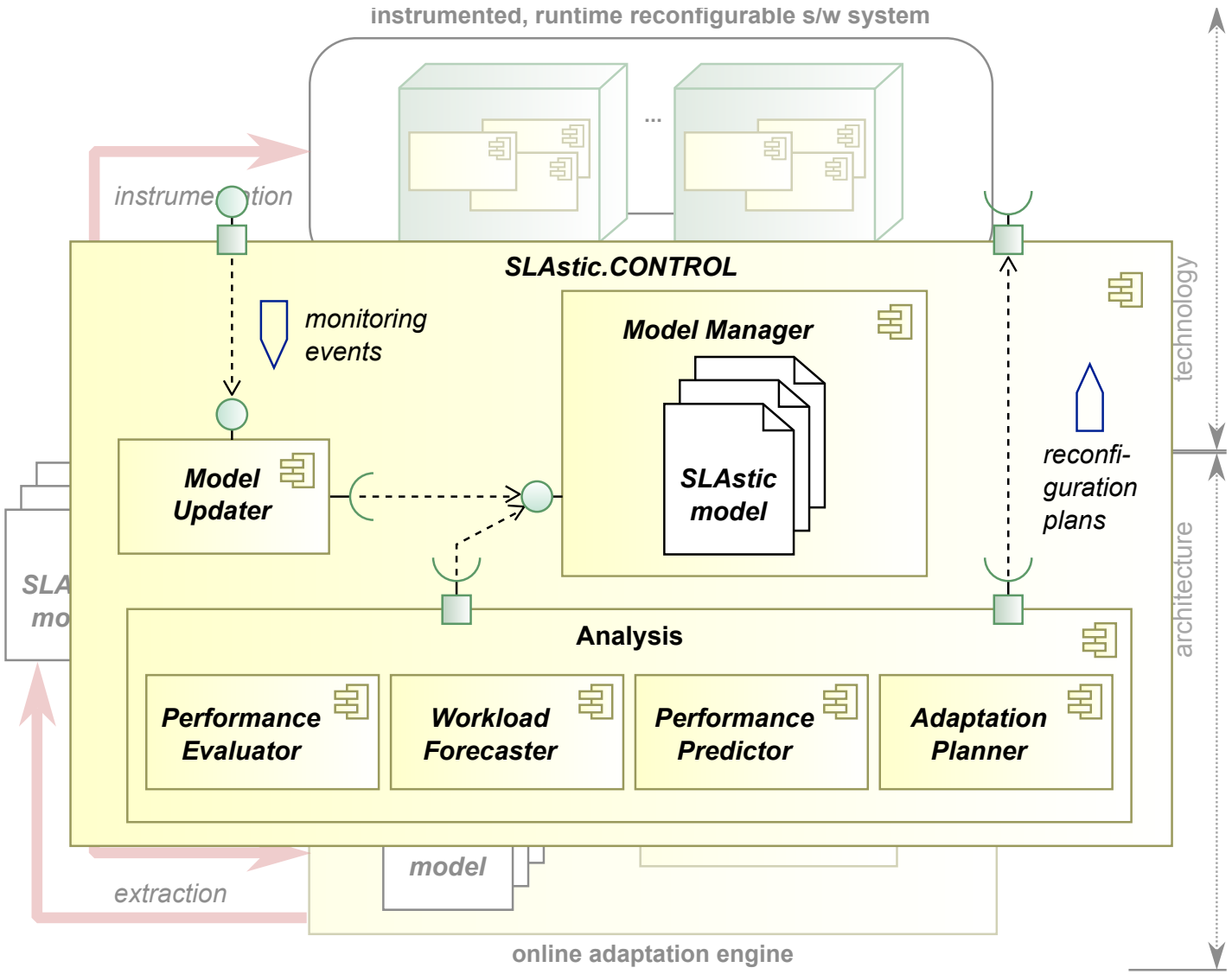
Christian-Albrechts-Universität zu Kiel



Extraction of Architectural Models (Dyn. Analysis)



SLAstic.CONTROL (zoom-in)

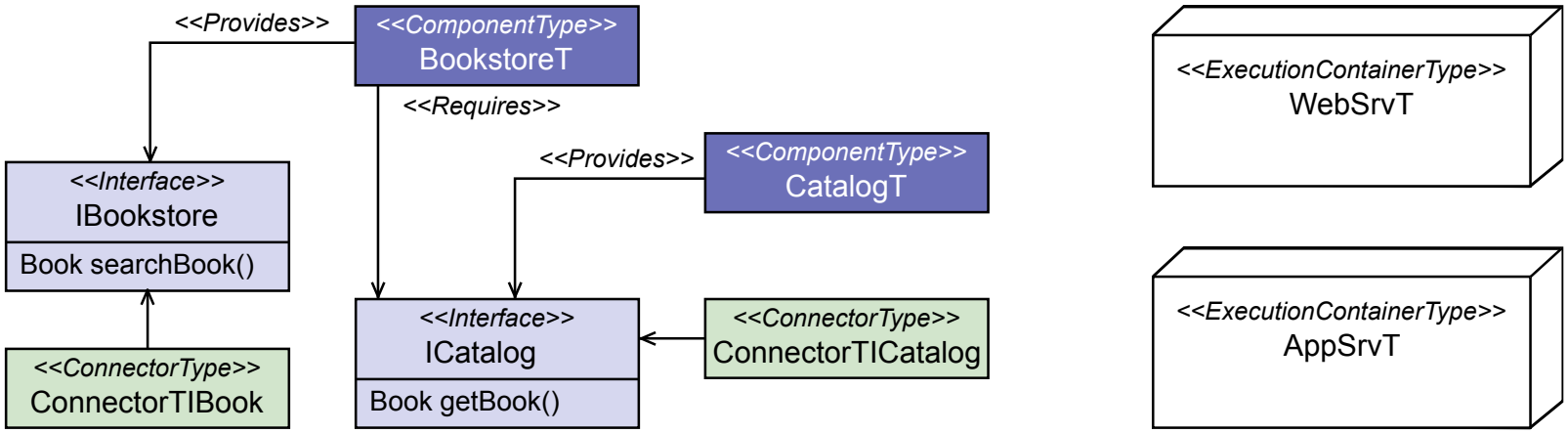


System Partition (also used as runtime model)

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 $\rightarrow$ containers)

System Partition (also used as runtime model)

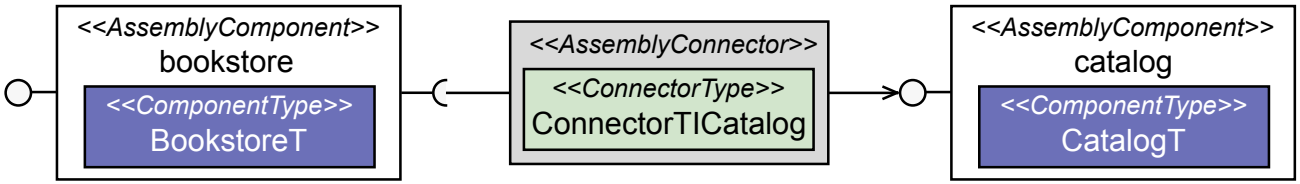
1. Type repository (e.g., component types, interfaces, connector types, execution container types)
2. Component assembly (e.g., assembly of components via connectors)
3. Execution environment (e.g., execution containers and interconnection via links)
4. Component deployment (mapping: assembly components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



Example type repository

System Partition (also used as runtime model)

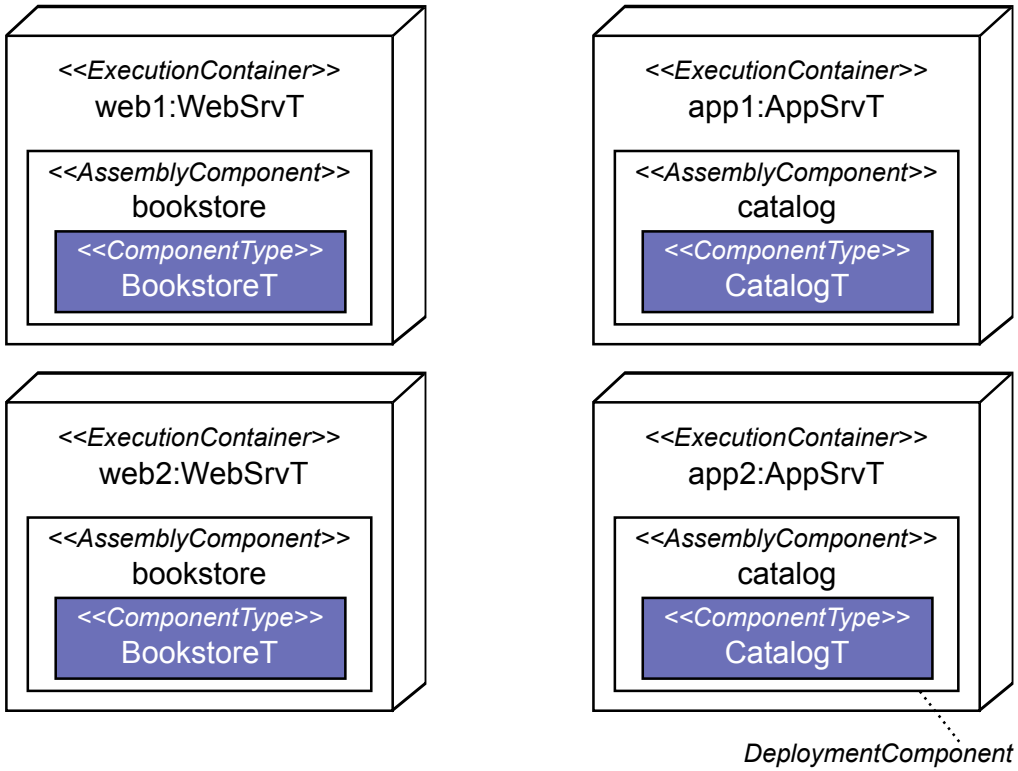
1. Type repository (e.g., component types, interfaces, connector types, execution container types)
2. Component assembly (e.g., assembly of components via connectors)
3. Execution environment (e.g., execution containers and interconnection via links)
4. Component deployment (mapping: assembly components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



Example component assembly

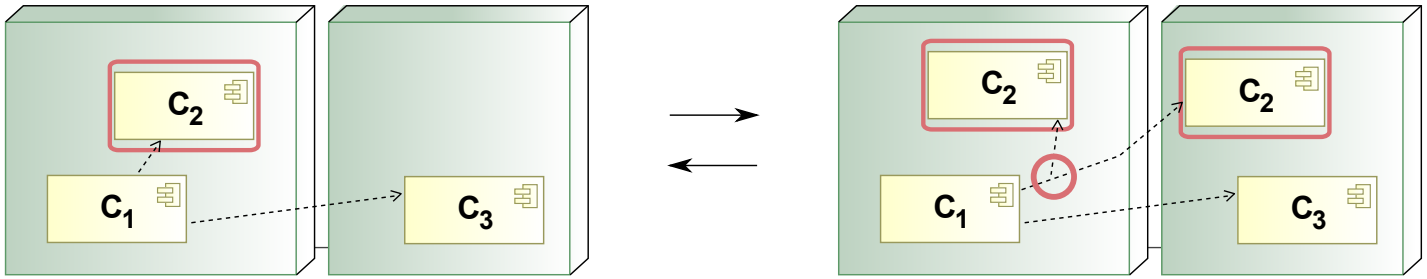
System Partition (also used as runtime model)

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

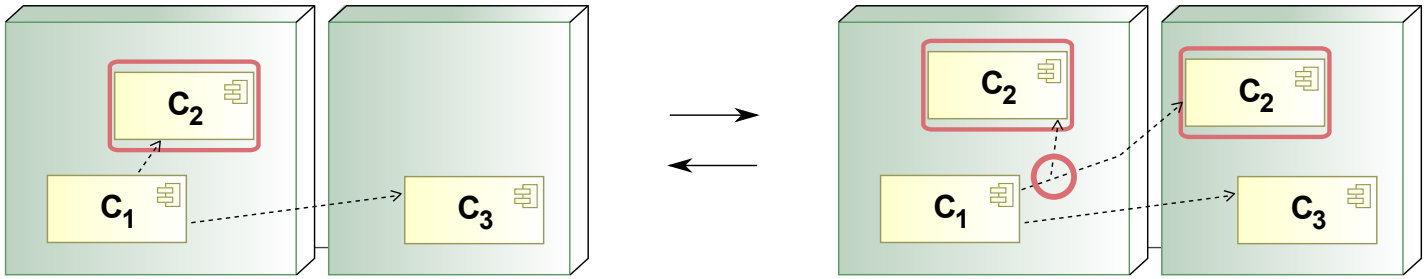
1 (De-)Replication of Software Components



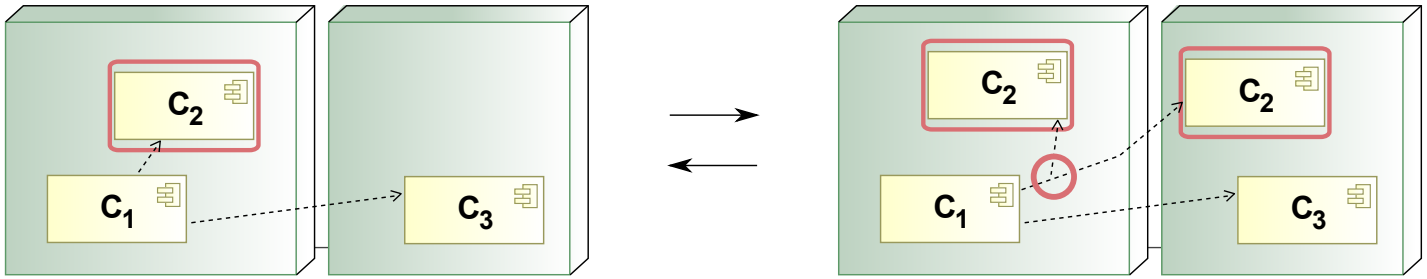
1 (De-)Replication of Software Components

- replicate (component: AssemblyComponent, to: ExecutionContainer)
- dereplicate (component: DeploymentComponent)

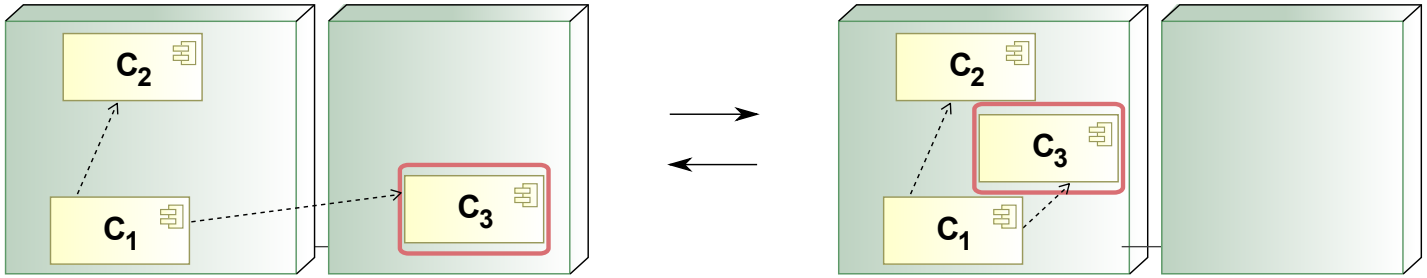
1 (De-)Replication of Software Components



1 (De-)Replication of Software Components



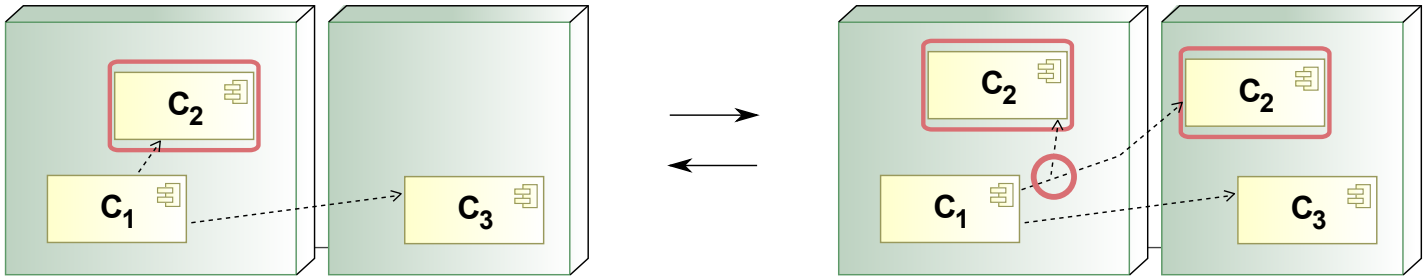
2 Migration of Software Components



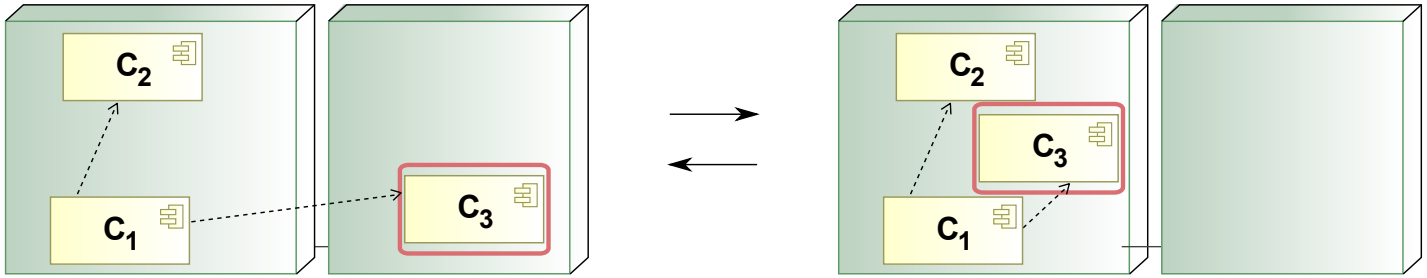
SLAstic Runtime Reconfiguration Operations

Christian-Albrechts-Universität zu Kiel

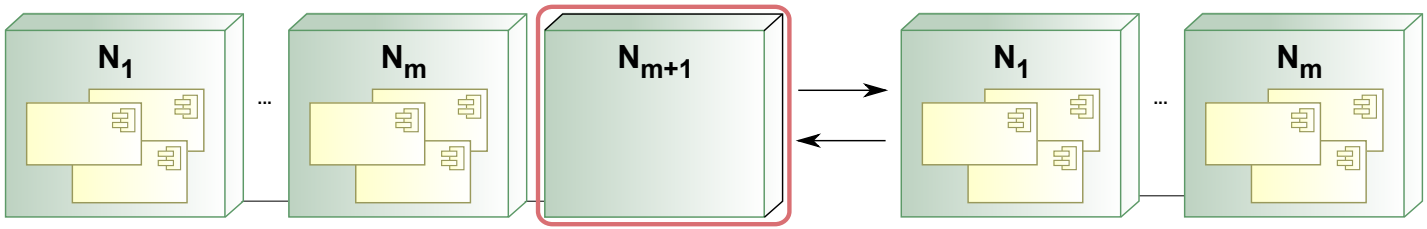
1 (De-)Replication of Software Components



2 Migration of Software Components



3 (De-)Allocation of Execution Containers

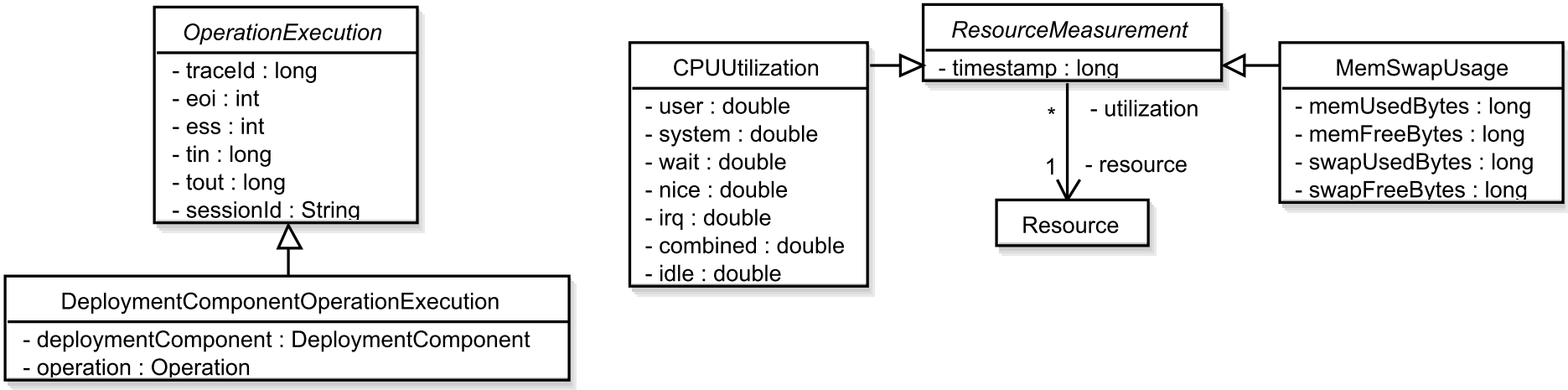


Completions/Decorations

- Adaptation / Reconfiguration (e.g., plans, operations, capabilities, properties)
- Measurement (e.g., workload, timing, utilization)
- Usage (e.g., operation call frequencies, calling relationships)
- . . .

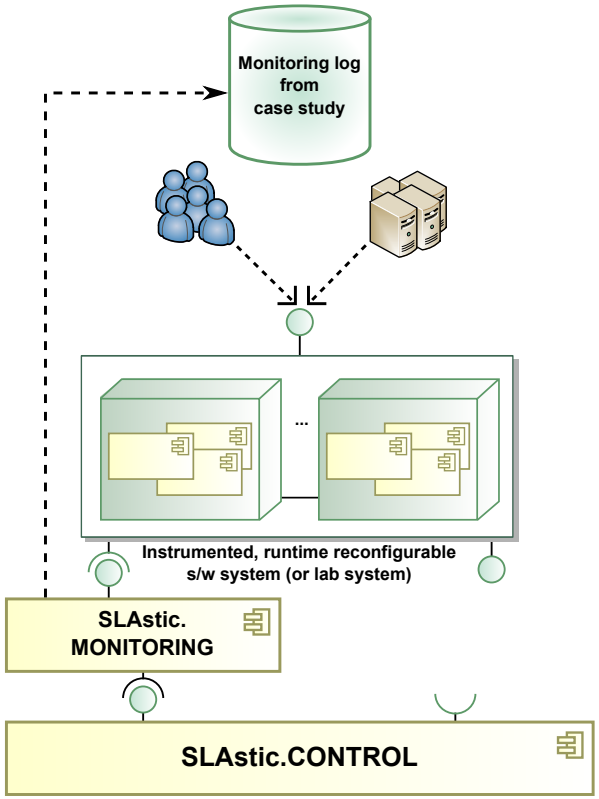
Completions/Decorations

- Adaptation / Reconfiguration (e.g., plans, operations, capabilities, properties)
- Measurement (e.g., workload, timing, utilization)
- Usage (e.g., operation call frequencies, calling relationships)
- . . .



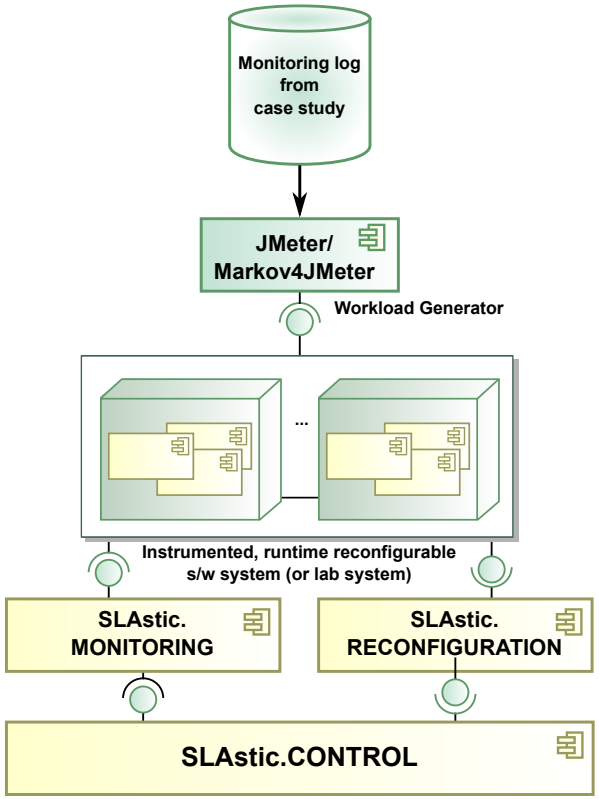
Measurement event types

Evaluation Methodology



1. Online analysis (production/lab system)

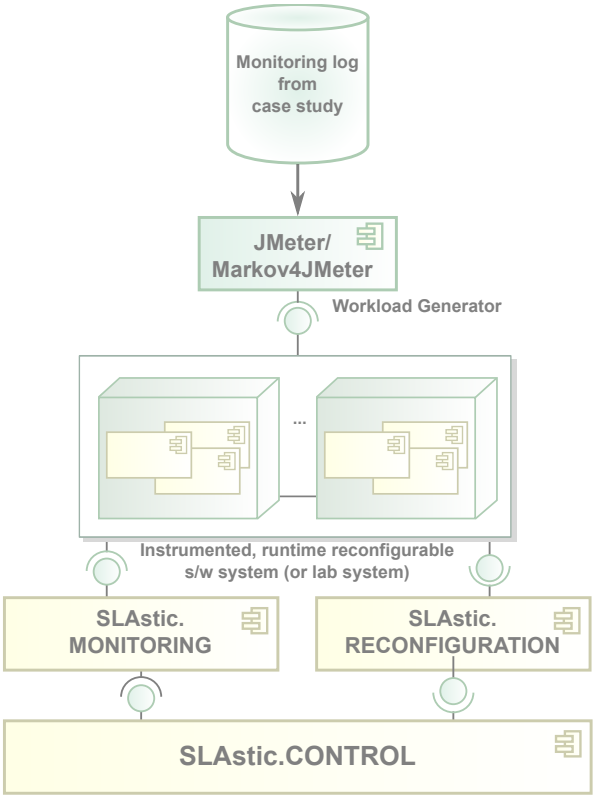
Evaluation Methodology



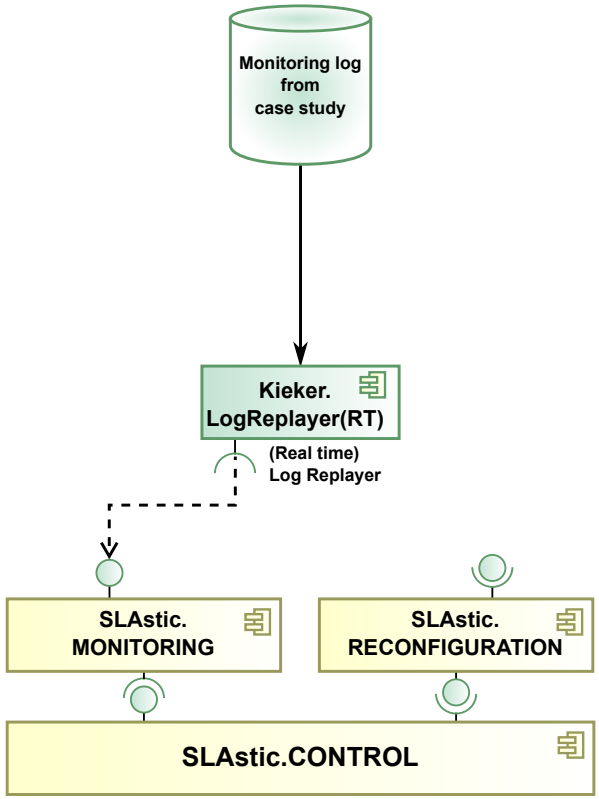
1. Online analysis (production/lab system)

Evaluation Methodology

Christian-Albrechts-Universität zu Kiel



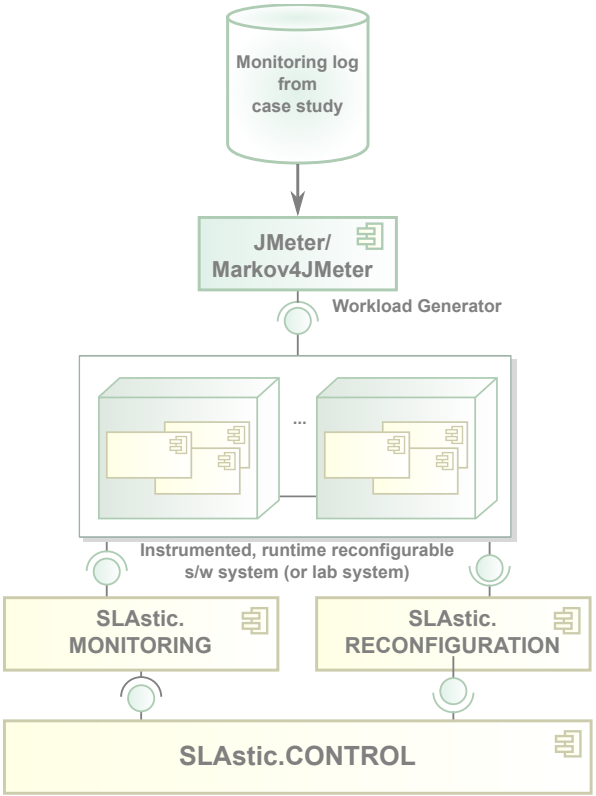
1 Online analysis (production/lab system)



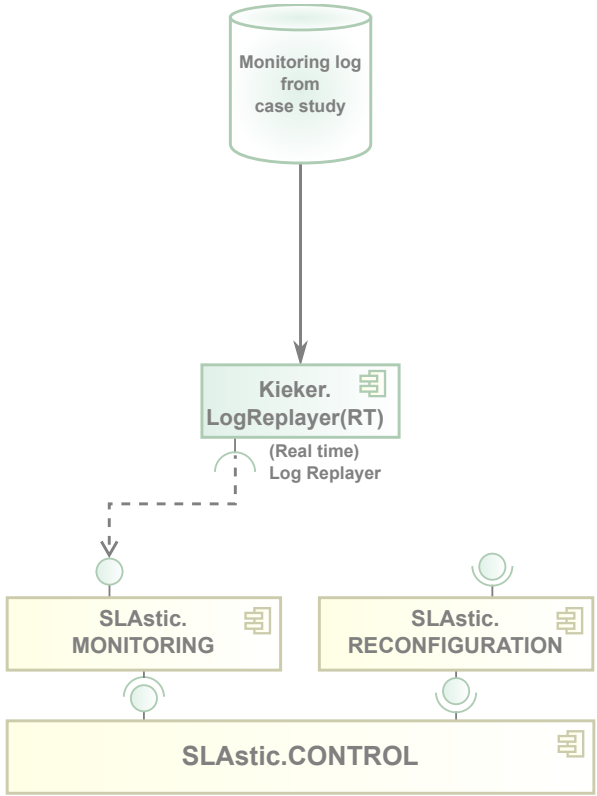
2 Offline analysis (log replay)

Evaluation Methodology

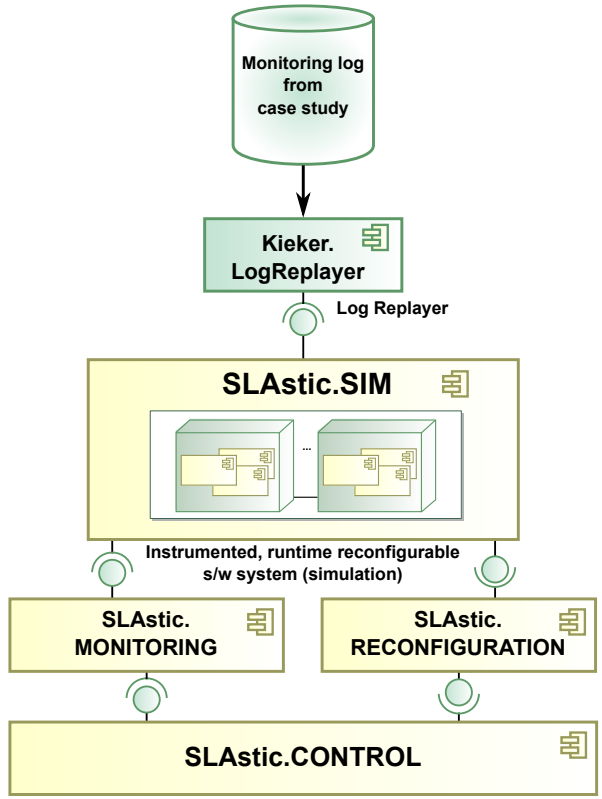
Christian-Albrechts-Universität zu Kiel



1 Online analysis (production/lab system)



2 Offline analysis (log replay)



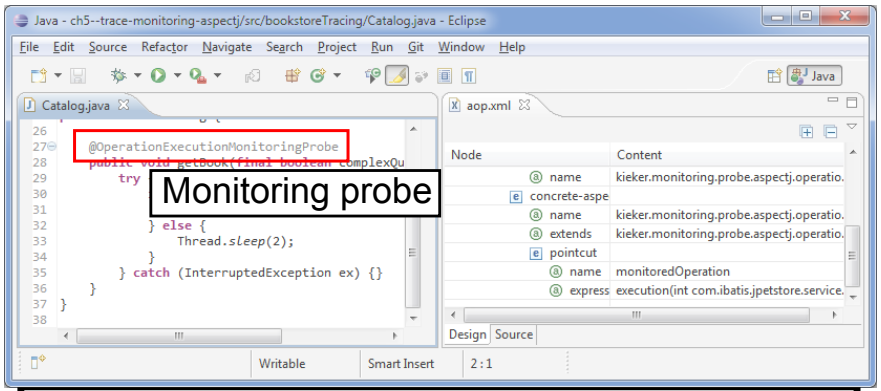
3 Simulation-based analysis

Agenda

Kieker: Example Workflow and Use Cases

[van Hoorn et al. 2012]

Christian-Albrechts-Universität zu Kiel

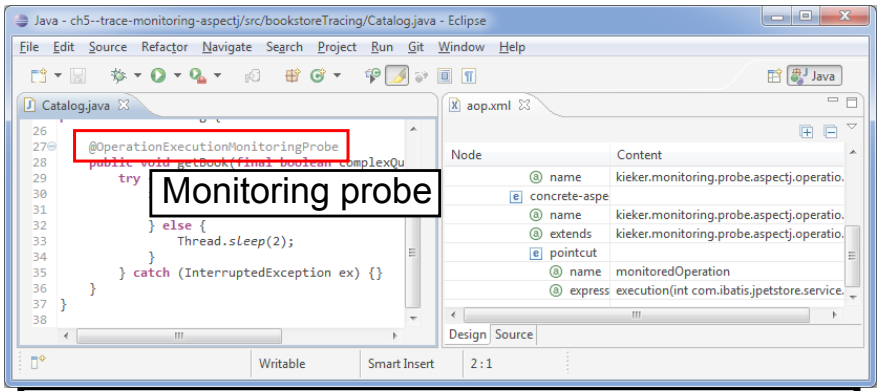


Software system with monitoring instrumentation

Kieker: Example Workflow and Use Cases

[van Hoorn et al. 2012]

Christian-Albrechts-Universität zu Kiel



Software system with monitoring instrumentation

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>		

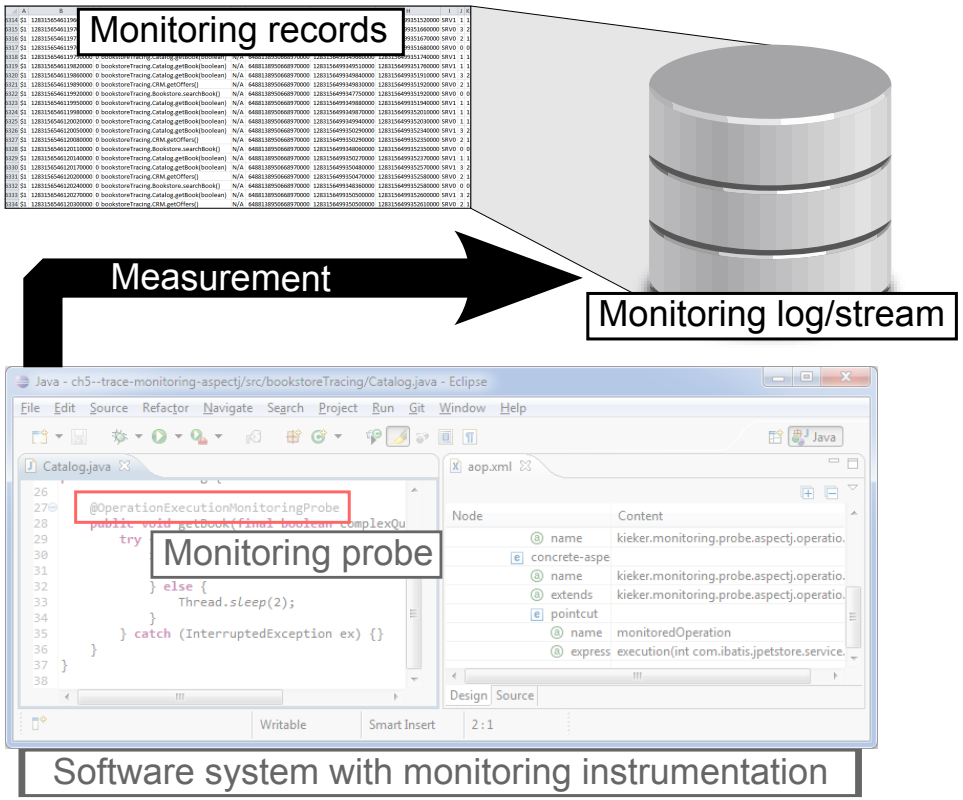
+ basic adapters for

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

Kieker: Example Workflow and Use Cases

[van Hoorn et al. 2012]

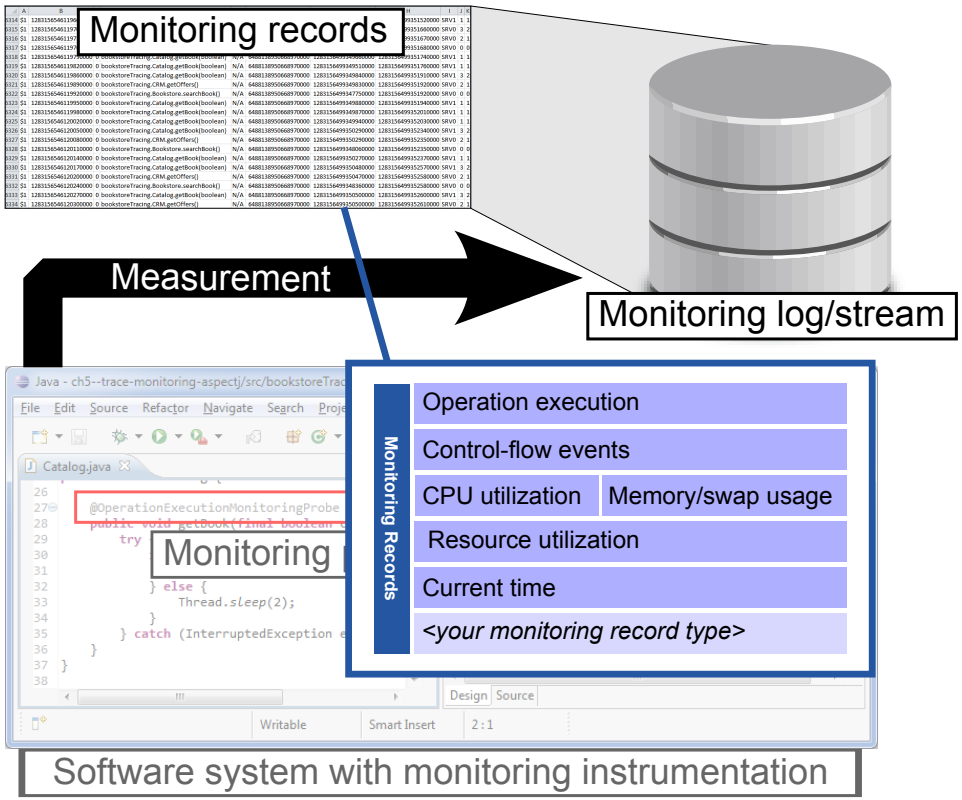
Christian-Albrechts-Universität zu Kiel



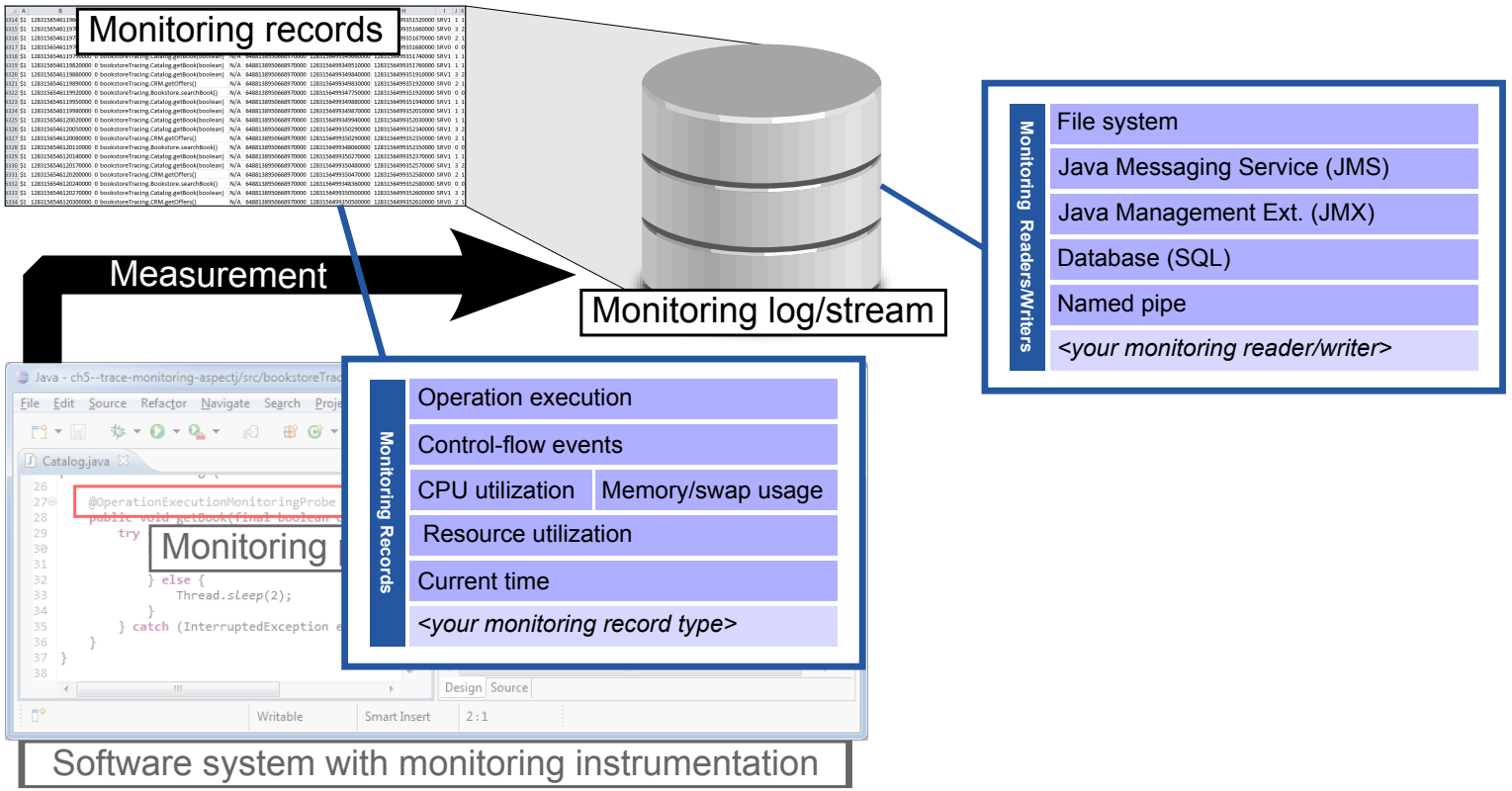
Kieker: Example Workflow and Use Cases

[van Hoorn et al. 2012]

Christian-Albrechts-Universität zu Kiel



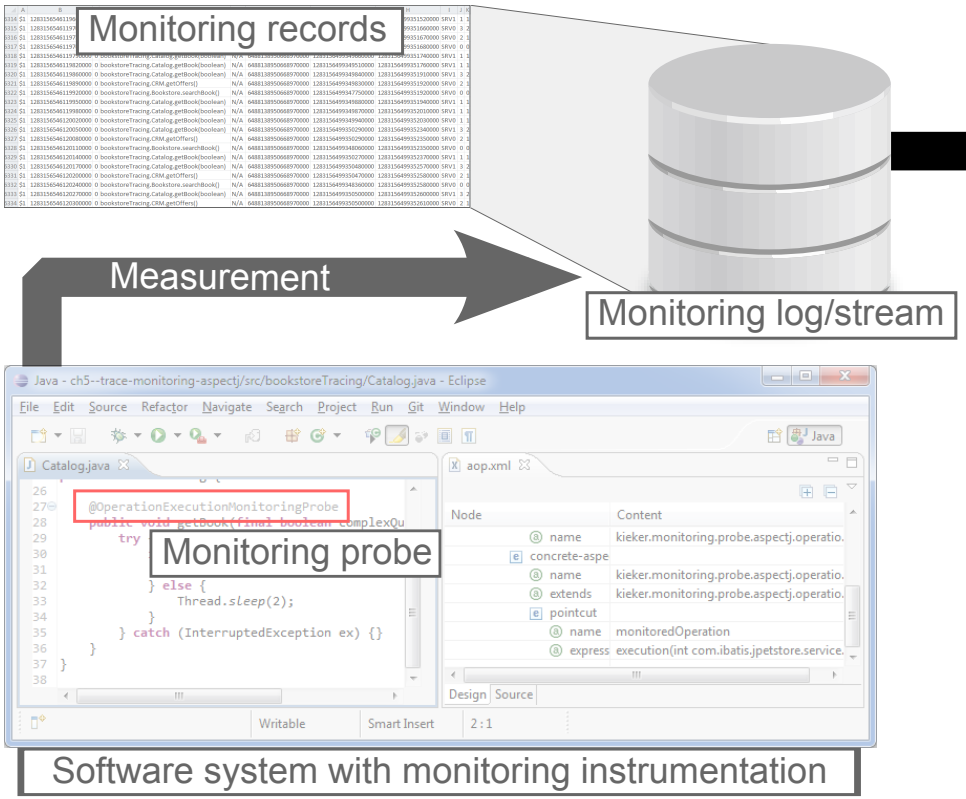
Christian-Albrechts-Universität zu Kiel

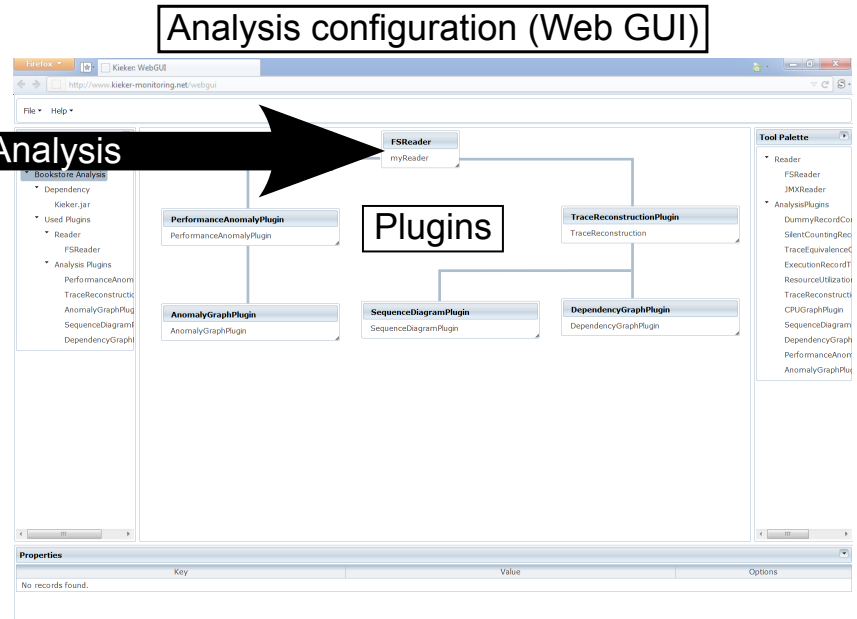


Kieker: Example Workflow and Use Cases

[van Hoorn et al. 2012]

Christian-Albrechts-Universität zu Kiel

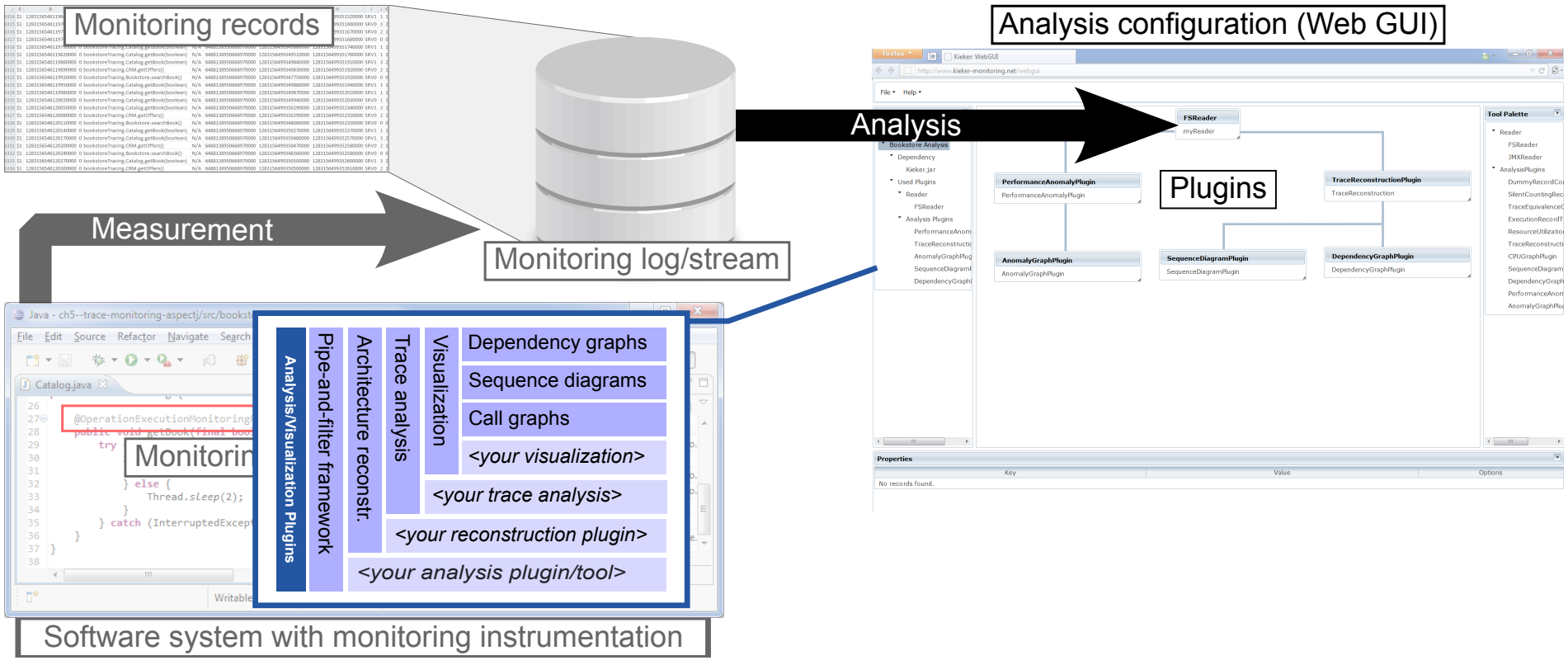




Kieker: Example Workflow and Use Cases

[van Hoorn et al. 2012]

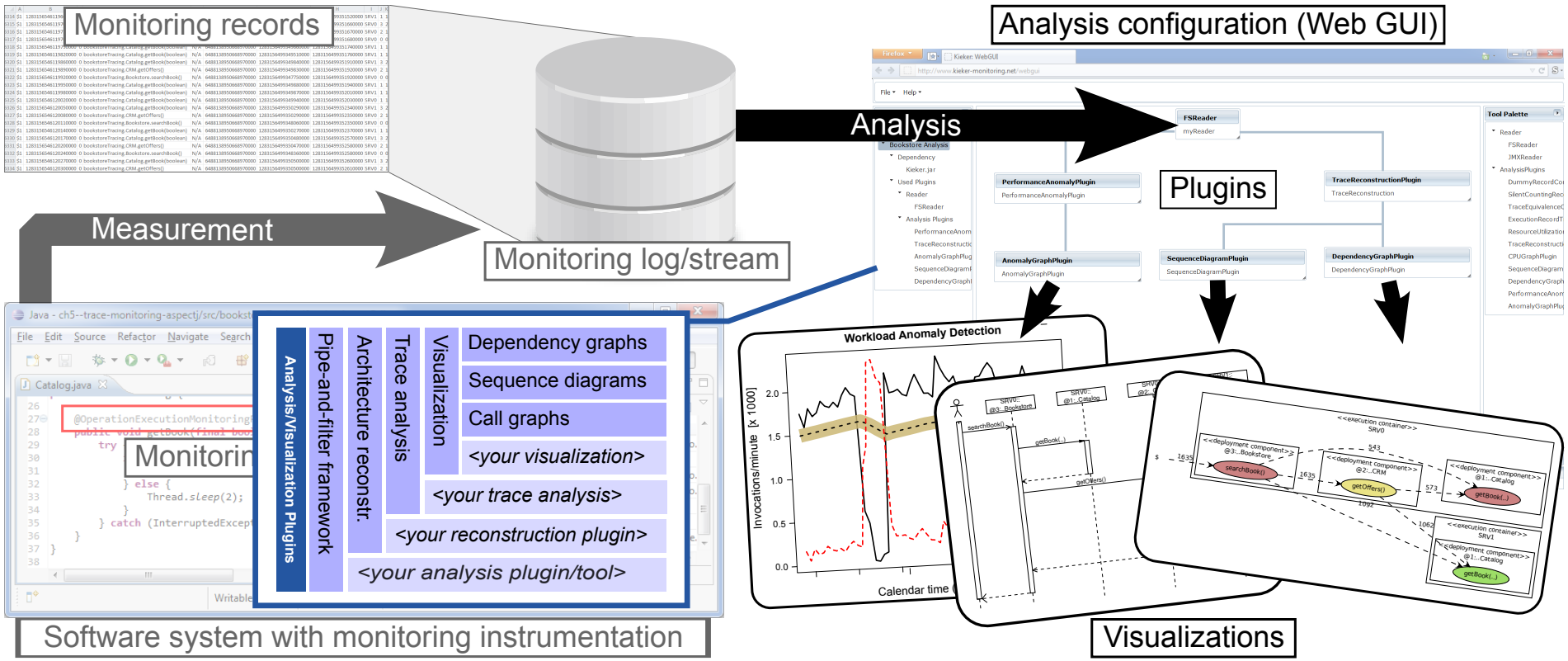
Christian-Albrechts-Universität zu Kiel

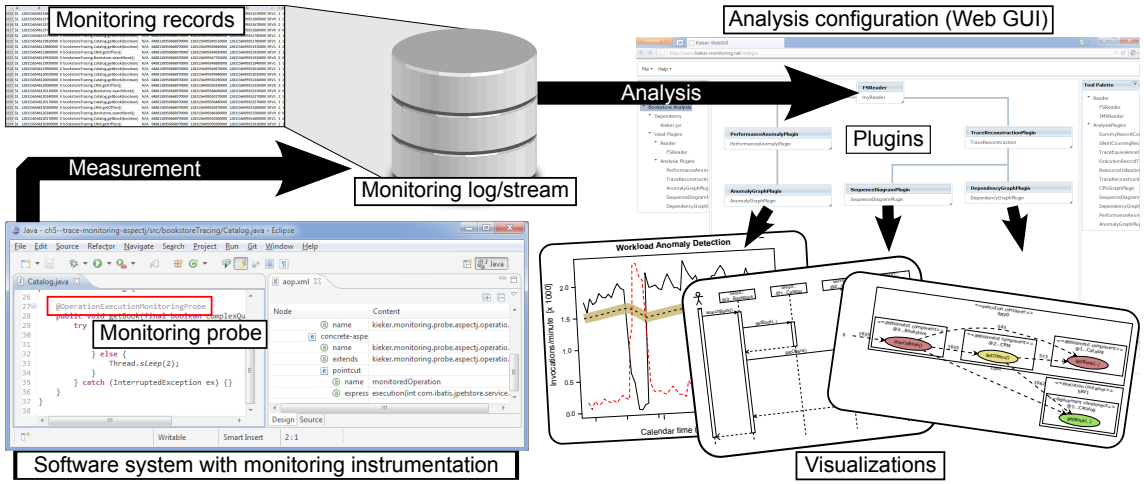


Kieker: Example Workflow and Use Cases

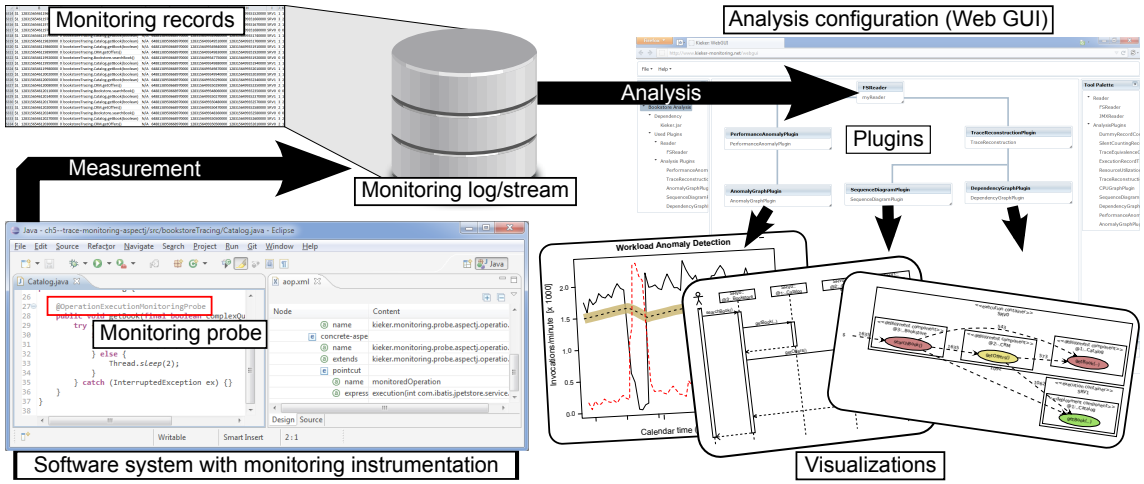
[van Hoorn et al. 2012]

Christian-Albrechts-Universität zu Kiel



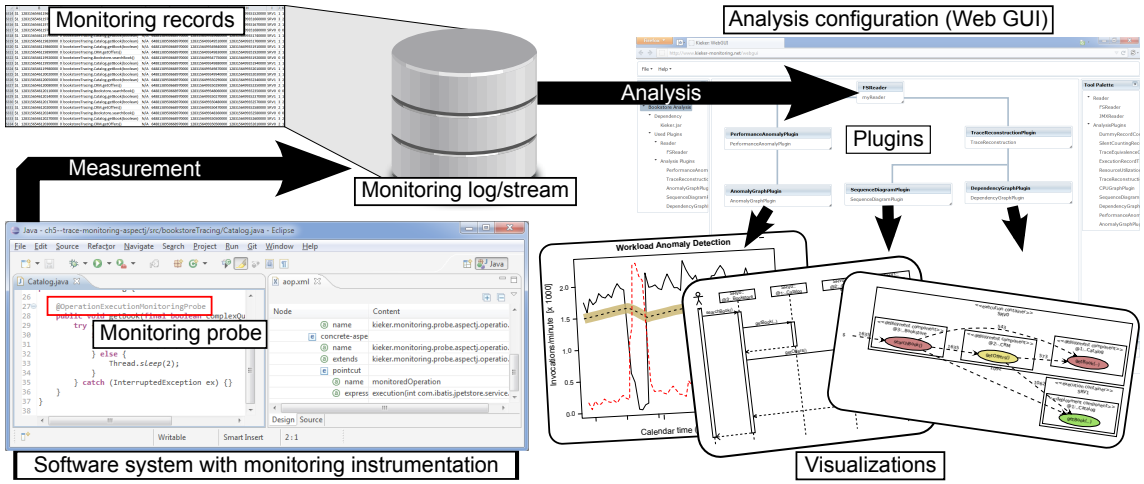


Selected use cases in research and industrial practice (+ teaching):



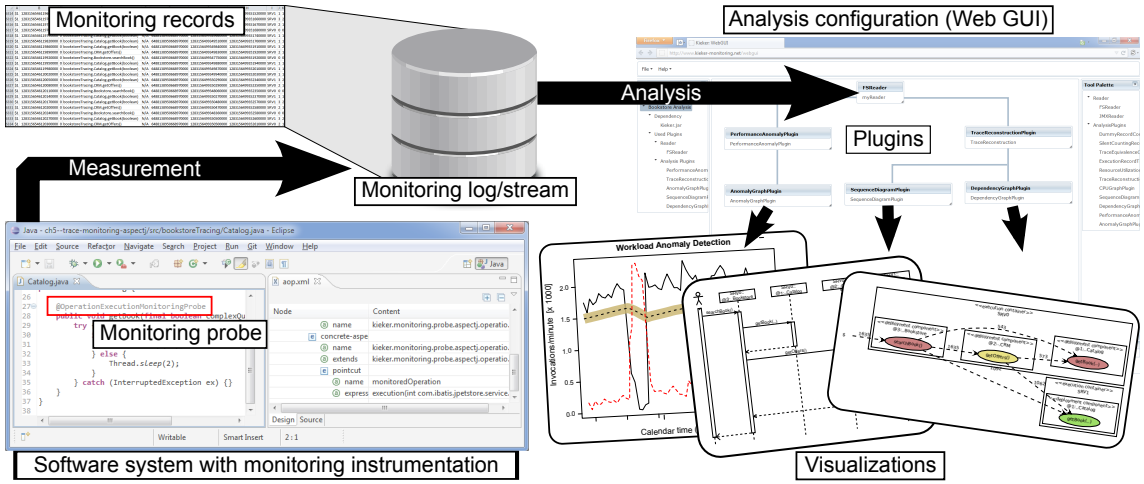
Selected use cases in research and industrial practice (+ teaching):

- Online/offline performance evaluation and feedback, e.g.,
  - Continuous monitoring of application behavior and usage
  - Performance anomaly detection and diagnosis
  - (Self-)adaptation control



Selected use cases in research and industrial practice (+ teaching):

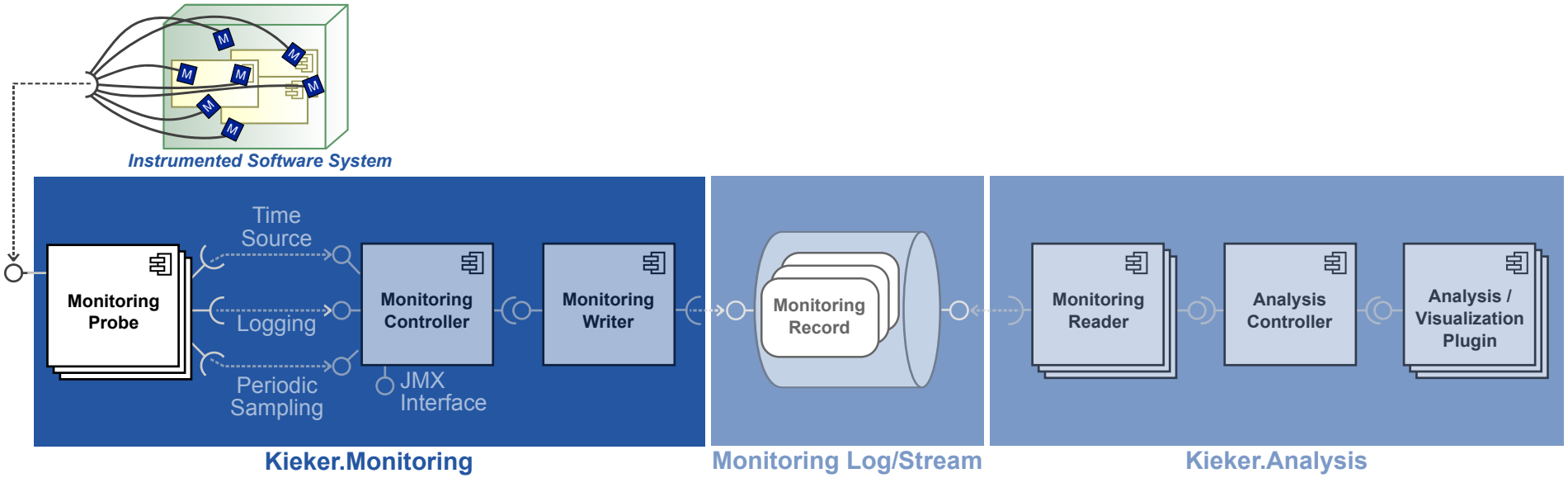
- Online/offline performance evaluation and feedback, e.g.,
  - Continuous monitoring of application behavior and usage
  - Performance anomaly detection and diagnosis
  - (Self-)adaptation control
- Extraction of software architectural (performance) models and visualizations



Selected use cases in research and industrial practice (+ teaching):

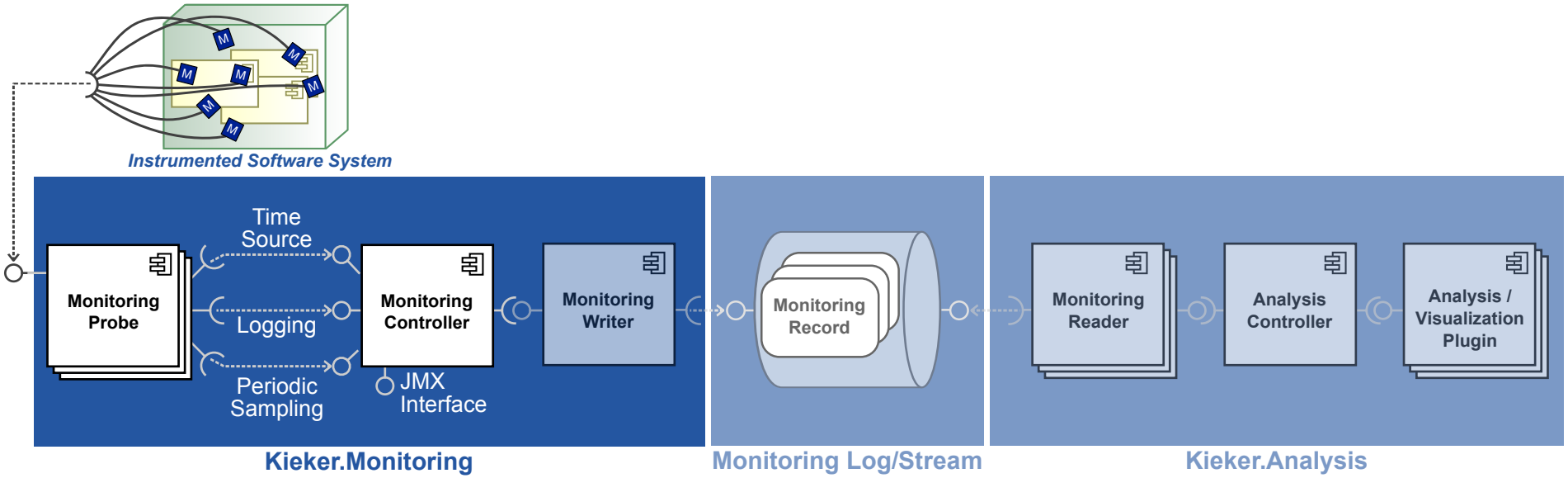
- Online/offline performance evaluation and feedback, e.g.,
  - Continuous monitoring of application behavior and usage
  - Performance anomaly detection and diagnosis
  - (Self-)adaptation control
- Extraction of software architectural (performance) models and visualizations
- Simulation (replaying previously monitored stimuli; measurement, logging, and analysis)

Core Kieker Framework Components



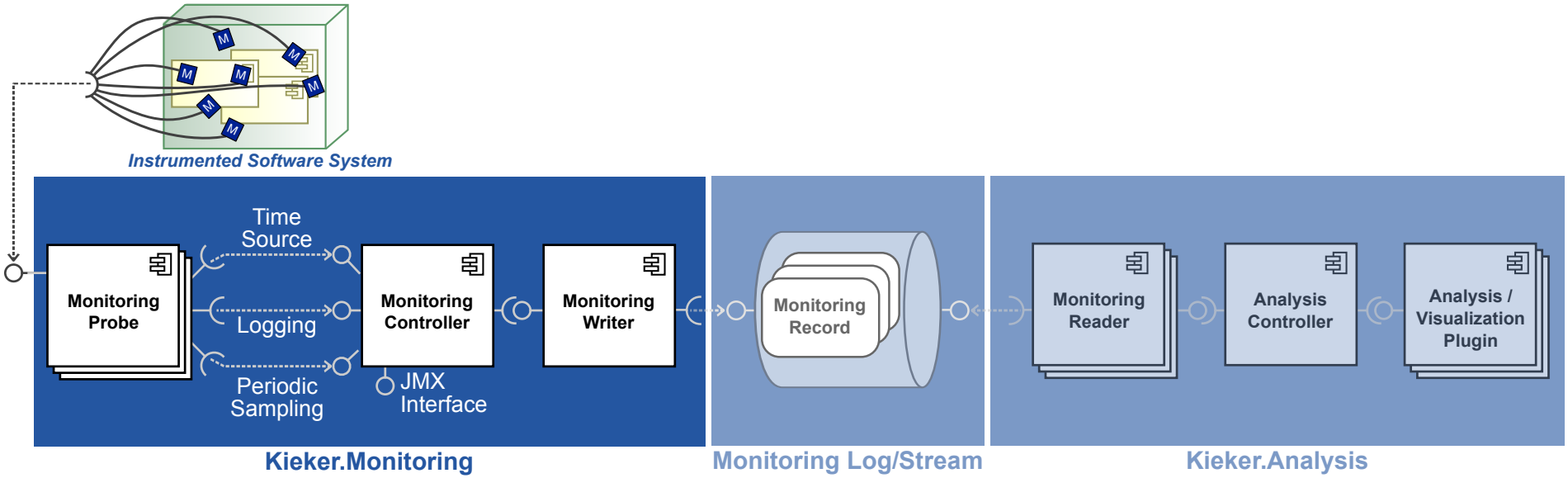
Core Kieker Framework Components

Christian-Albrechts-Universität zu Kiel



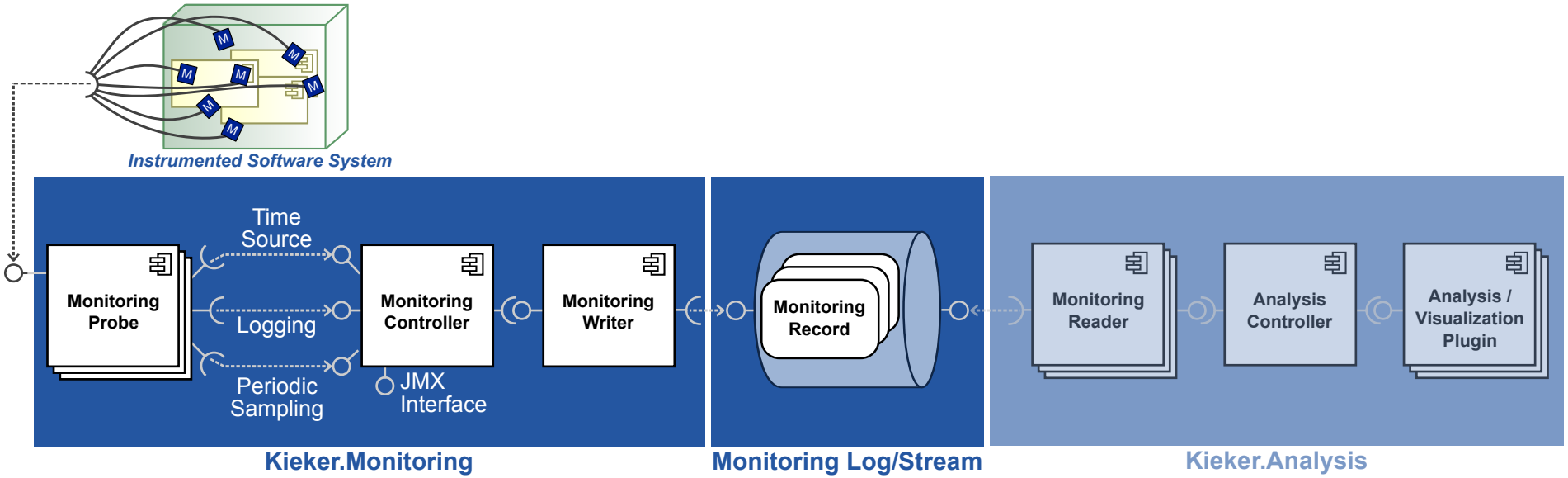
Core Kieker Framework Components

Christian-Albrechts-Universität zu Kiel

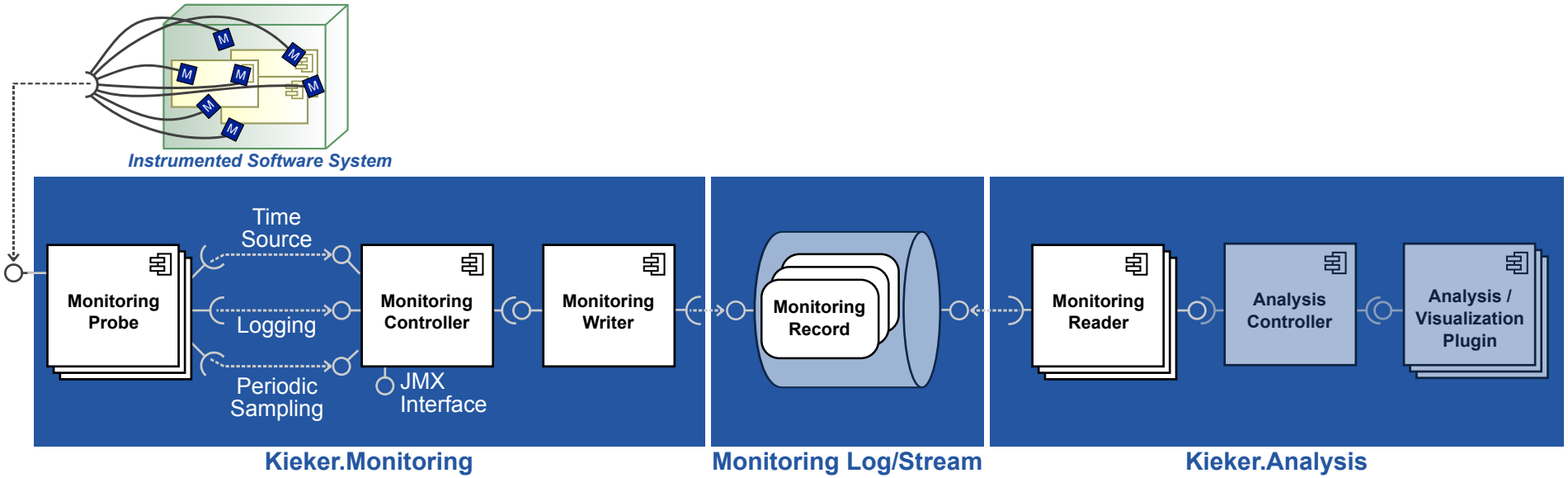


Core Kieker Framework Components

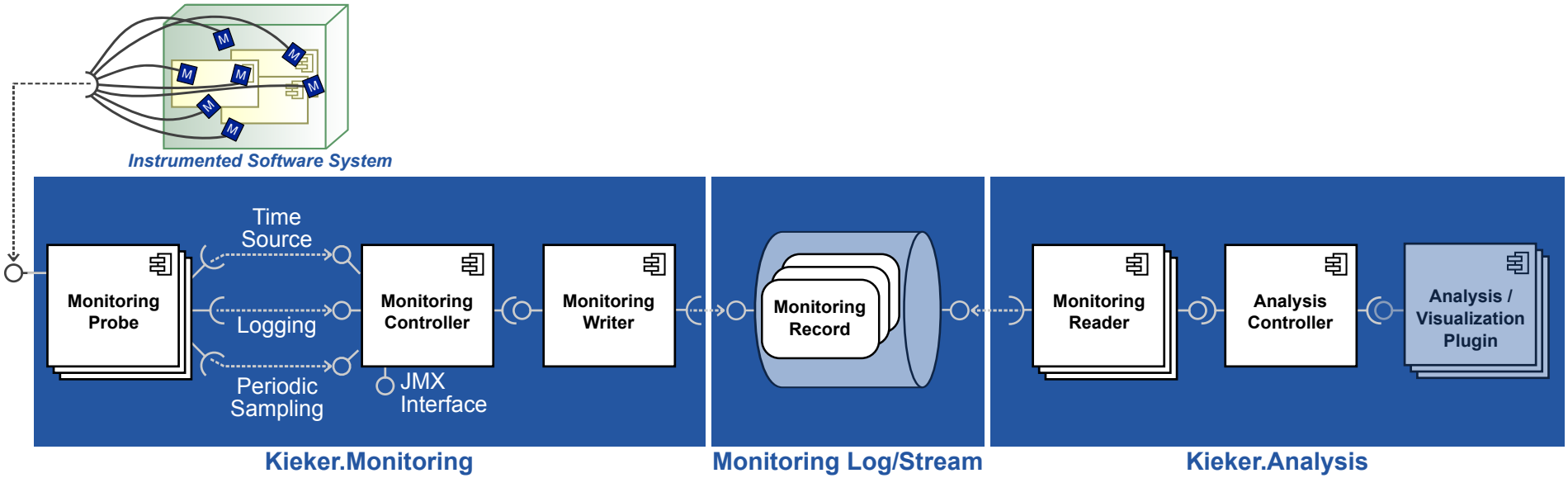
Christian-Albrechts-Universität zu Kiel



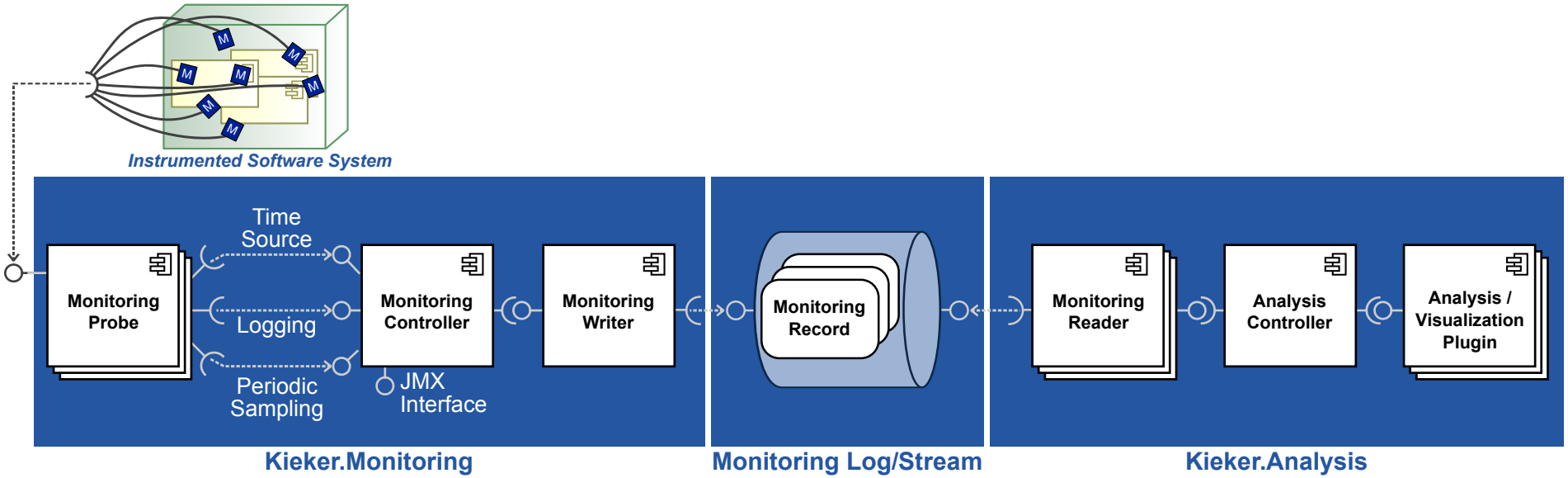
Core Kieker Framework Components



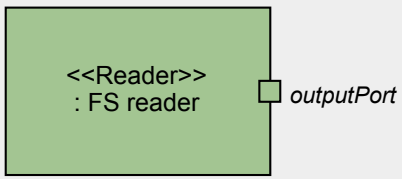
Core Kieker Framework Components



Core Kieker Framework Components



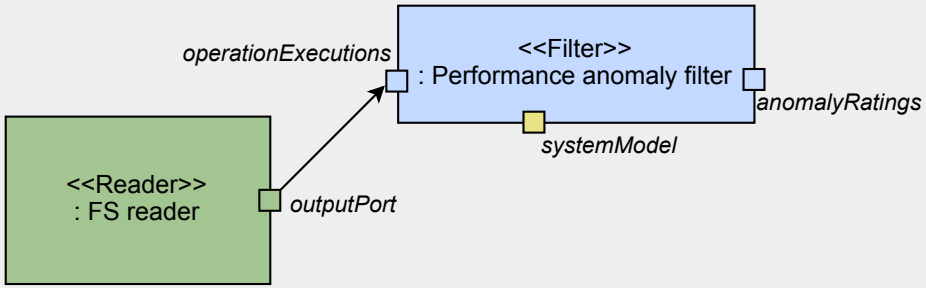
Example Pipe-and-Filter Configuration



Kieker.Analysis example pipes-and-filters configuration

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

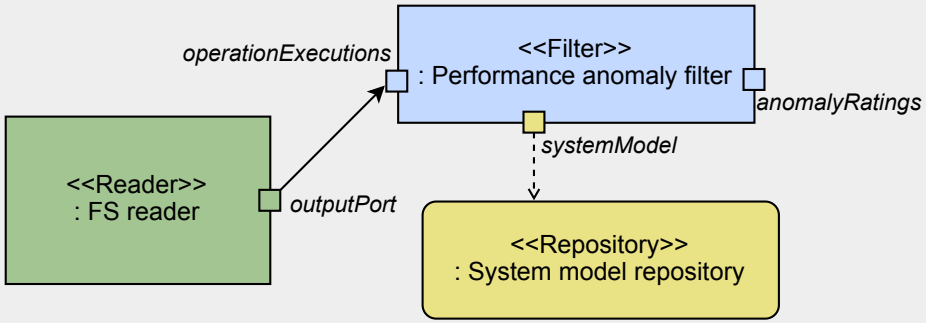
Example Pipe-and-Filter Configuration



Kieker.Analysis example pipes-and-filters configuration

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

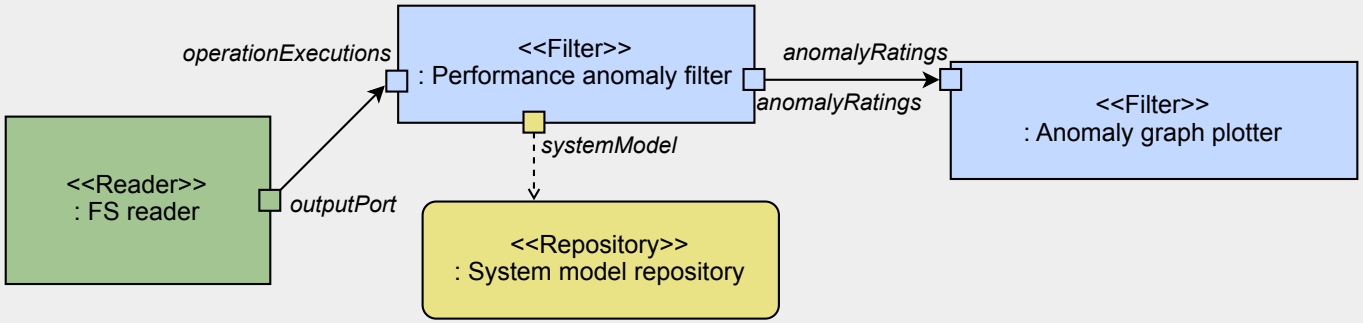
Example Pipe-and-Filter Configuration



Kieker.Analysis example pipes-and-filters configuration

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

Example Pipe-and-Filter Configuration

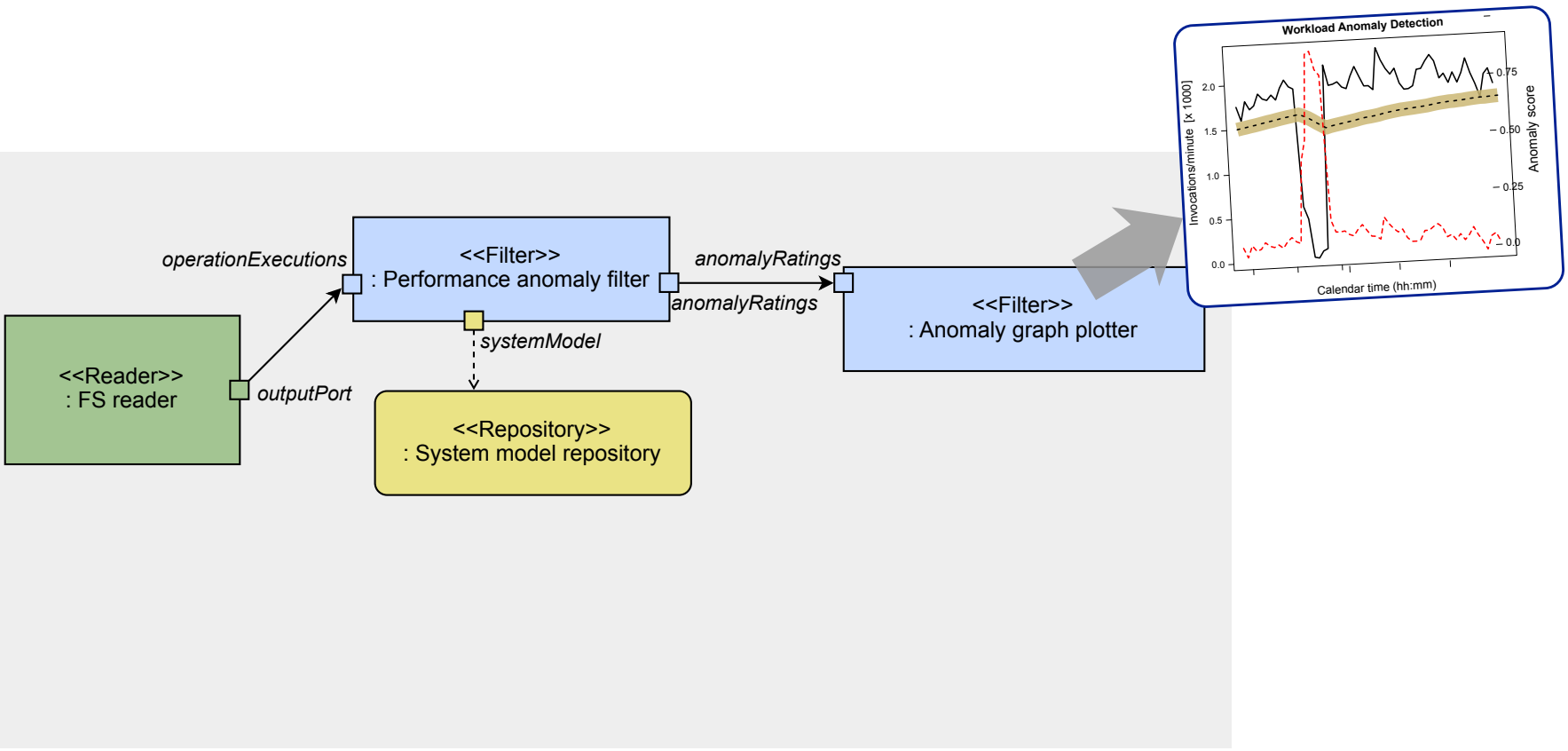


Kieker.Analysis example pipes-and-filters configuration

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

Example Pipe-and-Filter Configuration

Christian-Albrechts-Universität zu Kiel

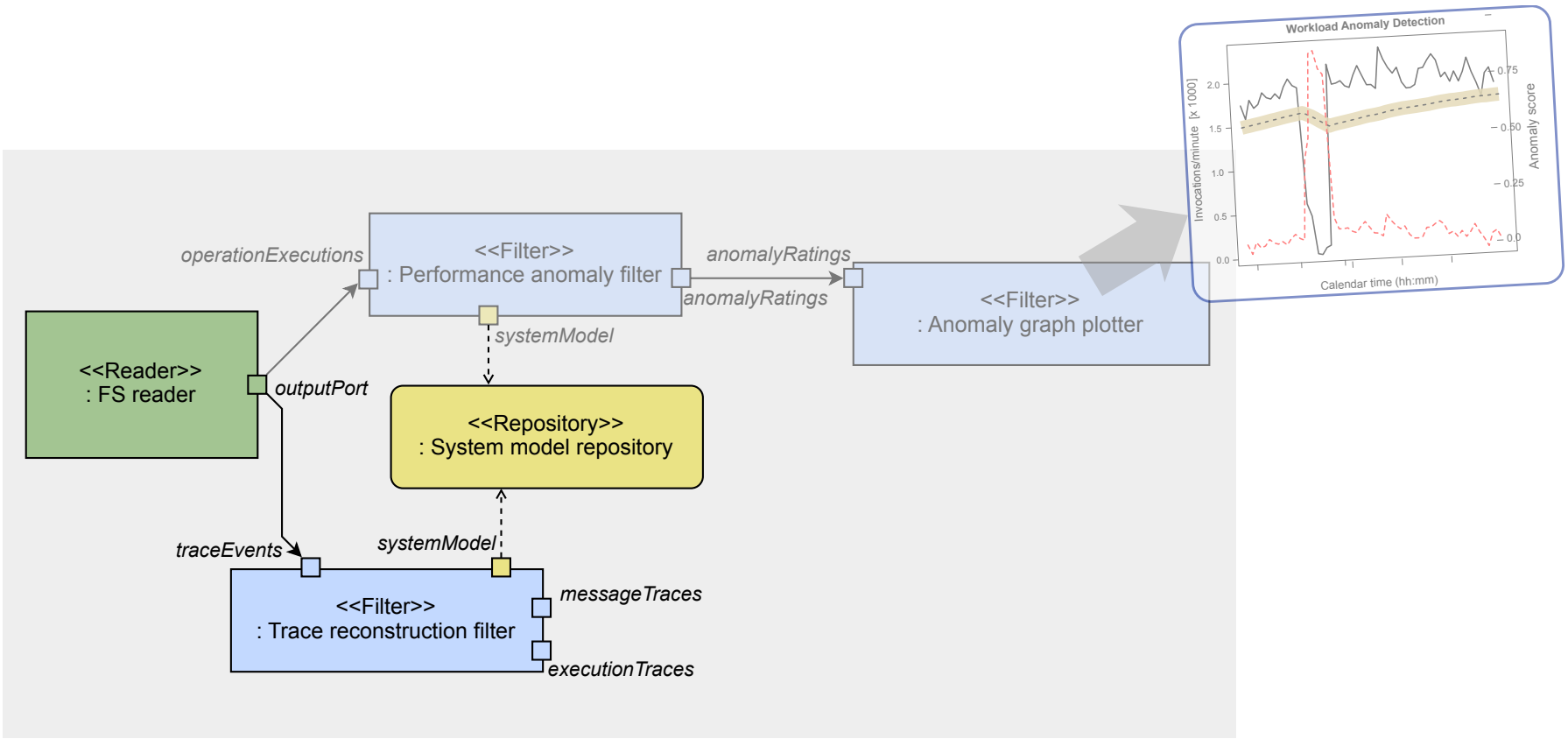


Kieker.Analysis example pipes-and-filters configuration

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

Example Pipe-and-Filter Configuration

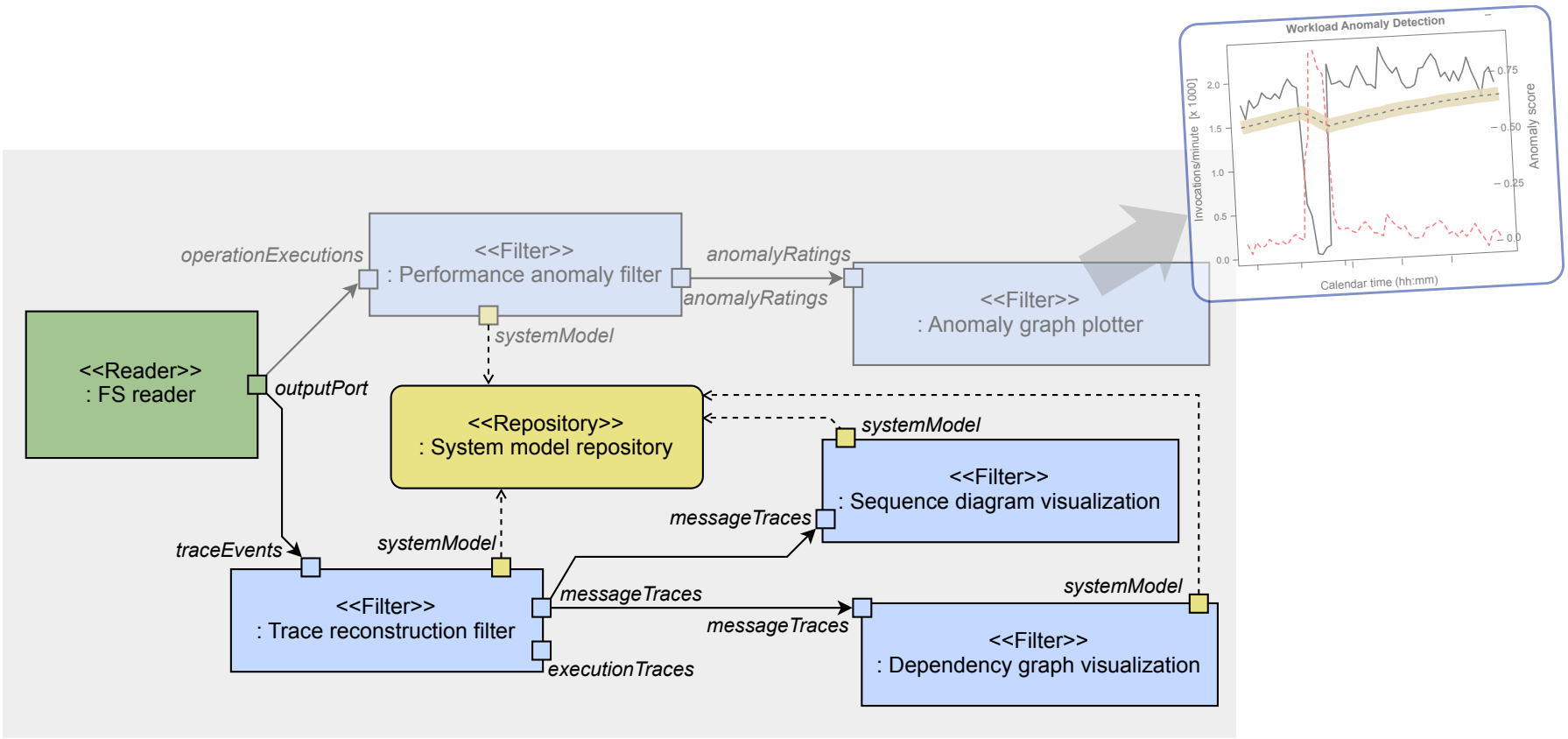
Christian-Albrechts-Universität zu Kiel



Kieker.Analysis example pipes-and-filters configuration

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

Example Pipe-and-Filter Configuration

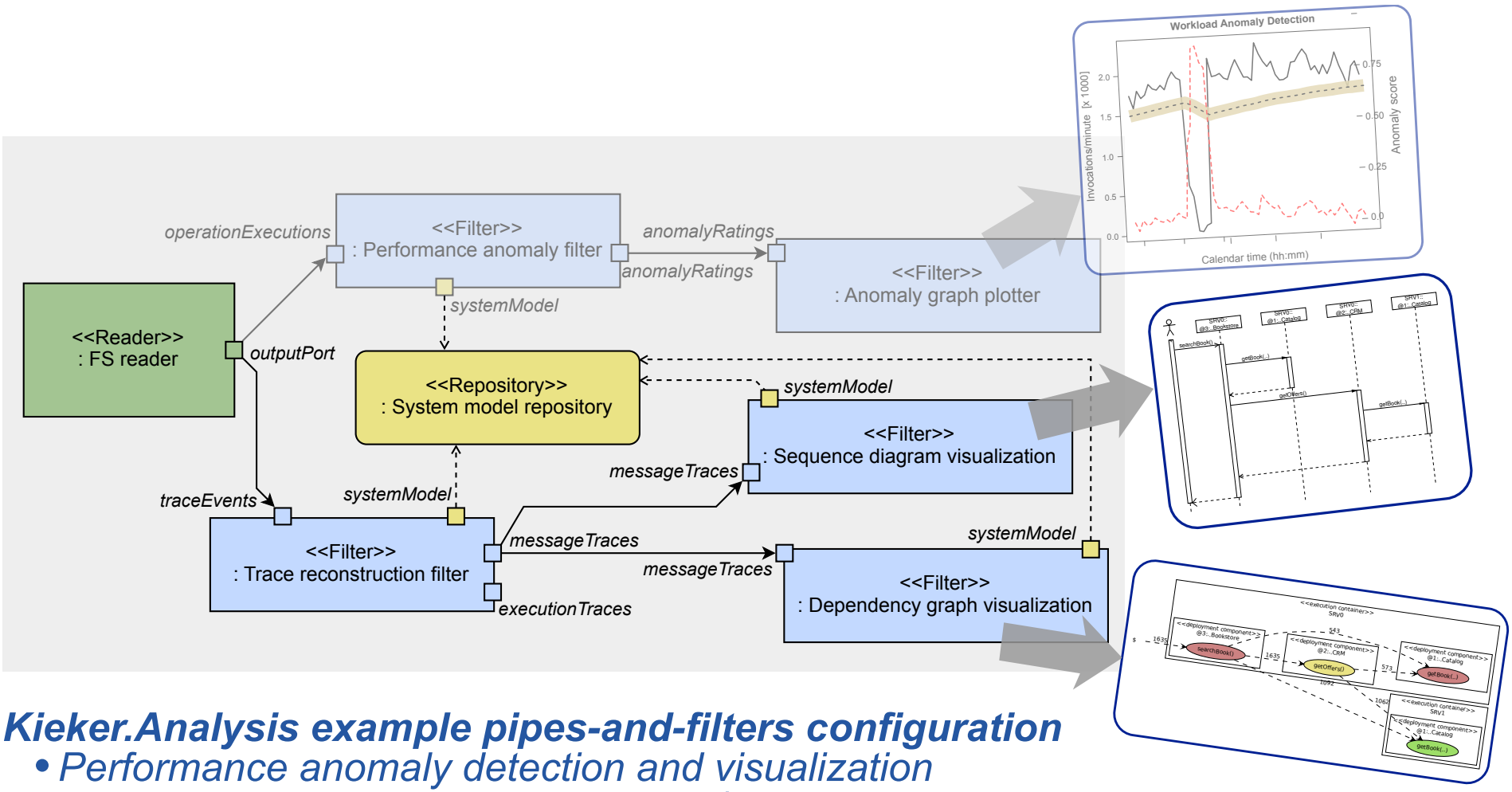


Kieker.Analysis example pipes-and-filters configuration

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

Example Pipe-and-Filter Configuration

Christian-Albrechts-Universität zu Kiel



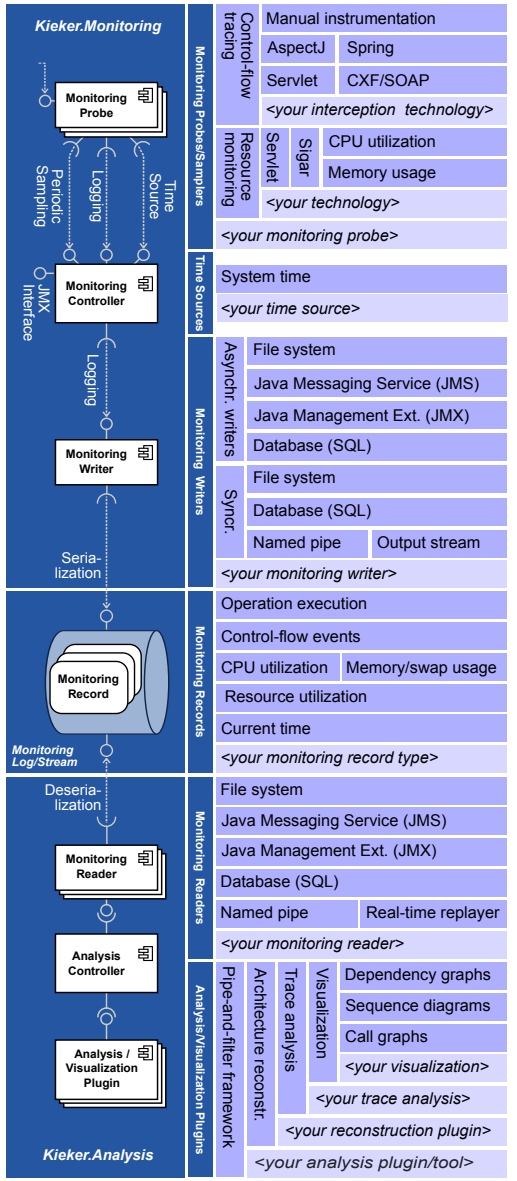
Kieker.Analysis example pipes-and-filters configuration

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

Kieker

- Modular, flexible, and extensible architecture (Probes, records, readers, writers, filters etc.)
- Pipes-and-filters framework for analysis configuration
- Distributed tracing (logging, reconstruction, visualization)
- Low overhead (designed for continuous operation)
- Evaluated in lab and industrial case studies

b+m business IT management · cewe color · dataport · EWE · HSH NORDBANK · XING



Framework Features & Extension Points

Christian-Albrechts-Universität zu Kiel

Kieker

- Modular, flexible, and extensible architecture (Probes, records, readers, writers, filters etc.)
- Pipes-and-filters framework for analysis configuration
- Distributed tracing (logging, reconstruction, visualization)
- Low overhead (designed for continuous operation)
- Evaluated in lab and industrial case studies

b+m business IT management · cewe color · dataport · EWE · HSH NORDBANK · XING

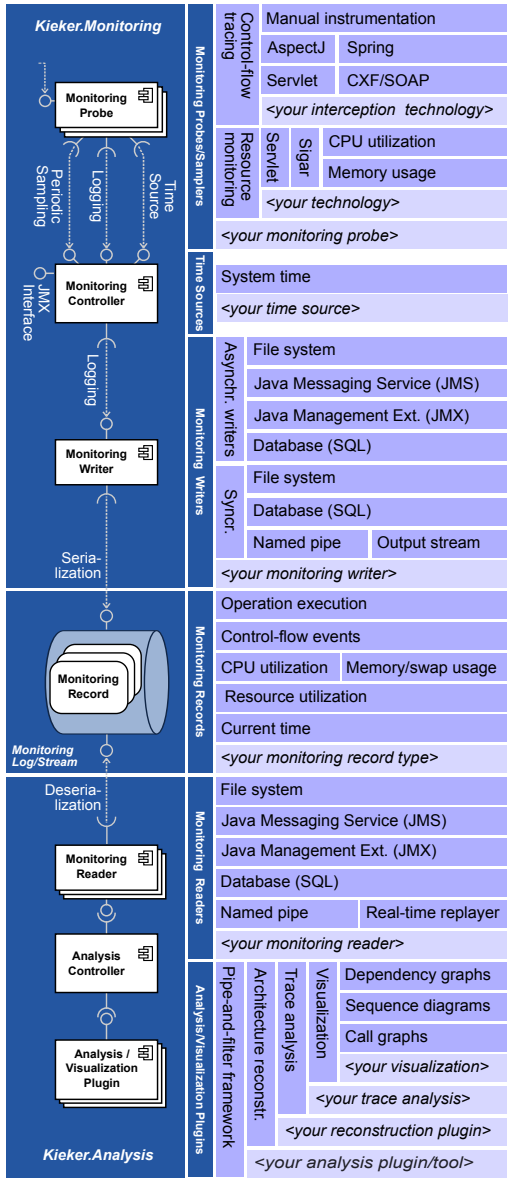
Kieker is open-source software (Apache License, V. 2.0)

`http://kieker-monitoring.net`

Kieker is distributed as part of SPEC® RG's repository of peer-reviewed tools for quantitative system evaluation and analysis

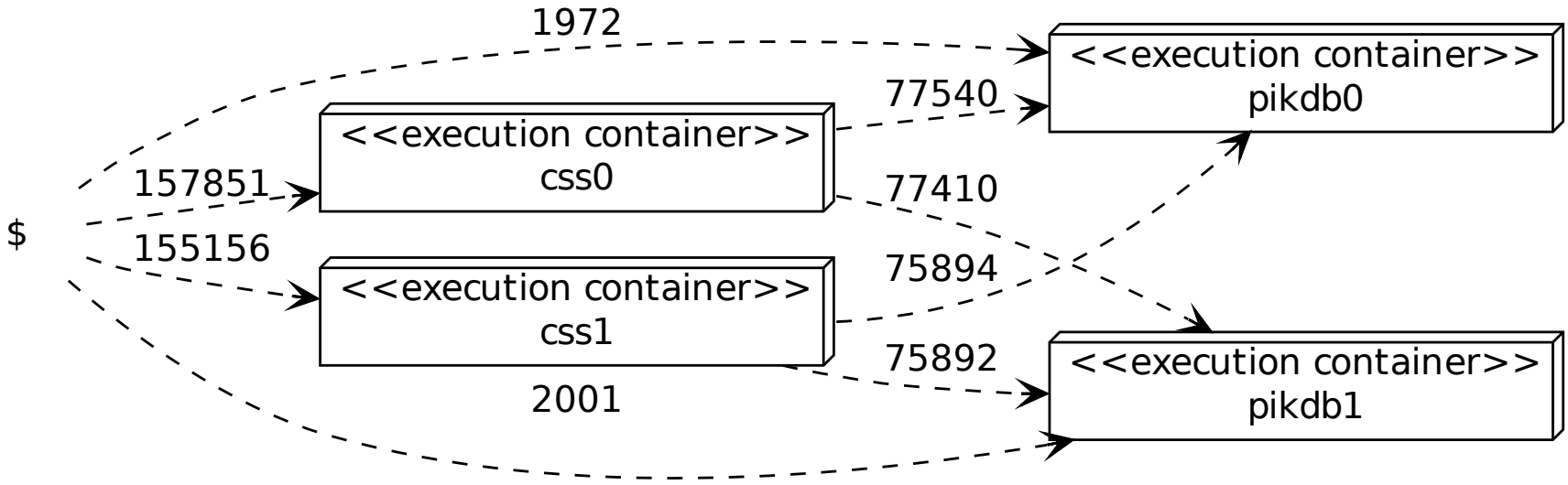
`http://research.spec.org/projects/tools.html`

spec Research



Telecommunication provider, distributed setting

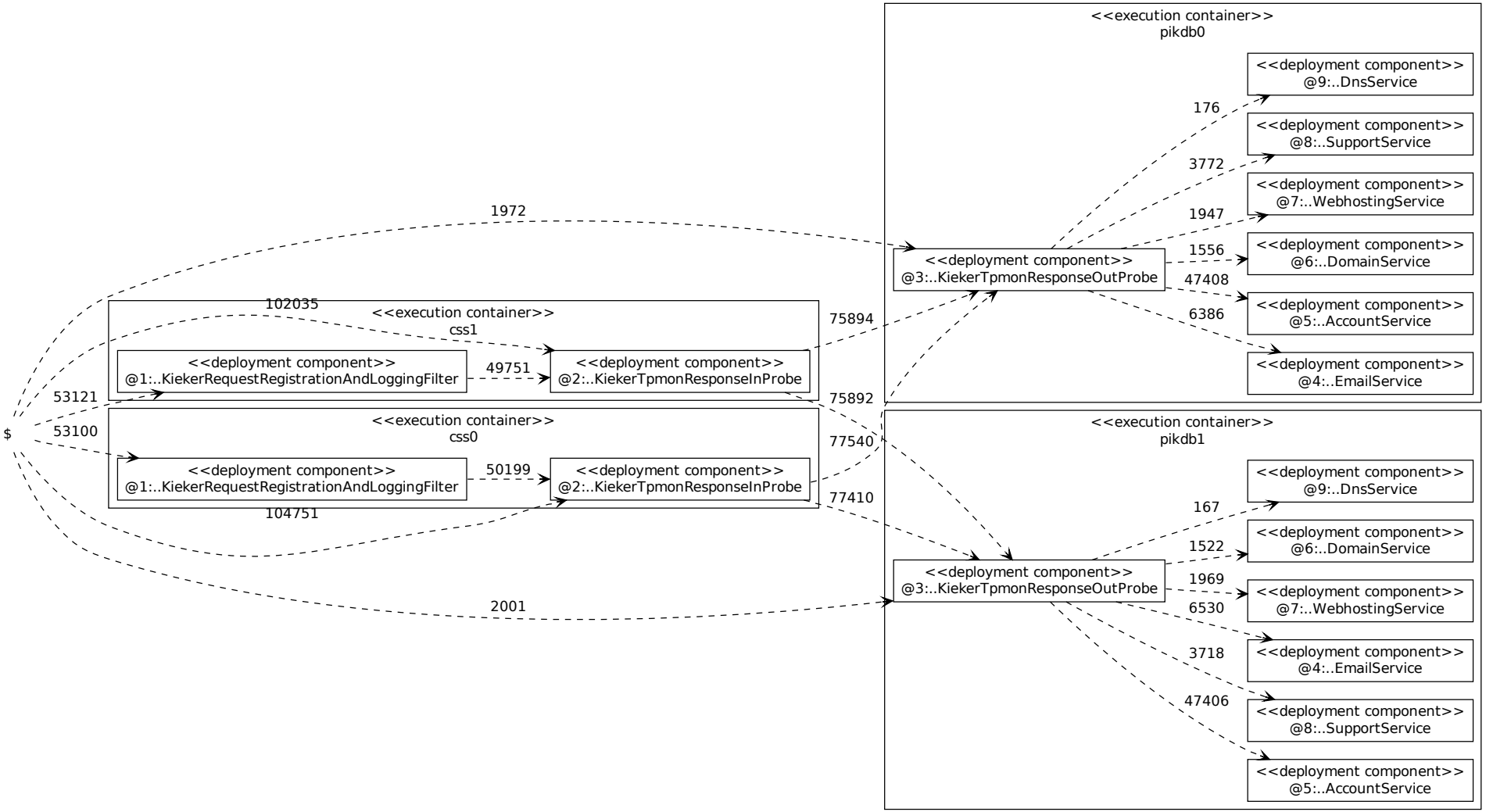
- Goal: Workload characterization and performance evaluation
- Java-based technology: Servlet, Spring, and CXF/SOAP
- Continuous Monitoring (utilizing Kieker):
  - Probes for collecting distributed application-level trace and performance data
  - Continuous monitoring in production use since 12/2009
- Model extraction examples (316,980 traces):



Container dependency graph (with calling frequencies)

Case Study: Model Extraction Examples (cont'd)

Deployment-level component dependency graph (with calling frequencies):



What I've Talked About

Christian-Albrechts-Universität zu Kiel

SLAstic

- Self-adaptive capacity management based on application-level workload
- Adaptation goal: Increased resource efficiency while meeting SLAs
- Means for adaptation: Architectural runtime reconfiguration
- Use of architectural (performance) models at development & runtime
- Domain: Business-critical distributed component-based software systems

Kieker

- Modular and extensible architecture (Probes, records, readers, writers, filters etc.)
- Pipes-and-filters framework for analysis configuration
- Distributed tracing (logging, reconstruction, visualization)
- Low overhead (designed for continuous operation)
- Evaluated in lab and industrial case studies

Feel free to contact me: `avh@informatik.uni-kiel.de`

N. Günther. Modellbasierte Laufzeit-Performance-Vorhersage für komponentenbasierte Softwarearchitekturen (“Model-based online performance prediction for component-based software architectures”, in german), Nov. 2011. Diploma Thesis, University of Kiel.

A. van Hoorn. *Model-Driven Online Capacity Management for Resource Efficient Component-Based Software Systems*. PhD thesis, Department of Computer Science, University of Kiel, Kiel, Germany, 2012. work in progress.

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. URL `http://www.informatik.uni-kiel.de/uploads/tx_publication/vanhoorn_tr0921.pdf`.

A. van Hoorn, S. Frey, W. Goerigk, W. Hasselbring, H. Knoche, S. Köster, H. Krause, M. Porembski, T. Stahl, M. Steinkamp, and N. Wittmüss. DynaMod project: Dynamic analysis for model-driven software modernization. In A. Fuhr, W. Hasselbring, V. Riediger, M. Bruntink, and K. Kontogiannis, editors, *Joint Proceedings of the 1st International Workshop on Model-Driven Software Migration (MDSM 2011) and the 5th International Workshop on Software Quality and Maintainability (SQM 2011)*, volume 708 of *CEUR Workshop Proceedings*, pages 12–13, Mar. 2011. Invited paper.

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 2012)*, pages 247–248. ACM, Apr. 2012.

R. von Massow, A. van Hoorn, and W. Hasselbring. Performance simulation of runtime reconfigurable component-based software architectures. In I. Crnkovic, V. Gruhn, and M. Book, editors, *Proceedings of the 5th European Conference on Software Architecture (ECSA ’11)*, volume 6903 of *Lecture Notes in Computer Science*, pages 43–58. Springer, Sept. 2011.