Performance Simulation of Runtime Reconfigurable Component-Based Software Architectures

Robert von Massow, **André van Hoorn**,
and Wilhelm Hasselbring

Software Engineering Group
`http://se.informatik.uni-kiel.de/`

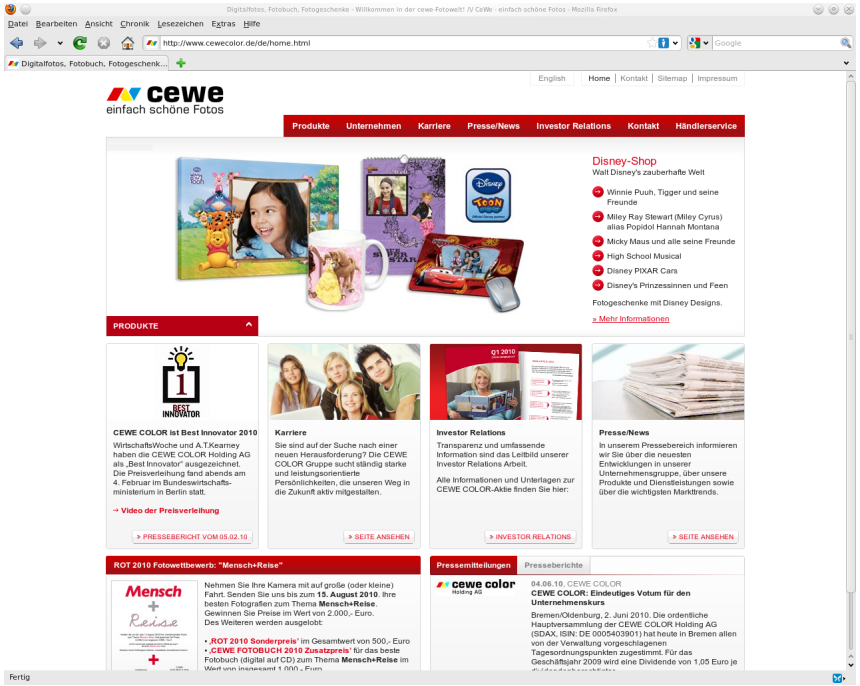
Dept. of Computer Science
Christian Albrechts University of Kiel

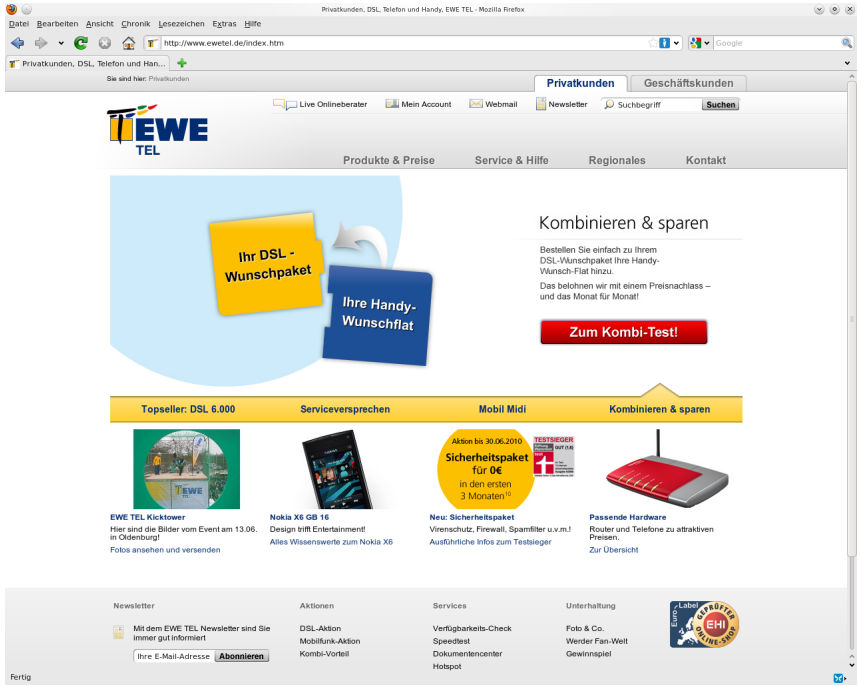
Sept. 14, 2011 @ ECSA 2011, Essen, Germany

Context of this Work

Online Capacity Management for Increased Resource Efficiency

Introduction





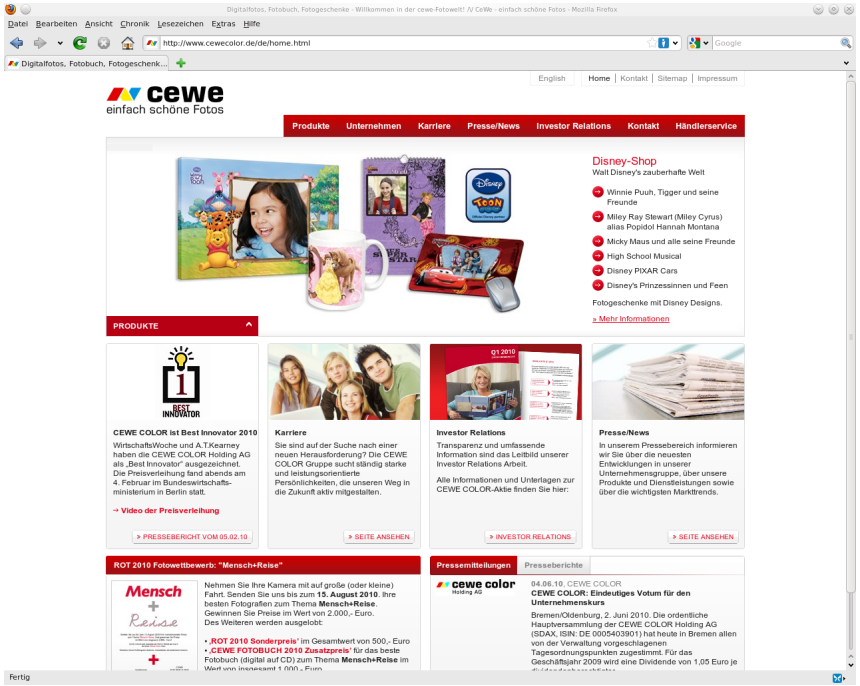
- Business-critical software systems

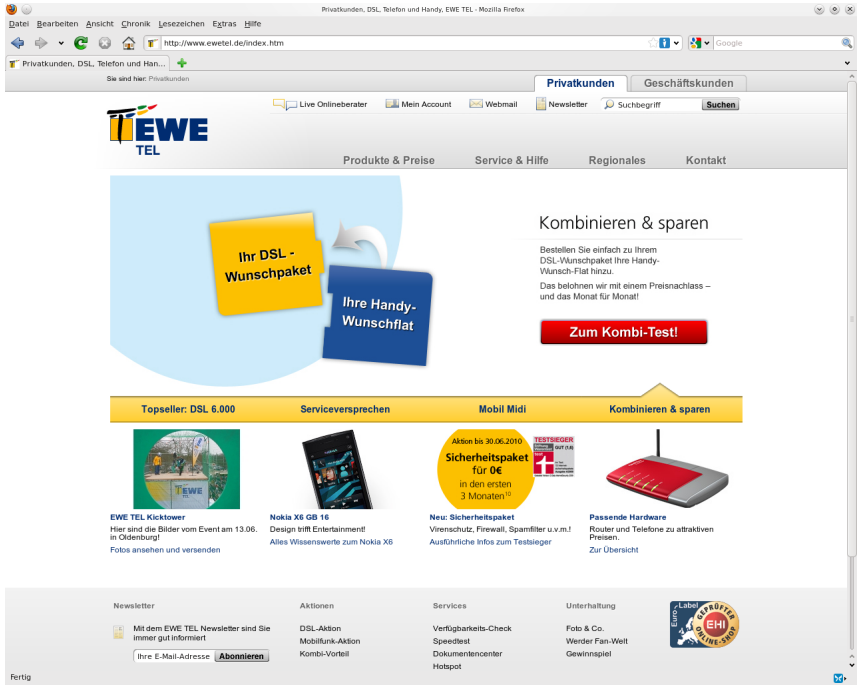
Context of this Work

Online Capacity Management for Increased Resource Efficiency

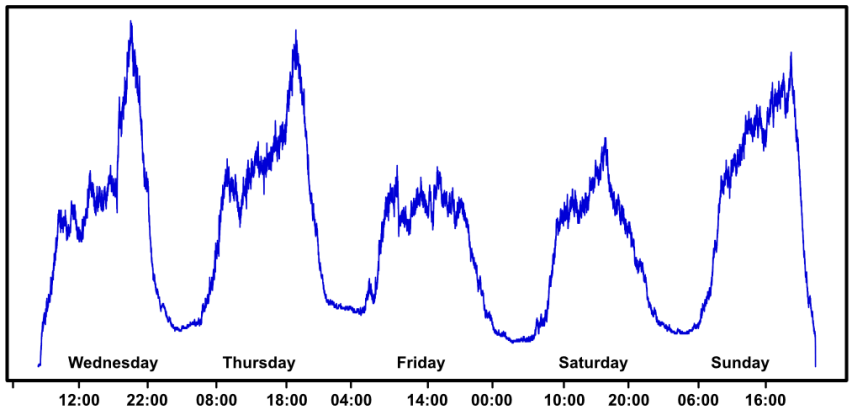
Introduction

Christian-Albrechts-Universität zu Kiel



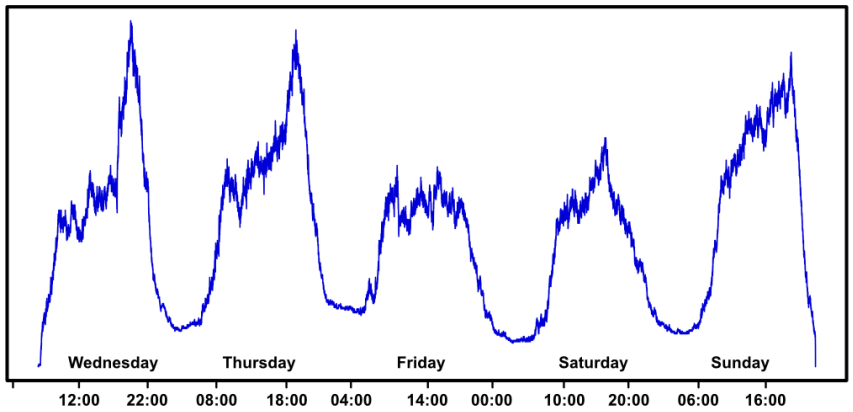


- Business-critical software systems
- Quality of service (performance, availability, . . .)



- Business-critical software systems
- Quality of service (performance, availability, . . .)
- Varying workloads + static capacity management

Christian-Albrechts-Universität zu Kiel



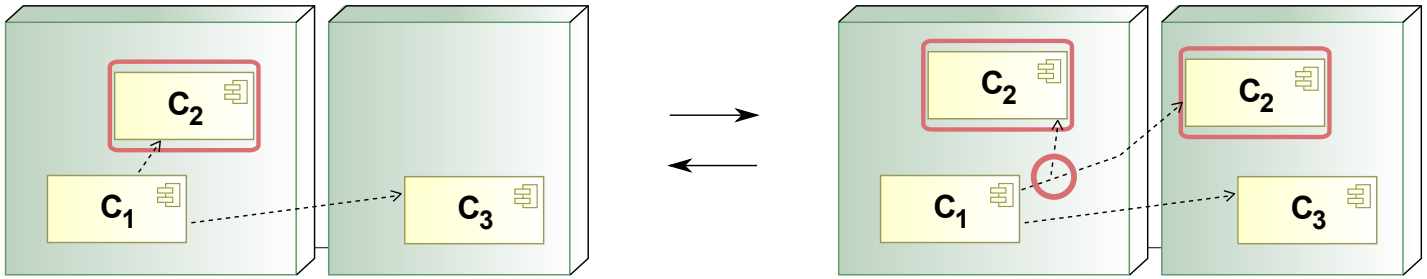
Problem: Overprovisioning — unnecessarily high operating costs

Underutilized resources during medium/low workload periods

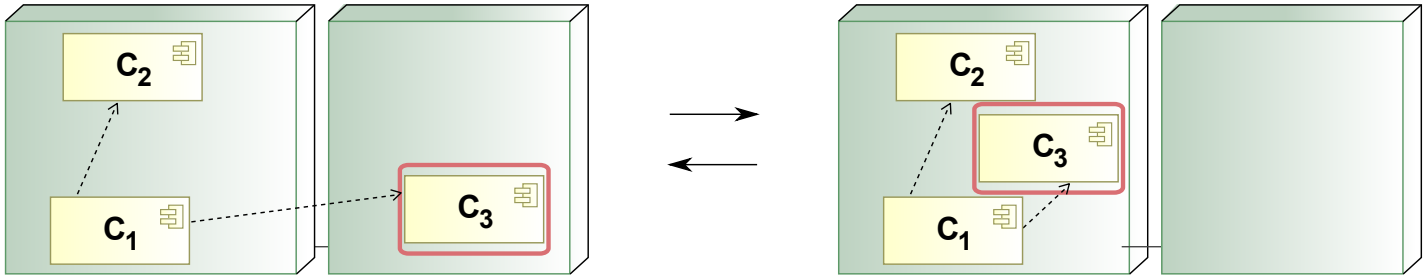
Goal: Increase resource efficiency while meeting SLAs

▷ **SLAstic** [vHRGH09, vH11]: Online capacity management employing runtime reconfiguration

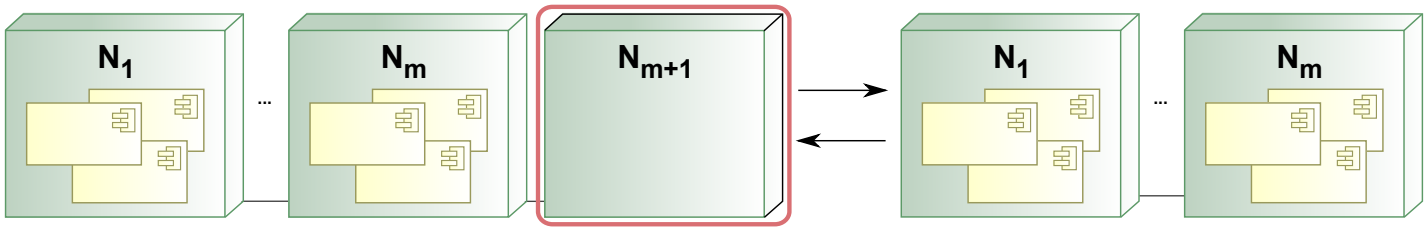
1. (De-)Replication of Software Components



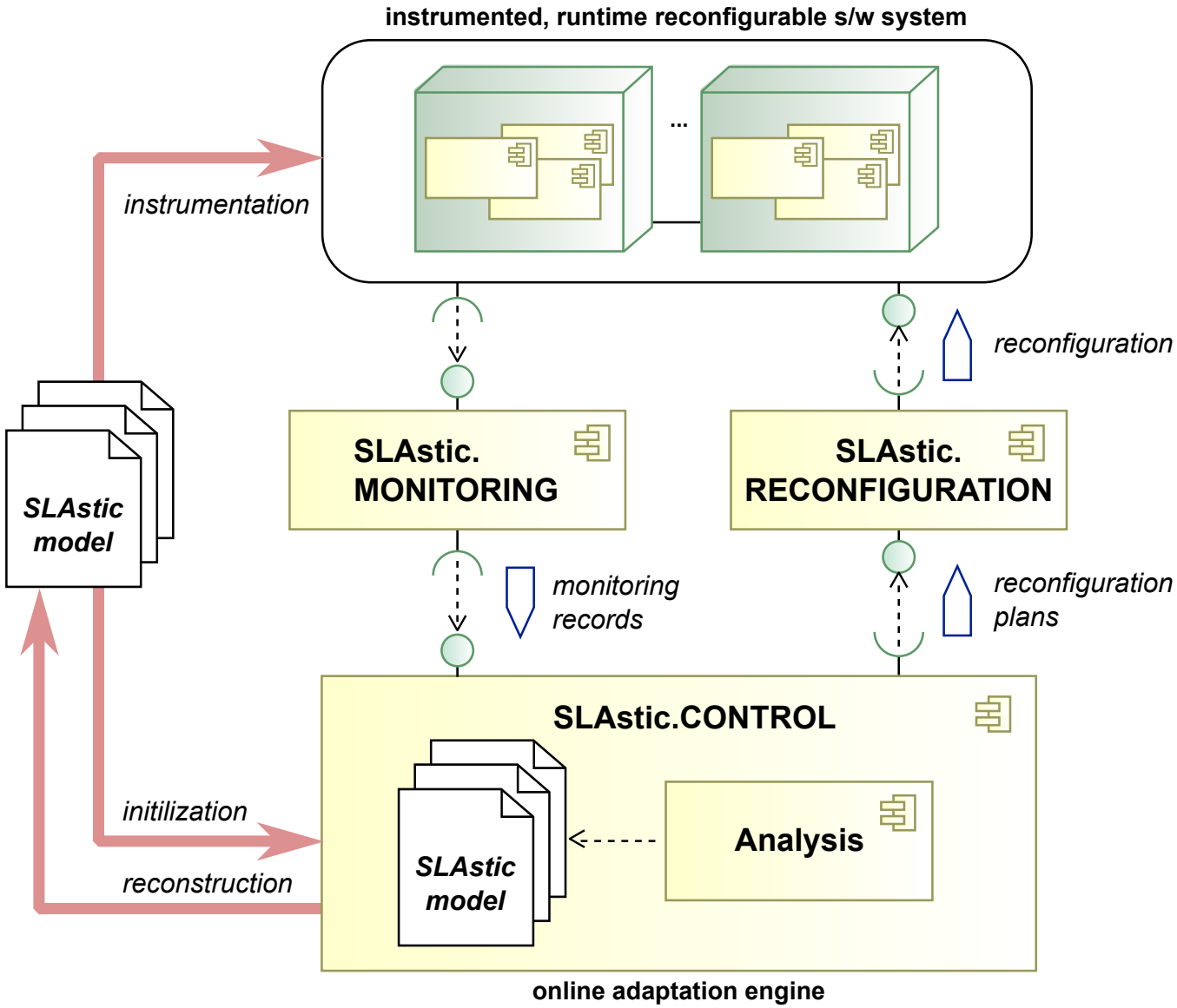
2. Migration of Software Components

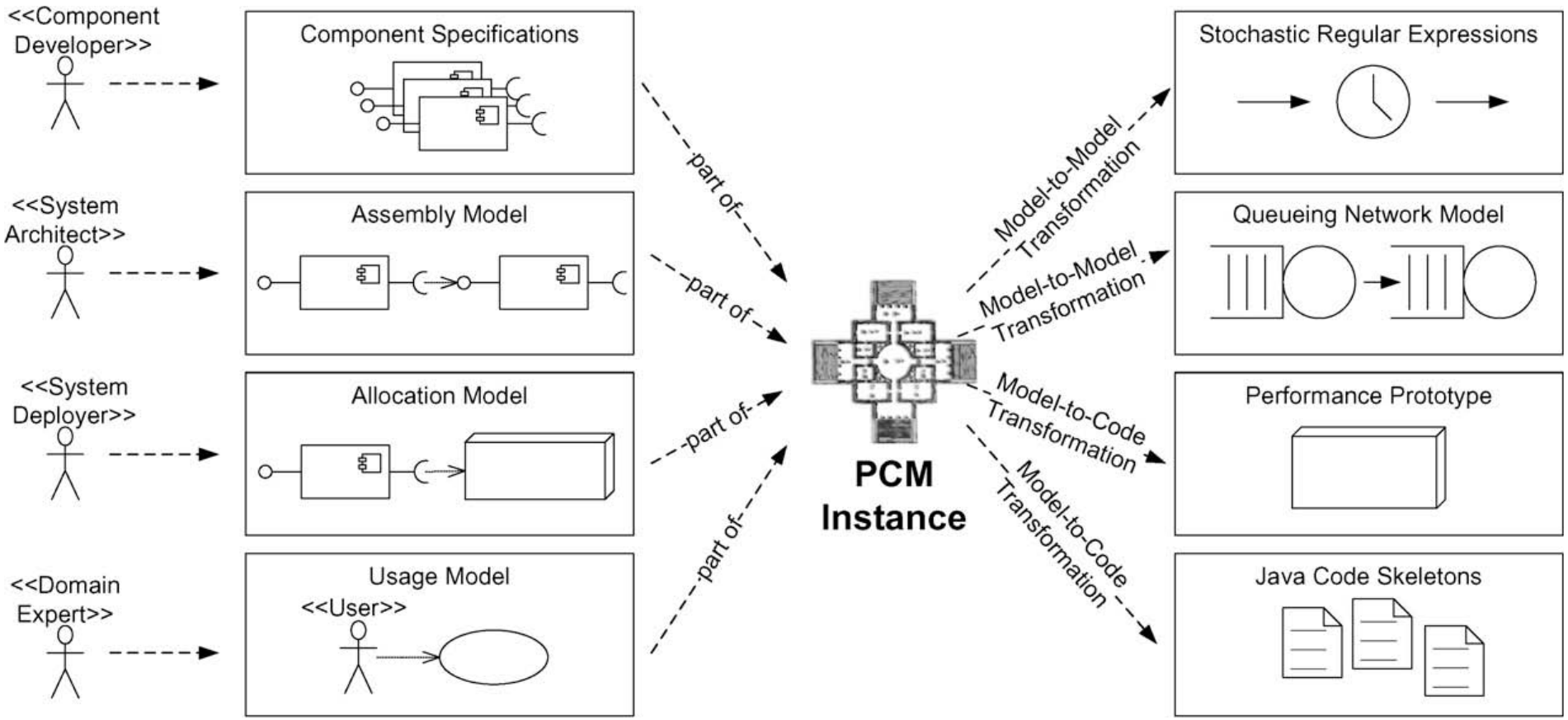


3. (De-)Allocation of Execution Containers



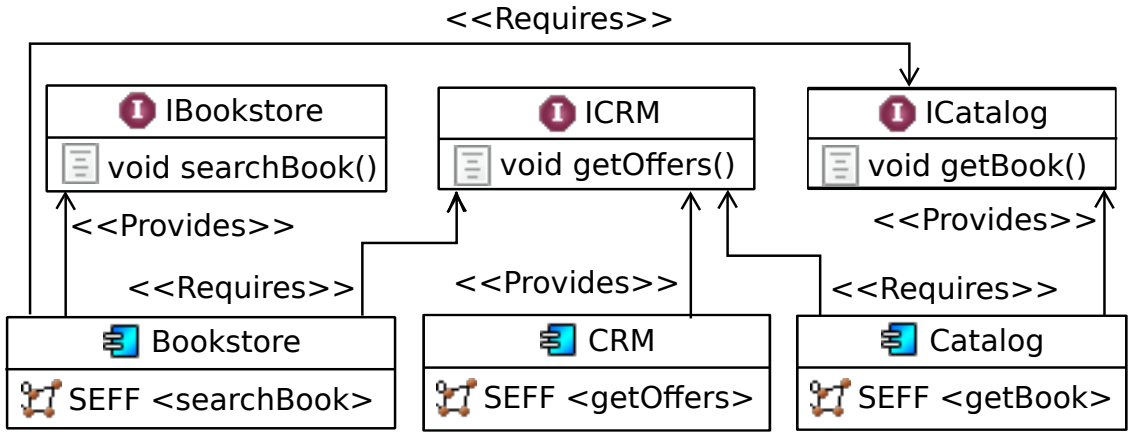
Agenda


instrumented, runtime reconfigurable s/w system
...
instrumentation
reconfiguration
SLAstic.
MONITORING
SLAstic.
RECONFIGURATION
SLAstic
model
monitoring
records
reconfiguration
plans
SLAstic.CONTROL
SLAstic
model
Analysis
initilization
reconstruction
online adaptation engine


<<Component Developer>>
Component Specifications
<<System Architect>>
Assembly Model
<<System Deployer>>
Allocation Model
<<Domain Expert>>
Usage Model
<<User>>
part of
part of
part of
part of
PCM Instance
Model-to-Model Transformation
Model-to-Model Transformation
Model-to-Code Transformation
Model-to-Code Transformation
Stochastic Regular Expressions
Queueing Network Model
Performance Prototype
Java Code Skeletons

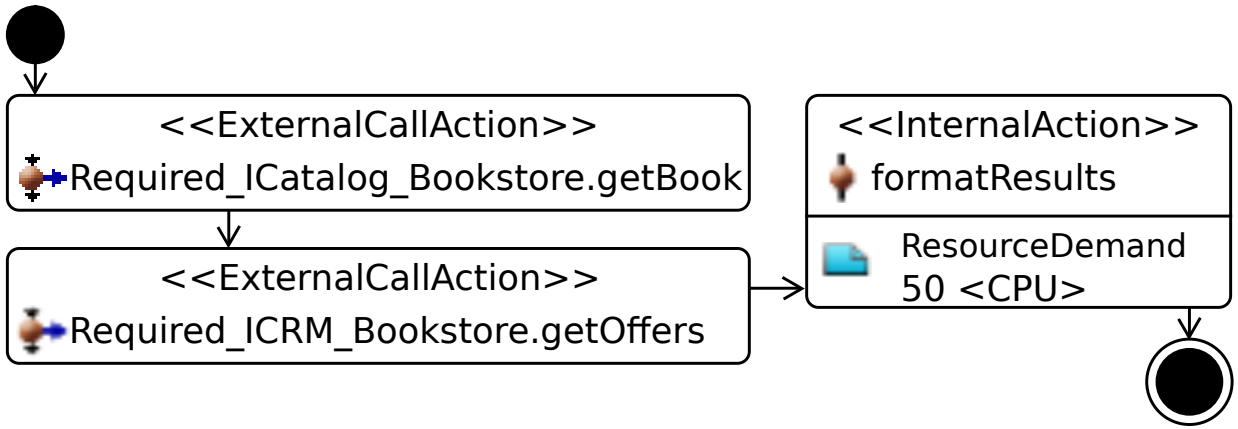
[BKR09]

1. PCM Repository Model
   - Components & Interfaces
   - RDSEFFS
2. PCM System Model, Resource Env. & Allocation
3. Scenario searchBook()



Christian-Albrechts-Universität zu Kiel

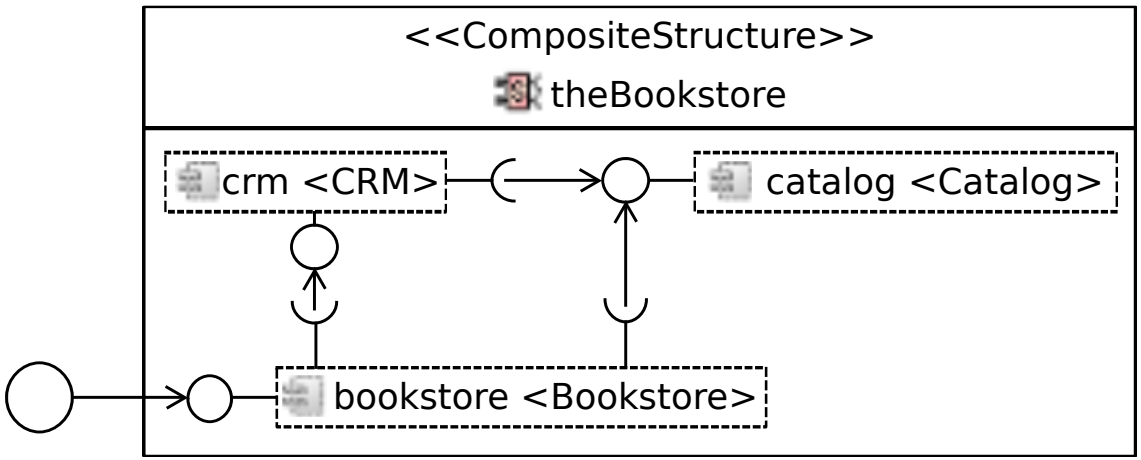
1. PCM Repository Model
   - Components & Interfaces
   - RDSEFFS
2. PCM System Model, Resource Env. & Allocation
3. Scenario searchBook()



RDSEFF of `Bookstore.searchBook()`

- `Bookstore.searchBook()`: External calls: `getBook()` & `getOffers`; CPU demand: 50
- `Catalog.getBook()`: External call: `getBook()`; CPU demand: 20
- `CRM.getOffers()`: CPU demand: 15

1. PCM Repository Model
   - Components & Interfaces
   - RDSEFFS
2. PCM System Model, Resource Env. & Allocation
3. Scenario searchBook()

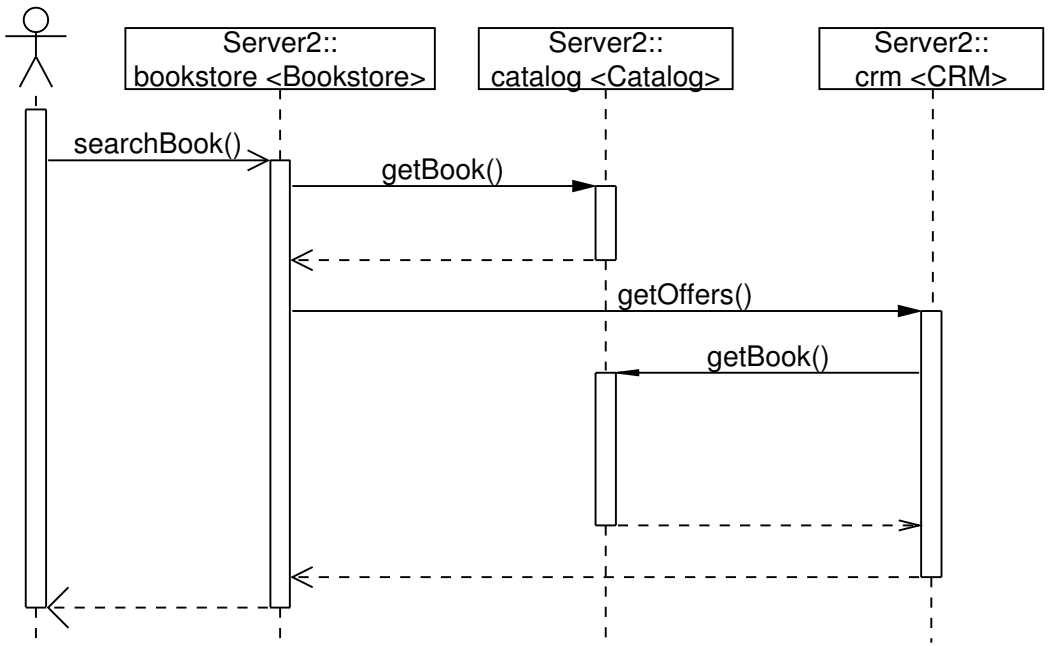


System Model

- Resource Environment: 2 resource containers (`Server1` & `Server2`)
- (Initial) Allocation: 1 allocation ctx. per assembly ctx. on `Server2`; `Server1` empty

Christian-Albrechts-Universität zu Kiel

1. PCM Repository Model
   - Components & Interfaces
   - RDSEFFS
2. PCM System Model, Resource Env. & Allocation
3. Scenario searchBook()

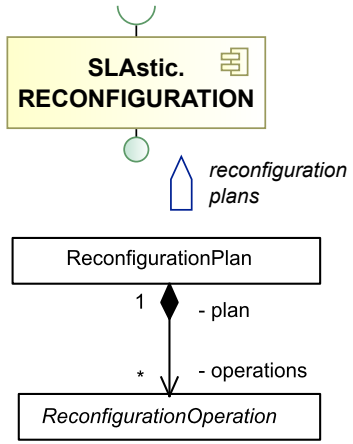


(reconstructed from simulation data)

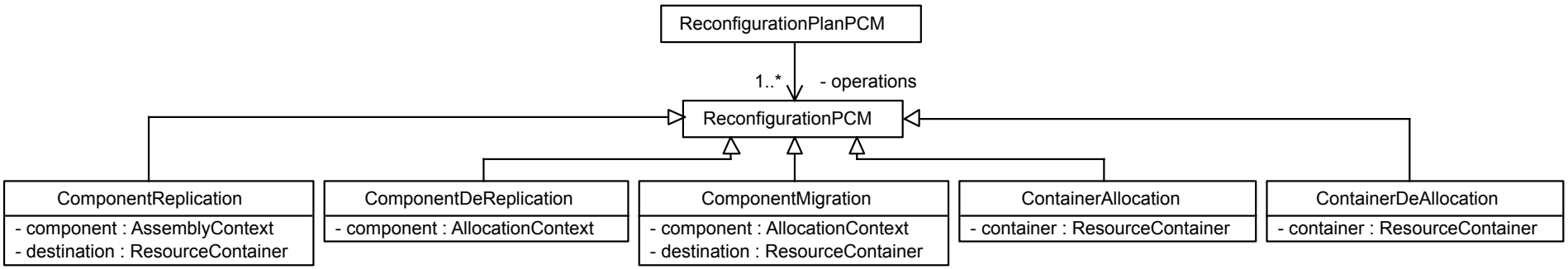
Reconfiguration Operations (SLAstic/PCM)

Architecture Level Operation Signatures (based on SLAstic Meta-Model):

1. (De-)Replication of Software Components
   - replicate (component: AssemblyComponent, to: ExecutionContainer)
   - dereplicate (component: DeploymentComponent)
2. Migration of Software Components
   - migrate (component: DeploymentComponent, to: ExecutionContainer)
3. (De-)Allocation of Execution Containers
   - allocate (containerType: ExecutionContainerType)
   - deallocate (container: ExecutionContainer)



Technology-specific Operation Signatures (here: PCM [BKR09]):



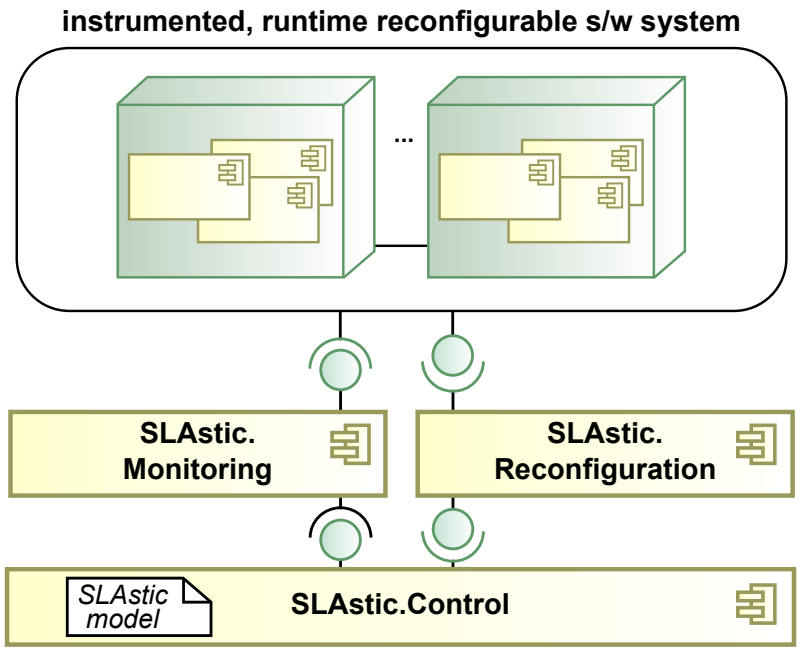
Rationale: Simulation-based . . .

1. Study (transient & stationary) effects of reconfigurations (implementation “costly”)
2. Online performance prediction (e.g., proactive problem detection, online planning)
3. Offline performance prediction (e.g., evaluation of adaptation strategies/techniques)
4. Evaluation of overall approach (in addition to lab & case studies)

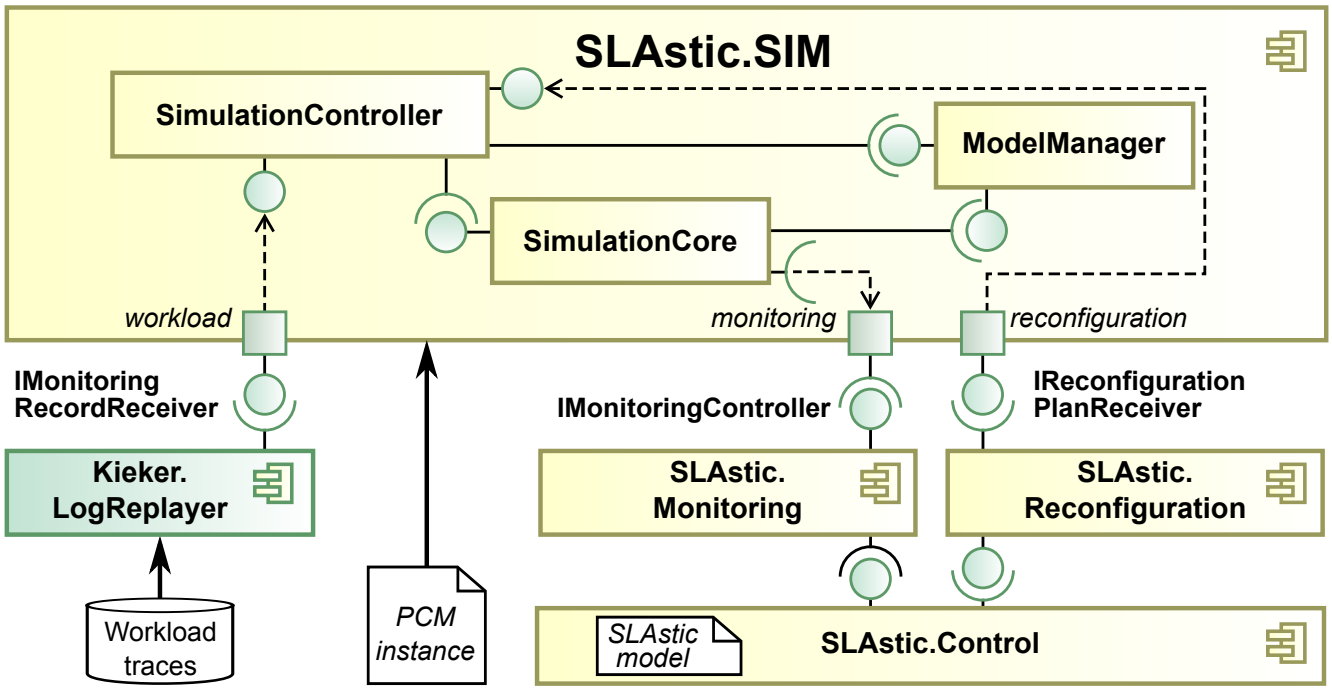
Our Requirements for a Performance Simulator:

1. Architectural style: C-B software systems (assembly, deployment, . . .)
2. Workload: Replay traces, support varying workload
3. Runtime reconfiguation: Support SLAstic operations
4. Measures: Performance (timing, CPU utilization)

Agenda


instrumented, runtime reconfigurable s/w system
...
SLAstic.
Monitoring
SLAstic.
Reconfiguration
SLAstic
model
SLAstic.Control

SLAstic.SIM — Architecture & Integration



- (Initial) transformation of PCM instance into internal representation
- SLAstic.SIM employs Desmo-J[1] [PK05] discrete-event simulation engine
- Kieker used for providing workload traces and for monitoring simulation data
- Probes injected with Google Guice[2]

[1] `http://desmoj.sourceforge.net/` [2] `http://code.google.com/p/google-guice/`

Features/Limitations

- Workload
  - Driven by externally provided workload events (e.g., replayed monitoring logs)
  - *Allows integration with workload generators*
- Reconfiguration
  - Triggered by externally provided PCM-specific reconfiguration plans
  - Supports the presented operations
- Hardware resource scheduling
  - CPU: Procesor-sharing (time slicing)
  - Disk: First-come/first-served
  - *Scheduler interface allows custom disciplines*
- Measurements
  - Executions (incl. timing & control-flow information)
  - CPU utilization
  - # Concurrent (system-level) transactions
- Unsupported PCM features:
  - (Parametric) stochastic expressions $\longrightarrow$ *planned*
  - Middleware model (as supported by SimuCom)

Initialization & Runtime Phase

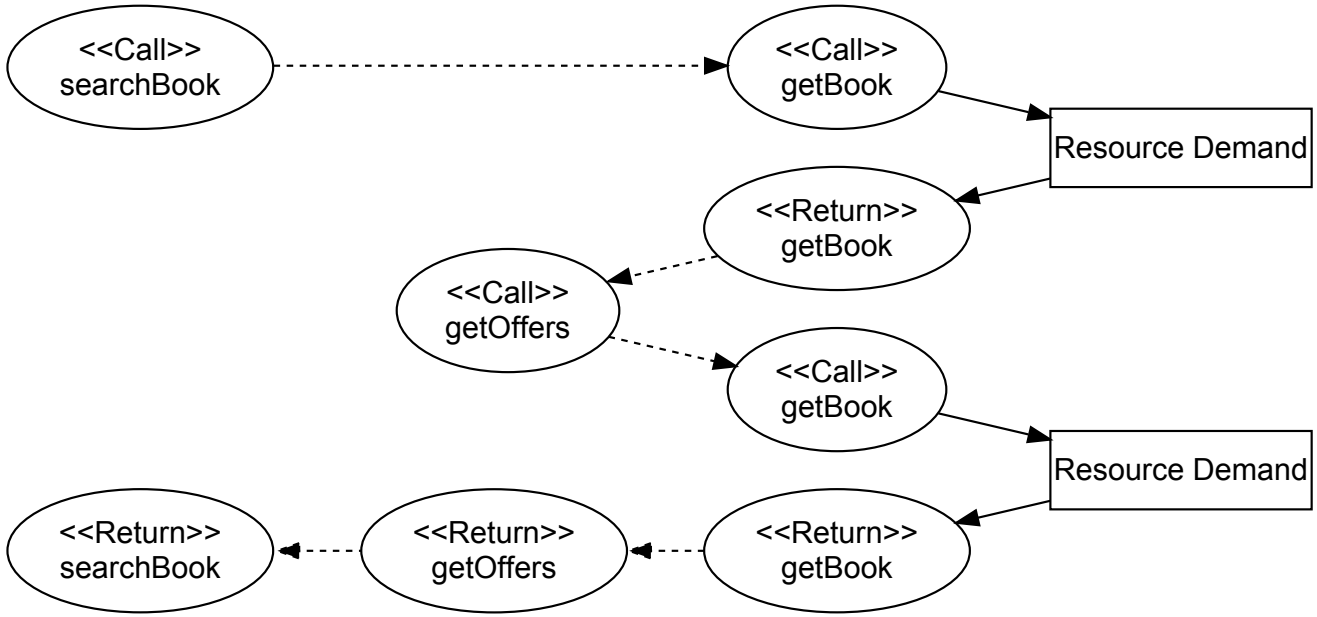
Initialization of the dynamic simulation model

1. store `Components` and their `Services` for lookup
2. initialize `AssemblyContexts` (i. e. component instatiation)
3. initialize `ResourceContainers` (CPUs, schedulers, . . .)
4. initialize `AllocationContexts` and deallocate remaining `ResourceContainers`

On occurrence of an entry level system call

1. the corresponding RDSEFF is looked up
2. the control flow is evaluated (i. e. the `AbstractActions`) and translated into simulatable events
3. the first event is scheduled

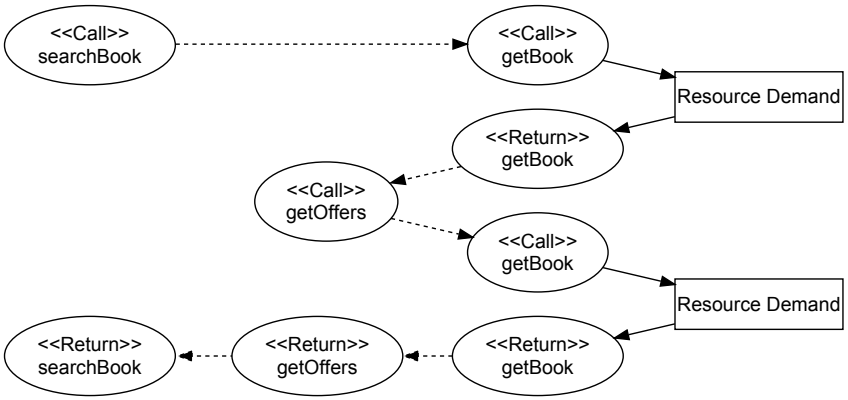
Initialization & Runtime Phase

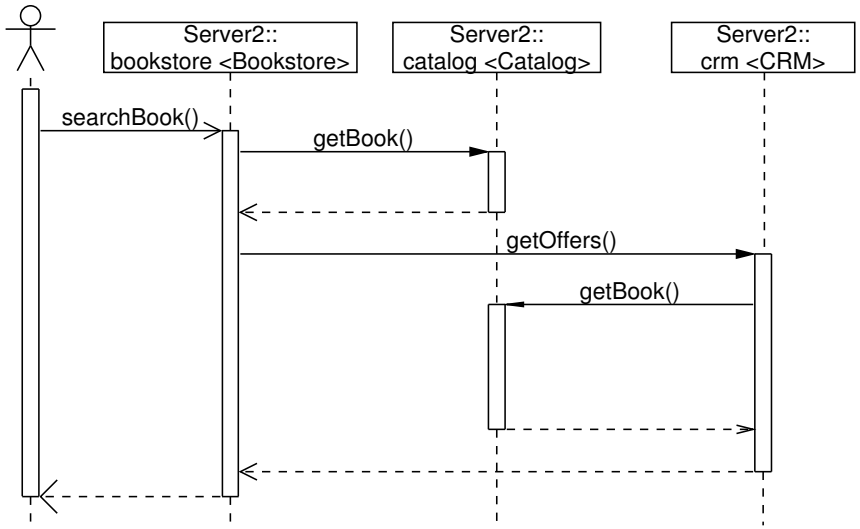


On occurrence of an entry level system call

1. the corresponding RDSEFF is looked up
2. the control flow is evaluated (i. e. the `AbstractActions`) and translated into simulatable events
3. the first event is scheduled

Control Flow Events and Sequence Diagram





Evaluation Goals & Methodology

Goals

- Evaluating SLAstic.SIM's performance
- Comparison with SimuCom (performance & results)
- Evaluating effect of reconfiguration on the simulated system

Methodology

- Use of Bookstore sample application *(PCM instance described in the beginning)*
- **Evaluation scenarios**:
  1. Constant workload, comparison with SimuCom *[results in the paper]*
  2. Varying workload intensity w/o reconfiguration
  3. Varying worklaod intensity w/ reconfiguration

Hardware and Software Setup

CPU	Intel Core i5, hyper-threading enabled
RAM	4 GB
OS	Ubuntu Generic Linux kernel 2.6.32-22 SMP
Java	Sun Java Version 1.6.0_20
Heap space	1 GB for SLAstic.SIM / 2 GB for SimuCom 3.0

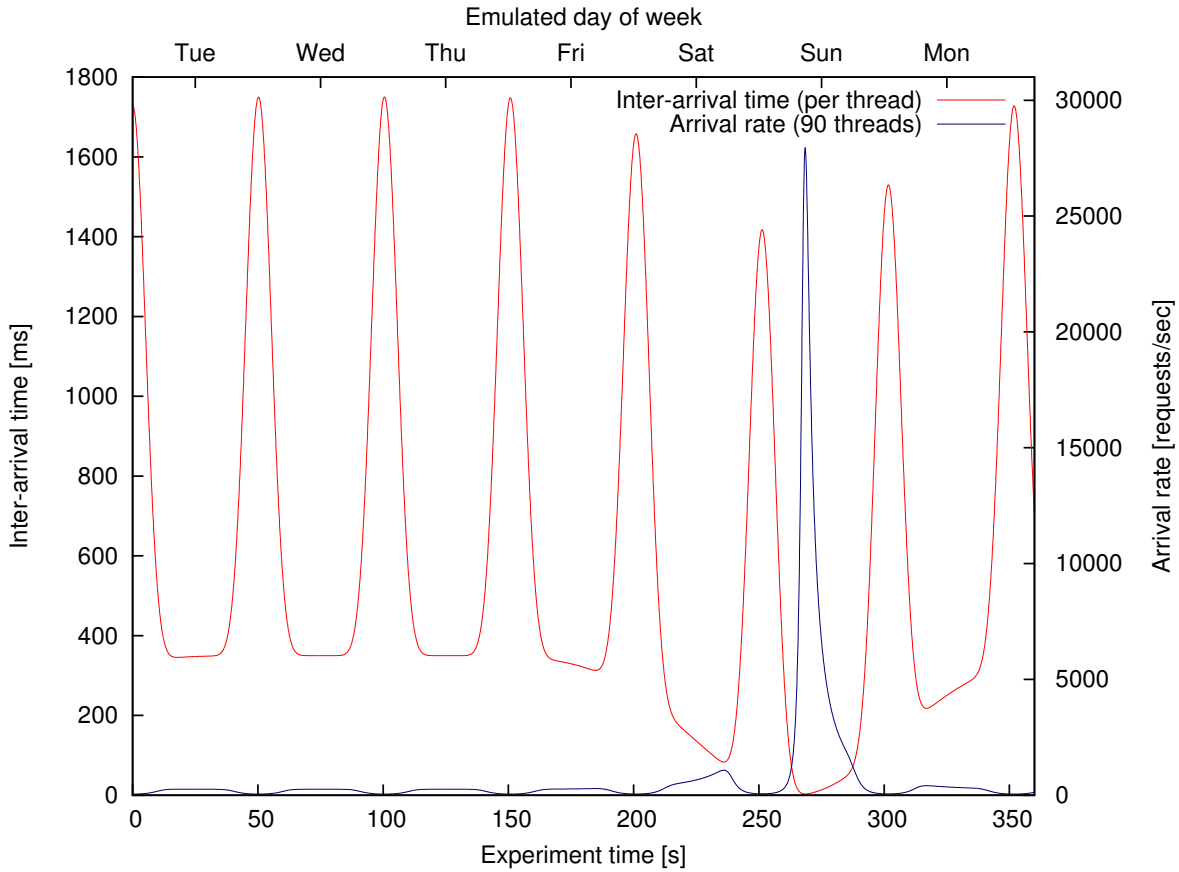
Simulation of Varying/Bursty Workload (Scenario 2)

Setup

- Workload generated (offline) by Apache JMeter[2] + own plug-in[3]
- Varying inter-arrival times according to an input function
- 68 653 traces
- Simulation time: 360 time units

[2] `http://jakarta.apache.org/jmeter/` [3] `http://code.google.com/p/delayfunction/`

Setup



Varying workload intensity (Scenarios 2 & 3)

Simulation of Varying/Bursty Workload (Scenario 2)

Setup

- Workload generated (offline) by Apache JMeter[2] + own plug-in[3]
- Varying inter-arrival times according to an input function
- 68 653 traces
- Simulation time: 360 time units

Goal

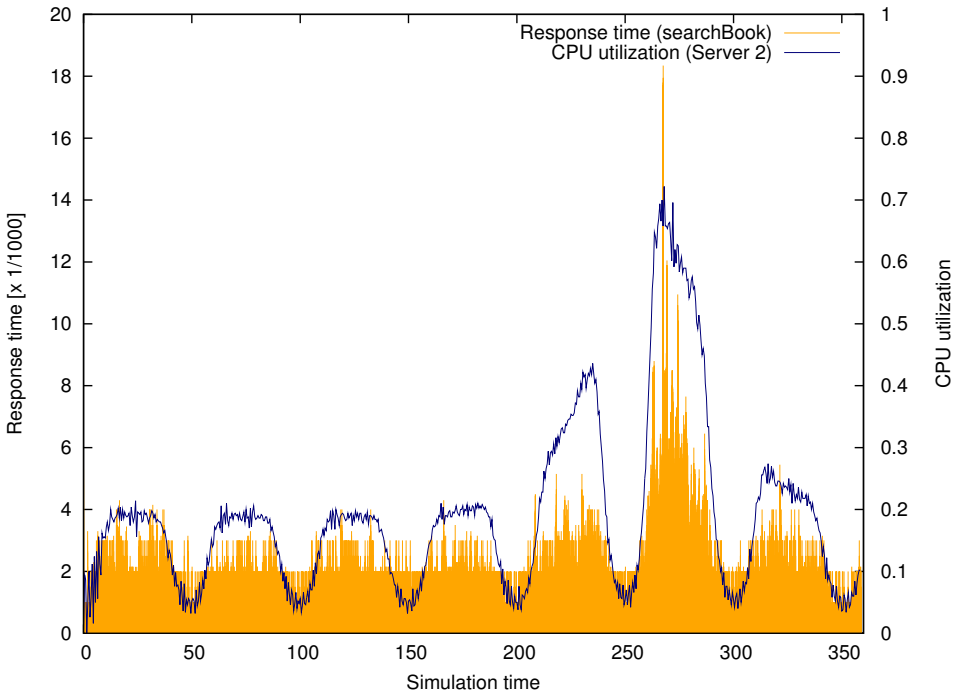
- Demonstrating the performance simulation driven by bursty workload

Results

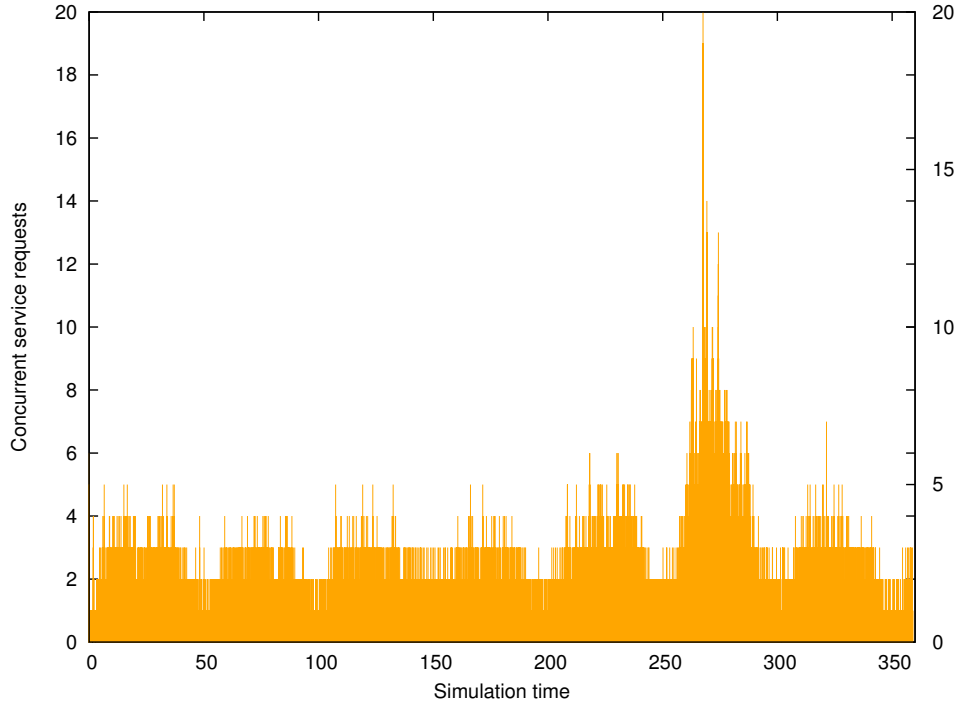
- Simulation took around 18.6 seconds
- Peak in input workload resulted in peaks of both, response times & CPU utilization

[2] `http://jakarta.apache.org/jmeter/` [3] `http://code.google.com/p/delayfunction/`

Results (cont'd)



Response times (`searchBook`) & CPU utilization



Concurrent transactions

Varying Workload w/ Reconfiguration (Scenario 3)

Christian-Albrechts-Universität zu Kiel

Setup

- Reused varying workload from Scenario 2
- Simulation time: 360 time units
- Reconfigurations at $t = 200$ and $t = 300$
  - $t = 200$: Increase capacity ("weekend plan")
    1. Allocation of `Server1`
    2. Replicating `CRM` to `Server1`
    3. Migrating `Catalog` from `Server2` to `Server1`
  - $t = 300$: Decrease capacity (inverse plan)
    1. De-replication of `CRM` from `Server1`
    2. Migration of `Catalog` from `Server1` to `Server2`
    3. De-allocation of `Server1`

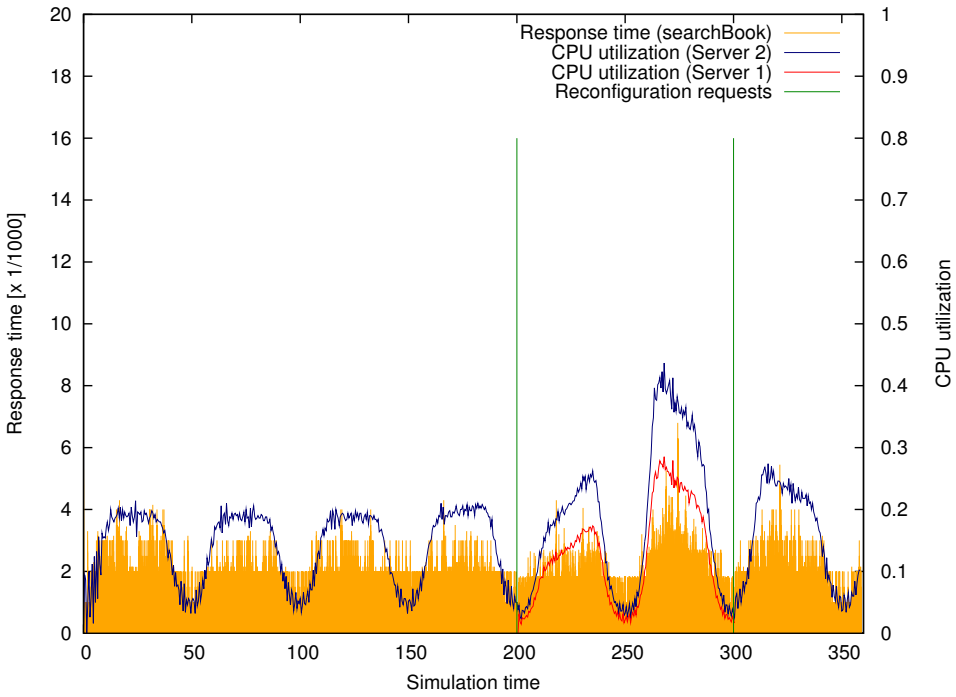
Goal

- Demonstrating the effect of reconfiguration

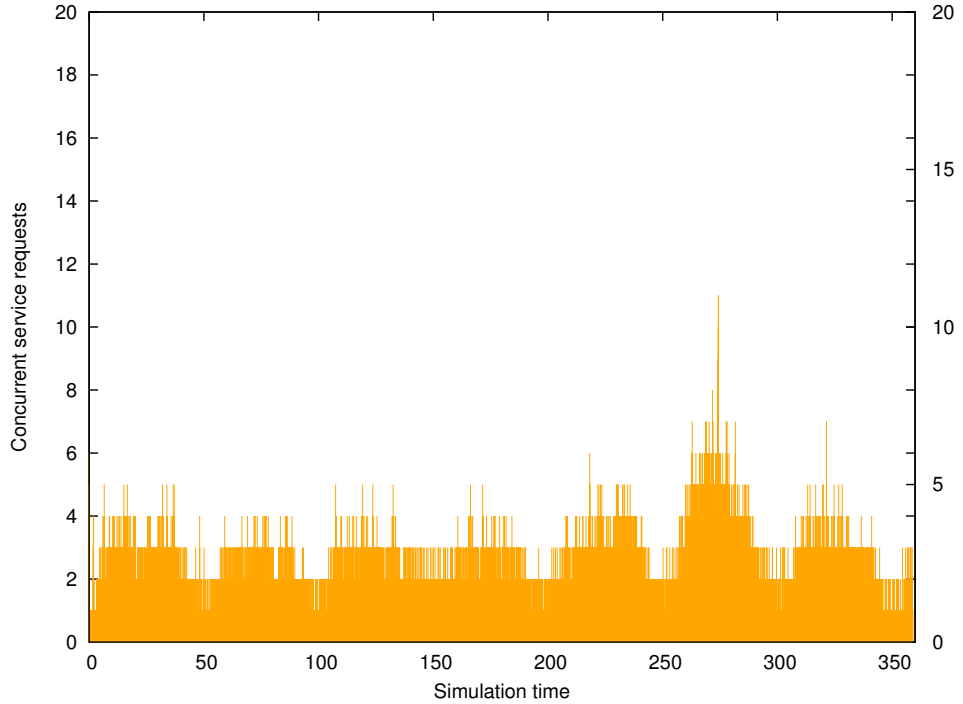
Results

- Simulation took around 18.1 seconds
- Reconfiguring the system lowered the peaks

Results (cont'd)

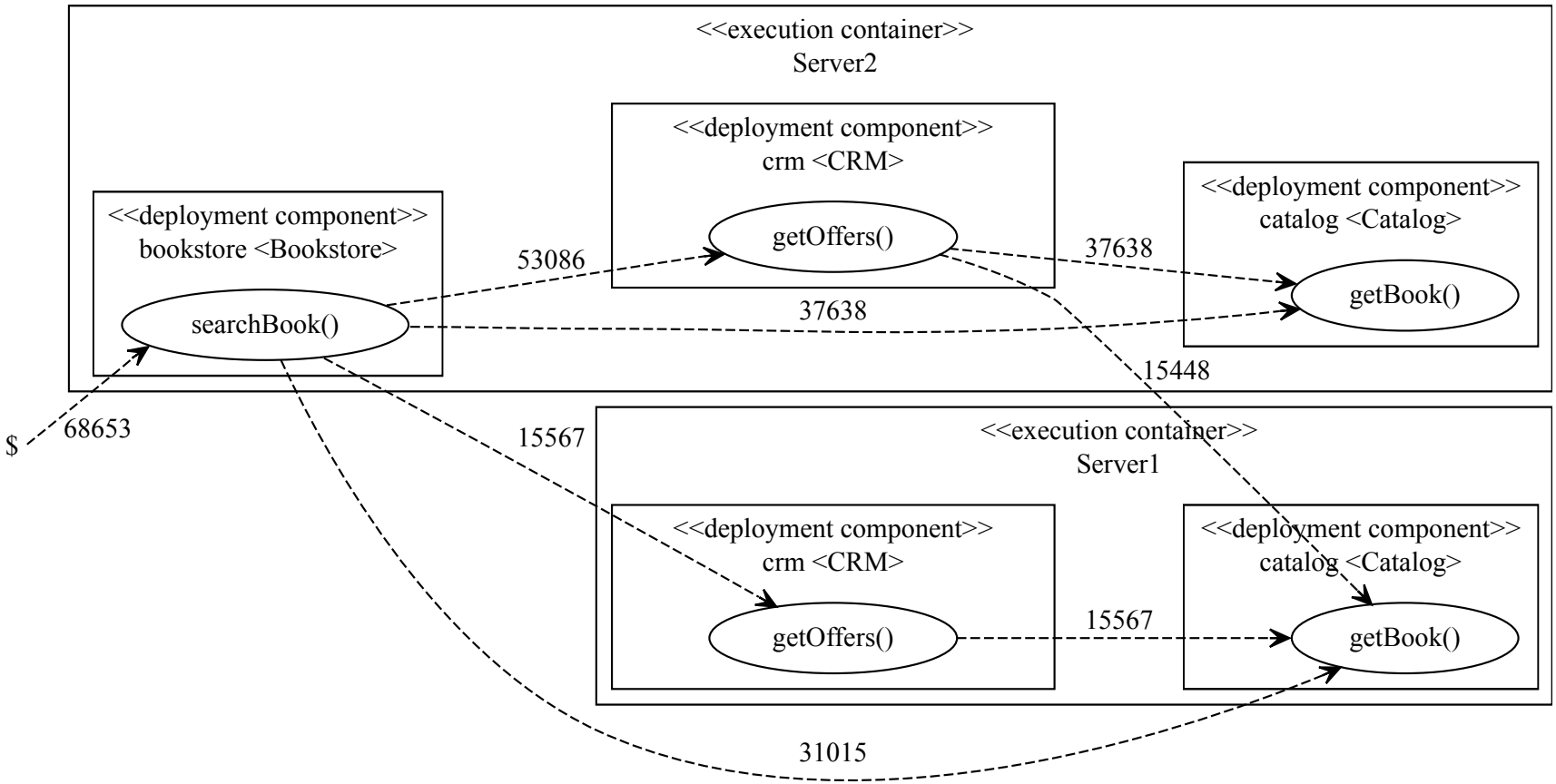


Response times (`searchBook`) & CPU utilization



Concurrent transactions

Results (cont'd)

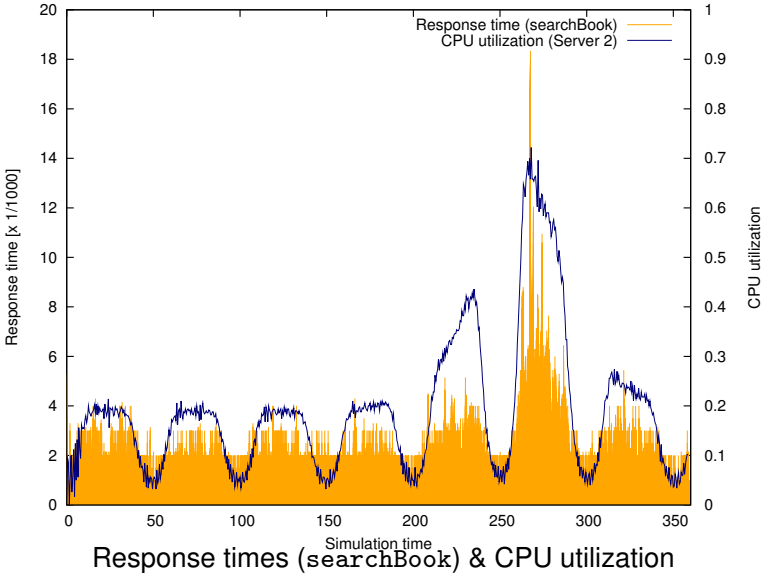


Reconstructed operation dependency graph (deployment view)

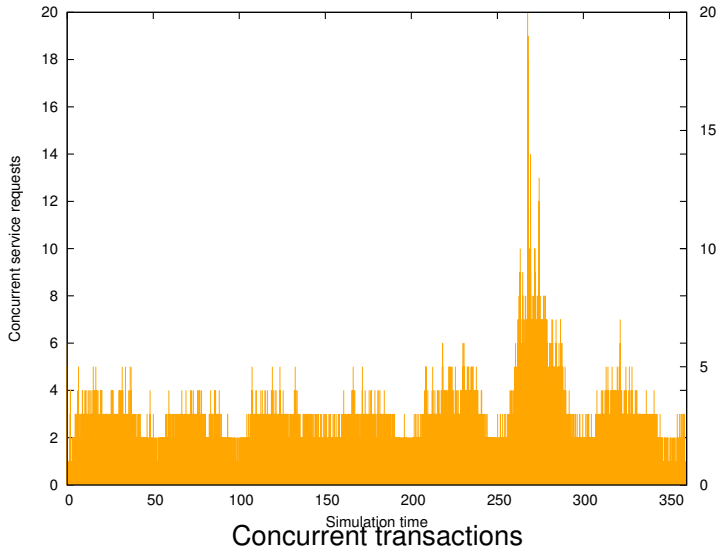
Comparison between Scenarios 1 & 2

Christian-Albrechts-Universität zu Kiel

Scenario 2 (w/o Reconfiguration):

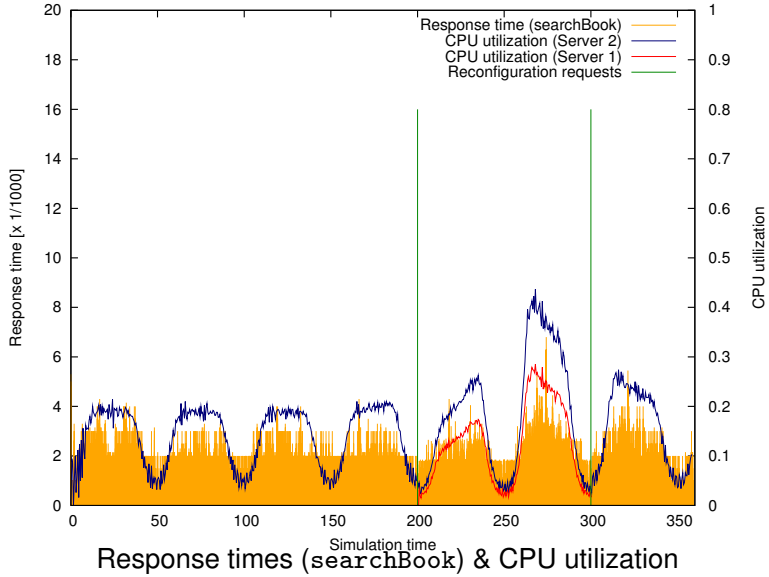


Response times (`searchBook`) & CPU utilization

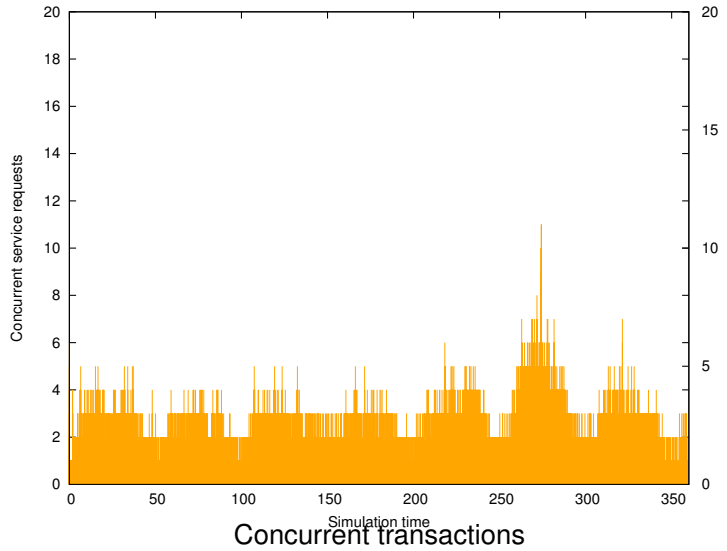


Concurrent transactions

Scenario 3 (w/ Reconfiguration):



Response times (`searchBook`) & CPU utilization



Concurrent transactions

Future Work

- Implement missing PCM features: Parametric stochastic expressions
- Evaluation with more complex ...
  - Models
  - Workloads (from production systems)
- New features
  - Cost model for reconfigurations (e.g., delays for allocation/replication/...)
  - Additional strategies for dispatching requests among replicas
  - Additional runtime reconfiguration operations: (e.g., replace implementation [Mat09, Bun08])
  - Additional QoS properties (e.g., reliability of execution containers)
- Dependency injection (Guice): heavier use & external configuration
- As usual: Performance ;-)

Literature

Steffen Becker, Heiko Koziolek, and Ralf Reussner.

The Palladio Component Model for model-driven performance prediction.
Journal of Systems and Software, 82(1):3–22, 2009.

Sven Bunge.

Transparentes Redeployment in komponentenbasierten Softwaresystemen (“transparent redeployment in component-based software systems”, in German), December 2008.
Diploma Thesis, University of Oldenburg.

Jasminka Matevska.

Architekturbasierte erreichbarkeitsoptimierte Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit.
PhD thesis, Department of Computer Science, University of Oldenburg, Oldenburg, Germany, July 2009.

Bernd Page and Wolfgang Kreutzer, editors.

The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java.
Shaker Verlag, 1. edition, 2005.

Ralf Reussner, Steffen Becker, Jens Happe, Heiko Koziolek, Klaus Krogmann, and Michael Kuperberg.

The Palladio Component Model.
Technical report, Univ. Karlsruhe (TH), June 2007.
`http://sdqweb.ipd.uka.de/wiki/Palladio_Component_Model`.

André van Hoorn.

Online Capacity Management for Increased Resource Efficiency of Software Systems.
PhD thesis, Dept. Comp. Sc., Univ. Oldenburg, Germany, 2011.
work in progress.

André van Hoorn, Matthias Rohr, Asad Gul, and Wilhelm Hasselbring.

An adaptation framework enabling resource-efficient operation of software systems.
In *Proc. Warm-Up Workshop for ACM/IEEE ICSE 2010 (WUP '09)*, pages 41–44. ACM, April 2009.

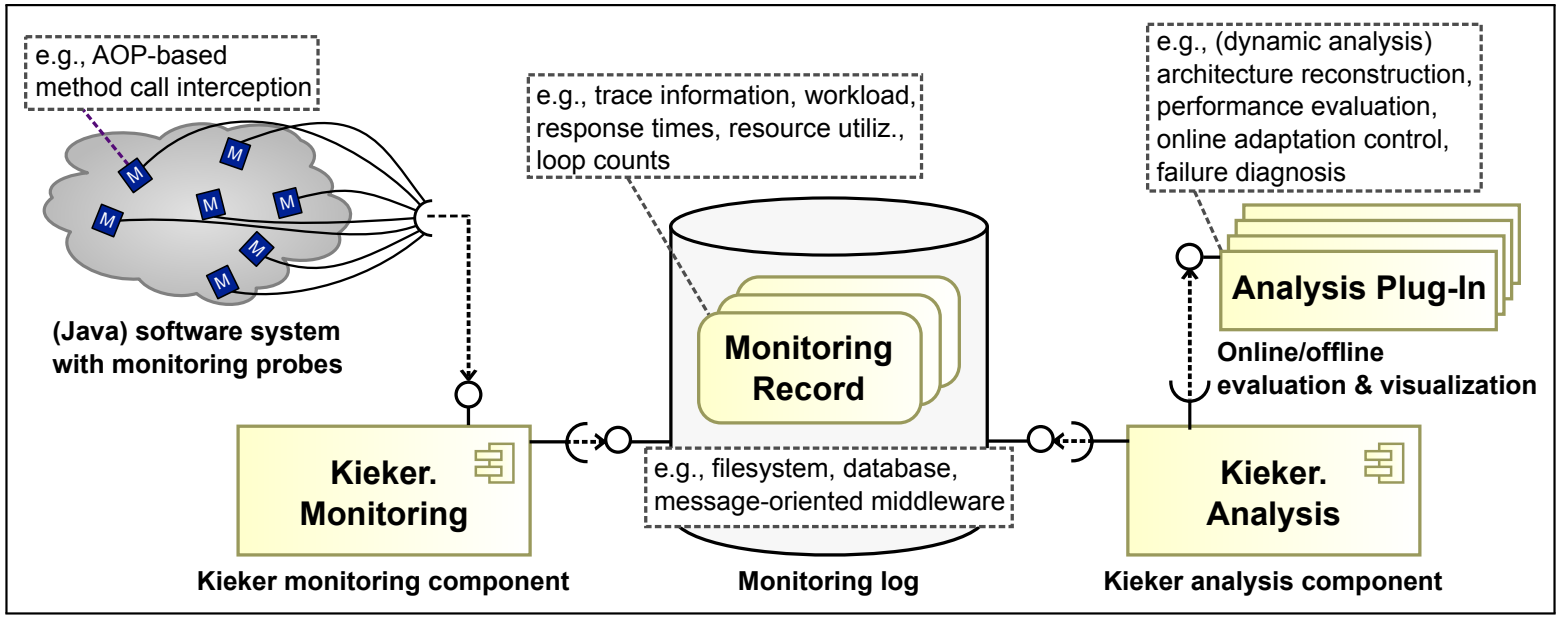
Literature (cont'd)

André van Hoorn, Matthias Rohr, Wilhelm Hasselbring, Jan Waller, Jens Ehlers, Sören Frey, and Dennis Kieselhorst.

Continuous monitoring of software services: Design and application of the Kieker framework.
Technical Report TR-0921, Dept. Comp. Sc., Univ. Kiel, Germany, November 2009.
http://www.informatik.uni-kiel.de/uploads/tx_publication/vanhoorn_tr0921.pdf.

Robert von Massow.

Performance simulation of runtime reconfigurable software architectures, April 2010.
Diploma Thesis, Univ. Oldenburg, Germany.



Core Characteristics [vHRH+09]

- Flexible architecture (custom *probes*, *readers*, *writers*, *analysis plug-ins*)
- Integrated & extensible *record* type model for monitoring & analysis
- Logging, reconstruction, analysis/visualization of (distributed) traces
- Low overhead (designed for continuous operation in multi-user systems)
- Evaluated in industry case studies

Kieker

http://kieker.sourceforge.net

Comparison with SimuCom (Scenario 1)

Setup

- Constant workload (log generated by script)
- Open workload with inter-arrival time 0.1
  $\longrightarrow$ yielding 100% simulated CPU utilization
- Simulation time: 1000 time units

Goal

- Comparison of results/performance between SLAstic.SIM & SimuCom

Results (Duration [ms] of 50 simulation runs)

	Min.	*Median*	*Mean*	*Max.*	*Dev.*
SimuCom	6434	7179	7199	7873	287.24
SLAstic.SIM	4864	5325	5333	5833	161.25

(Duration excl. code generation (SimuCom) & initialization (SLAstic.SIM))