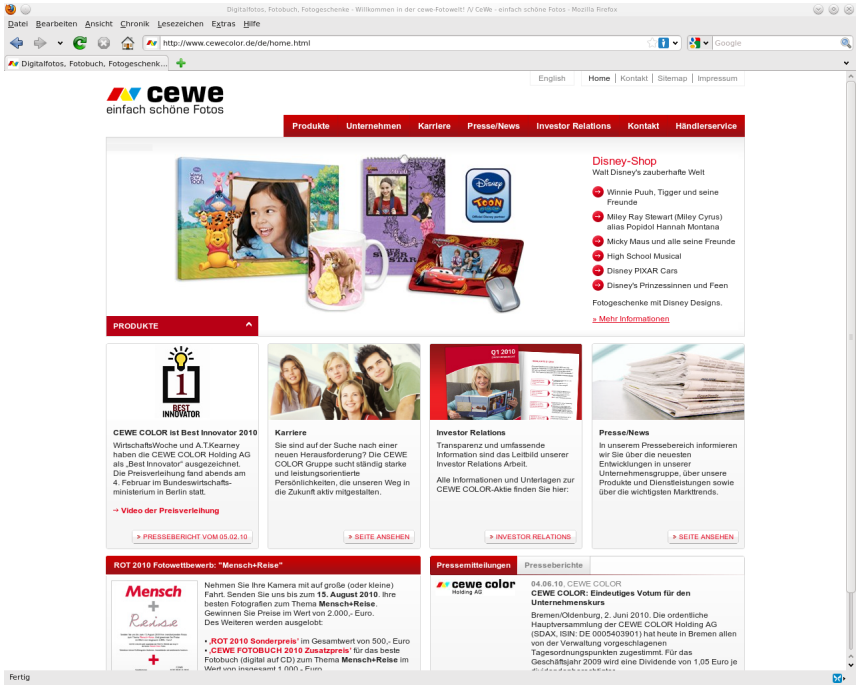
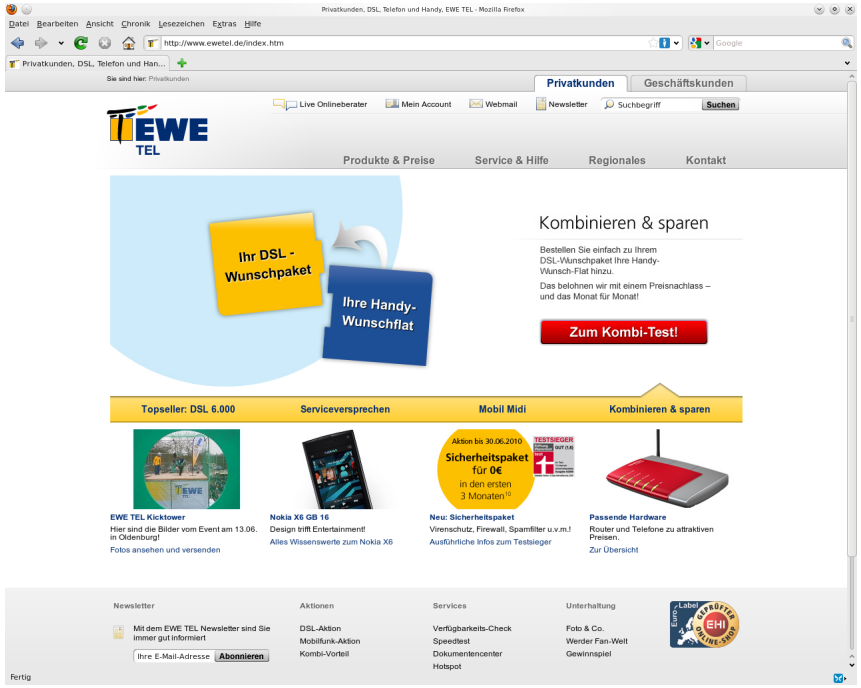
Utilizing PCM for Online Capacity Management of Component-Based Software Systems

André van Hoorn

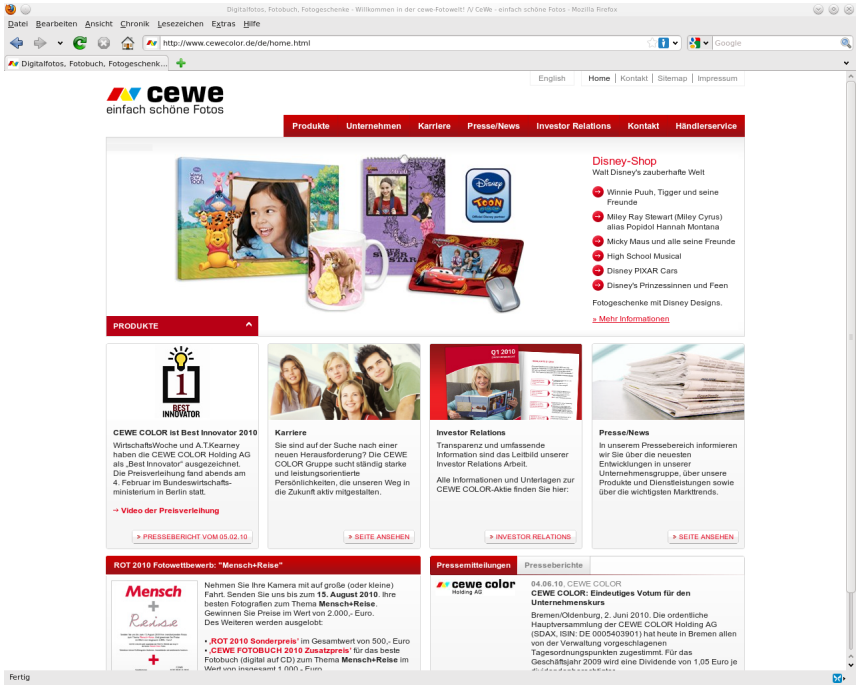
Software Engineering Group, University of Kiel
`http://se.informatik.uni-kiel.de/`

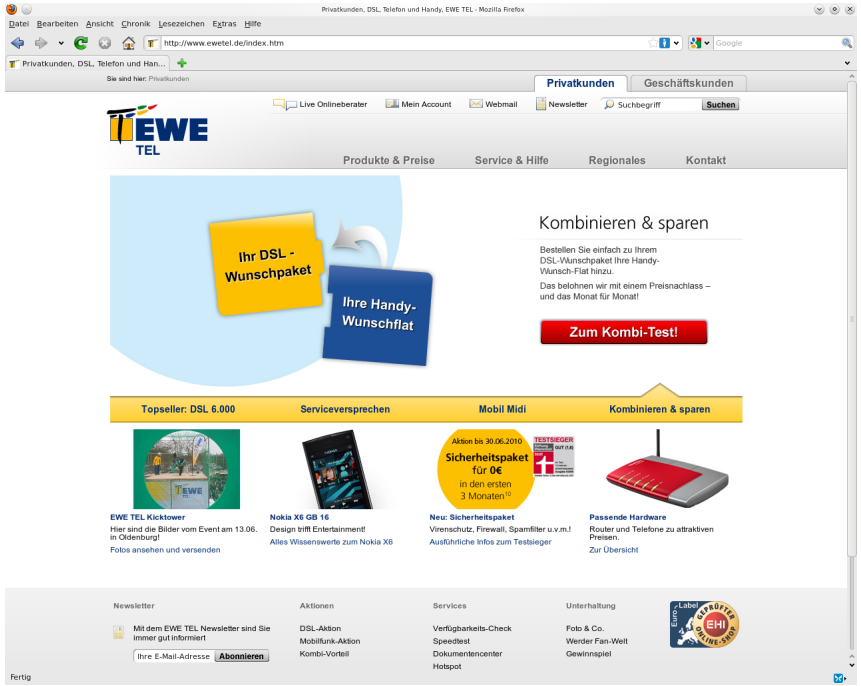
Nov. 18, 2011 @ Palladio Days, Karlsruhe



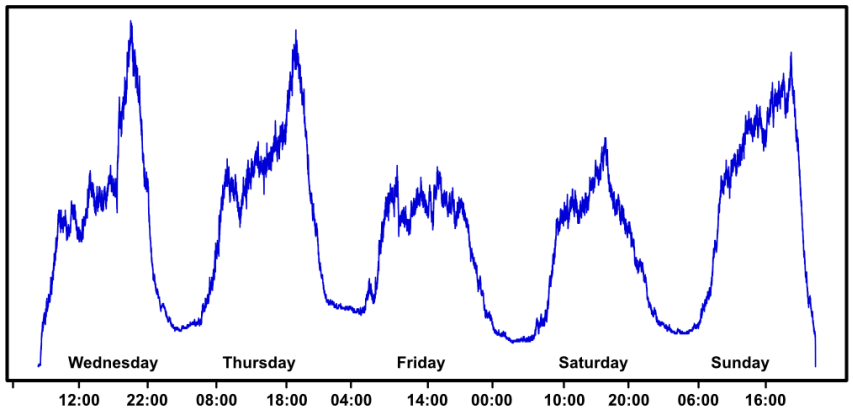


- Business-critical software systems



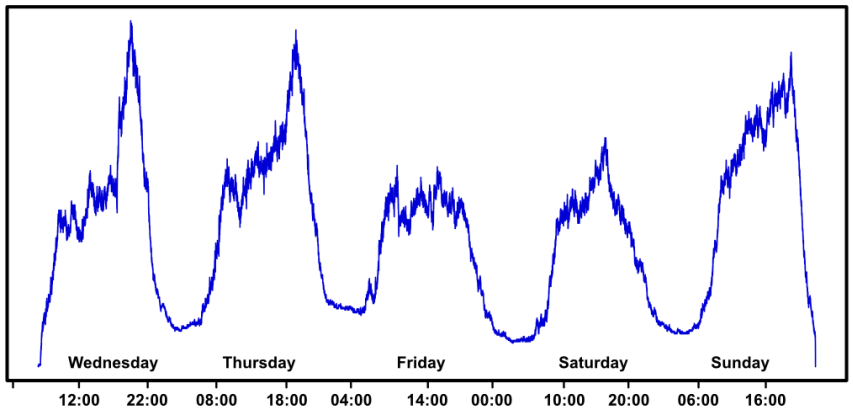


- Business-critical software systems
- Quality of service (performance, availability, . . .)



- 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

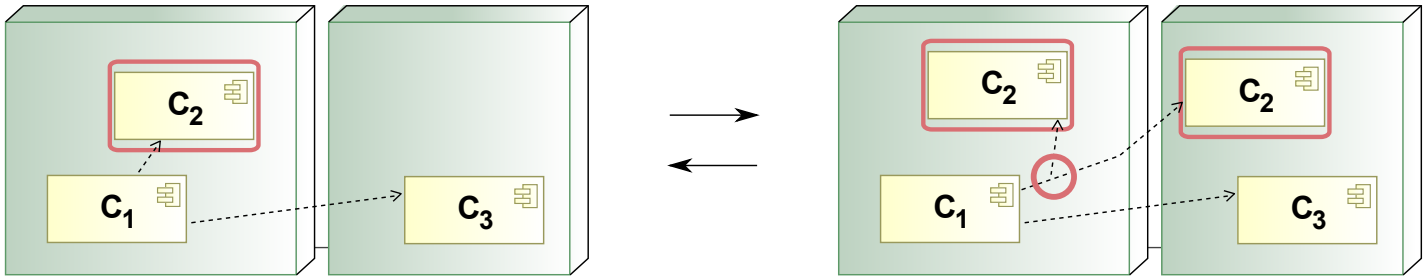
- **SLAstic** [van Hoorn et al. 2009a, van Hoorn 2011]:
  Online capacity management employing architecture-based runtime reconfiguration

Agenda

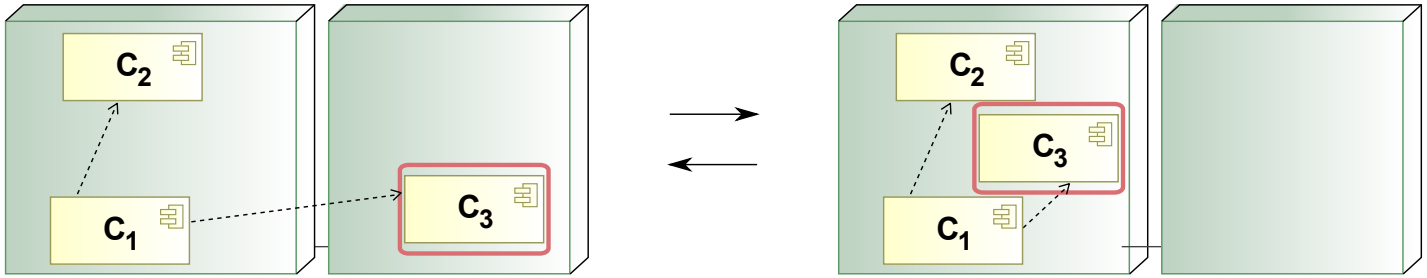
SLAstic Runtime Reconfiguration Operations

Christian-Albrechts-Universität zu Kiel

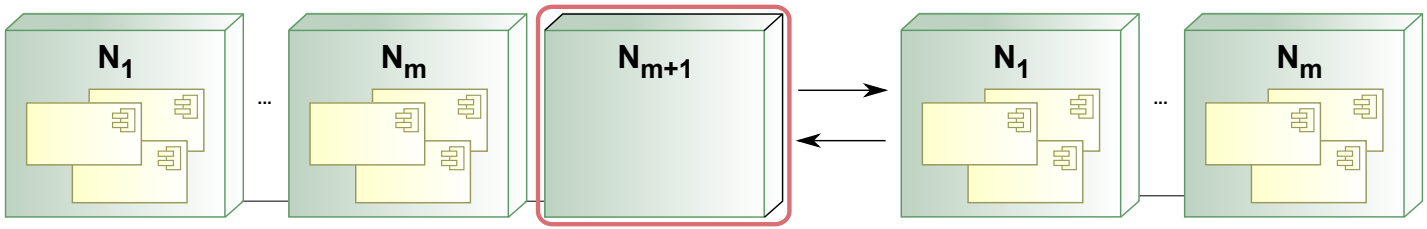
1 (De-)Replication of Software Components



2 Migration of Software Components



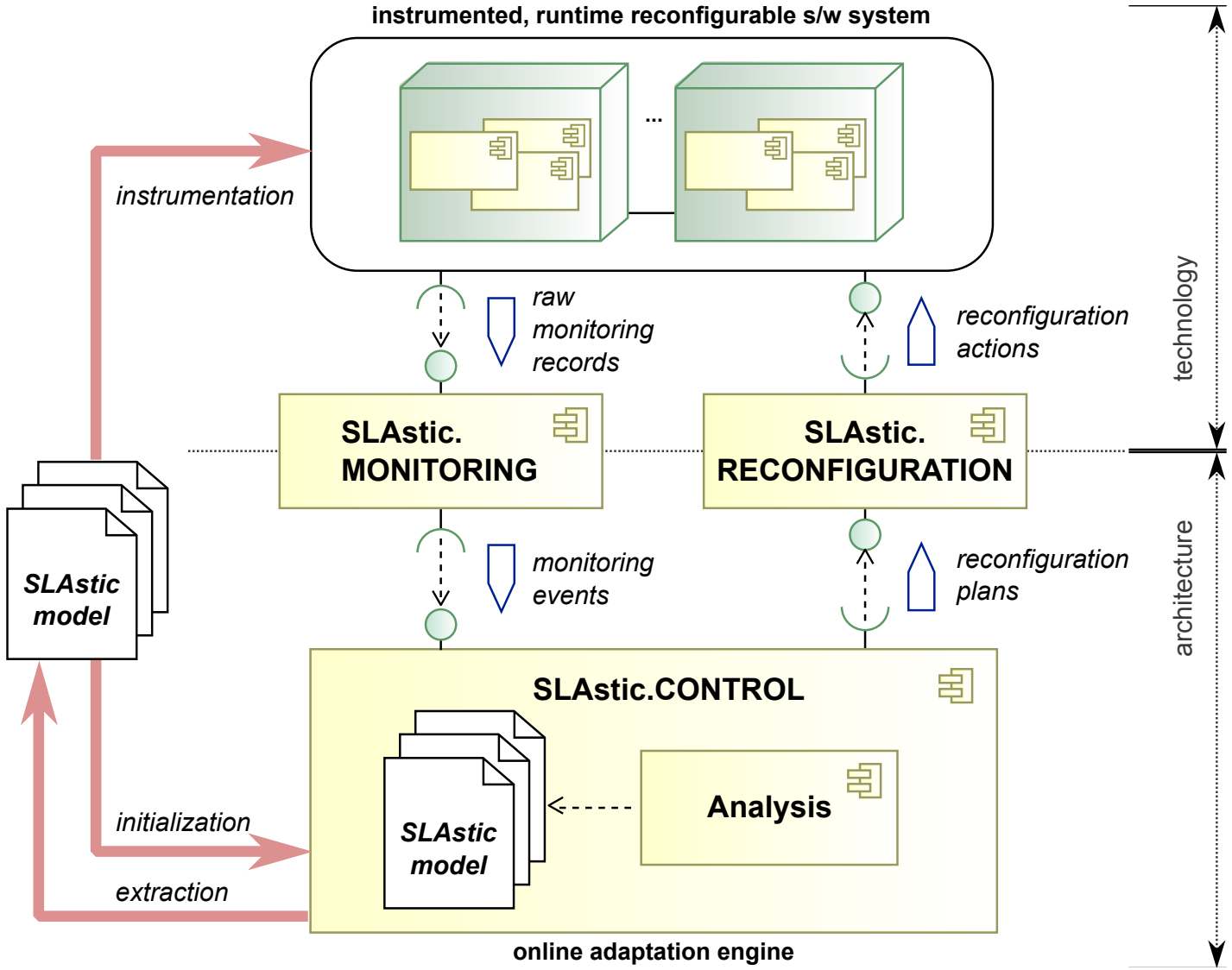
3 (De-)Allocation of Execution Containers

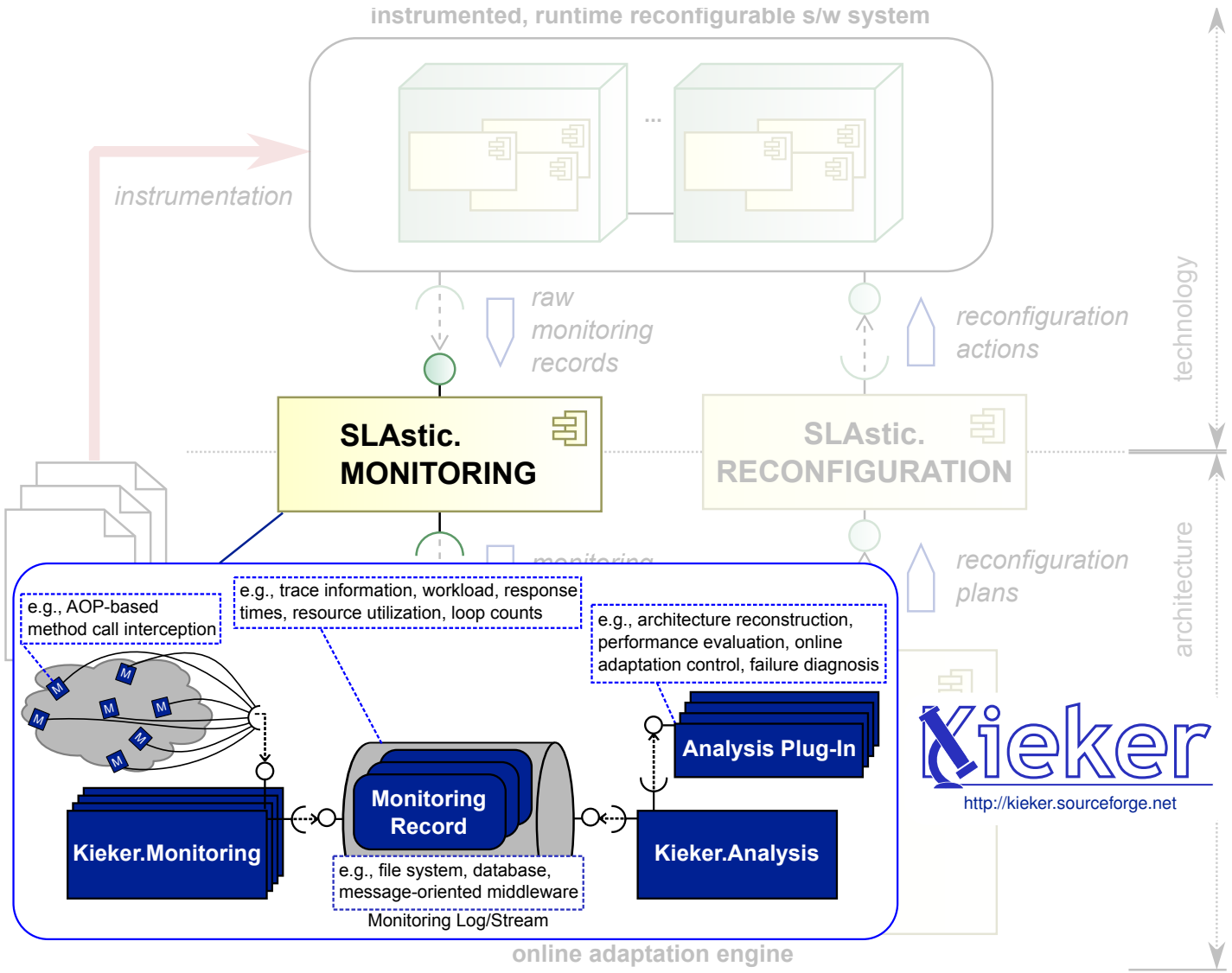


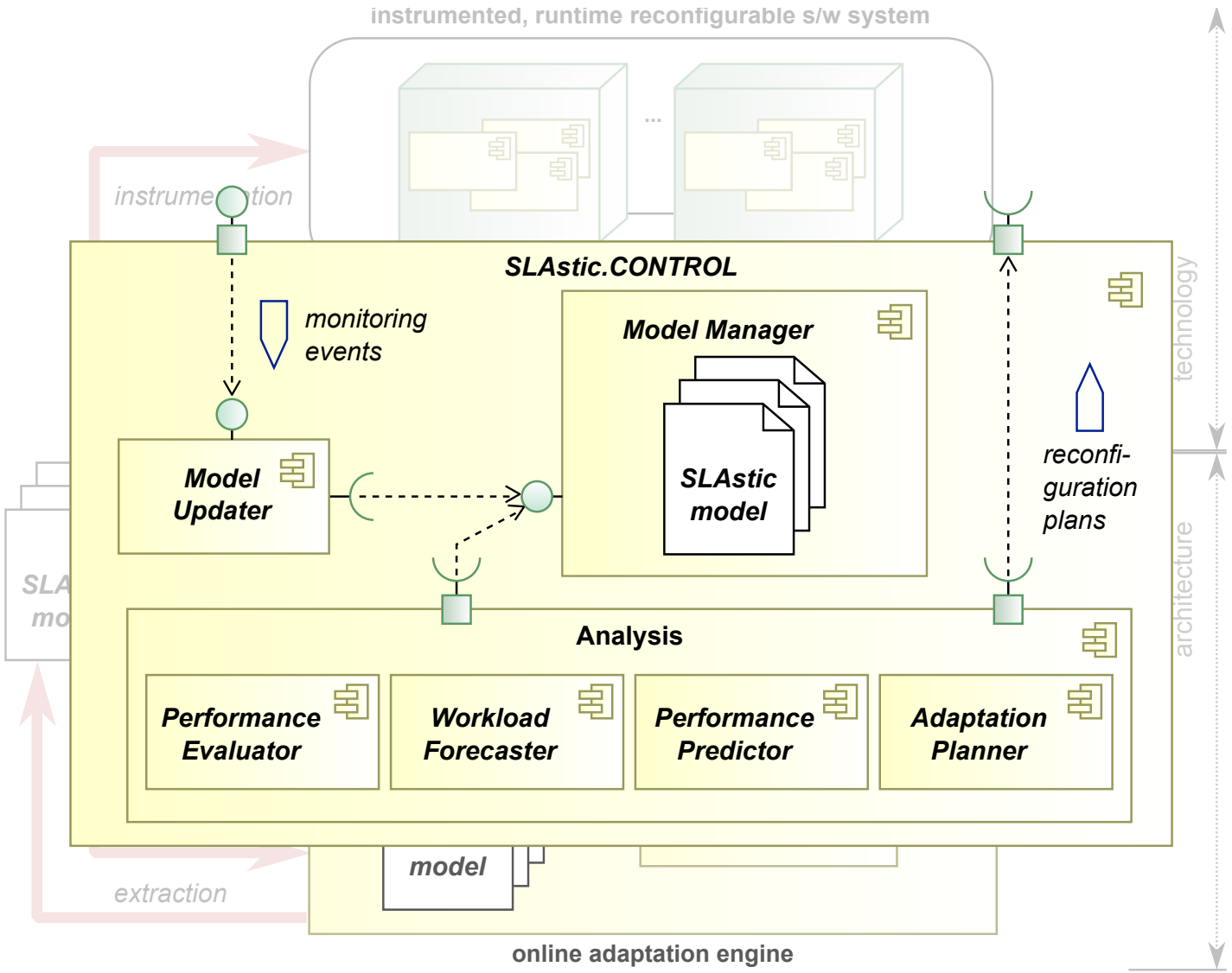
SLAstic Runtime Reconfiguration Operations

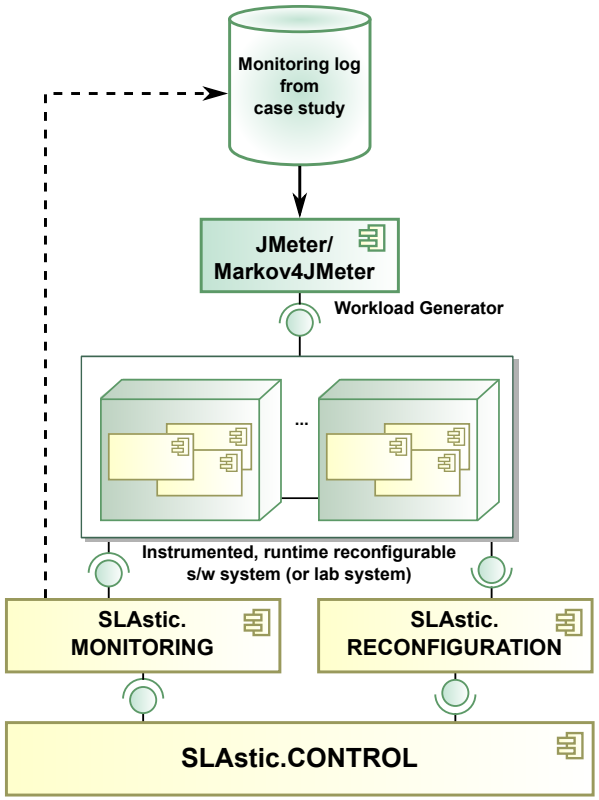
Christian-Albrechts-Universität zu Kiel

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
   - deallocate (container: ExecutionContainer)
   - allocate (containerType: ExecutionContainerType)

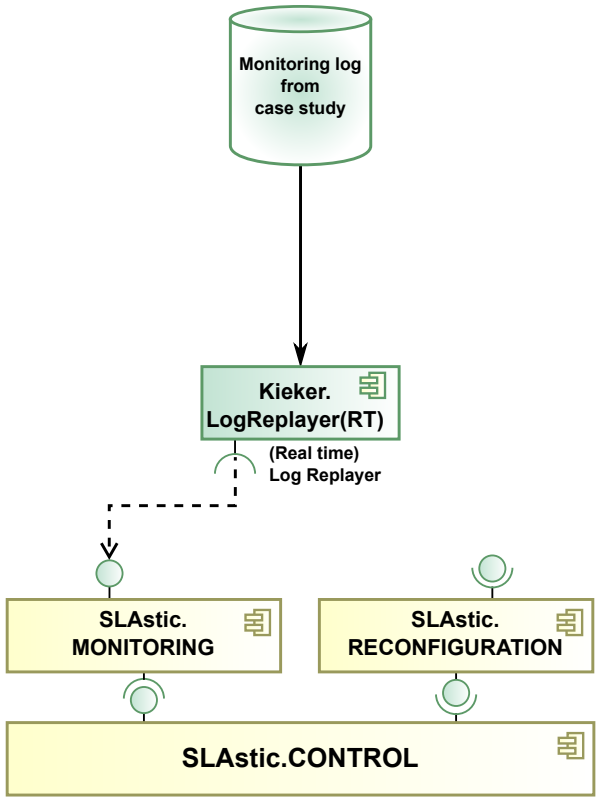

instrumented, runtime reconfigurable s/w system
...
instrumentation
raw monitoring records
reconfiguration actions
technology
SLAstic. MONITORING
SLAstic. RECONFIGURATION
SLAstic model
monitoring events
reconfiguration plans
architecture
SLAstic.CONTROL
SLAstic model
Analysis
initialization
extraction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


instrumented, runtime reconfigurable s/w system
instrumentation
raw monitoring records
reconfiguration actions
SLAstic. MONITORING
SLAstic. RECONFIGURATION
technology
reconfiguration plans
architecture
e.g., AOP-based method call interception
e.g., trace information, workload, response times, resource utilization, loop counts
e.g., architecture reconstruction, performance evaluation, online adaptation control, failure diagnosis
Analysis Plug-In
Kieker
http://kieker.sourceforge.net
Kieker.Monitoring
Monitoring Record
Kieker.Analysis
e.g., file system, database, message-oriented middleware
Monitoring Log/Stream
online adaptation engine

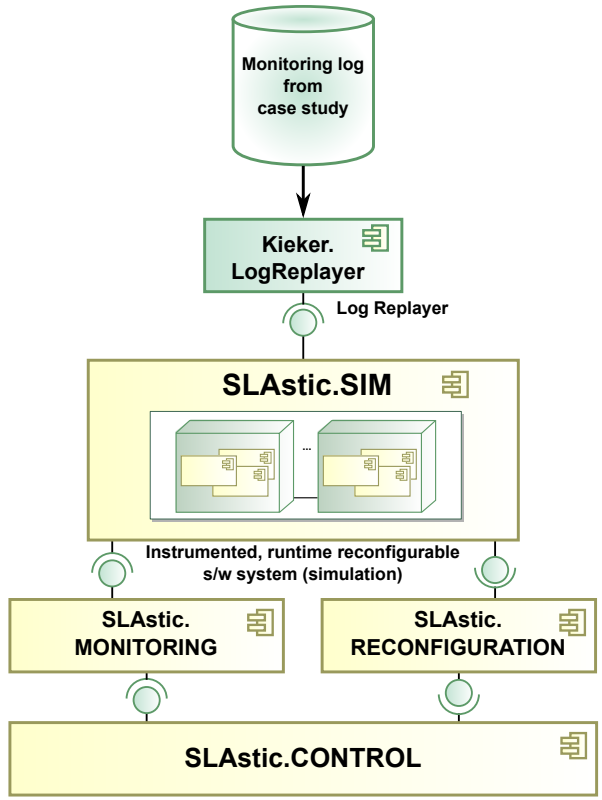

instrumented, runtime reconfigurable s/w system
...
instrumentation
SLAstic.CONTROL
monitoring events
Model Manager
SLAstic model
Model Updater
reconfi-guration plans
technology
architecture
Analysis
Performance Evaluator
Workload Forecaster
Performance Predictor
Adaptation Planner
model
extraction
online adaptation engine



1 Online analysis (production/lab system)



2 Offline analysis (log replay)

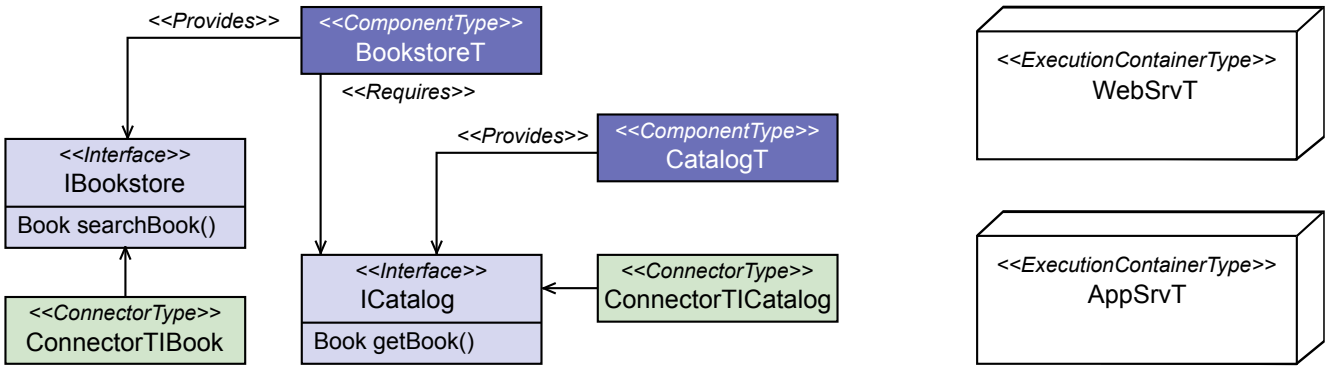


3 Simulation-based analysis

Christian-Albrechts-Universität zu Kiel

System Partition (also used as runtime model)

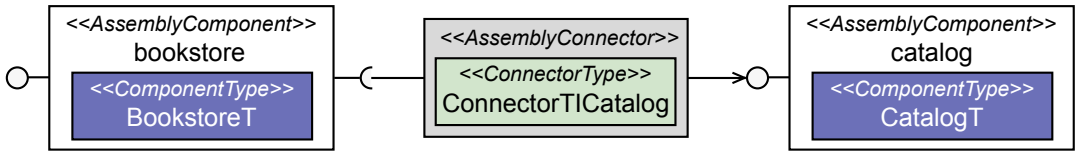
1. Type repository (e.g., component types, interfaces, connector types, execution container types)



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)

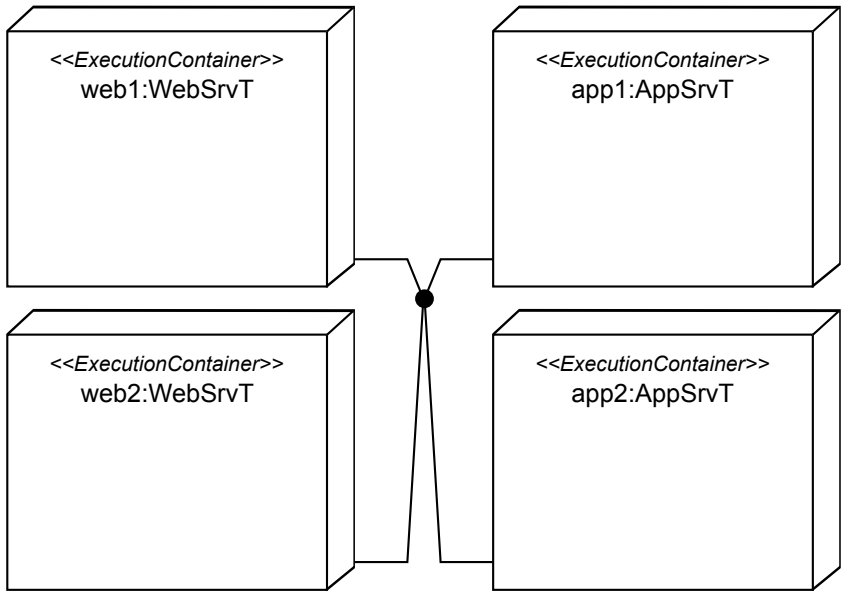


Example component assembly

Christian-Albrechts-Universität zu Kiel

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)

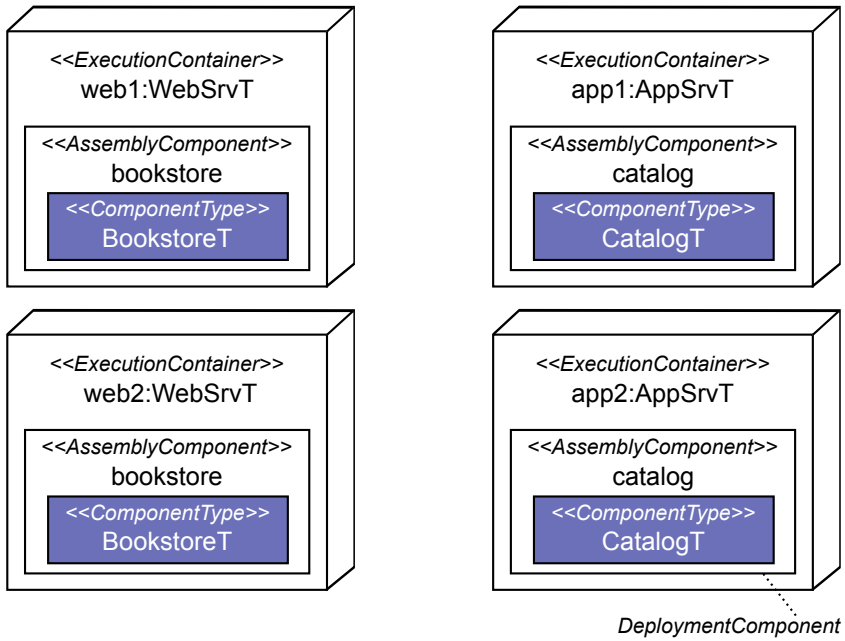


Example execution environment

Christian-Albrechts-Universität zu Kiel

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

Christian-Albrechts-Universität zu Kiel

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)

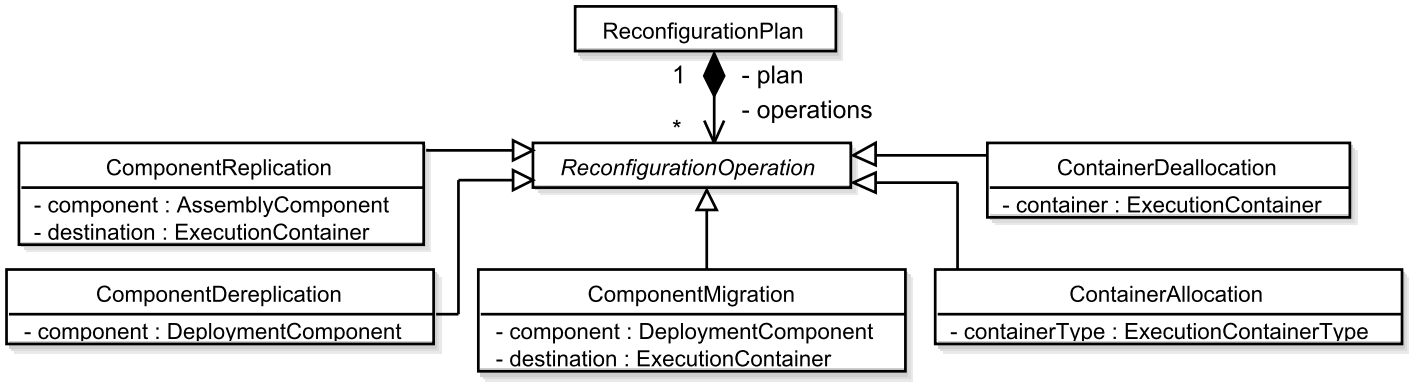
SLAstic2PCM transforms SLAstic model to PCM [Günther 2011]

—

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.

Completions/Decorations

- Adaptation / Reconfiguration (e.g., plans, operations, capabilities, properties)



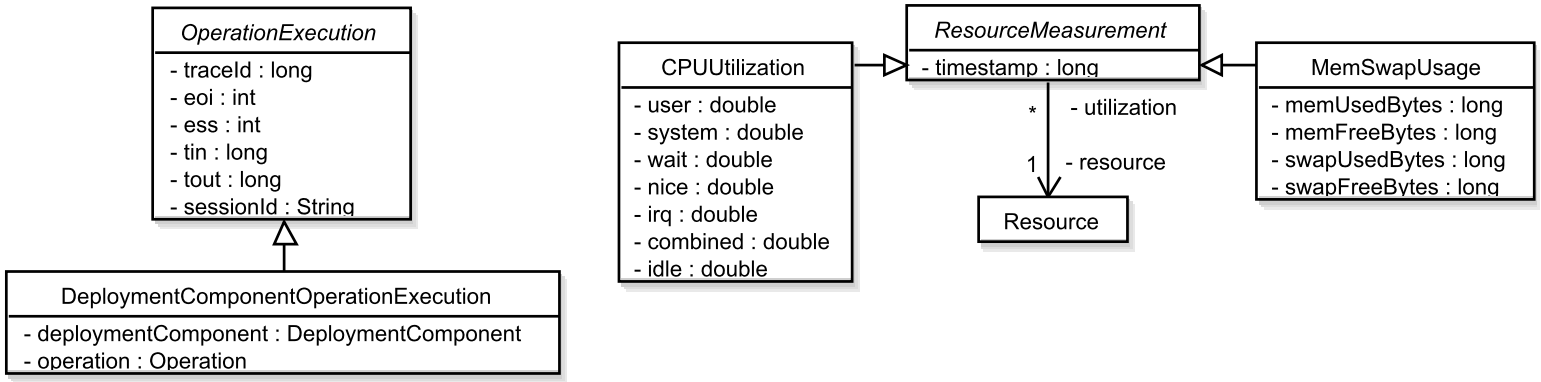
PCM-specific implementation of operations part of SLAstic.SIM [von Massow et al. 2011]

—

R. von Massow, A. van Hoorn, and W. Hasselbring. *Performance simulation of runtime reconfigurable component-based software architectures.* In Proc. ECSA '11, vol. 6903 of LNCS, pages 43–58. Springer, Sept. 2011
(As presented at Palladio Days 2010)

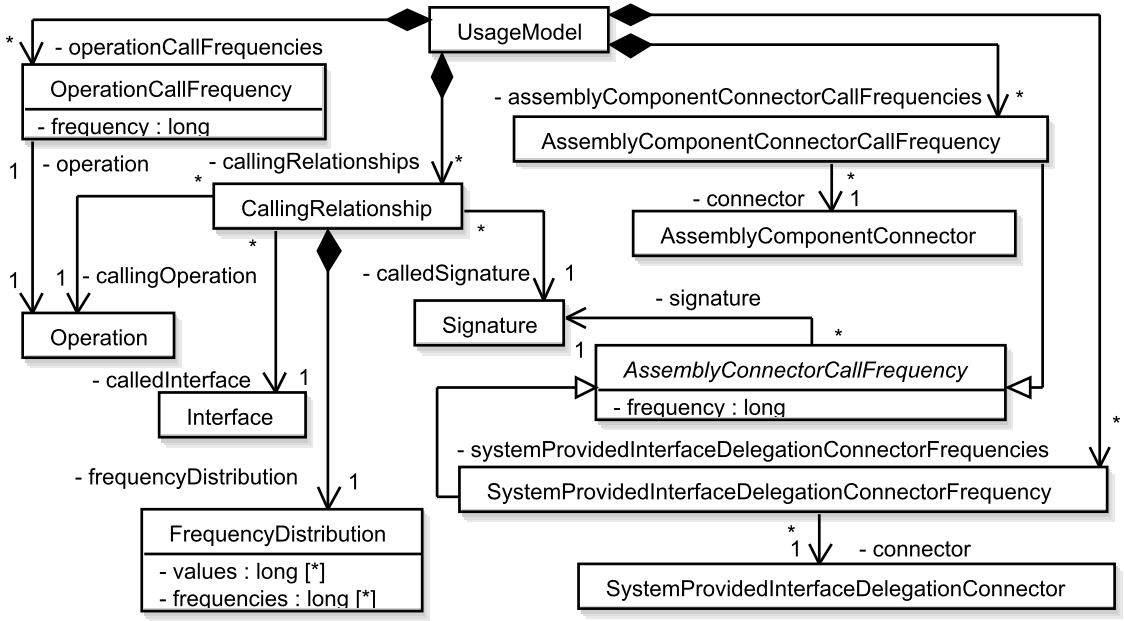
Completions/Decorations

- Adaptation / Reconfiguration (e.g., plans, operations, capabilities, properties)
- Measurement (e.g., workload, timing, utilization)



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

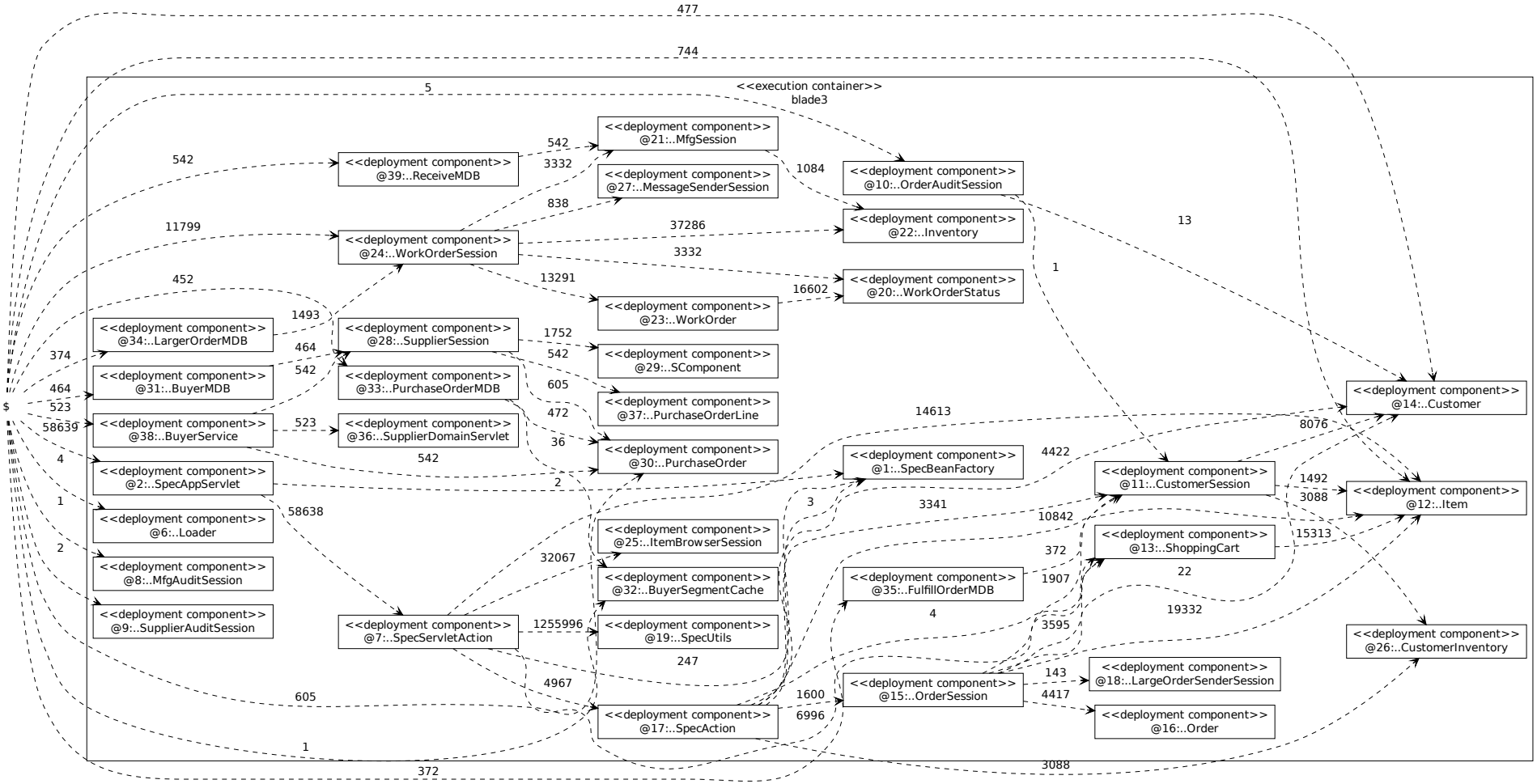
- Adaptation / Reconfiguration (e.g., plans, operations, capabilities, properties)
- Measurement (e.g., workload, timing, utilization)
- Usage (e.g., operation call frequencies, calling relationships)
- . . . — e.g., “cost” profiles (not yet implemented)

Agenda

Model Extraction by Dynamic Analysis

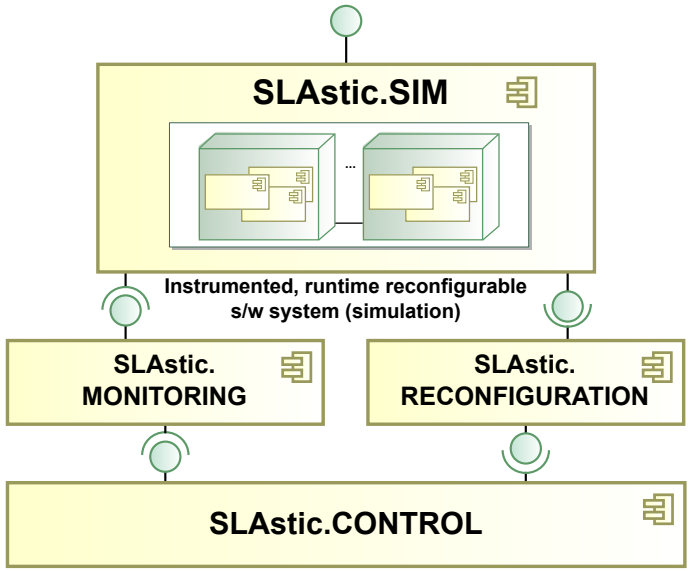
C A U
Christian-Albrechts-Universität zu Kiel

1. Raw monitoring records delivered by Kieker, e.g.
   - `OperationExecutionRecord` (incl. timing & tracing information)
   - `ResourceUtilizationRecord` and `MemSwapUsageRecord`
2. Processing of records by SLAstic.MONITORING
   1. Possible abstraction of component/operation names etc.
   2. Lookup/creation of architectural entities using `ModelManager`
   3. Transformation of raw monitoring records into monitoring events
   4. Send monitoring events to complex-event processing (CEP) engine
3. Trace reconstruction
   1. `TraceReconstructor` registers match/recognize CEP statement to collect executions grouped by trace ID (online)
   2. Send each reconstructed message trace to CEP engine
4. Update of system and usage model
   1. `UsageAndSystemModelUpdater` registers CEP statement to collect valid message traces
   2. Process each trace and update models using `ModelManager`


477
744
5
<<execution container>>
blade3
<<deployment component>>
@21:..MfgSession
<<deployment component>>
@27:..MessageSenderSession
<<deployment component>>
@39:..ReceiveMDB
<<deployment component>>
@10:..OrderAuditSession
<<deployment component>>
@22:..Inventory
<<deployment component>>
@24:..WorkOrderSession
<<deployment component>>
@20:..WorkOrderStatus
<<deployment component>>
@23:..WorkOrder
<<deployment component>>
@34:..LargerOrderMDB
<<deployment component>>
@28:..SupplierSession
<<deployment component>>
@29:..SComponent
<<deployment component>>
@31:..BuyerMDB
<<deployment component>>
@33:..PurchaseOrderMDB
<<deployment component>>
@37:..PurchaseOrderLine
<<deployment component>>
@38:..BuyerService
<<deployment component>>
@36:..SupplierDomainServlet
<<deployment component>>
@30:..PurchaseOrder
<<deployment component>>
@2:..SpecAppServlet
<<deployment component>>
@1:..SpecBeanFactory
<<deployment component>>
@11:..CustomerSession
<<deployment component>>
@14:..Customer
<<deployment component>>
@12:..Item
<<deployment component>>
@6:..Loader
<<deployment component>>
@25:..ItemBrowserSession
<<deployment component>>
@13:..ShoppingCart
<<deployment component>>
@8:..MfgAuditSession
<<deployment component>>
@32:..BuyerSegmentCache
<<deployment component>>
@35:..FulfillOrderMDB
<<deployment component>>
@9:..SupplierAuditSession
<<deployment component>>
@7:..SpecServletAction
<<deployment component>>
@19:..SpecUtils
<<deployment component>>
@26:..CustomerInventory
<<deployment component>>
@18:..LargeOrderSenderSession
<<deployment component>>
@15:..OrderSession
<<deployment component>>
@17:..SpecAction
<<deployment component>>
@16:..Order
$
542
3332
1084
11799
838
37286
13
3332
452
13291
16602
1
1493
1752
374
464
542
542
605
464
523
472
14613
58639
523
36
8076
4
542
4422
2
1492
1
3
3341
3088
58638
10842
15313
2
32067
372
22
1907
4
19332
1255996
3595
247
4967
143
605
1600
4417
6996
1
372
3088

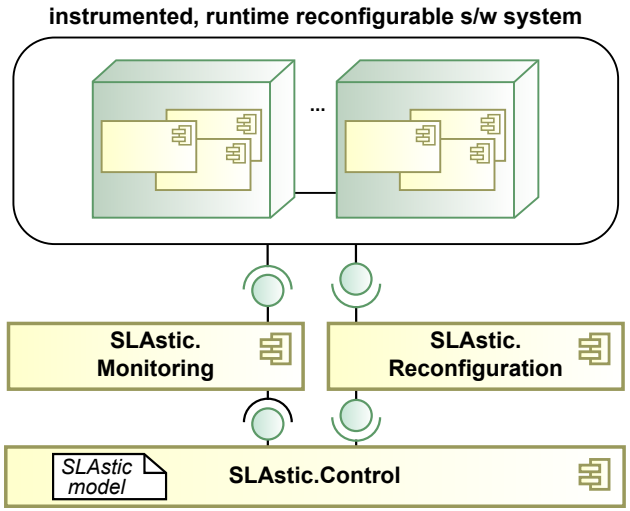
Utilizing PCM in SLAstic

1. Simulation-based analysis
   - SLAstic.SIM as substitute for real system
   - Evaluation of approach, adaptation plans etc.
2. PCM @ Runtime
   - PCM instance decorated by SLAstic model
3. Online performance prediction (e.g., SLAstic.SIM)



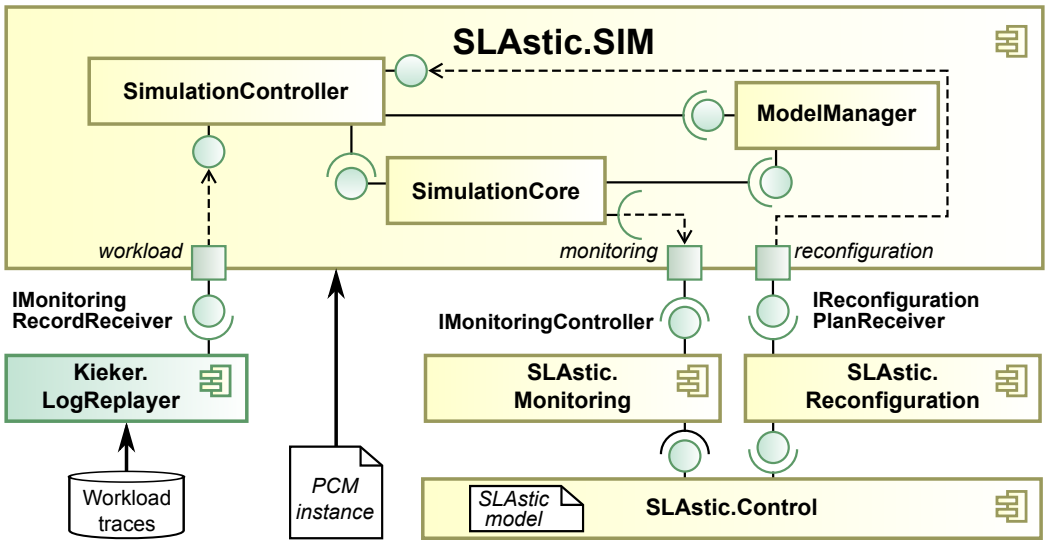
Semi-automated extraction of PCM instances

1. Extraction of SLAstic instance by dyn. analysis
2. SLAstic2PCM transformation [Günther 2011]
3. *Assumption:*
   Manual refinement/calibration (offline)



—

R. von Massow, A. van Hoorn, and W. Hasselbring. *Performance simulation of runtime reconfigurable component-based software architectures.* In Proc. ECSA '11, vol. 6903 of LNCS, pages 43–58. Springer, Sept. 2011
(As presented at Palladio Days 2010)

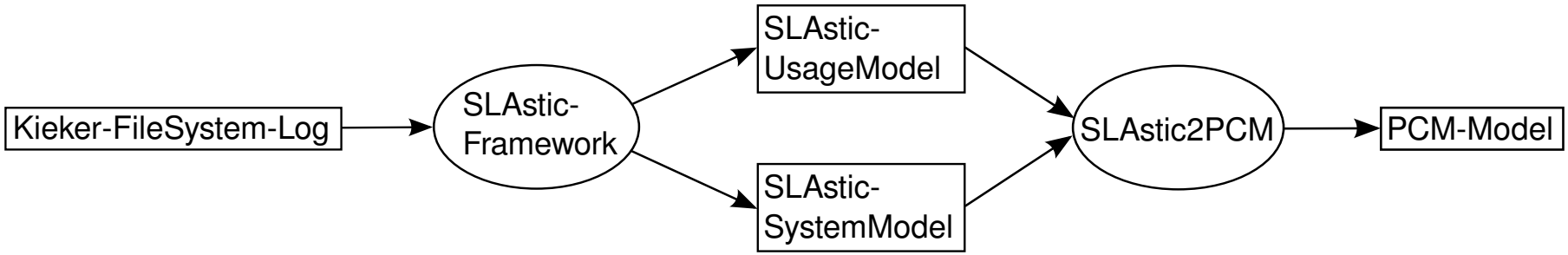


- Simulation driven by recorded (varying) workload
- Simulation of PCM models with SLAstic's reconfiguration capabilities
- (limitations in terms of supported PCM features)

—

R. von Massow, A. van Hoorn, and W. Hasselbring. *Performance simulation of runtime reconfigurable component-based software architectures.* In Proc. ECSA '11, vol. 6903 of LNCS, pages 43–58. Springer, Sept. 2011
(As presented at Palladio Days 2010)

- SLAstic2PCM implemented as an ATL-based M2M transformation

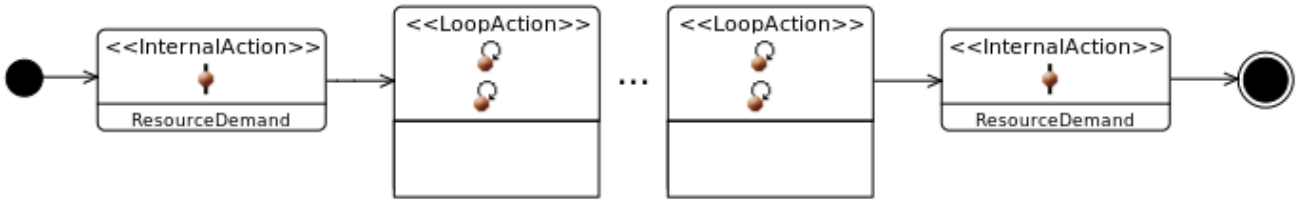


1. Transformation of SLAstic *System* Model to PCM counter parts pretty straight-forward

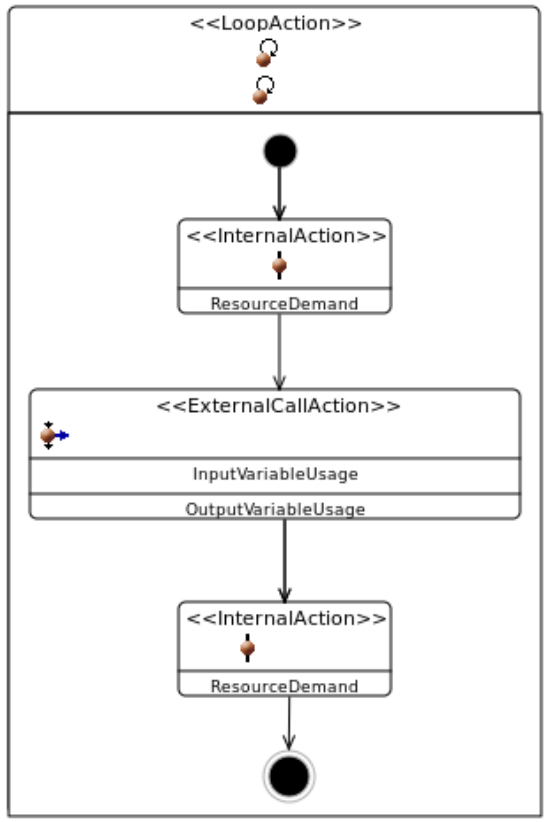
SLAstic	PCM
TypeRepository	Repository, ResourceRepository
ComponentAssembly	System
ExecutionEnvironment	ResourceEnvironment
ComponentDeployment	Allocation

2. Generation of RDSEFFs (detailed on next slide) and Usage Models
   - RDSEFFs: Generated from `CallingRelationships` and `OperationCallFrequencys`
   - Usage Scenario (open workload) based on `SystemProvidedInterfaceDelegationConnectorFrequencies` and observation period

For each `Operation`



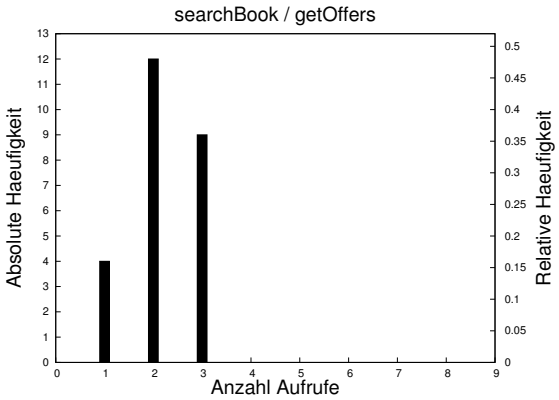
1. Create `StartAction`
2. Create `InternalAction` for initialization
3. Create `LoopAction` for each `Interface Signature` called by `Operation`
   1. Create `StartAction` + `InternalAction` for initialization
   2. Create `ExternalCallAction`
   3. Create `InternalAction` for termination + `StopAction`
4. Create `InternalAction` for termination
5. Create `StopAction`

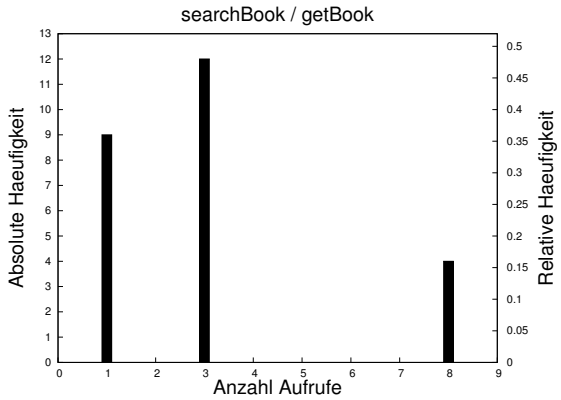


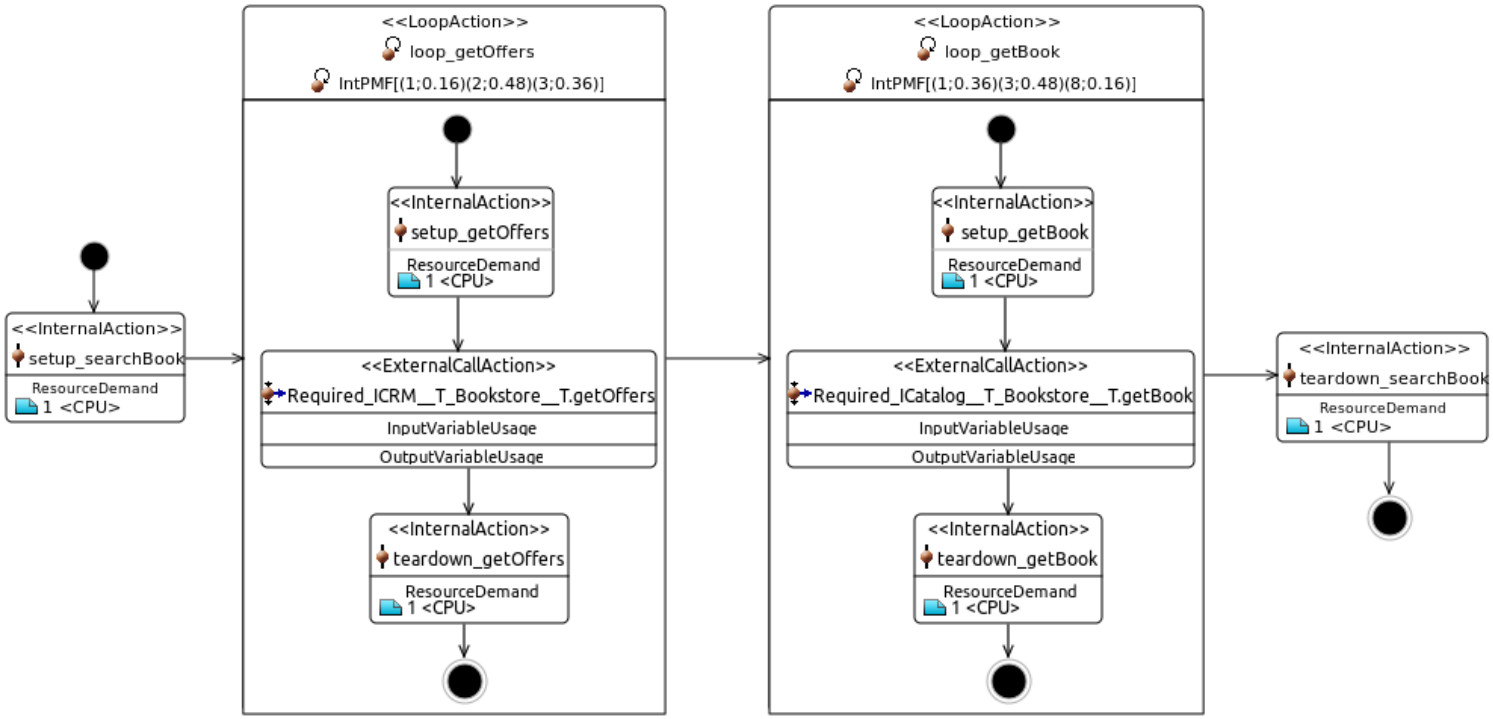
Extraction of RDSEFFs (Example)

SLAstic2PCM Transformation (cont'd)

Christian-Albrechts-Universität zu Kiel







Scalability of the Transformation

- Kieker file system log

System	Size of Monitoring Log	# Operation Executions	# Traces
Bookstore	892 KB	6,540	1,635
JPetStore	63 MB	259,852	48,720
Customer Portal	164 MB	847,758	316,980
SPECjEnterprise	380 MB	189,4830	75,018

- SLAstic model metrics

System	# Components	# Signatures	# Containers	# CallingRelationships	# value pairs
Bookstore	3	6	2	3	3
JPetStore	9	42	1	25	25
Customer Portal	11	212	4	97	174
SPECjEnterprise	39	272	1	198	463

- Durations of SLAstic2PCM transformations

System	Mean duration (seconds)	Std dev.
Bookstore	0.9985	0.049
JPetStore	1.177	0.043
Customer Portal	2.2608	0.032
SPECjEnterprise	4.9192	0.101

Summary

- Outlined SLAstic approach: Online capacity management of CB systems
- Architectural models used @ Runtime (PCM by decoration)
- Extraction of SLAstic ($\rightarrow$ SLAstic2PCM $\rightarrow$ PCM) models via dynamic analysis
- Extraction step evaluated for larger systems already
- Supported languages (thanks to Kieker's language support):
  - Java,
  - .NET,
  - Visual Basic 6
  - (in progress: COBOL, . . .)

Future Work

- Extend SLAstic2PCM to produce SLAstic/PCM decorator model
- Basic extraction of resource demands for PCM RDSEFFs
- Evaluate quality of extracted models
- Resolve some of SLAstic.SIM's limitations (e.g., PMF support in loops)

S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1): 3–22, 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2008.03.066. Special Issue: Software Performance – Modeling and Analysis.

P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley / Pearson Education, 2002. ISBN 0201703726.

D. Garlan, S.-W. Cheng, and B. R. Schmerl. Increasing system dependability through architecture-based self-repair. In R. de Lemos, C. Gacek, and A. B. Romanovsky, editors, *Architecting Dependable Systems*, volume 2677 of *Lecture Notes in Computer Science*, pages 61–89. Springer, 2003. ISBN 978-3-540-40727-0. doi: 10.1007/3-540-45177-3_3.

I. Gorton, Y. Liu, and N. Trivedi. An extensible and lightweight architecture for adaptive server applications. *Softw. Pract. Exper.*, 38(8):853–883, 2008. ISSN 0038-0644. doi: http://dx.doi.org/10.1002/spe.v38:8.

V. Grassi, R. Mirandola, and A. Sabetta. A model-driven approach to performability analysis of dynamically reconfigurable component-based systems. In *Proceedings of the 6th International Workshop on Software and Performance (WOSP '07)*, pages 103–114. ACM, 2007. ISBN 1-59593-297-6. doi: 10.1145/1216993.1217011.

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.

C. Hofmeister. *Dynamic Reconfiguration of Distributed Applications*. PhD thesis, University of Maryland, 1993.

IBM. IBM Autonomic Computing. http://www.ibm.com/autonomic/, 2010.

IEEE. IEEE recommended practice for architectural description of software-intensive systems—std. 1471-2000, 2000.

R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, New York, 1991.

J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, Jan. 2003.

S. Kounev, F. Brosig, N. Huber, and R. Reussner. Towards self-aware performance and resource management in modern service-oriented systems. In *Proceedings of the 7th IEEE International Conference on Services Computing (SCC 2010)*. IEEE Computer Society, 2010.

J. Kramer and J. Magee. The evolving philosophers problem: Dynamic change management. *IEEE Transactions on Software Engineering*, 16(11): 1293–1306, 1990.

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

N. Medvidovic and R. N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Trans. Softw. Eng.*, 26(1): 70–93, 2000. ISSN 0098-5589. doi: 10.1109/32.825767.

D. A. Menascé and V. A. Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. New Jersey: Prentice Hall, 2002. ISBN 0-13-065903-7.

D. A. Menascé, M. N. Bennani, and H. Ruan. On the use of online analytic performance models in self-managing and self-organizing computer systems. In Ö. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A. P. A. van Moorsel, and M. van Steen, editors, *Self-star Properties in Complex Information Systems*, volume 3460 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2005. ISBN 3-540-26009-9. doi: 10.1007/11428589_9.

R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture: Foundations, Theory and Practice*. John Wiley & Sons, Inc., 2009. ISBN 978-0-470-16774-8.

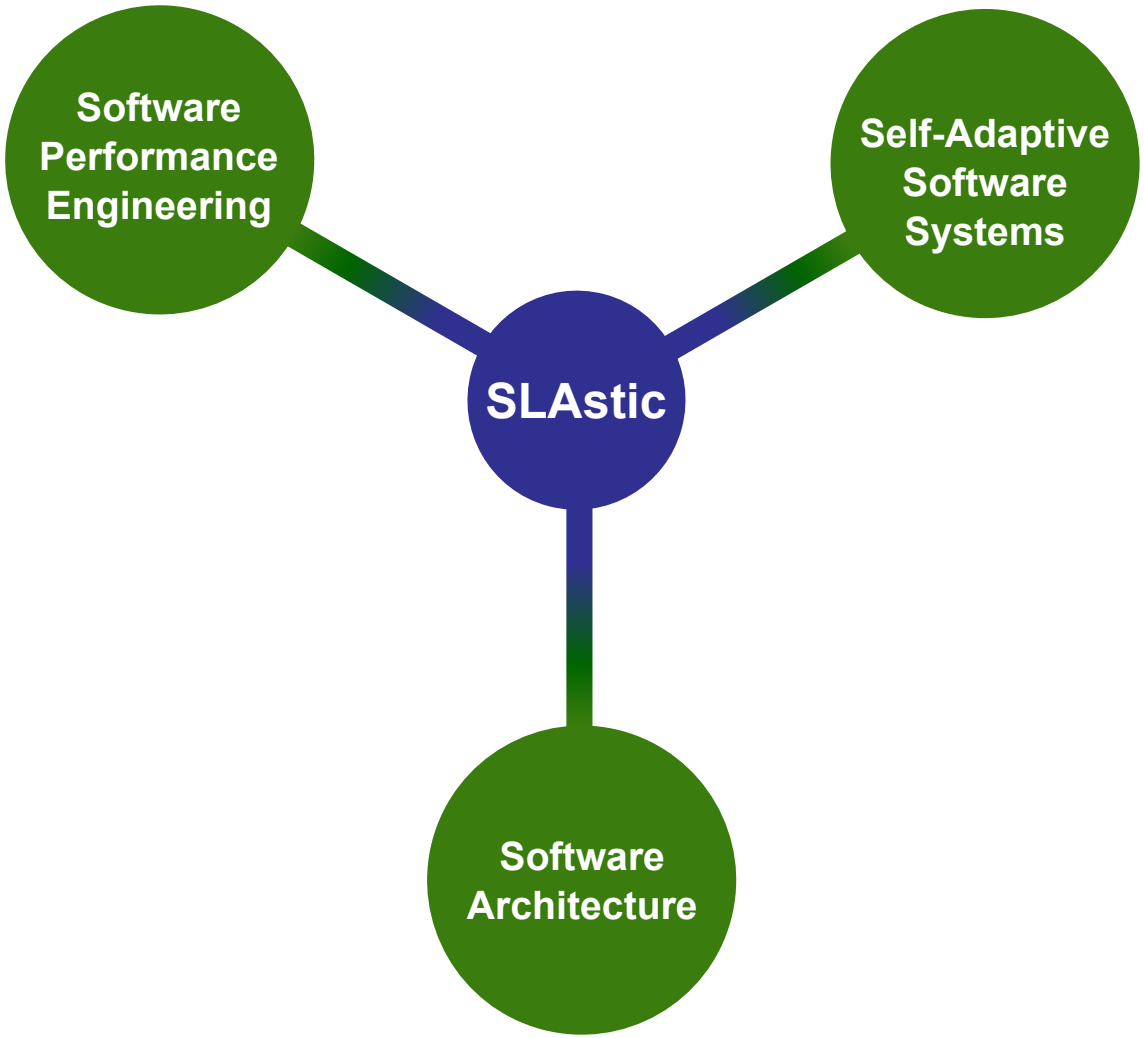
A. van Hoorn. *Online Capacity Management for Increased Resource Efficiency of Component-Based Software Systems*. PhD thesis, Department of Computer Science, University of Oldenburg, Oldenburg, Germany, 2011. work in progress.

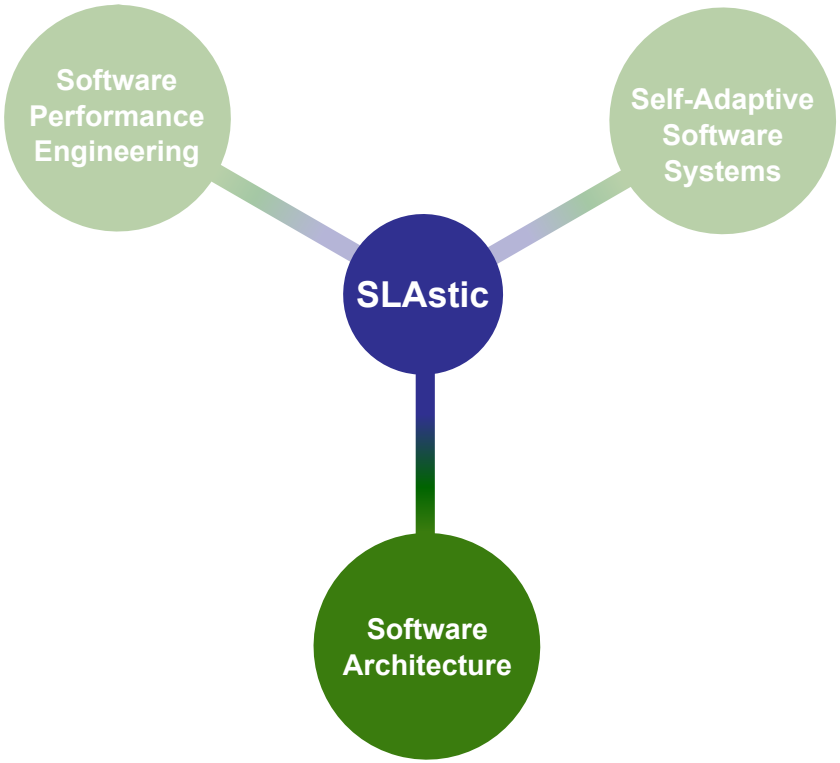
A. van Hoorn, M. Rohr, A. Gul, and W. Hasselbring. An adaptation framework enabling resource-efficient operation of software systems. In N. Medvidovic and T. Tamai, editors, *Proceedings of the 2nd Warm-Up Workshop for ACM/IEEE ICSE 2010 (WUP '09)*, pages 41–44. ACM, Apr. 2009a. ISBN 978-1-60558-565-9. doi: 10.1145/1527033.1527047.

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. 2009b. URL `http://www.informatik.uni-kiel.de/uploads/tx_publication/vanhoorn_tr0921.pdf`.

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. doi: 10.1007/978-3-642-23798-0_5.

M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *2007 Future of Software Engineering (FOSE '07)*, pages 171–187. IEEE, 2007.

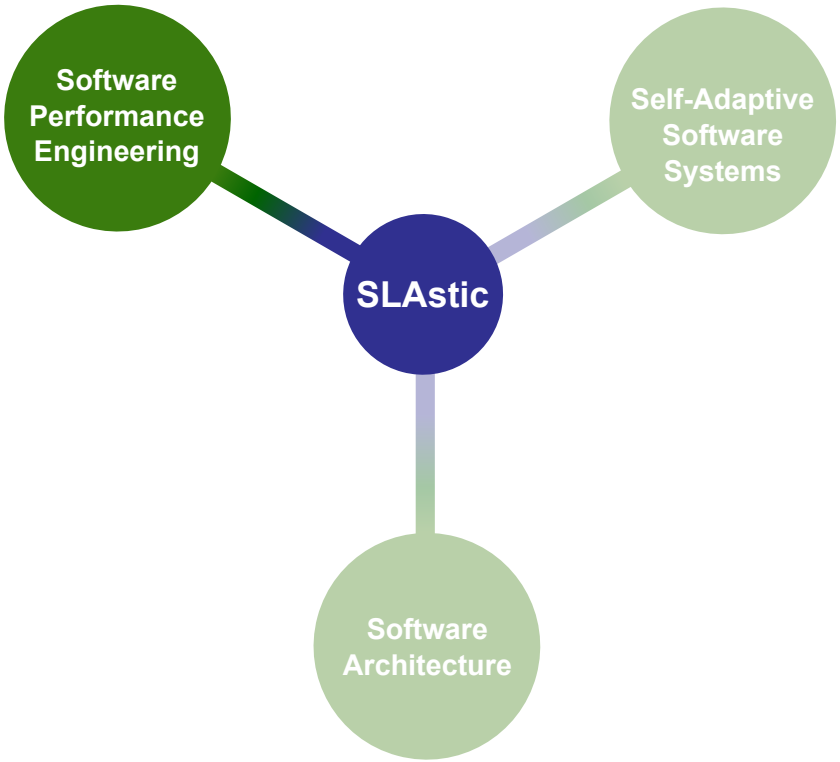

Software Performance Engineering
Self-Adaptive Software Systems
SLAstic
Software Architecture



Software Architecture (IEEE Def. [IEEE 2000])

"The fundamental organization of a system embodied in
- its components,
- their relationships to each other and to the environment,
- and the principles guiding its design and evolution."

- Views & Viewpoints
  [Clements et al. 2002]
  - Structural
  - Behavioral
- ADLs
  [Medvidovic and Taylor 2000, Taylor et al. 2009]
- Architectural styles [Taylor et al. 2009]
  - Component-based
  - Service-oriented, etc.
- Runtime reconfiguration/ Change management
  [Kramer and Magee 1990, Hofmeister 1993, Matevska 2009]



Software Performance Engineering [Woodside et al. 2007]

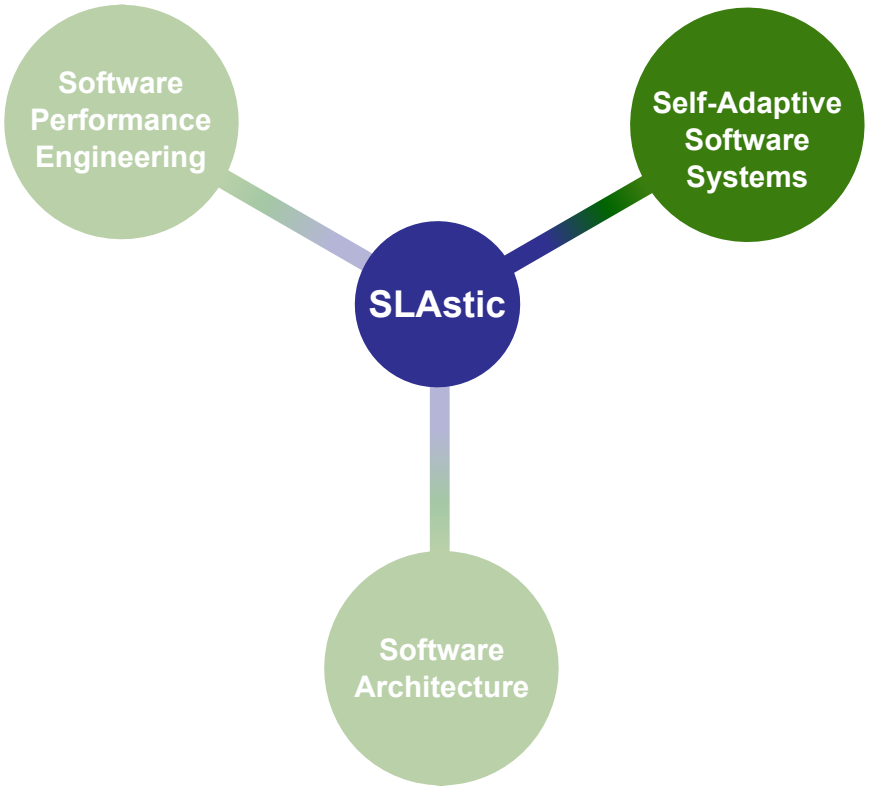
"Represents the entire collection of software engineering activities and related analyses used throughout the software development cycle which are directed to meeting performance requirements."

Performance

- Concerned with
  - Timing
  - Resource usage
- Metrics [Jain 1991]
  - Response time,
  - Throughput,
  - Utilization, etc.
- Model- vs. Measurement-based
- Capacity Planning

  [Menascé and Almeida 2002]
  - SLAs/SLOs
  - Workload Characterization
  - Workload Forecasting
  - Performance Prediction

Christian-Albrechts-Universität zu Kiel



IBM's autonomic computing initiative [Kephart and Chess 2003, IBM 2010]

"Systems manage themselves according to an administrator's goals."

Performance Modeling of Component-Based Software Systems

- S. Becker, H. Koziolek and R. Reussner. The Palladio component model for model-driven performance prediction. JSS, 2009.
- V. Grassi, R. Mirandola and A. Sabetta. A model-driven approach to performability analysis of dynamically reconfigurable component-based systems. WOSP '07, ACM, 2007.

Architecture-Based Runtime Reconfiguration

- D. Garlan, S.-W. Cheng and B. Schmerl. Increasing system dependability through architecture-based self-repair. ADS, Springer, 2003.
- J. Matevska. Architekturbasierte erreichbarkeitsoptimierte Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit, Dissertation, Dept. Comp. Sc., Univ. Oldenburg, 2009

Online Performance & Resource Management

- A. Diaconescu and J. Murphy. Automating the performance management of component-based enterprise systems through the use of redundancy. ASE '05, ACM, 2005.
- I. Gorton, Y. Liu and N. Trivedi. An extensible and lightweight architecture for adaptive server applications. Softw. Pract. Exper., 2008.
- D. A. Menascé, M. H. Bennani, and H. Ruan. On the use of online analytic performance models in self-managing and self-organizing Computer Systems. Self-Star '04, Springer, 2004.
- S. Kounev, F. Brosig, N. Huber, and R. Reussner. Towards self-aware performance and resource management in modern service-oriented systems. SCC '10, IEEE, 2010.