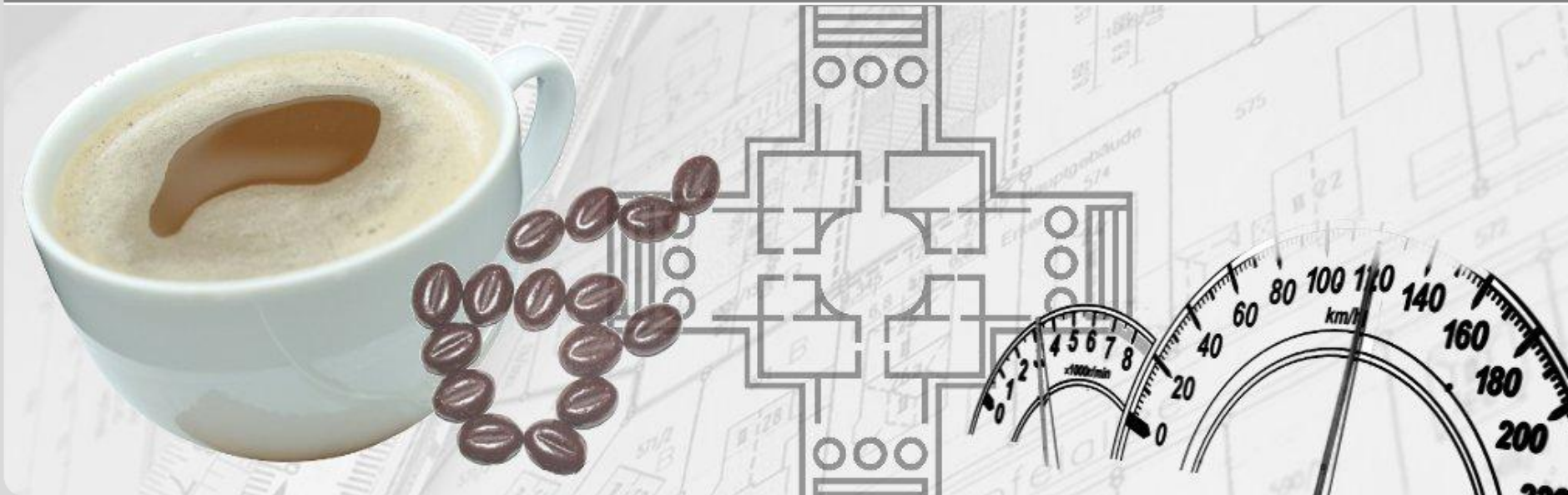


Automated Extraction of Palladio Component Models from Running Enterprise Java Applications

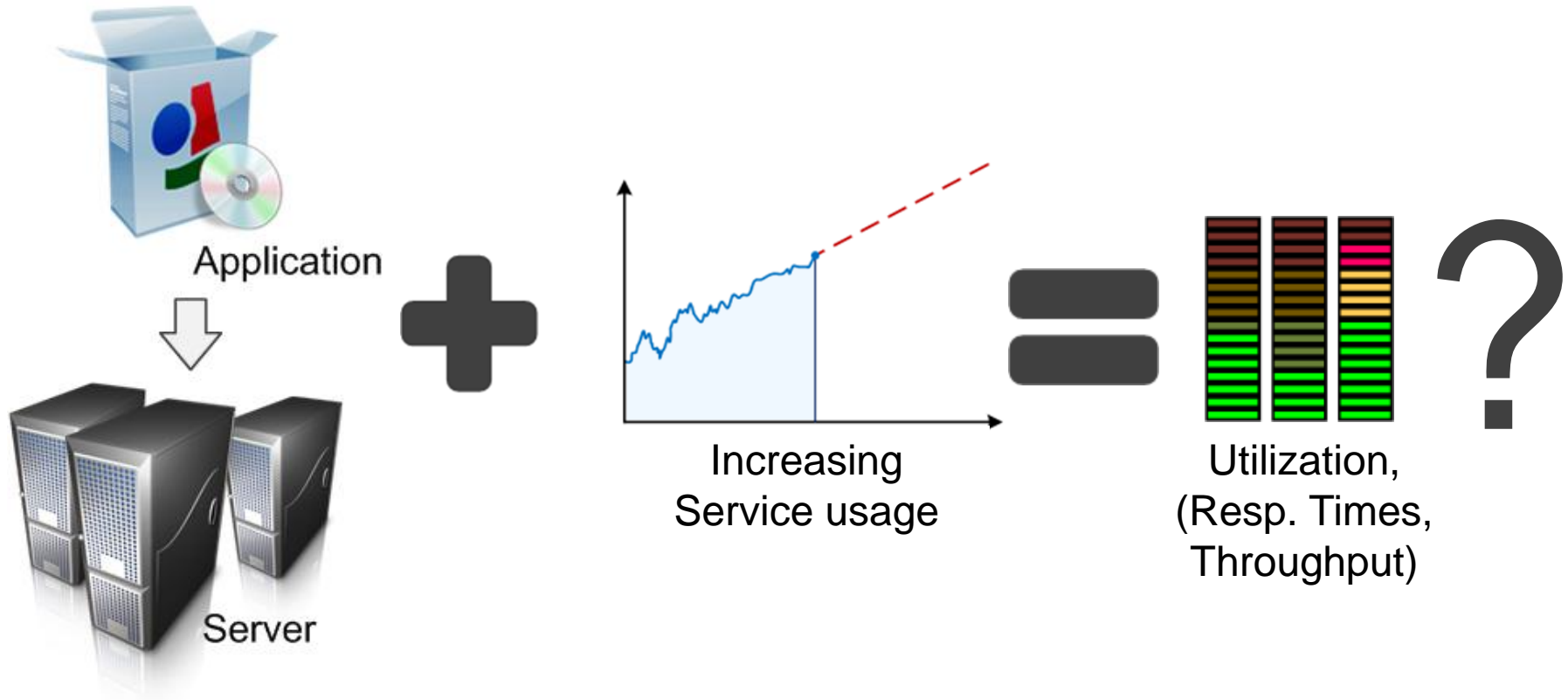
Fabian Brosig, Samuel Kounev, Klaus Krogmann
Karlsruhe Institute of Technology (KIT), {firstname.lastname}@kit.edu

October 19, 2009

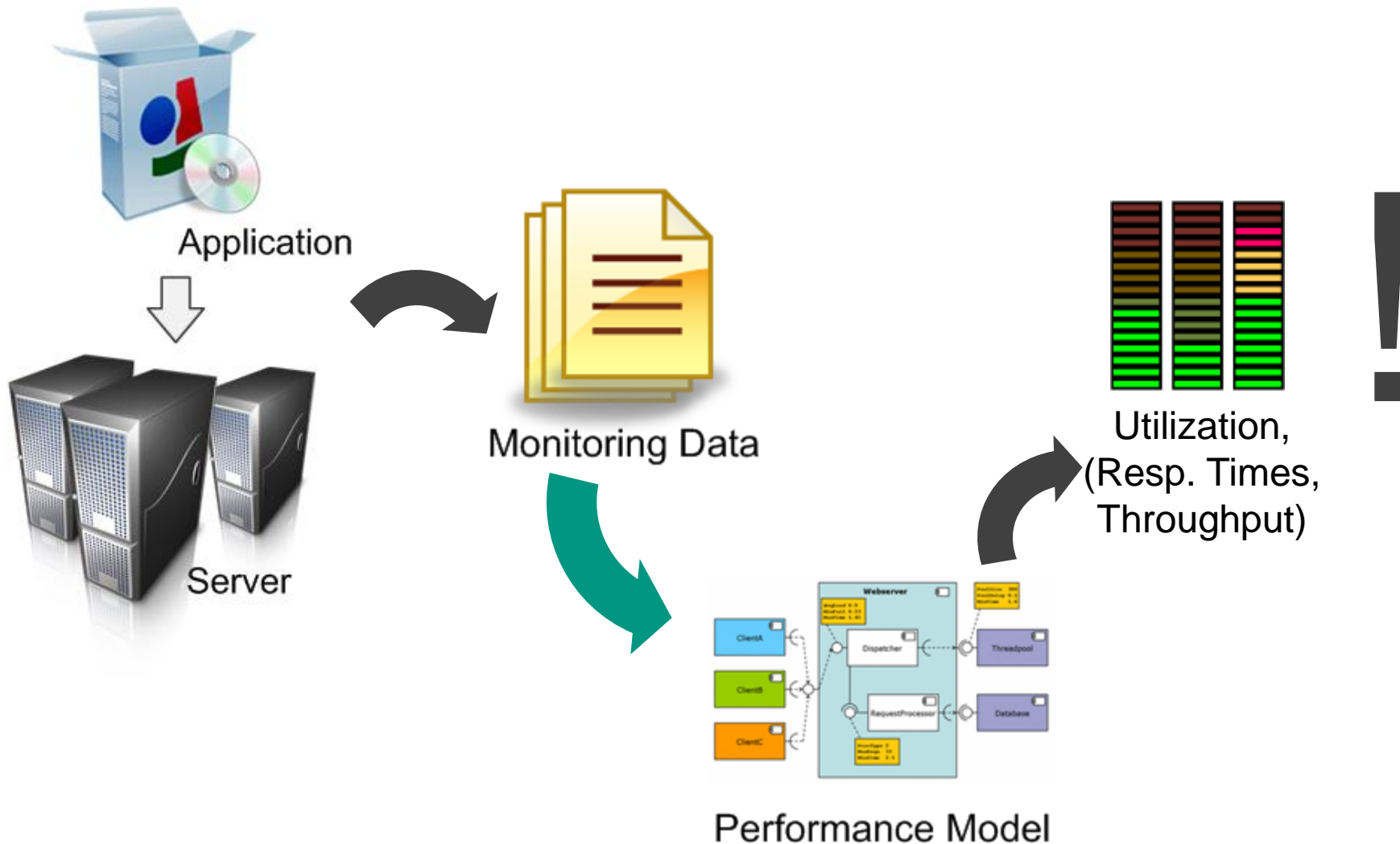
SOFTWARE DESIGN AND QUALITY GROUP



Performance Prediction during Operation

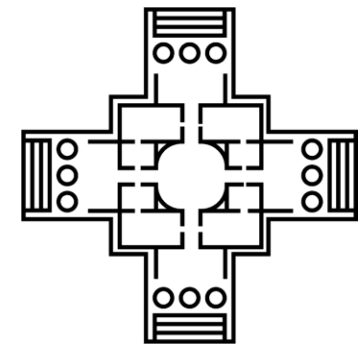


Performance Prediction during Operation



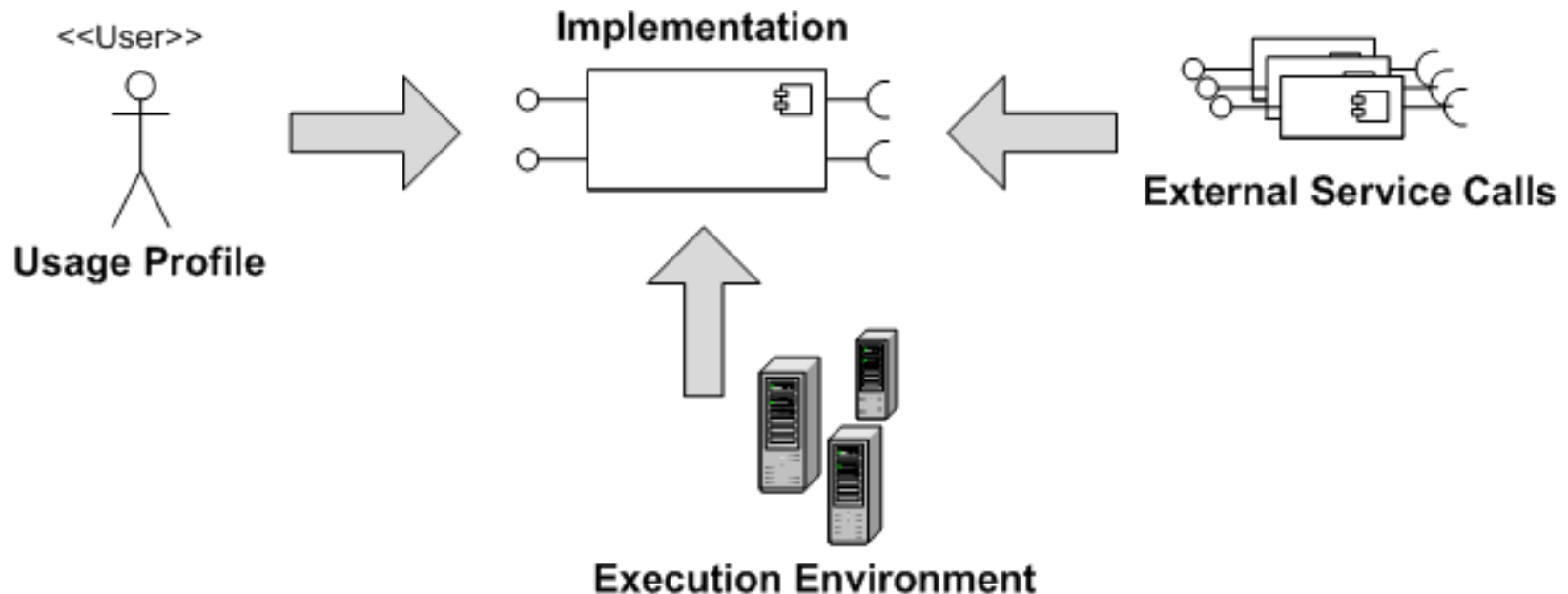
Approach and Contributions

- Automatically extract
 - a Palladio Component Model (PCM) instance
 - from a *running* enterprise Java (Java EE) application
 - by means of *online monitoring data*
 - collected by *state-of-the-art monitoring tools*
- Model extraction method, gap between performance models and run-time information can be closed
- Validate method with a representative system



Background – Palladio Component Model

- Palladio Component Model (PCM) [BKR09]
 - Meta Model: Performance-relevant aspects of component-based software architectures



- Model Solving:
 - Queueing Network based simulation (amongst others)

Background

- Oracle WebLogic Server (WLS)
- WebLogic Diagnostics Framework (WLDF)
 - Instrumentation engine
 - Monitor runtime information
(e.g., size of database connection pool)



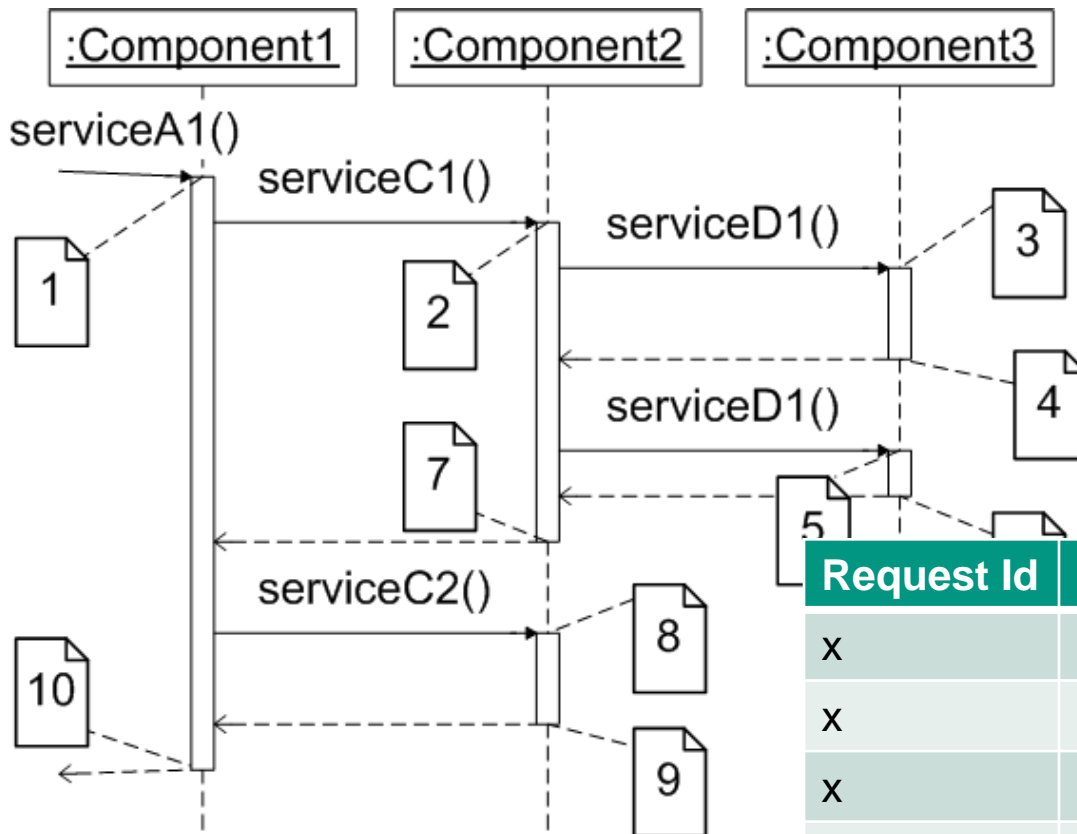
Model Extraction Method

Extracting a PCM Instance from a Running EJB Application

Model Extraction Method - Overview

- Scope: Enterprise JavaBeans (EJB) 3.0, Java Persistence API (JPA)
- Identifying Component Boundaries
- Extracting Inter-Component Control Flow
 - Call Path Tracing
- Extracting Intra-Component Control Flow
 - Call Path Tracing
- Extracting Resource Demands
 - Resource Consumption Monitoring

Call Path Tracing



Request Id	Record Id	Service	Location
x	1	serviceA1	entry
x	2	serviceC1	entry
x	3	serviceD1	entry
x	4	serviceD1	exit
x	5	serviceD1	entry
...
x	10	serviceA1	Exit

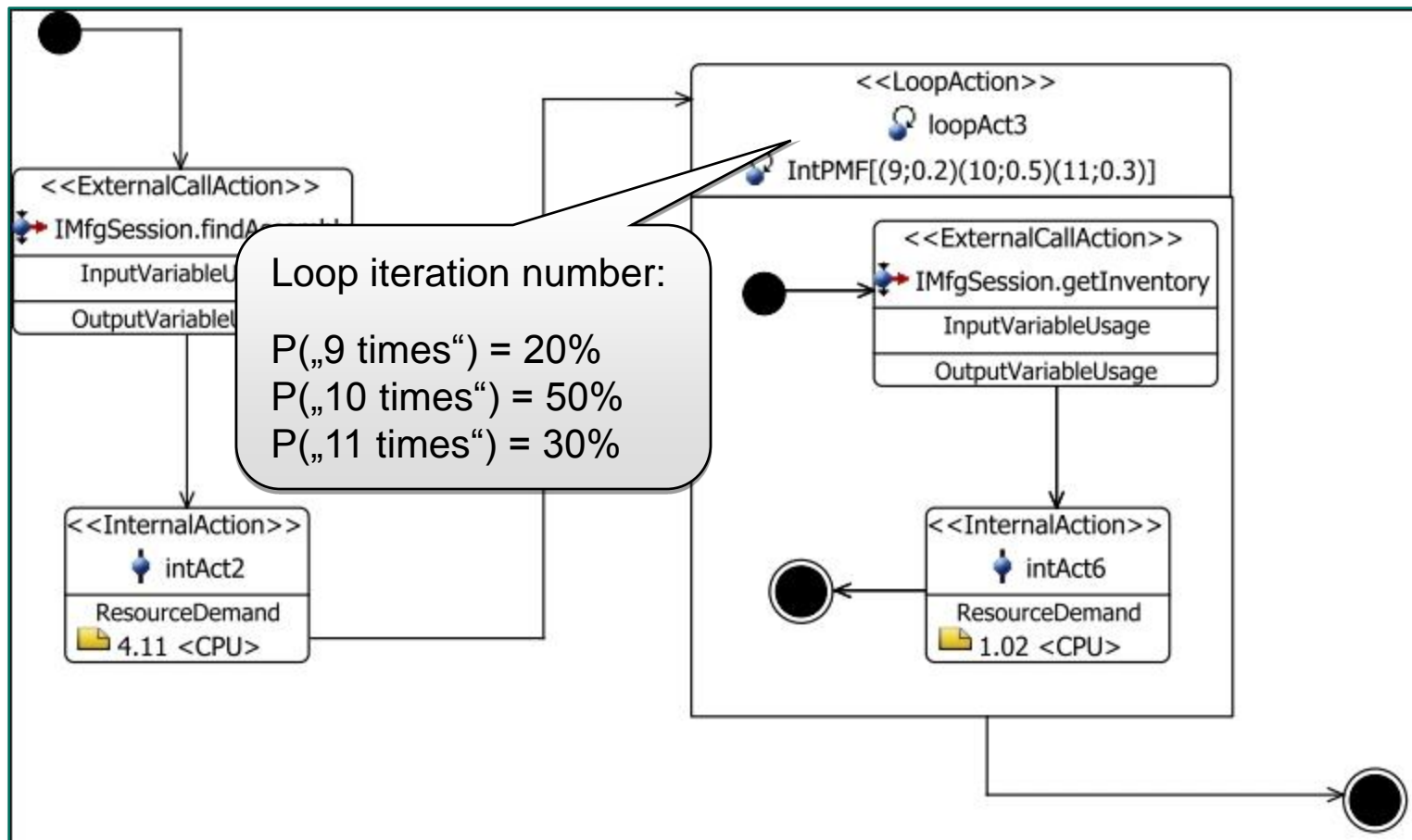
Extracting Control Flow

- Extracting Inter-Component Control Flow
- Extracting Intra-Component Control Flow

```
class LargeOrderSession {  
    ...  
    public LargeOrder createLargeOrder(String assemblyId, ...) {  
        // extracted external call  
        Assembly assembly = createLargeOrder_1_externalcallaction_1(assemblyId);  
        // extracted internal action  
        LargeOrder largeOrder = createLargeOrder_1_internalaction_2(assembly, ...);  
        return largeOrder;  
    }  
  
    private LargeOrder createLargeOrder_1_internalaction_2(Assembly assembly, ...) {  
        /* compute something */  
        LargeOrder largeOrder = new LargeOrder(assembly, ...);  
        entityManager.persist(largeOrder);  
        return largeOrder;  
    }  
  
    private Assembly createLargeOrder_1_externalcallaction_1(String assemblyId) {  
        Assembly assembly = mfgSession.findAssembly(assemblyId);  
        return assembly;  
    }  
}
```

Example: Extracted Service Behavior

- Extracted resource demanding behavior (exemplary service)

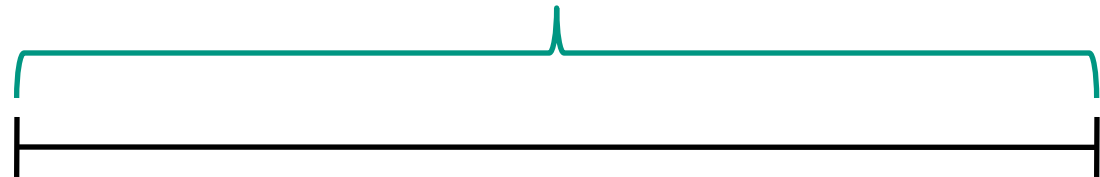


Extracting Resource Demands

- CPU resource demands
- Approach 1
 - Approximate resource demands with measured response times
- Approach 2
 - Estimate resource demands based on utilization and throughput data
- Partitioning between
 - app. server CPU (WLS CPU)
 - and database server CPU (DBS CPU) ?

Extracting Resource Demands - Example

■ Measurement Interval



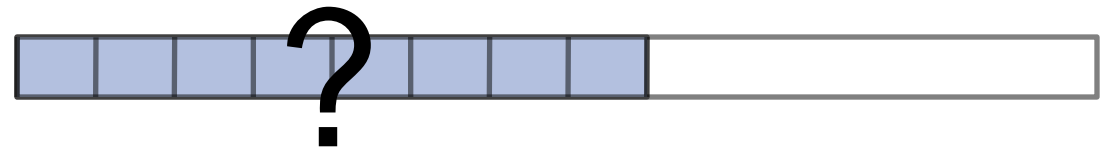
■ CPU Utilization



■ Action1 executed 8 times, Action2 executed 4 times

■ Apply Service Demand Law:

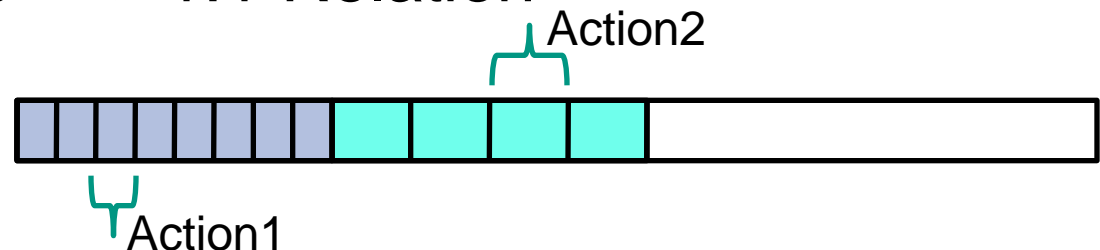
How to partition processing time?



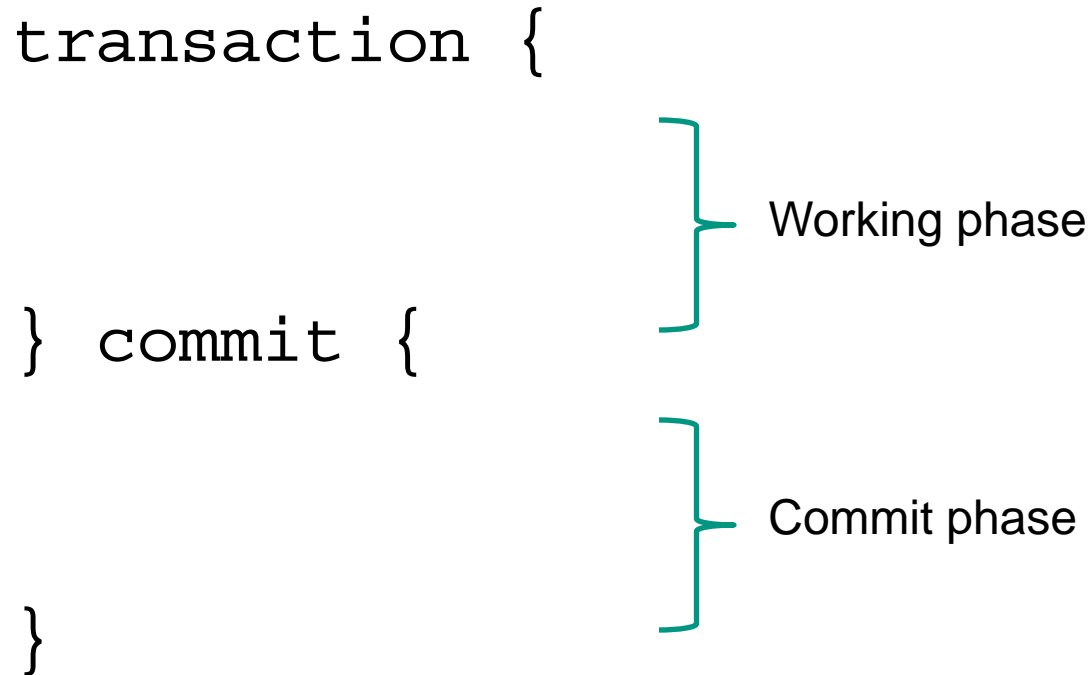
■ Response Times: Action1 = 1 sec, Action2 = 2 sec

■ $8 * 1 \text{ sec} = 4 * 2 \text{ sec} \implies 1:1 \text{ Relation}$

■ Resource Demands



WLS CPU demands and DBS CPU demands



- Working phase \Rightarrow WLS CPU
- Commit phase \Rightarrow DBS CPU
- DB read vs. DB write?

Related Work I

- Trace-based approaches
 - Hrischuk [HWRI99], Israr [IWF07]
 - Extraction of Layered Queueing Network (LQNs) structures
 - Not component-based, no explicit control flow in components
 - Dynatrace Tool [DYNT]
 - Transaction call tree in distributed JavaEE / .NET environments
 - Briand [BLL06]
 - UML sequence diagrams
- Java application monitoring at run-time
 - Carrera [CGT+03]: hotspot and bottleneck analysis
 - Compas [MM02]: generates interaction models

Related Work II

- PCM extraction
 - Java2PCM [KKKR08]
 - Component-level control-flow based on static code analysis
 - Krogmann [KKR08a, KKR08b]
 - Static analysis (code) and dynamic analysis (run-time)
 - Not focused on extraction during system operation
 - Abstracts from concrete timing values (Java bytecode operations)
 - Extraction by means of machine learning techniques
 - ArchiRec [CKK08]
 - provides component boundaries based on static code analysis

Case Study

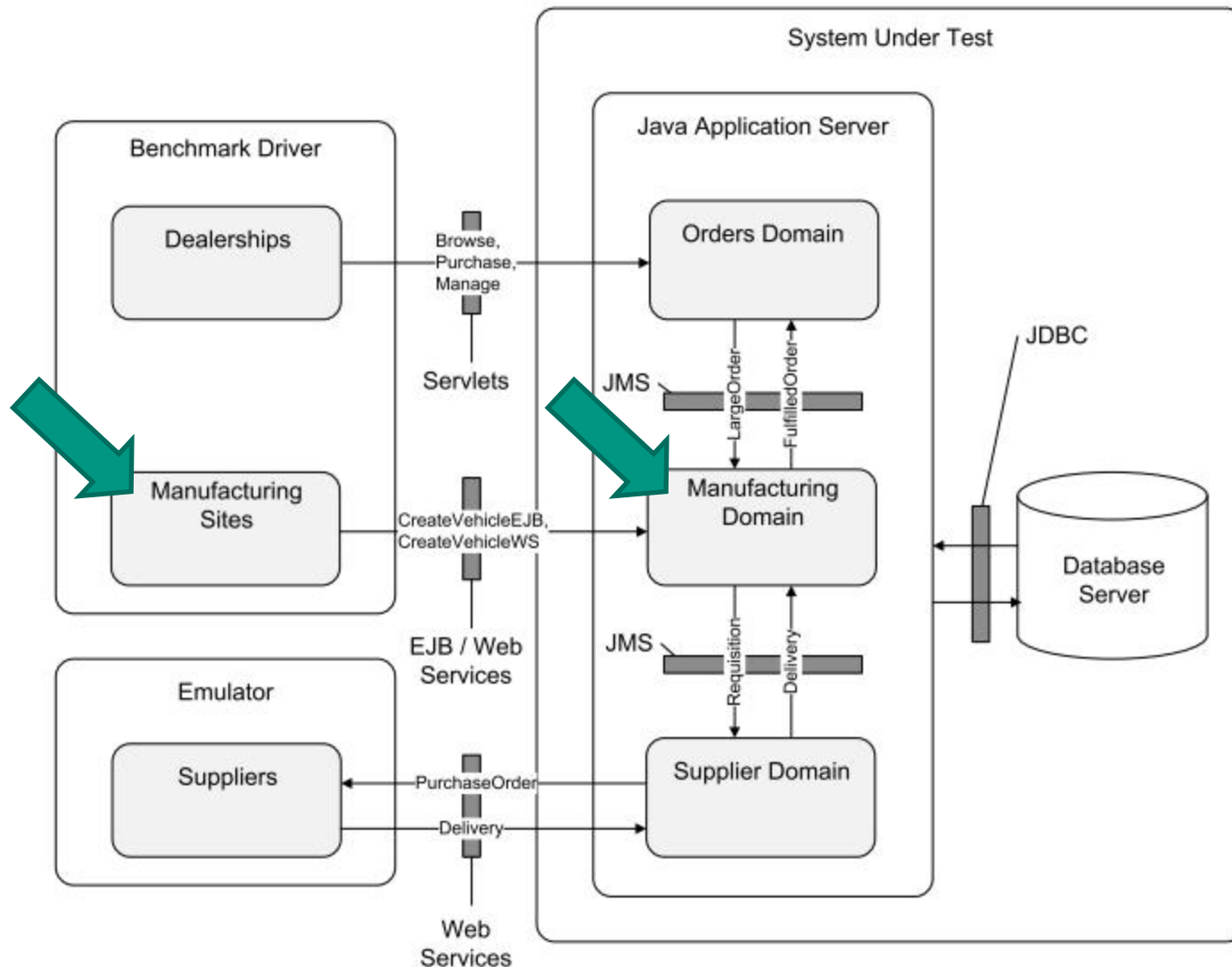
With a prototype of SPECjEnterprise2010



SPECjEnterprise2010

- State-of-the-art, industry-standard benchmark
- Realistic, three-tier, Java EE application
- Representative workload

SPECjEnterprise2010

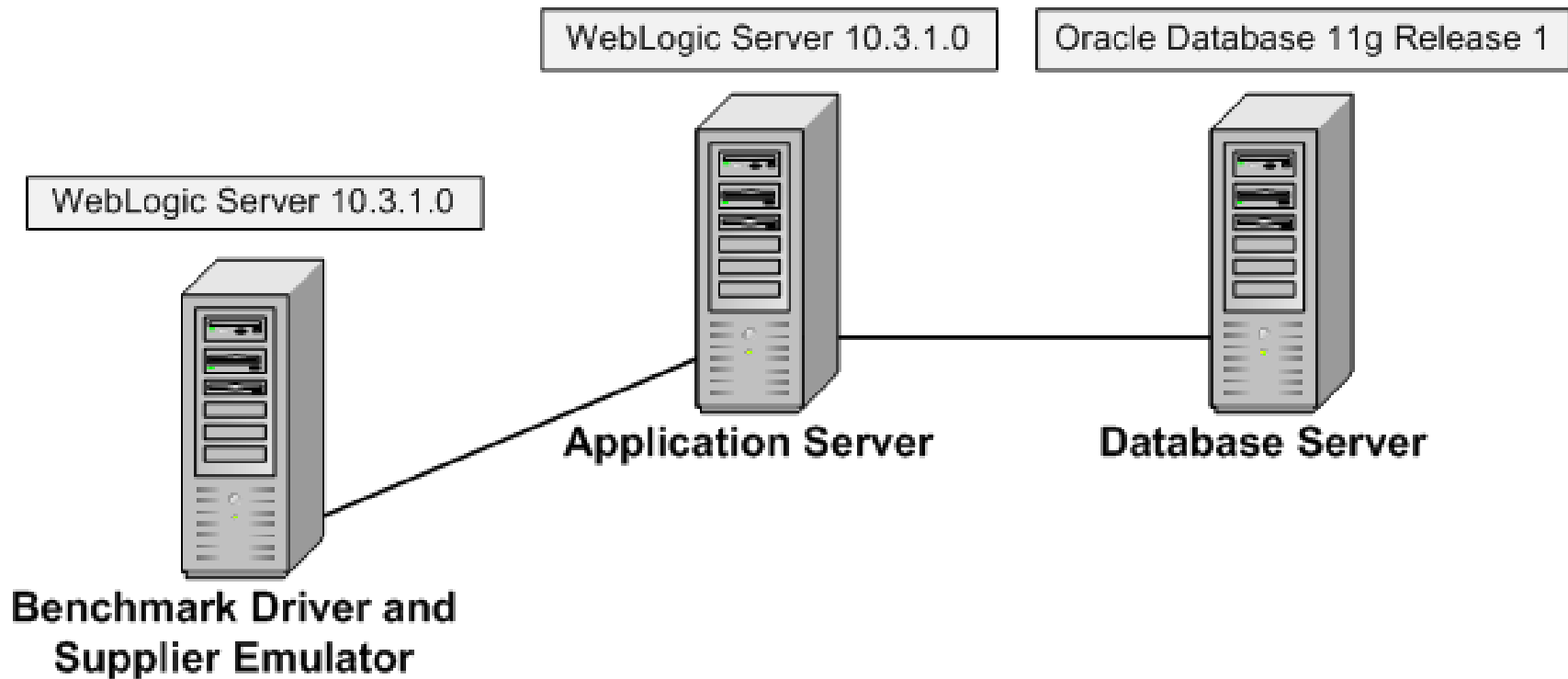


System Environment

- Benchmark deployed on three machines

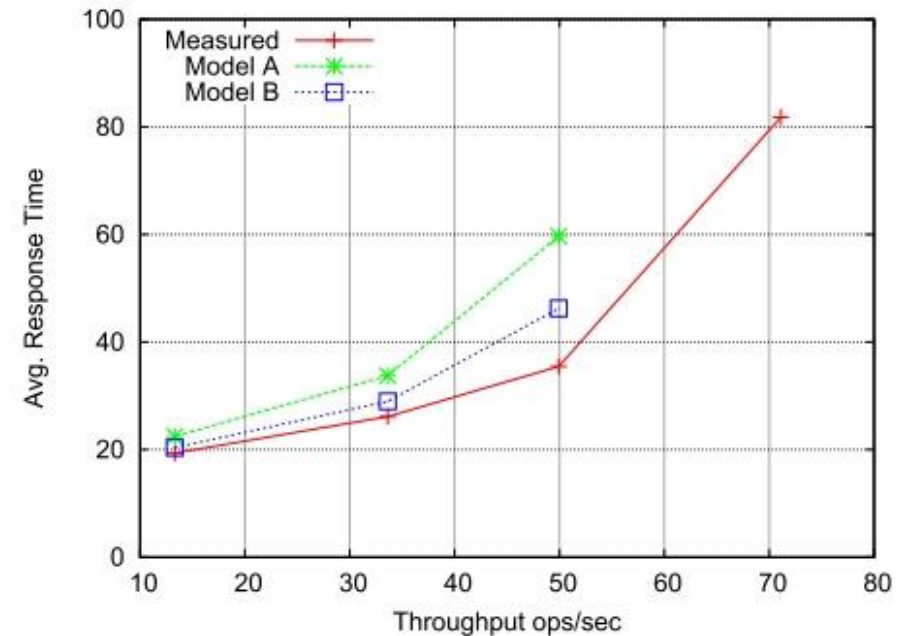
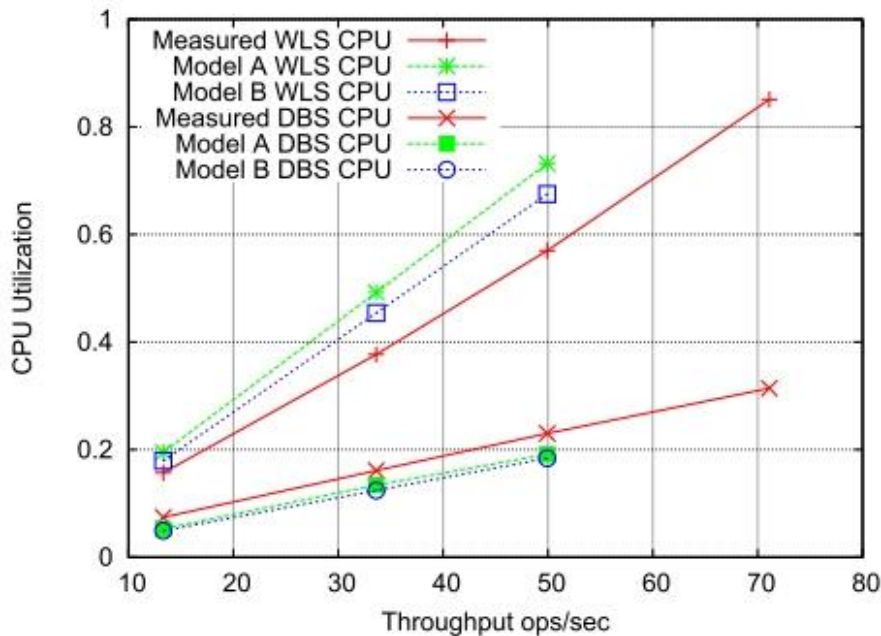
ORACLE
WebLogic Server®

ORACLE® 11^g
DATABASE



Results – Scenario 1: „Schedule Work Order“

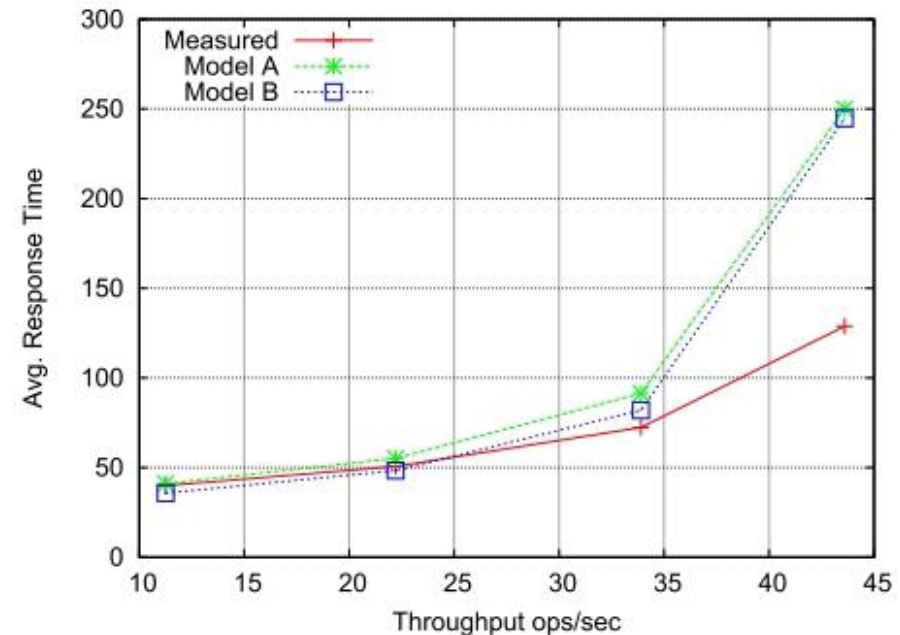
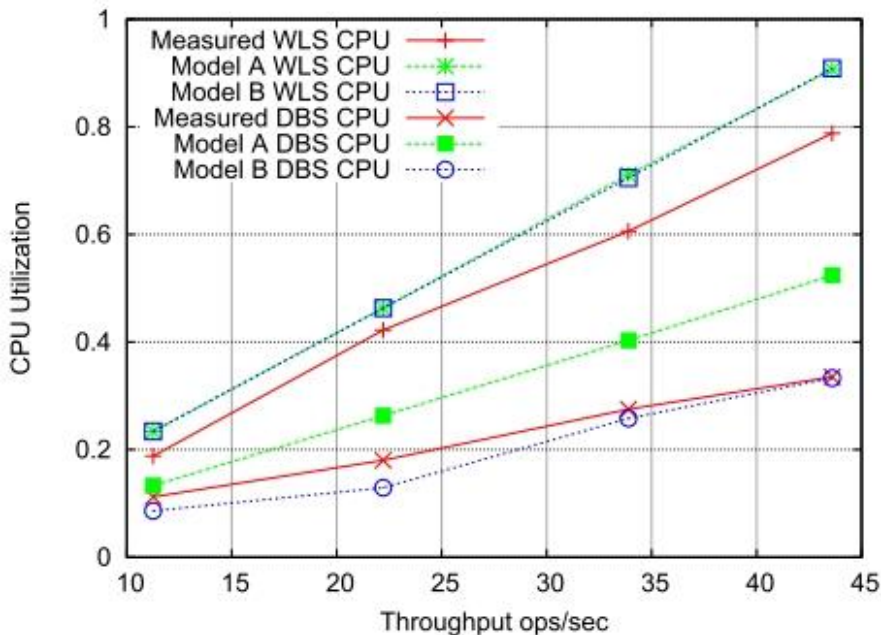
- Model A: Resource demands approximated with measured response times
- Model B: Resource demands estimated based on utilization and throughput data



Model A: $U_{WLS_CPU} = 0.12$, Model B: $U_{WLS_CPU} = 0.81$, Steady State Time: 1020 sec

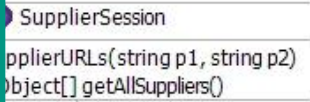
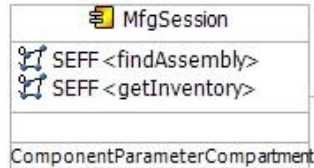
Results – Scenario 2a: „Create Vehicle“

- Model A: Resource demands approximated with measured response times
- Model B: Resource demands estimated based on utilization and throughput data



Model A: $U_{WLS_CPU} = 0.09$, Model B: $U_{WLS_CPU} = 0.75$, Steady State Time: 1140 sec

Scenario 2a: „Crea



<<Provides>>

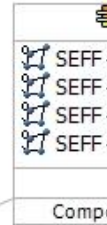


<<Provides>>

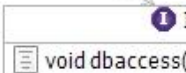


<<Provides>>

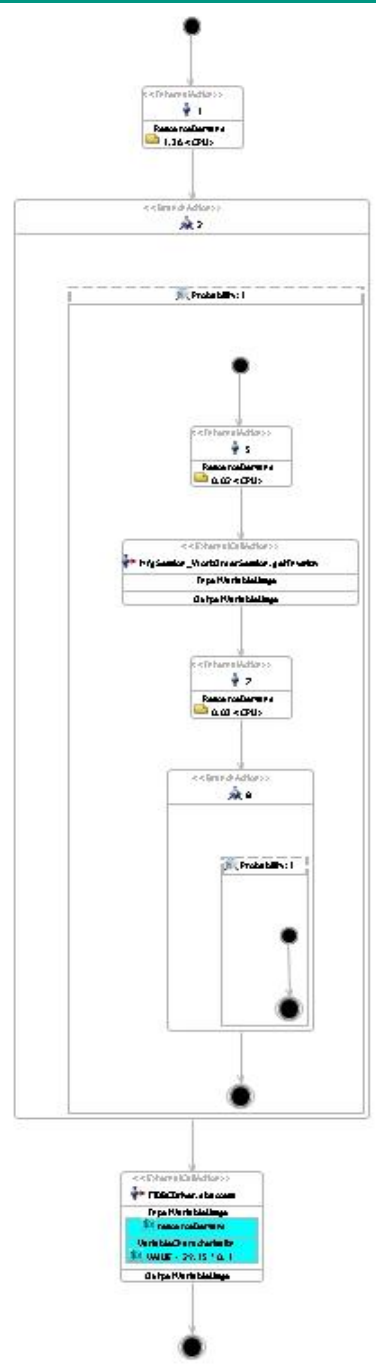
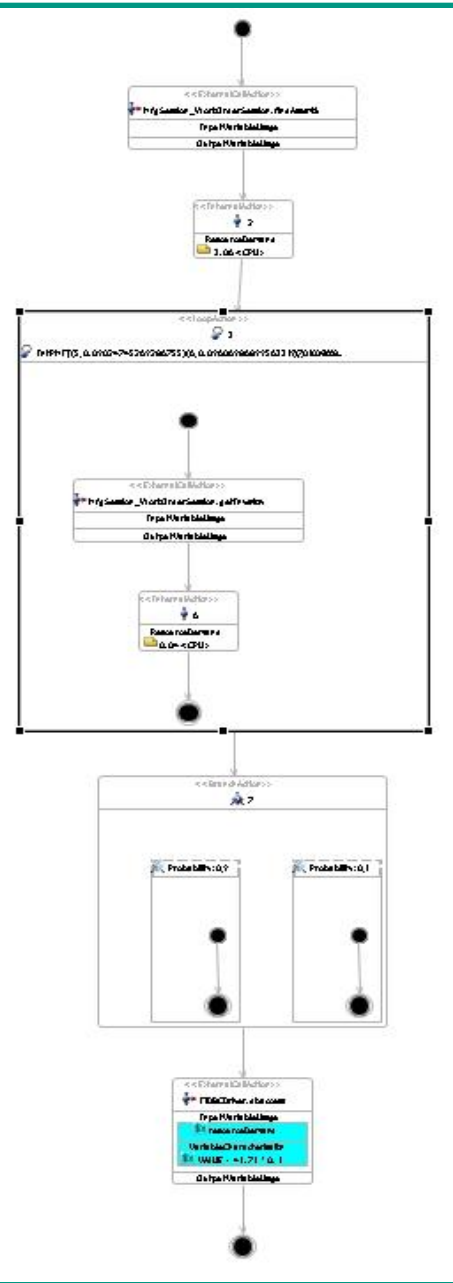
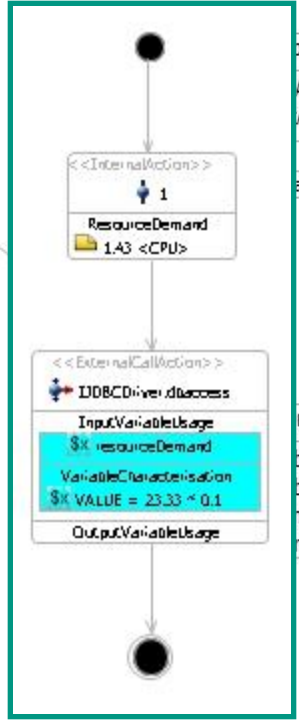
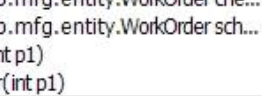
<<Requires>>



<<Requires>>



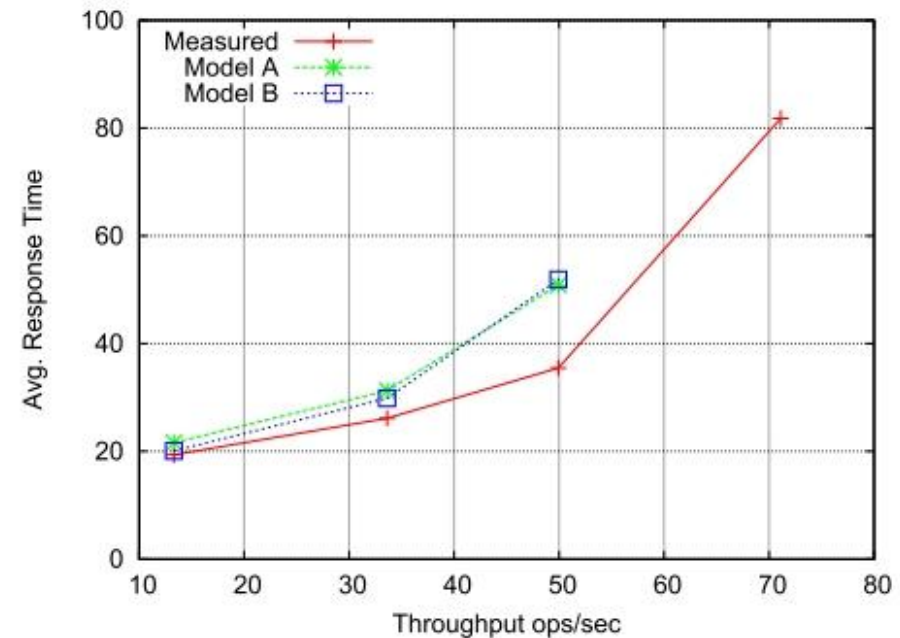
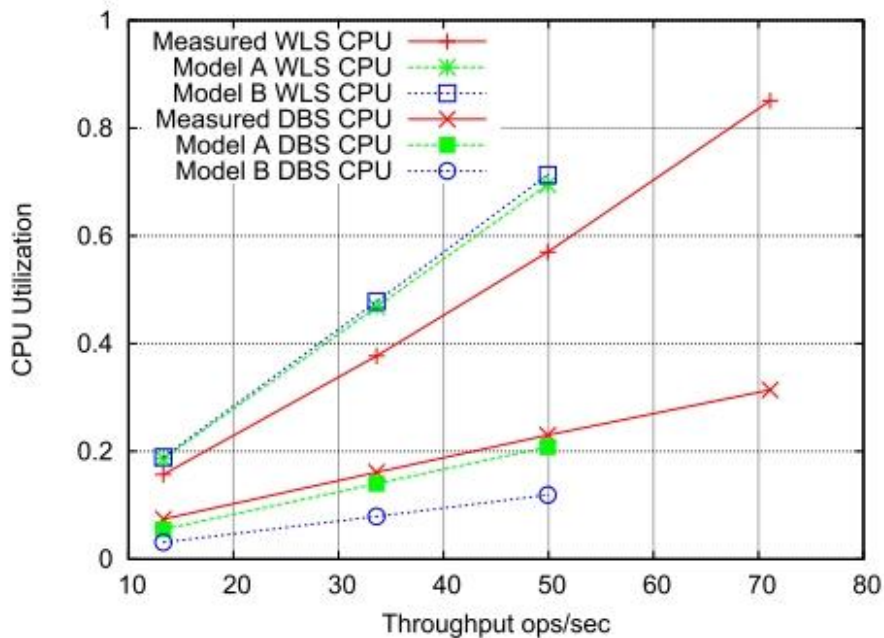
<<Provides>>



MODEL A. $U_{WLS_CPU} = 0.09$, MODEL B. U_{WLS}

Results – Scenario 2b: „Schedule Work Order“

- Model A: Resource demands approximated with measured response times
- Model B: Resource demands estimated based on utilization and throughput data



Conclusion

Summary and Future Work

Concluding Remarks - Summary

- Automated end-to-end model extraction
 - Extracts PCM instance from EJB application
 - Components, Component connections
 - Component internals
 - Resource demands
 - During application run-time
- Used WLDF to implement tool prototype
- Case study with prototype of SPECjEnterprise2010
 - Feasibility, Proof-of-concept

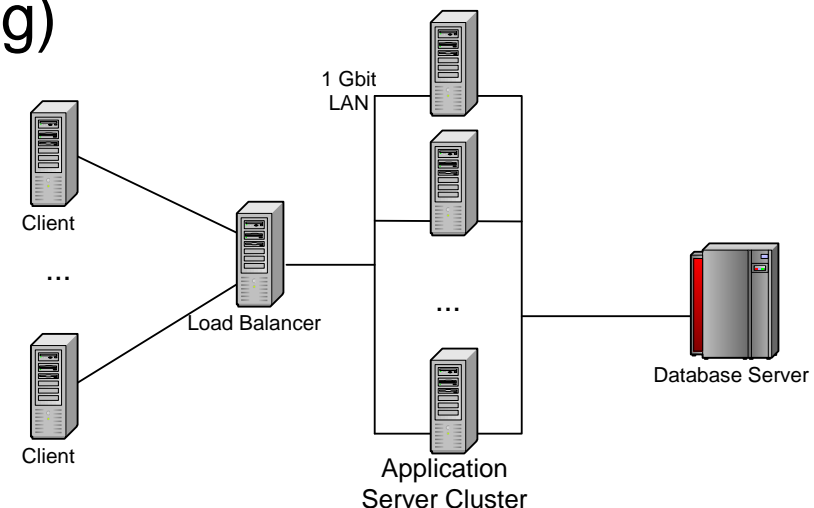


Concluding Remarks – Future Work

- Extend extraction method
 - Asynchronous Messaging
 - Extraction of component internals [KKR08a, KKR08b]
 - Resource demand estimation (e.g., statistical regression [Pac08, Rol95], JRockit method sampling)



- Extend case study
 - SPECjEnterprise2010
 - Application Server Cluster



- Descartes vision

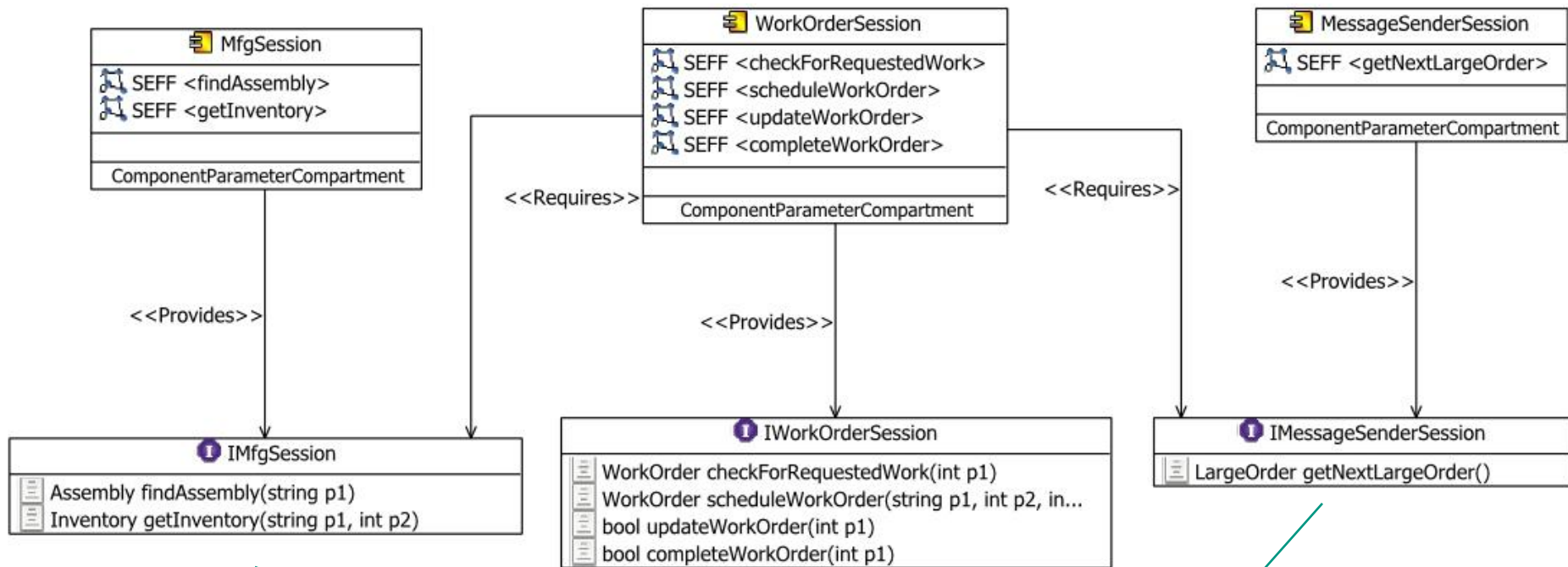
Questions?



Backup

Background – Palladio Component Model

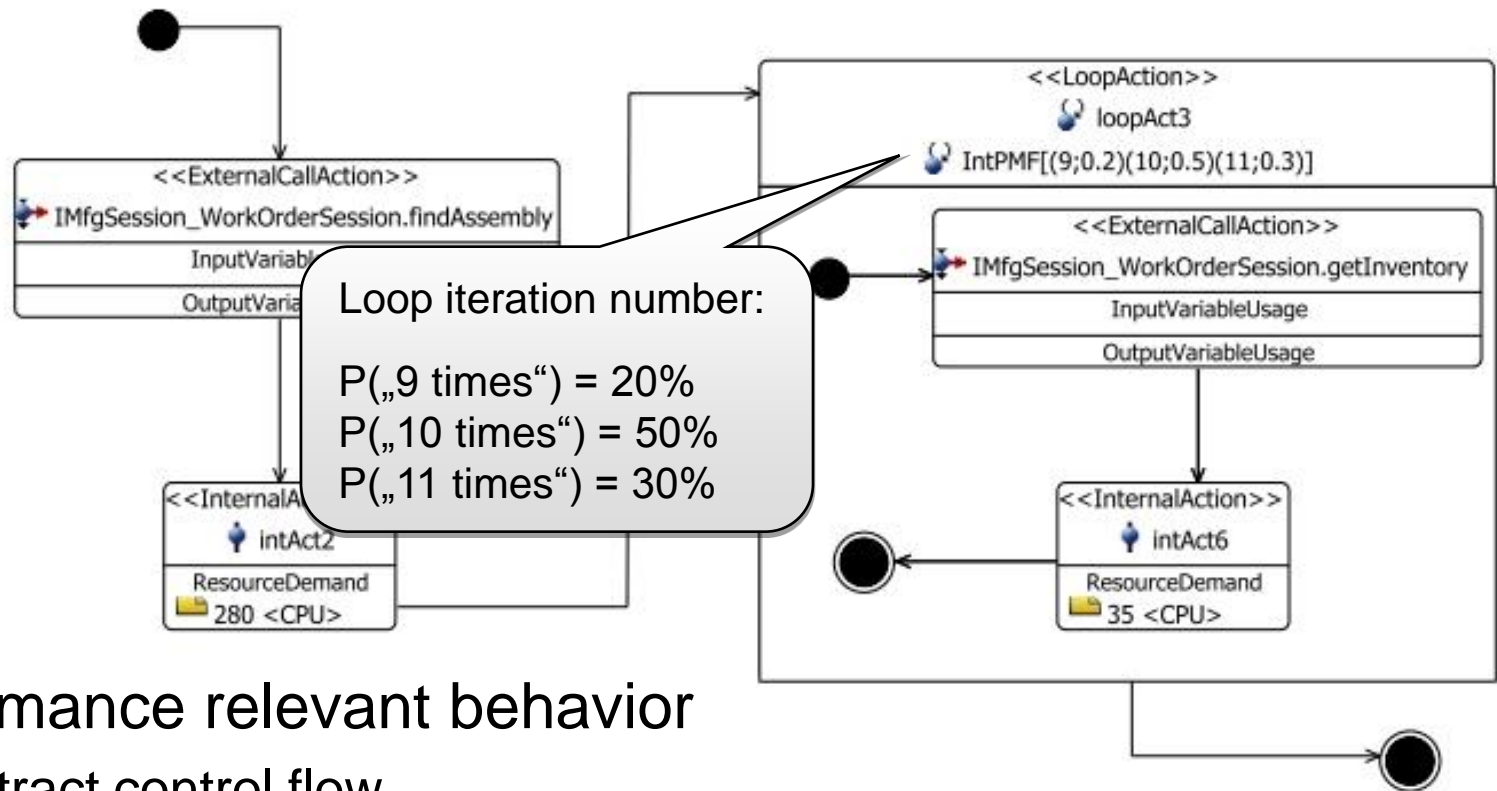
Components



Component Interfaces

Background – Palladio Component Model

- Service provided by a component described by: **Resource Demanding ServiceEffectSpecification (RDSEFF)**



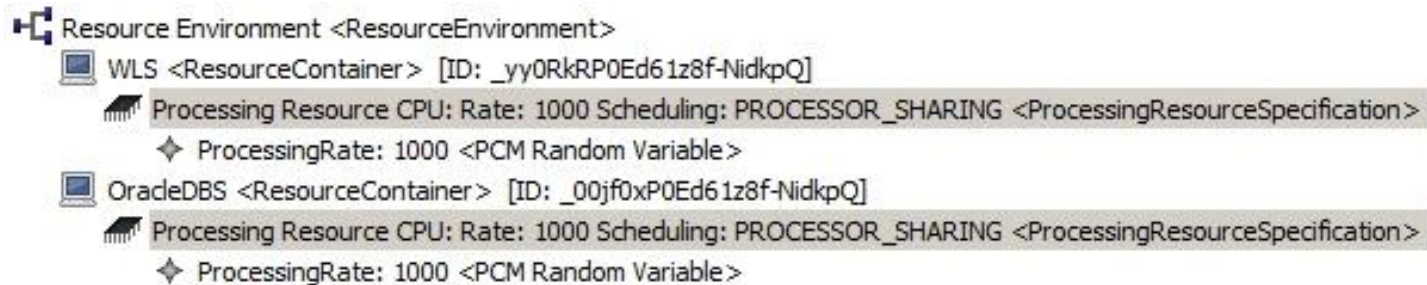
- Performance relevant behavior
 - Abstract control flow
 - Resource demands

Challenges

- Apportioning resource demands among different resources
- Stable system environment (repeatable experiments)
- Instrumentation Overhead
- Instrumentation vs. JRockit Optimization
- Benchmark driver measured incorrect response times
 - Usage of `System.currentTimeMillis` (Windows)
- Dynamic frequency/voltage scaling
 - EIST (SpeedStep), load-dependent resource

Scenario 1: Resource & Allocation Model

PCM Resource Environment Model



PCM Allocation Model



Identifying Component Boundaries

- Options to specify component boundaries
 - Every EJB is considered as a separate component.

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- This is a sample system description a user has to provide.
      It describes the SPECjAppServer2004_Next benchmark application. -->

<fb:system xmlns:fb="http://sdq.ipd.uni-karlsruhe.de/fabro"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <!-- Choose application with name specj. -->
  <fb:application>specj</fb:application>

  <!-- Choose to map each EJB to its own component. -->
  <fb:configmode>ejbtocomponent</fb:configmode>

</fb:system>
  
```

ed as

- Tool Prototype: Specification provided as XML doc conforming to an XML schema

Tool Prototype - Extraction Steps

■ Extraction Steps

- Configure WLDF to
 - extract the application's structure
 - extract performance-relevant control flow
- - Monitor running application -
- Configure WLDF to
 - extract resource demands
- - Monitor running application -
- Extract PCM repository model and PCM system model



Extracting Intra-Component Control Flow I

- Naming of methods, where performance-relevant actions are moved to

```

<methodname> = <pname> '_' <idA> '_' <action> '_' <idZ>
  <pname>     = The name of the parent method.
  <action>    = 'internalaction' | 'externalcallaction' |
               'loopaction' | 'loopbody' |
               'branchaction' | 'branchtransition'
  <idA>      = A number that, together with <pname>, uniquely identifies
               the parent method. In case of method overloading,
               the parent method's name alone could not be used as
               identifier.
  <idZ>      = A number that uniquely identifies the extracted method
               amongst those extracted methods with the same parent
               method. Note that the number only serves as identifier,
               it is not an index of a sequence.
  
```

References

- [BKR09] Steffen Becker, Heiko Kozirolek, and Ralf Reussner. The Palladio Component Model for Model-Driven Performance Prediction. *Journal of Systems and Software*, 82:3–22, 2009.
- [KKKR08] Thomas Kappler, Heiko Kozirolek, Klaus Krogmann, and Ralf Reussner. Towards Automatic Construction of Reusable Prediction Models for Component-Based Performance Engineering. In Korbinian Herrmann and Bernd Brügge, editors, *Software Engineering 2008*, volume 121 of *LNI*, pages 140–154, Munich, Germany, February 2008. Bonner Köllen Verlag.
- [KKR08a] Klaus Krogmann, Michael Kuperberg, and Ralf Reussner. Reverse Engineering of Parametric Behavioural Service Performance Models from Black-Box Components. In Ulrike Steffens, Jan Stefan Addicks, and Niels Streekmann, editors, *MDD, SOA und IT-Management (MSI 2008)*, pages 57–71, Oldenburg, September 2008. GITO Verlag.
- [KKR08b] Michael Kuperberg, Klaus Krogmann, and Ralf Reussner. Performance Prediction for Black-Box Components using Reengineered Parametric Behaviour Models. In *Proceedings of the 11th International Symposium on Component Based Software Engineering (CBSE 2008)*, Karlsruhe, Germany, 14th-17th October 2008, volume 5282 of *LNCS*, pages 48–63. Springer, Heidelberg, October 2008.
- [QIM] Q-ImPreSS project site - Quality impact prediction for evolving service-oriented software. Online. Last visit: 2009-05-19.
<http://www.q-impress.eu/Q-ImPreSS/CMS/Overview/index.html>.

References

- [QIM08] Project Deliverable D2.1 Service Architecture Meta-Model (SAMM). Online, September 2008. Last visit: 2009-05-19. http://www.q-impress.eu/Q-ImPrESS/CMS/Documents/D2.1-Service_architecture_meta-model.pdf
- [CKK08] Landry Chouambe, Benjamin Klatt, and Klaus Krogmann. Reverse Engineering Software-Models of Component-Based Systems. In Kostas Kontogiannis, Christos Tjortjis, and Andreas Winter, editors, *12th European Conference on Software Maintenance and Reengineering*, pages 93–102, Athens, Greece, April1–4 2008. IEEE Computer Society.
- [BLL06] Lionel C. Briand, Yvan Labiche, and Johanne Leduc. Toward the Reverse Engineering of UML Sequence Diagrams for Distributed Java Software. *IEEE Transactions on Software Engineering*, 32(9):642–663, September 2006.
- [HWRI99] Curtis E. Hrischuk, Murray Woodside, Jerome A. Rolia, and Rod Iversen. Trace-Based Load Characterization for Generating Performance Software Models. *IEEE Transactions on Software Engineering*, 25(1):122–135, 1999.
- [IWF07] Tauseef Israr, Murray Woodside, and Greg Franks. Interaction tree algorithms to extract effective architecture and layered performance models from traces. *J. Syst. Softw.*, 80(4):474–492, 2007.
- [DYNT] dynaTrace software. Tracing transactions across distributed Java/.NET application components at production-save overhead. Online. Last visit: 2009-05-19. <http://www.dynatrace.com/en/whitepapers/articles.aspx>

References

- [CGT+03] David Carrera, Jordi Guitart, Jordi Torres, Eduard Ayguade, and Jesus Labarta. Complete instrumentation requirements for performance analysis of Web based technologies. In *ISPASS '03: Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 166–175, Washington, DC, USA, 2003. IEEE Computer Society.
- [MM02] Adrian Mos and John Murphy. A Framework for Performance Monitoring, Modelling and Prediction of Component Oriented Distributed Systems. In *WOSP '02: Proceedings of the 3rd international workshop on Software and performance*, pages 235–236, New York, NY, USA, 2002. ACM.
- [Pac08] Pacifici, G., Segmuller, W., Spreitzer, M., Tantawi, A.N.: Cpu demand for web serving: Measurement analysis and dynamic estimation. *Perform. Eval.* 65(6-7) (2008) 531–553.
- [Rol95] Rolia, J., Vetland, V.: Parameter estimation for performance models of distributed application systems. In: *CASCON '95: Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research*, IBM Press (1995) 54.