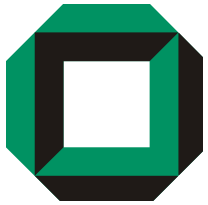
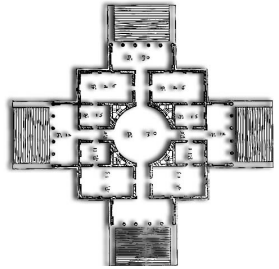


Universität Karlsruhe (TH)
Research University ▪ founded 1825

SDQ
Software Design and Quality

Palladio

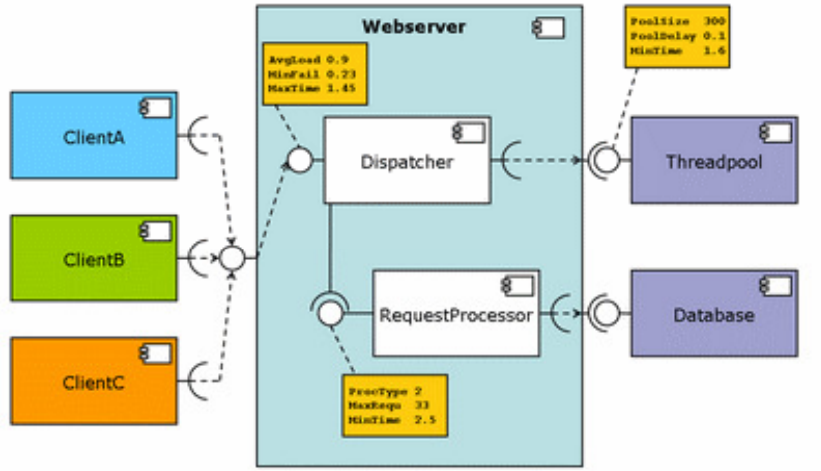


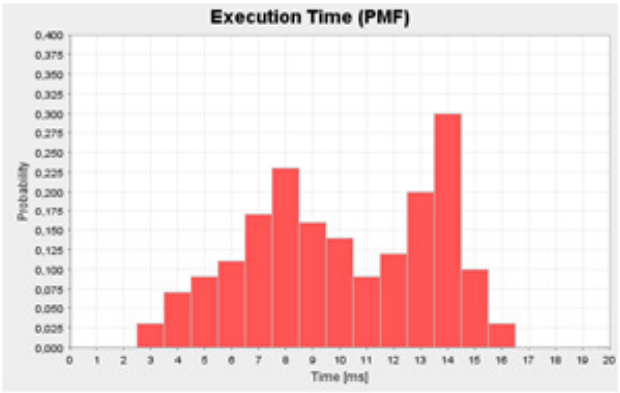
Prediction of Performance Properties

Heiko Koziolek (koziolek@ipd.uka.de)
Klaus Krogmann (krogmann@ipd.uka.de)
Ralf Reussner (reussner@ipd.uka.de)

Chair Software Design and Quality
Institute for Program Structures and Data Organization (IPD)
Faculty of Informatics, Universität Karlsruhe (TH)

Performance Prediction



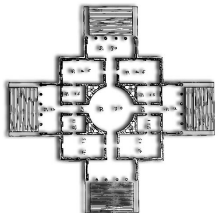


Performance model
of a component-based
software architecture

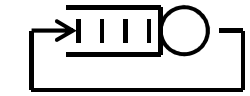
Performance data

- Execution time
- Throughput
- Resource utilisation

Palladio: The Approach

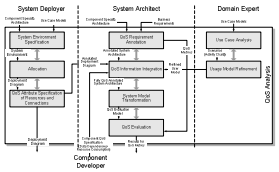


A Component Model

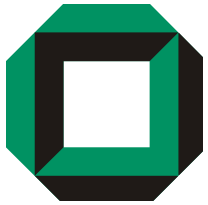


$$\sum_{i=0}^{N} p_l(i) \left(\circledast_{j=1}^{i} f_P \right)(t)$$

Multiple Analysis Methods



A Development Process

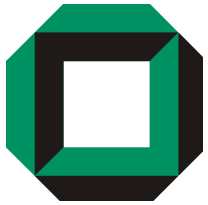


Agenda

SDQ Software Design and Quality

- Research Group
- Overview: The Role Concept
- The Palladio Component Model
- Analysis Methods & Transformations
- Excerpts from CoCoME Models & Prediction Results
- Conclusion

Photo: pixelio.de



Our Research Group

People & Topics

Klaus Krogmann:
Model-Reconstruction

Johannes Stammel:
Performance-Maintainability
-Trade-off

Steffen Becker:
Model-transformations
and Meta-Modelling

Jens Happe:
Analysismodel and Parallelism

Michael Kuperberg:
Resource Modelling

Henning Groenda:
Architecture Evaluation

Chris Rathfelder:
Architecture Evaluation

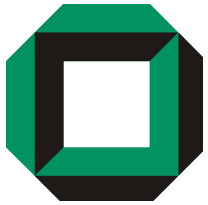
Elena Kienhöfer /
Elke Sauer:
Secretary

Ralf Reussner:
Head of group

Thomas Goldschmidt:
Testing Environment

Anne
Martens:
Dynamic
Architectures

Heiko
Koziolek:
Usage Model

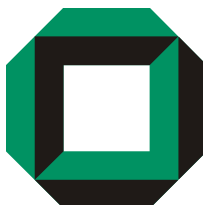


The Role Concept

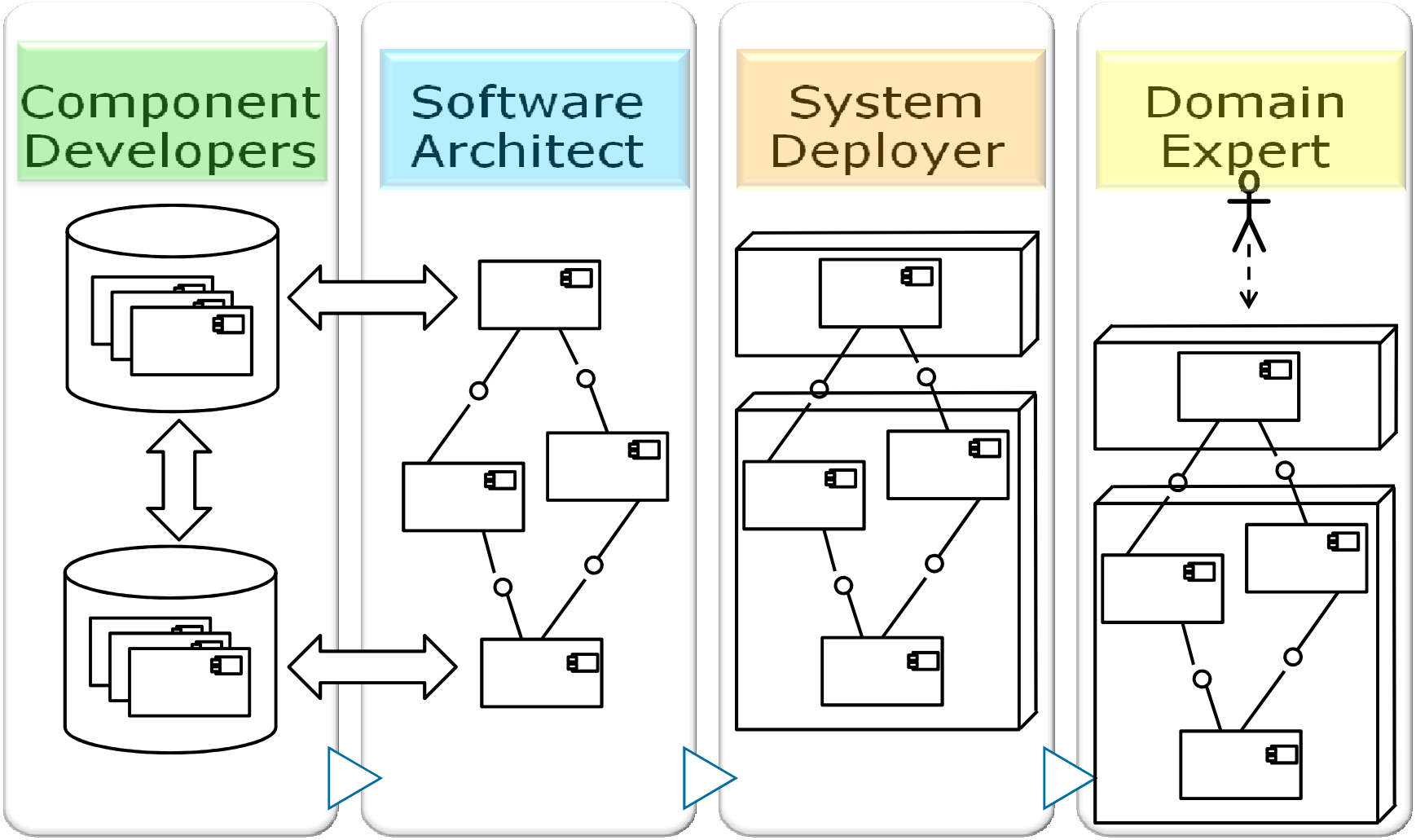
Overview

&

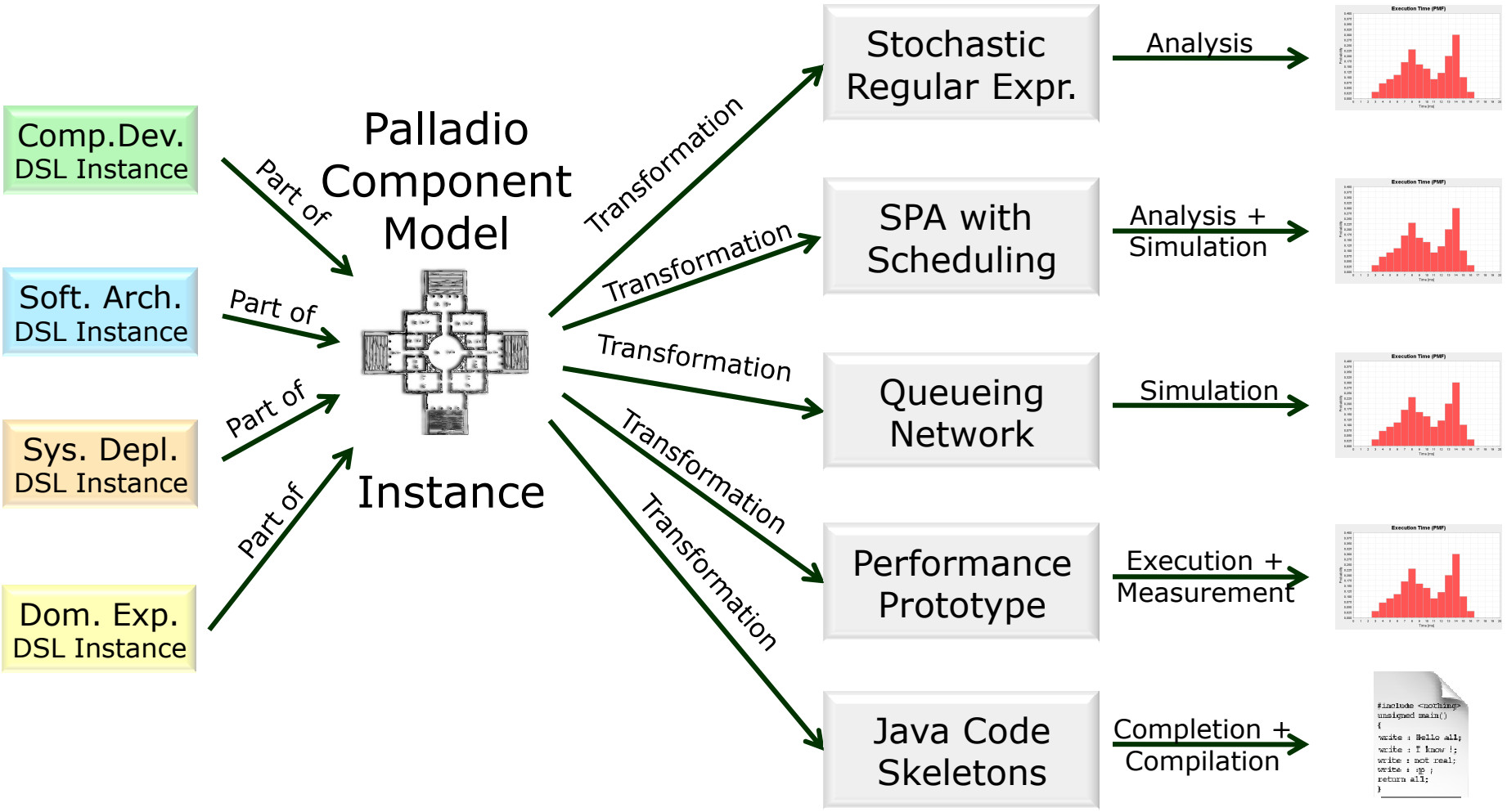
Developer Roles tied to the Palladio Component Model

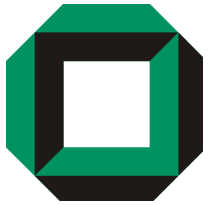


Development Roles



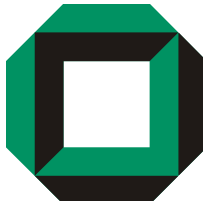
Models and Analyses





Palladio Component Model

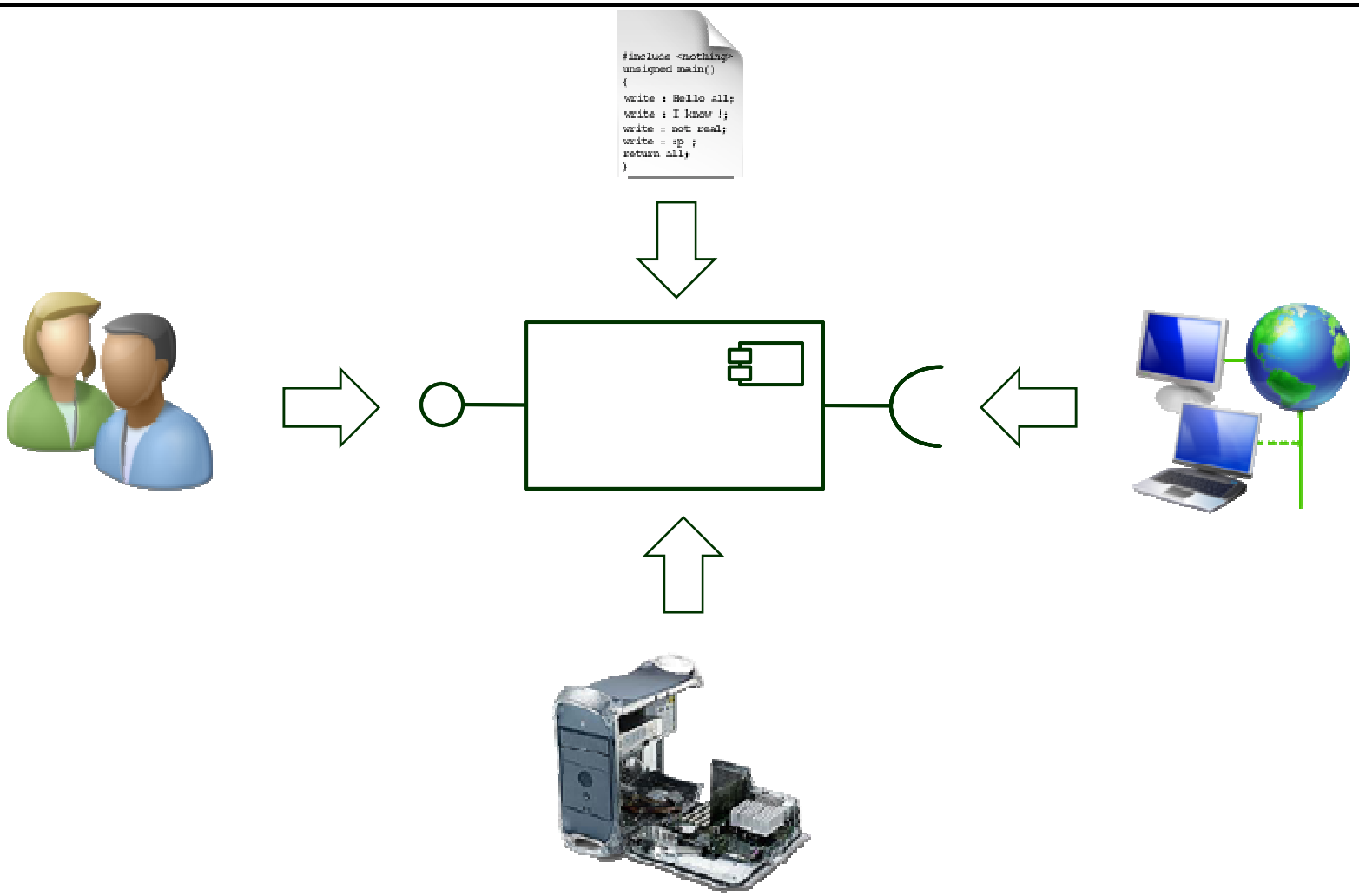
A Component Model for early Design Time Performance Predictions



Palladio Component Model (PCM)

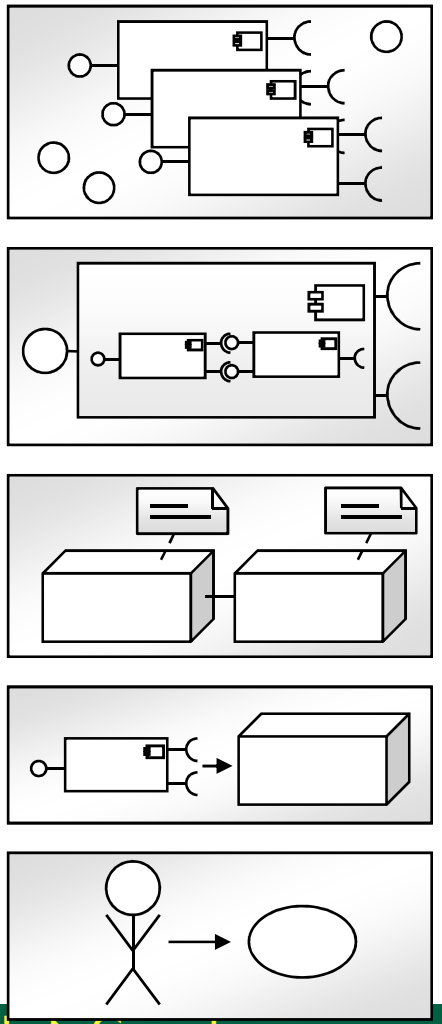
- Named after the Italian renaissance architect Andrea Palladio (1508–1580)
- Context-independent component specifications: Parameterised for re-use
- Split into sub-models:
  - Domain specific modelling languages
  - Specific for developer roles

Influences on Component Performance

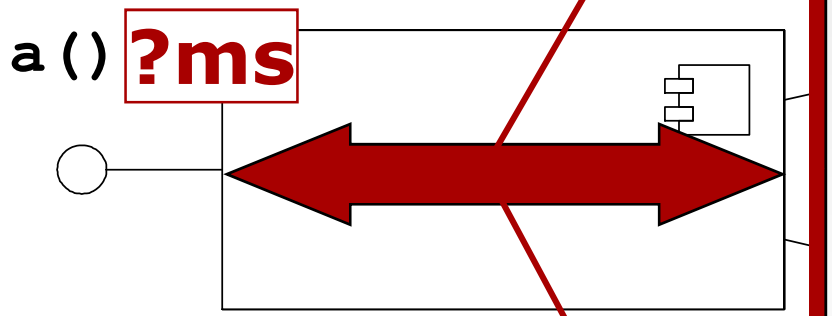


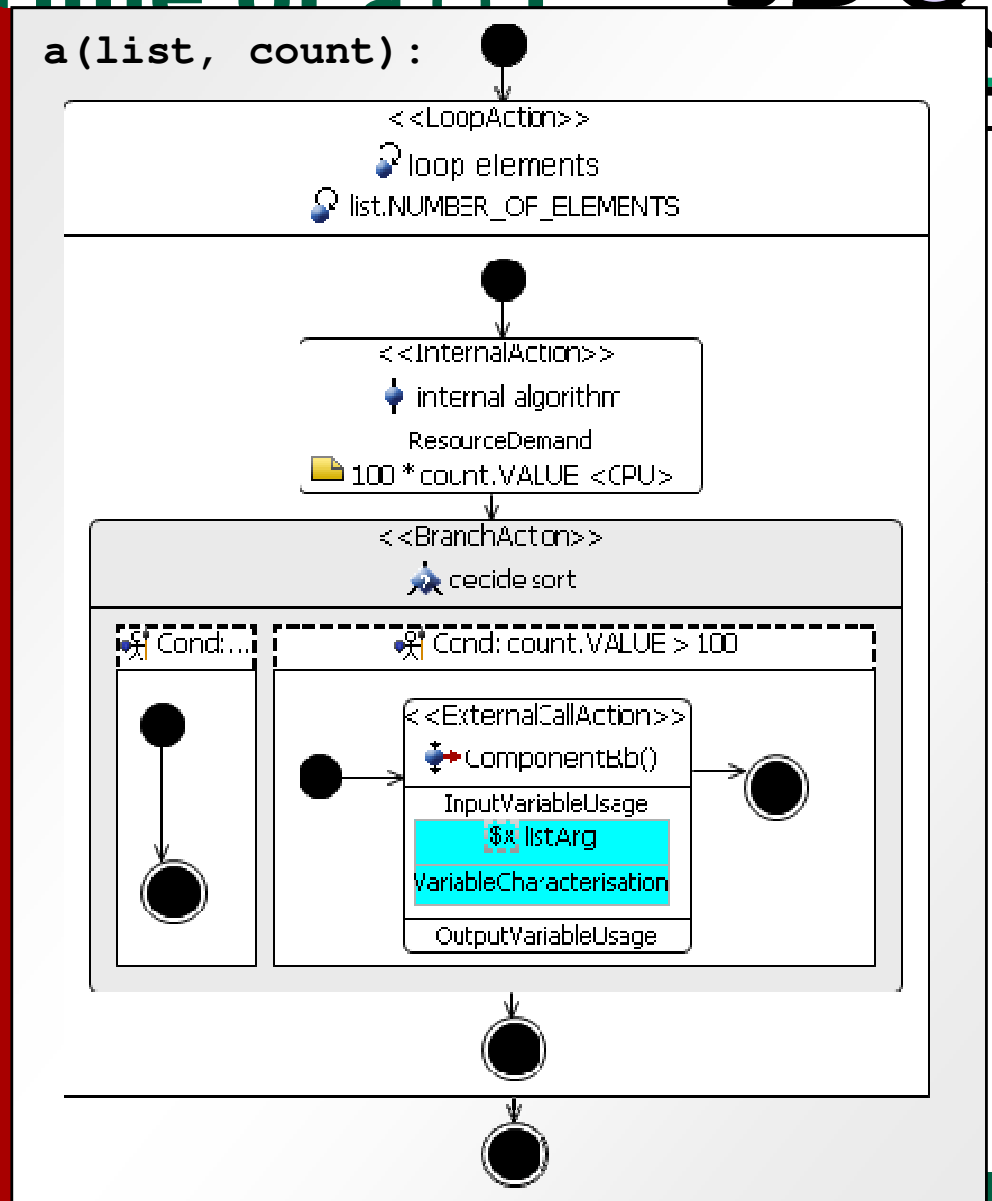
Sub-Models

- Repository model
  - Components and Interfaces
  - Service Effect Specification (SEFF)
- System model
  - Component Assembly
- Resource environment model
  - Resource types model
- Allocation model
- Usage model

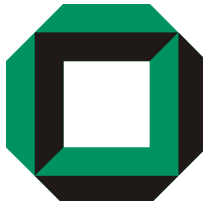


Execution Time of a()?



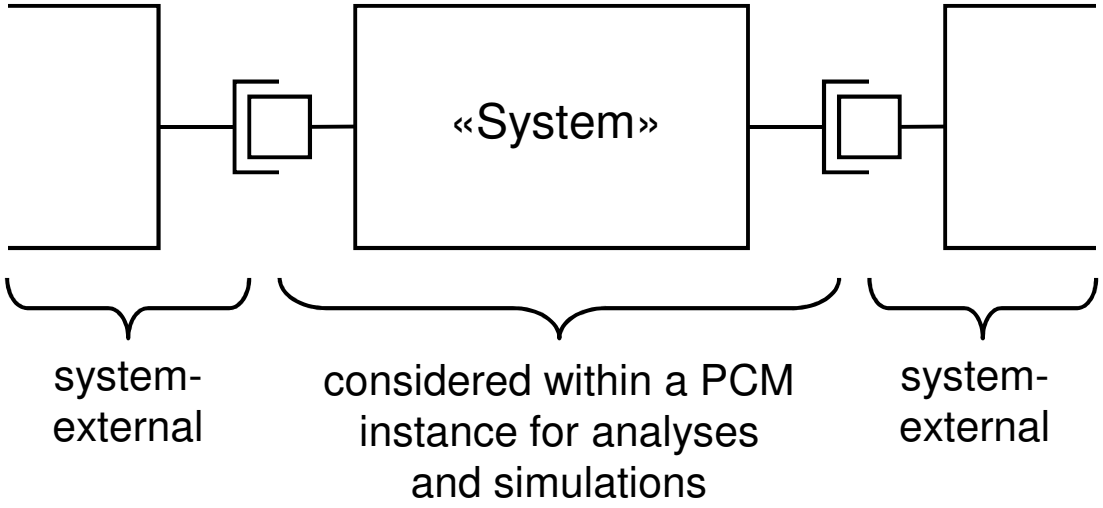


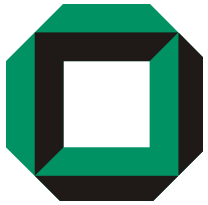
Service Effect Specification (SEFF)



System

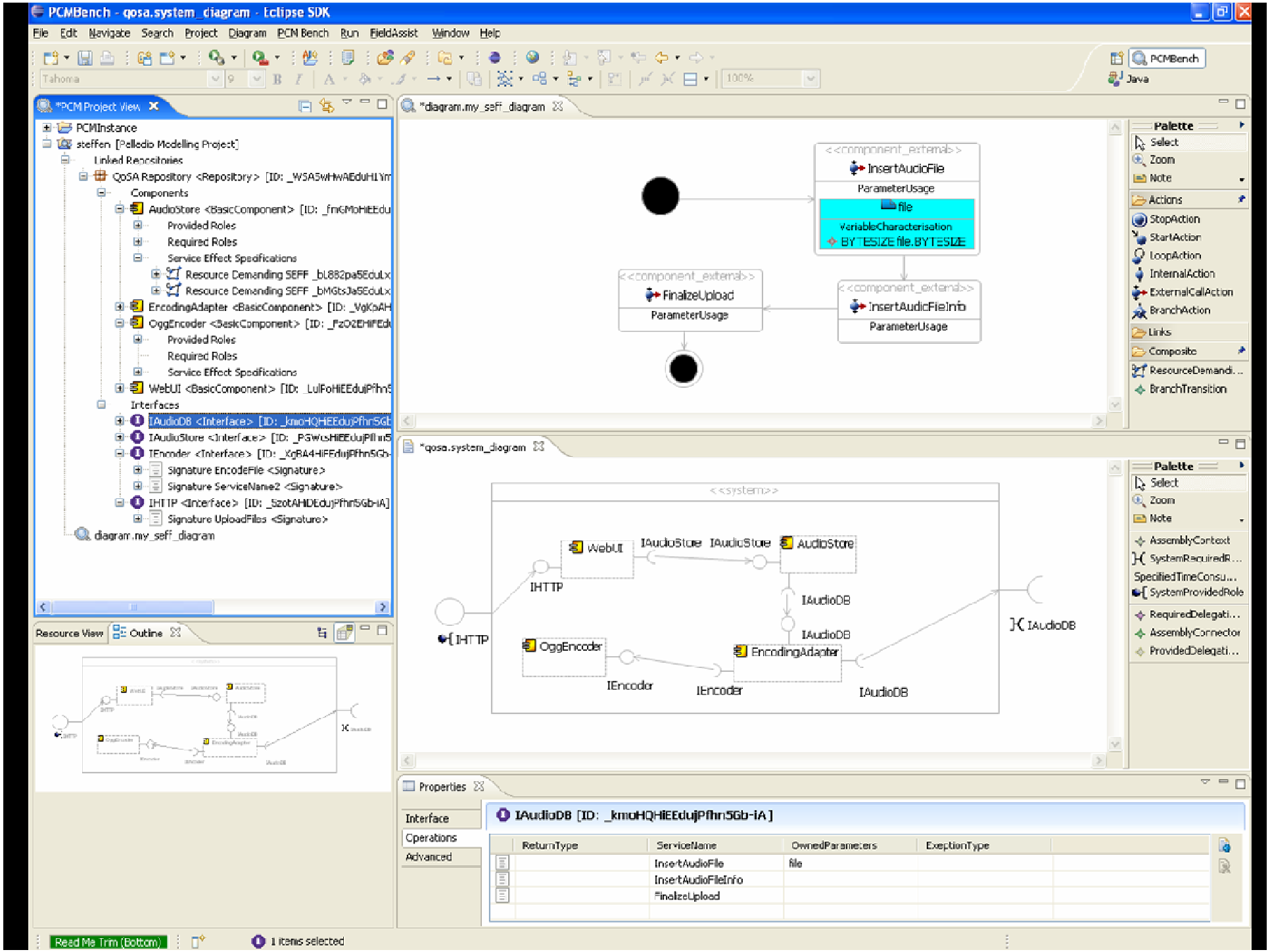
SDQ
Software Design
and Quality


«System»
system-
external
considered within a PCM
instance for analyses
and simulations
system-
external

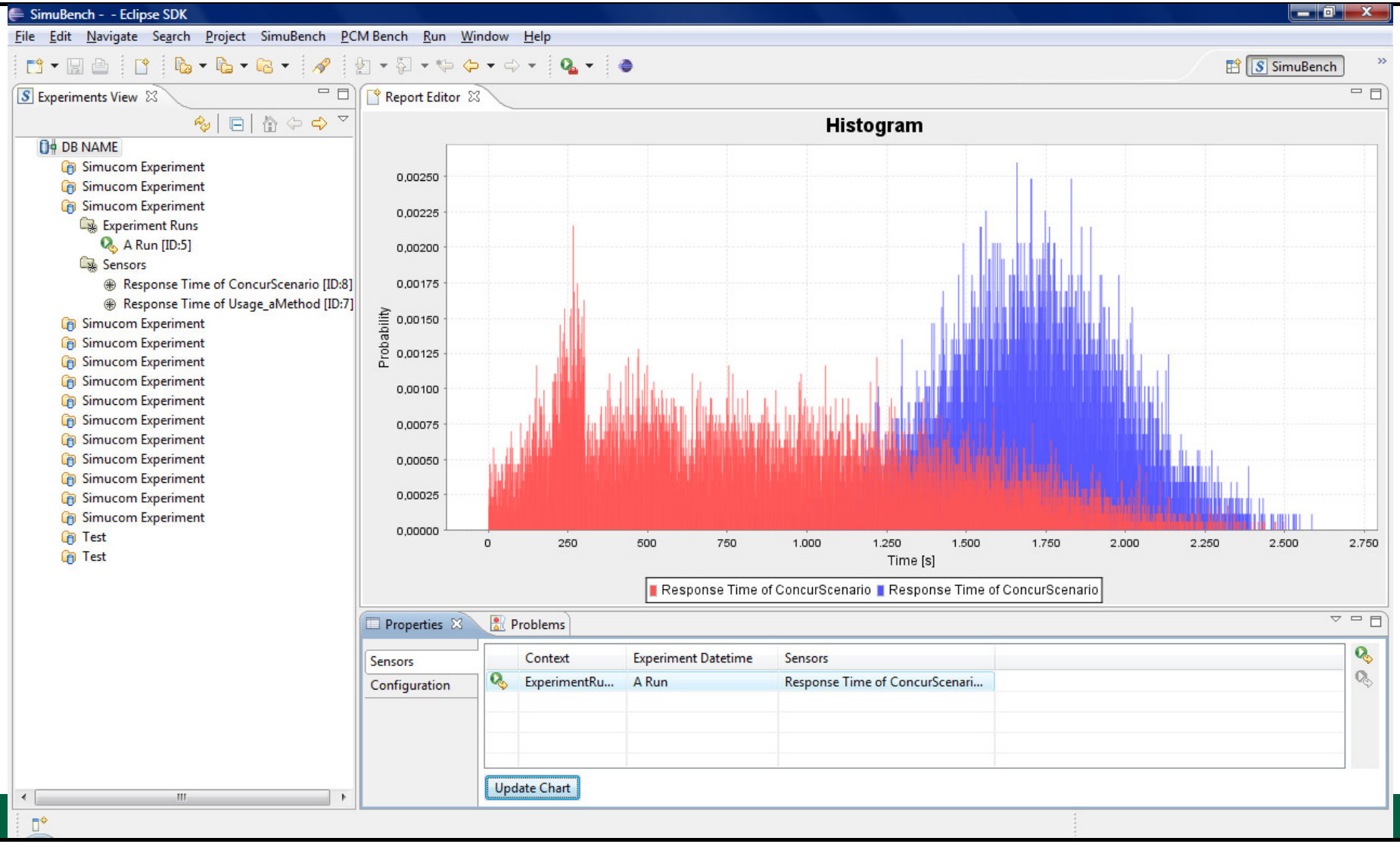


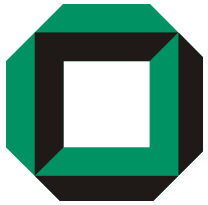
Meta-Model

- Syntax
  - Concrete syntax:
    Similar to UML 2 diagrams
  - Abstract syntax:
    - PCM is defined in the ECORE meta-meta-model
    - Different concepts than UML 2 meta-model
- Semantics
  - Static semantics: OCL constraints
  - Dynamic semantics: Technical report


PCMBench - qosa.system_diagram - Eclipse SDK
File Edit Navigate Search Project Diagram PCM Bench Run FieldAssist Window Help
*PCM Project View
PCMInstance
steffen [Palladio Modelling Project]
Linked Repositories
QoSA Repository <Repository> [ID: _WSA5wHWAEduHL1Ym
Components
AudioStore <BasicComponent> [ID: _fmKGMoHiEEdu
Provided Roles
Required Roles
Service Effect Specifications
Resource Demanding SEFF _bL882pa5EduLx
Resource Demanding SEFF _bMGtsJa5EduLx
EncodingAdapter <BasicComponent> [ID: _VgKoAH
OggEncoder <BasicComponent> [ID: _FzO2EHlFEdu
Provided Roles
Required Roles
Service Effect Specifications
WebUI <BasicComponent> [ID: _LulFoHiEEdujPfhn5
Interfaces
IAudioDB <Interface> [ID: _kmoHQHiEEdujPfhn5Gb
IAudioStore <Interface> [ID: _PSWtsHiEEdujPfhn5
IEncoder <Interface> [ID: _XgRA4HFEdujPfhn5Gh-
Signature EncodeFile <Signature>
Signature ServiceName2 <Signature>
IHTTP <Interface> [ID: _5zotAHDEdujPfhn5Gb-iA]
Signature UploadFiles <Signature>
diagram.my_seff_diagram
*diagram.my_seff_diagram
<<component_external>>
InsertAudioFile
ParameterUsage
file
VariableCharacterisation
BYTESIZE file.BYTESIZE
<<component_external>>
FinalizeUpload
ParameterUsage
<<component_external>>
InsertAudioFileInfo
ParameterUsage
Palette
Select
Zoom
Note
Actions
StopAction
StartAction
LoopAction
InternalAction
ExternalCallAction
BranchAction
Links
Composite
ResourceDemandi...
BranchTransition
*qosa.system_diagram
<<system>>
WebUI
IAudioStore
IAudioStore
AudioStore
IHTTP
IAudioDB
IAudioDB
IHTTP
IAudioDB
OggEncoder
EncodingAdapter
IEncoder
IEncoder
IAudioDB
Palette
Select
Zoom
Note
AssemblyContext
SystemRequiredR...
SpecifiedTimeConsu...
SystemProvidedRole
RequiredDelegati...
AssemblyConnector
ProvidedDelegati...
Resource View
Outline
Properties
Interface
Operations
Advanced
IAudioDB [ID: _kmoHQHiEEdujPfhn5Gb-iA]
ReturnType
ServiceName
OwnedParameters
ExceptionType
InsertAudioFile
file
InsertAudioFileInfo
FinalizeUpload
Read Me Trim (Bottom)
1 items selected

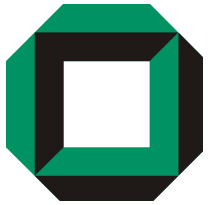
Tool Support





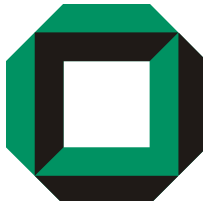
Analysis Methods & Transformations

Simulation of Queuing Networks & Stochastic Process Algebras



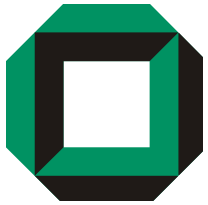
Analysis Methods

- Queuing Network
  - Simulation solution
  - Support of concurrency, scheduling strategies
- Stochastic Regular Expression
  - Analytical solution
  - No concurrency, but faster than simulation
- Stochastic Process Algebra
  - Hybrid solution (analysis + simulation)
  - High-level support of concurrency
- All Support
  - Parameterisation (usage, assembly, allocation, implementation)
  - Arbitrary distribution functions



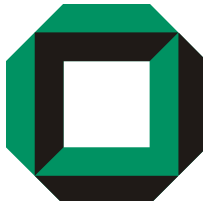
Transformations

- Used by analysis methods
- Model-2-model / model-2-text transformations use openArchitectureWare (oAW)
- Output
  - POJOs
  - EJB-System
- Supports
  - "QoS-Prototype"
  - Code skeletons



CoCoME

Palladio Models



Excerpts

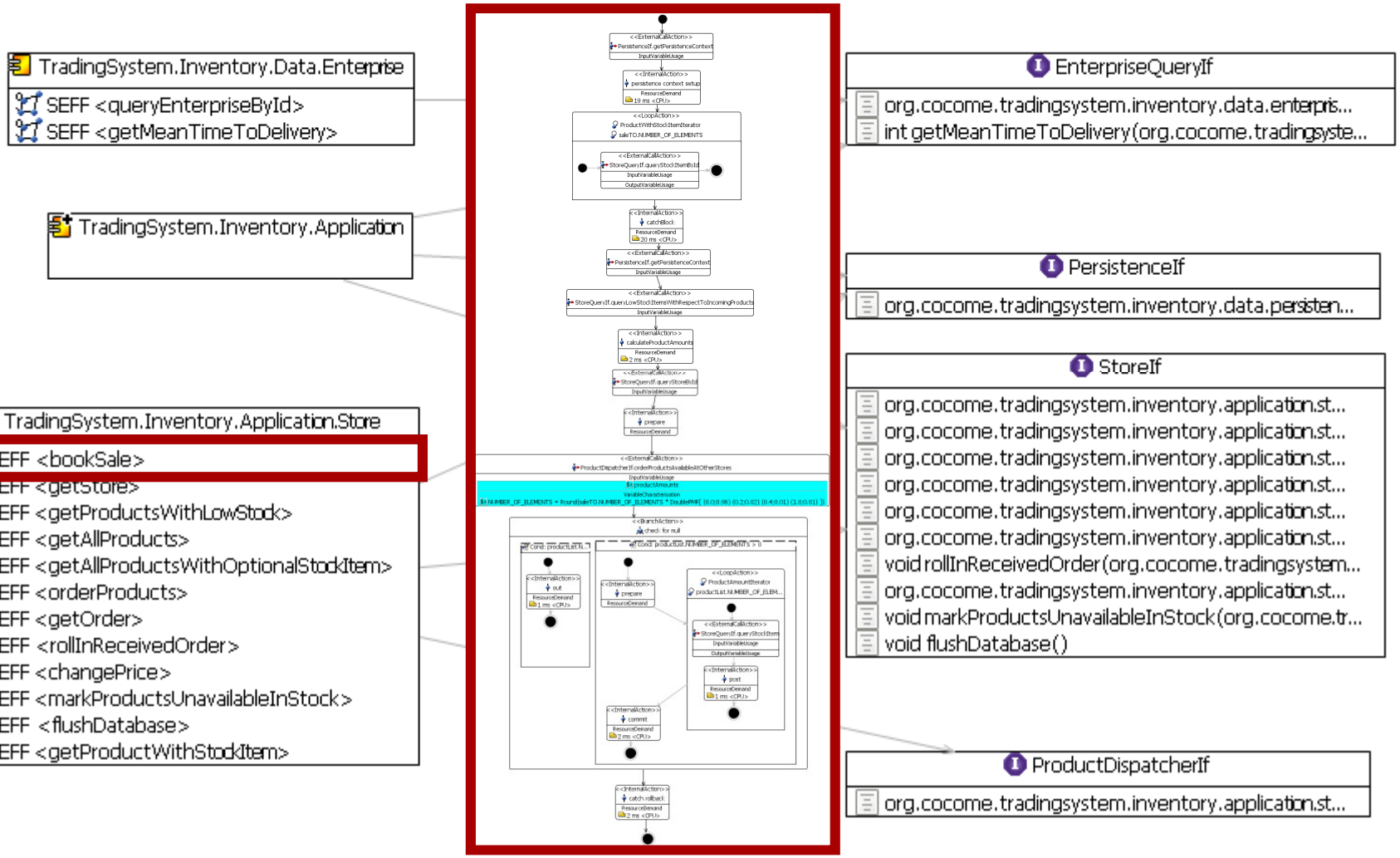
- Due to the size of CoCoME we will present only excerpts from our CoCoME models

- Repository
- SEFF
- Resource Environment
- Allocation
- Usage Model

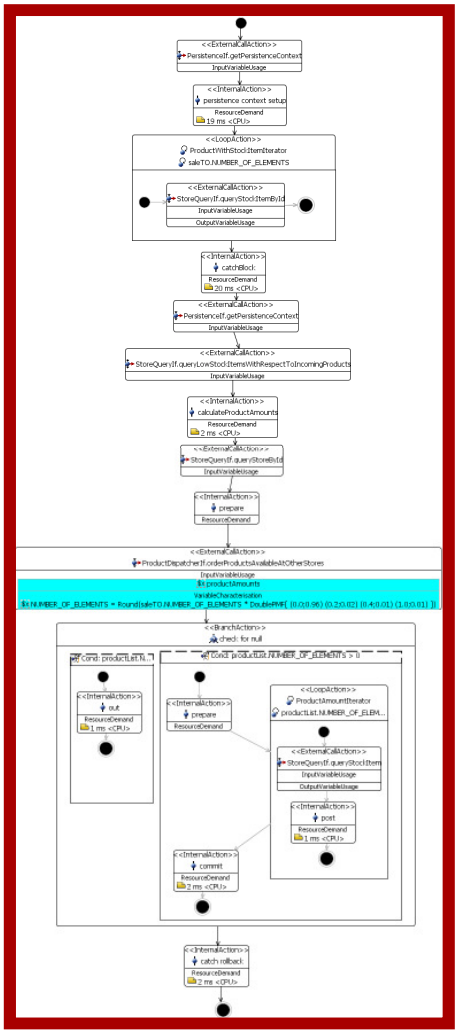
- Prediction results

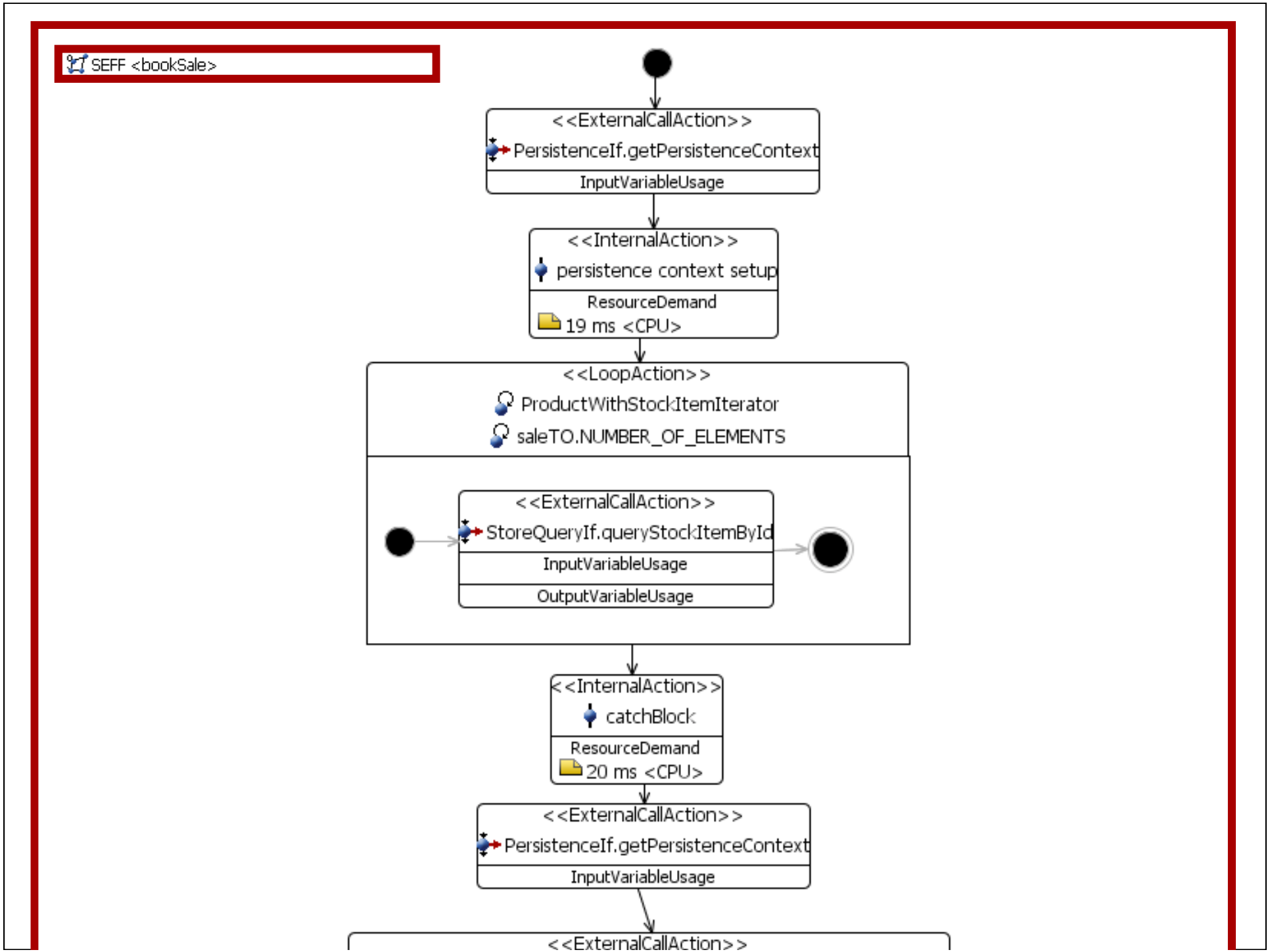
Repository Model

Repository Model

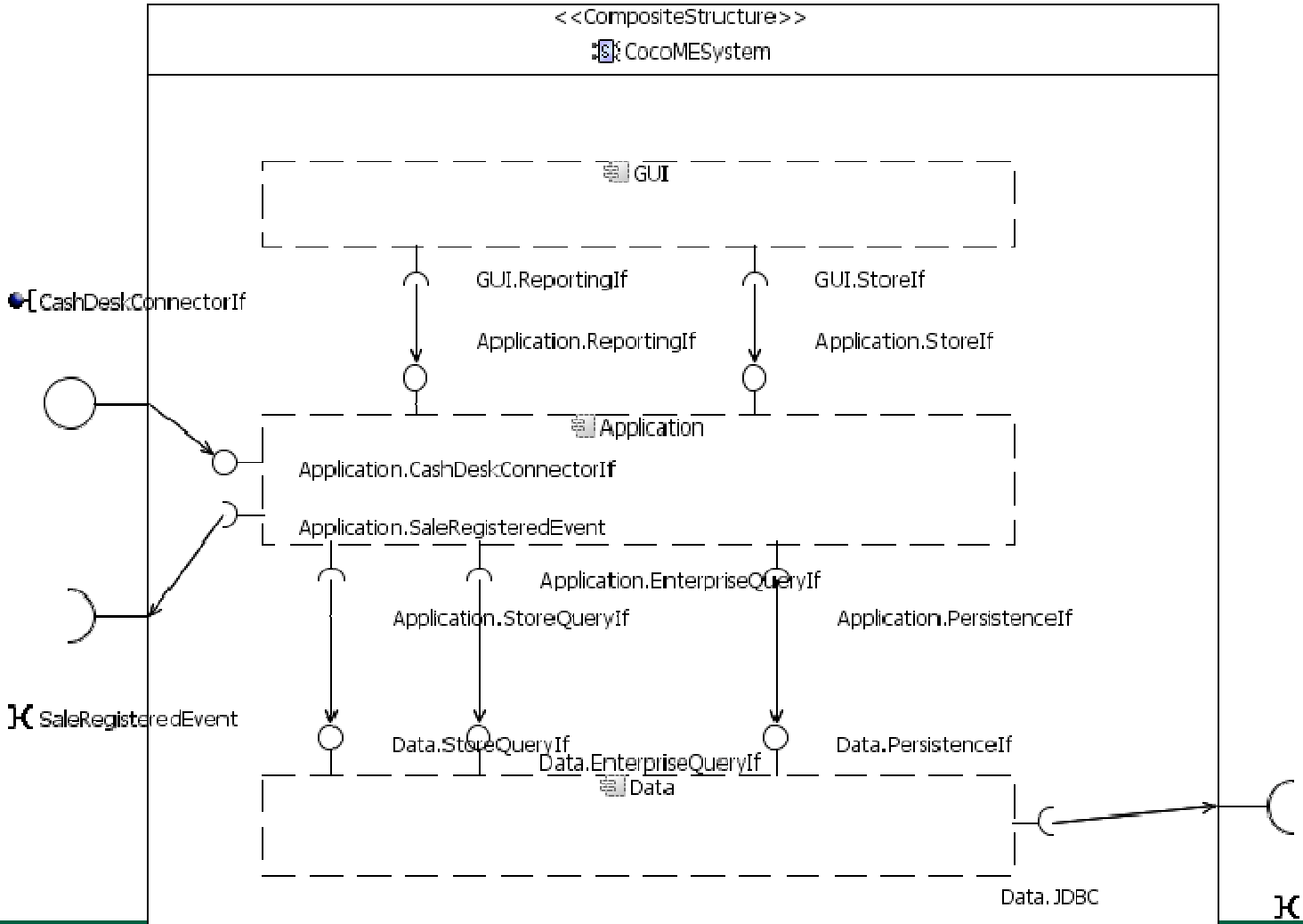


SEFF for bookSale()

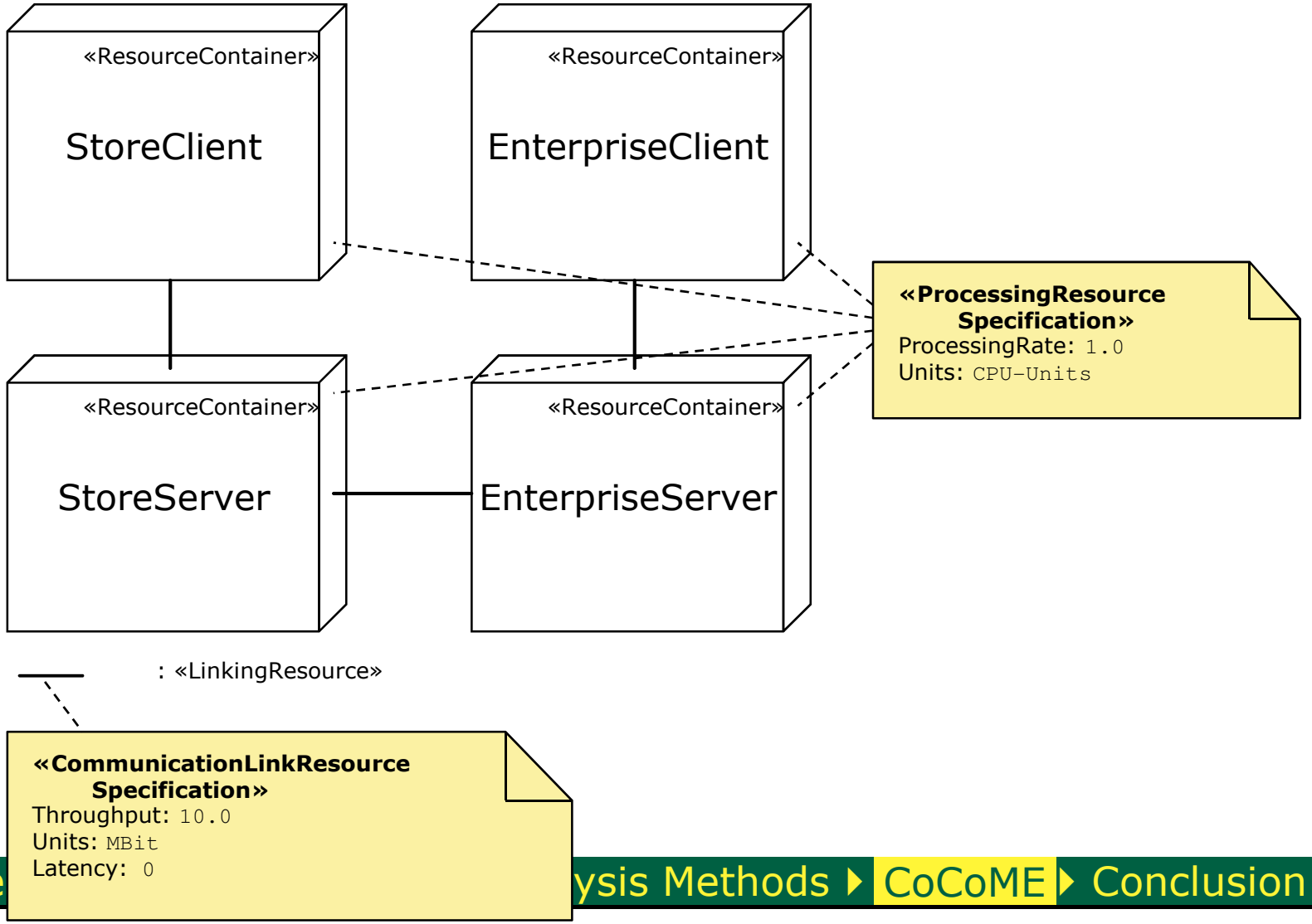



SEFF <bookSale>
<<ExternalCallAction>>
PersistenceIf.getPersistenceContext
InputVariableUsage
<<InternalAction>>
persistence context setup
ResourceDemand
19 ms <CPU>
<<LoopAction>>
ProductWithStockItemIterator
saleTO.NUMBER_OF_ELEMENTS
<<ExternalCallAction>>
StoreQueryIf.queryStockItemById
InputVariableUsage
OutputVariableUsage
<<InternalAction>>
catchBlock
ResourceDemand
20 ms <CPU>
<<ExternalCallAction>>
PersistenceIf.getPersistenceContext
InputVariableUsage
<<ExternalCallAction>>

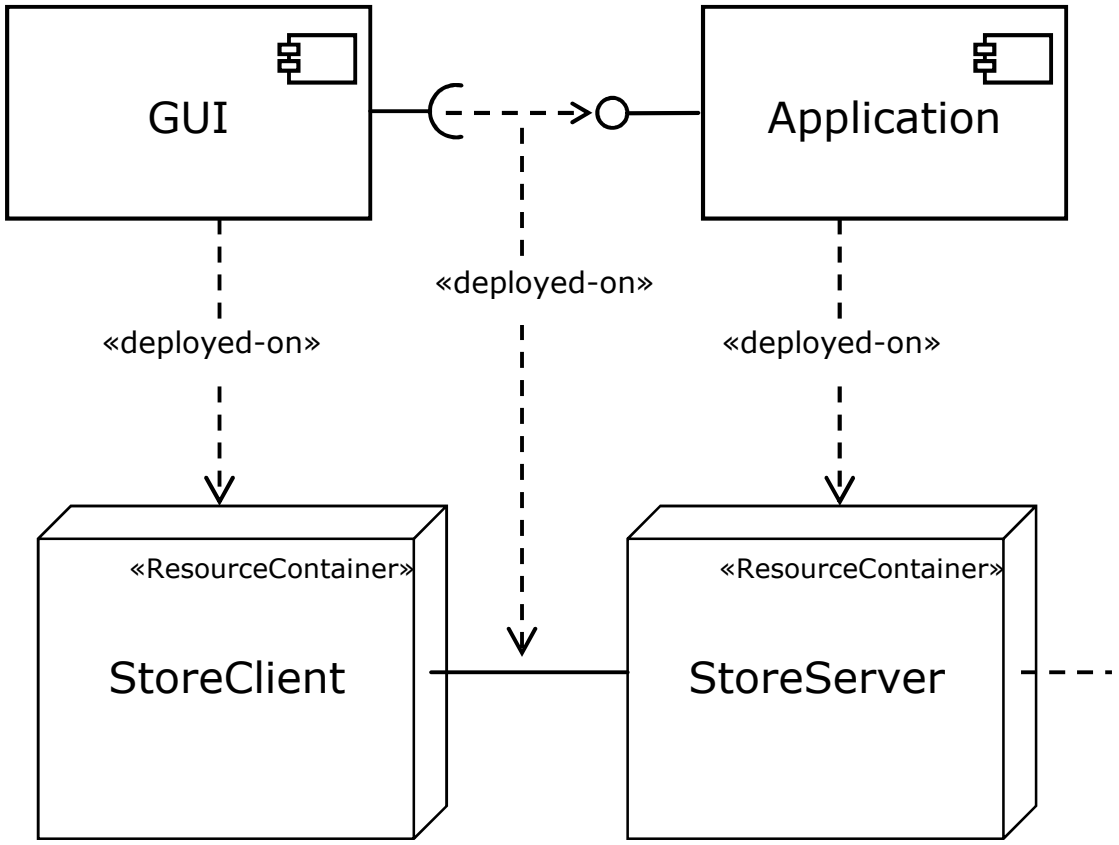
System Model



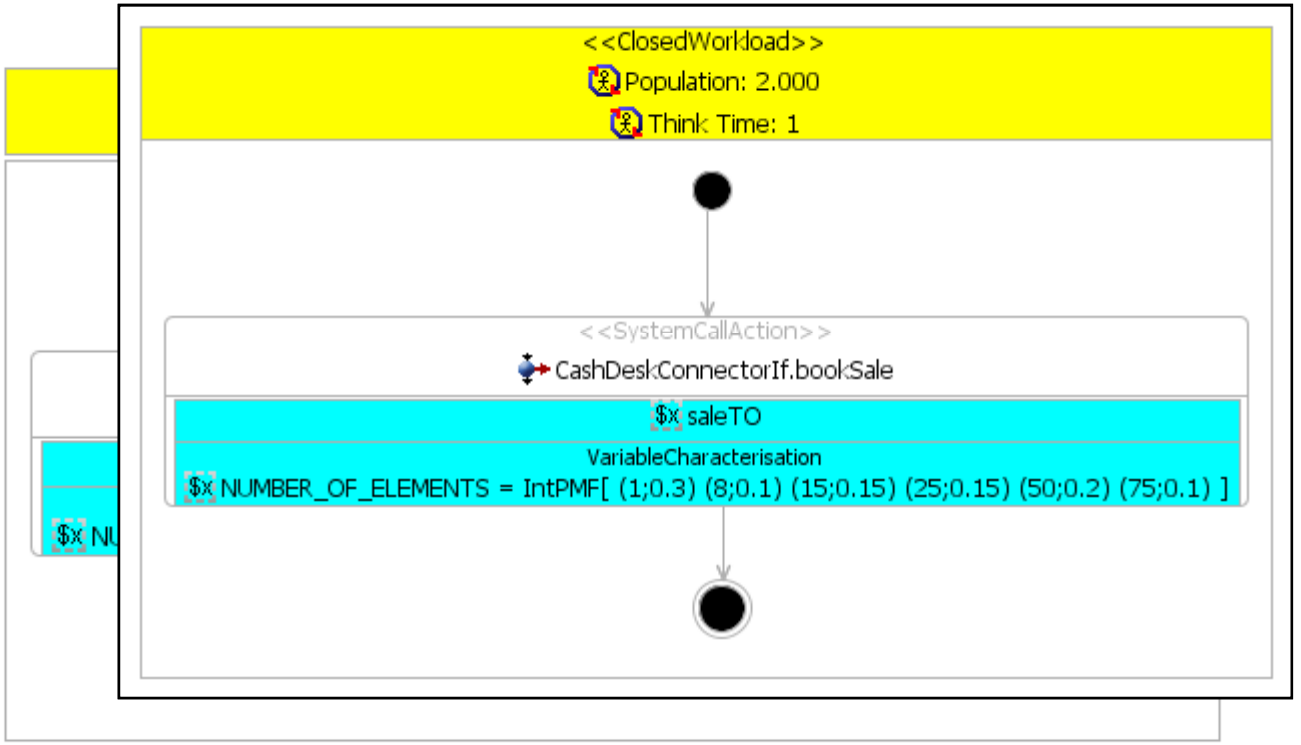
Resource Environment Model



Allocation Model

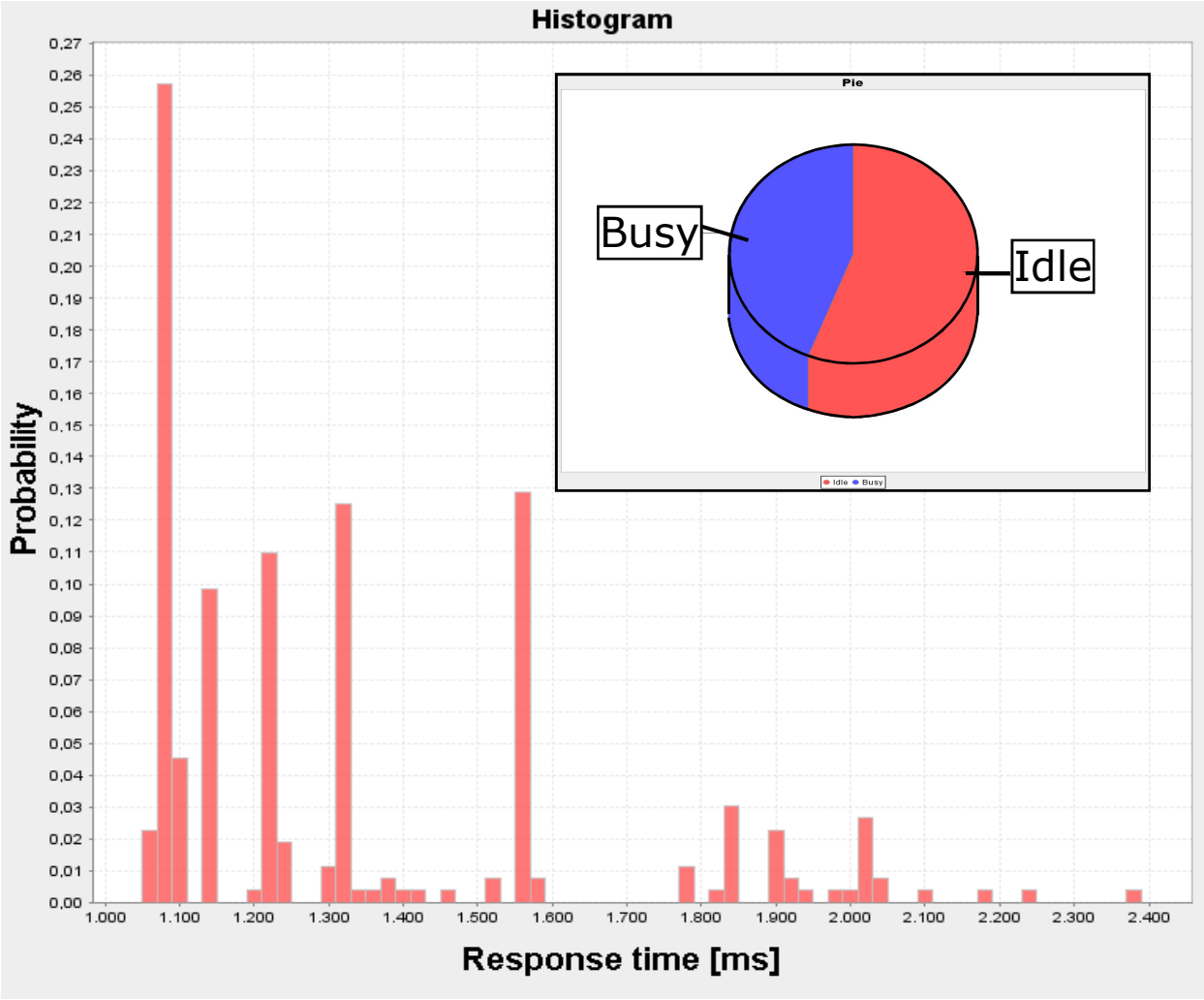


Usage Model


<<ClosedWorkload>>
Population: 2.000
Think Time: 1
<<SystemCallAction>>
CashDeskConnectorIf.bookSale
$x saleTO
VariableCharacterisation
$x NUMBER_OF_ELEMENTS = IntPMF[(1;0.3) (8;0.1) (15;0.15) (25;0.15) (50;0.2) (75;0.1)]

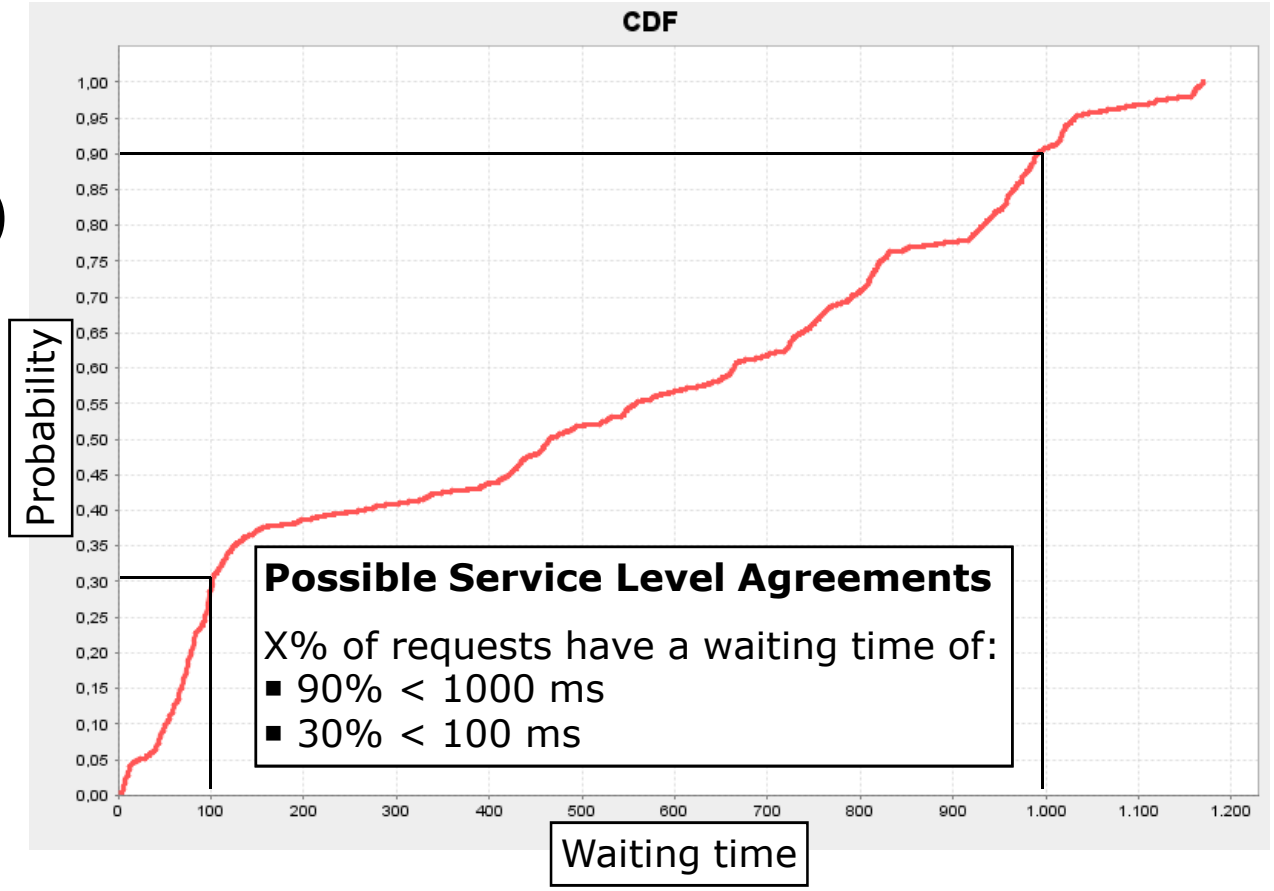
Prediction Results (1)

- **Response time**
- bookSale()
- **20** Stores
- Open workload
- Minimum: 1,050 ms
- Maximum: 2,400 ms



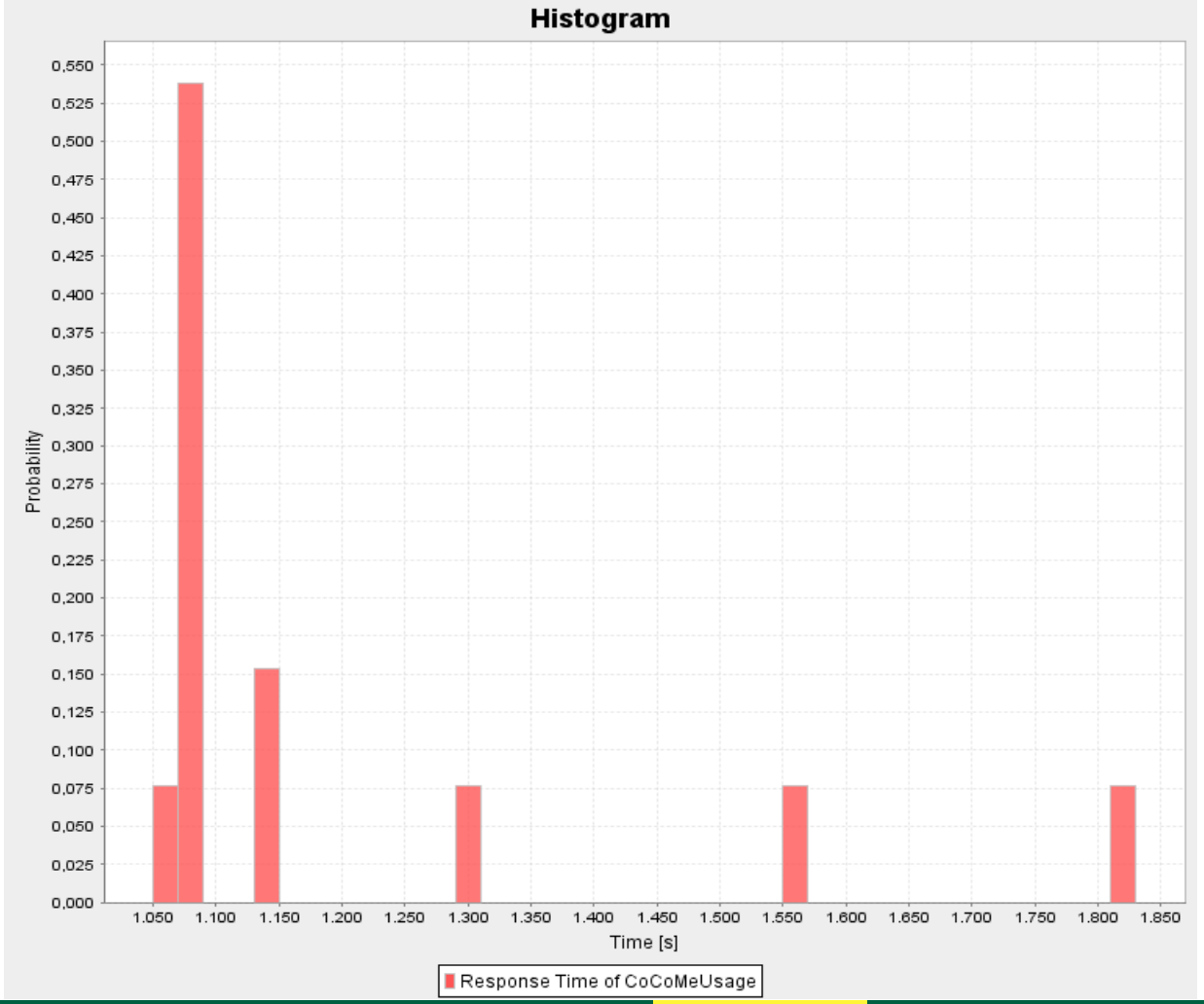
Prediction Results (3)

- **Waiting time**
- Enterprise Server (CPU)
- 20 Stores
- Open workload
- 30% of the requests are handled within 100 ms



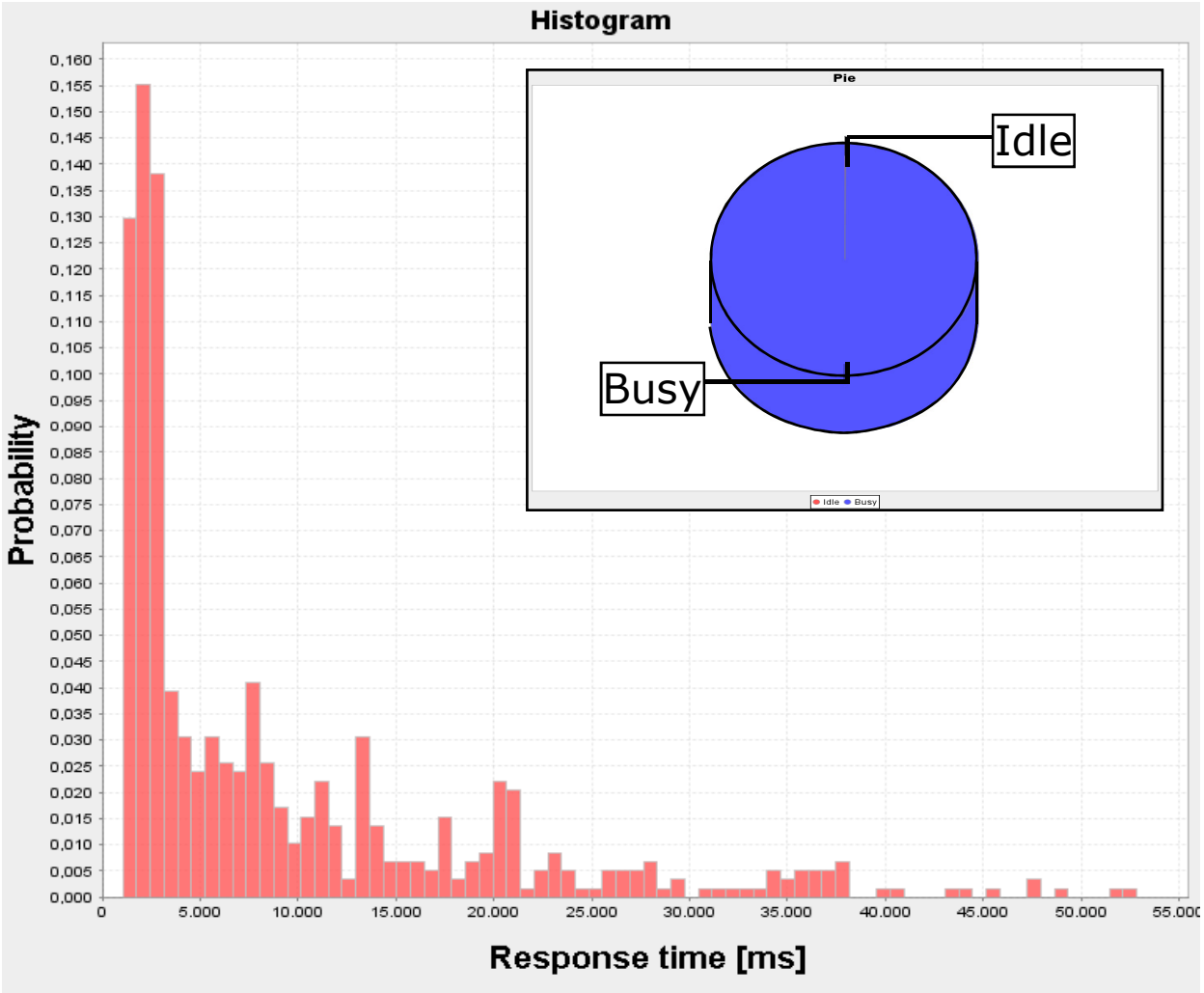
Prediction Results (2)

- Response time
- bookSale()
- **1** Store
- Open workload
- Minimum: 1,050 ms
- Maximum: 1,825 ms



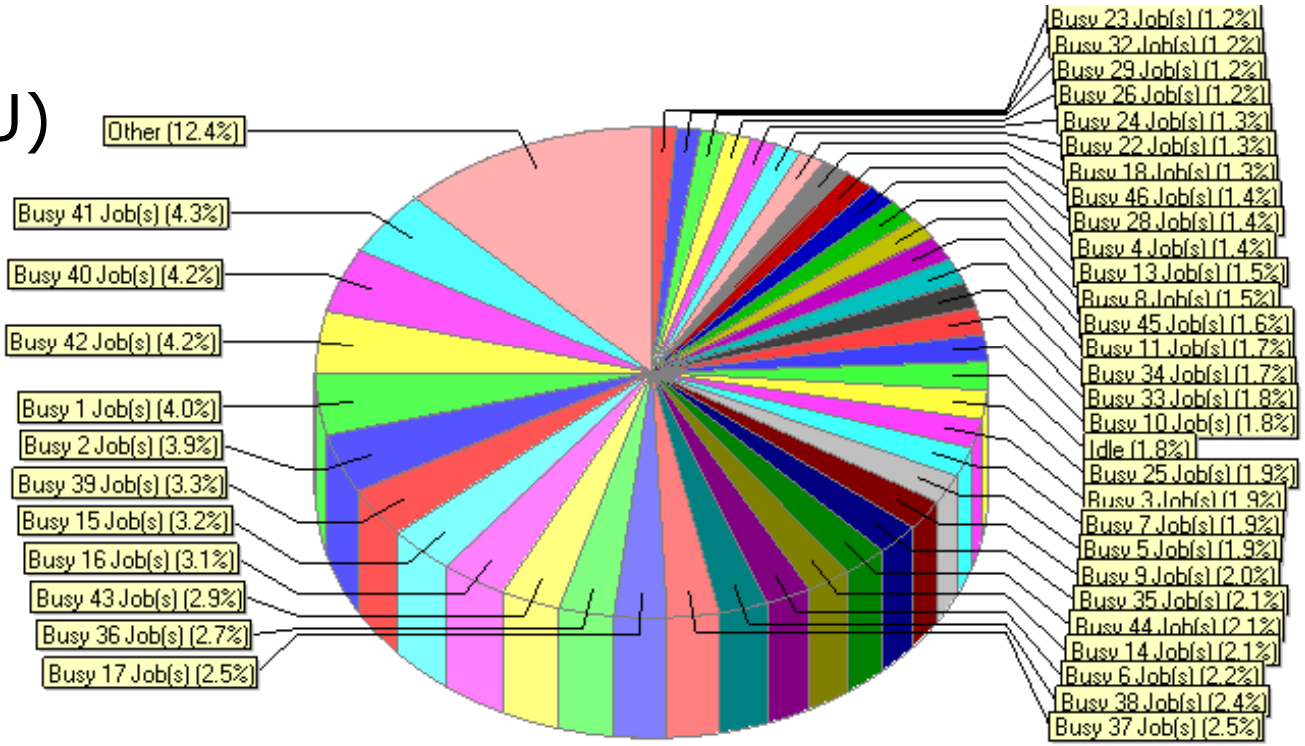
Prediction Results (2)

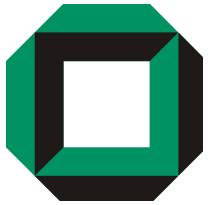
- **Response time**
- bookSale()
- **50** Stores
- Open workload
- Minimum: 1,000 ms
- Maximum: **52,000** ms
- CPU queue is overfull



Prediction Results (4)

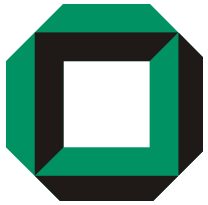
- **Utilisation**
- Enterprise Server (CPU)
- 50 Stores
- Open workload
- Up to 46 concurrent jobs
- Idle: 1.8%



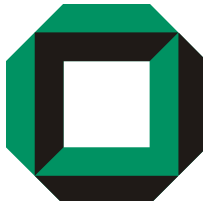


Results

- Simulation used
- All results base on the specification of the non-functional properties from the CoCoME chapter: Static time consumptions
- The results show that the enterprise server cannot handle 200 concurrently accessing store servers (as specified)

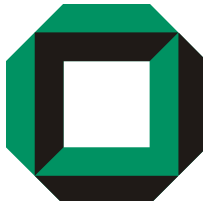


Conclusion



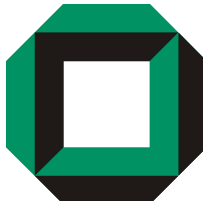
Limitations

- Current Palladio approach
  - No persistent component state
  - No dynamic architectures
  - Only one-to-one connectors
- Modelling CoCoME
  - Embedded part of the system (“POS”) was left out
  - Exceptions not modelled
  - POS and database were considered system-external



Lessons learned

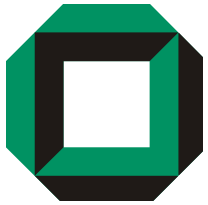
- Support of “sub-systems” apart from composite components would have been useful
- One-to-many connections and replication (store servers) should be supported by the PCM
- CoCoME: good debugging and testing system



Future Work

- Automation of model reconstruction for given source code
- "High-level-modelling": Concurrency, synchronisation
- Dynamic Architectures
- Resource Model

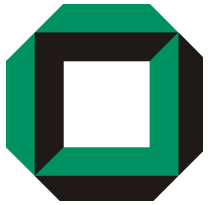
- Measure an adapted implementation of CoCoME
  → compare to the prediction results



More Information

http://sdqweb.ipd.uka.de/wiki/CoCoME-PCM

- CoCoME-Models
- Further prediction results
- Tools downloads: Modelling and Prediction
- Documentation of the Palladio Component Model
- Development Process in details



Palladio

SDQ Software Design and Quality

A Component Model

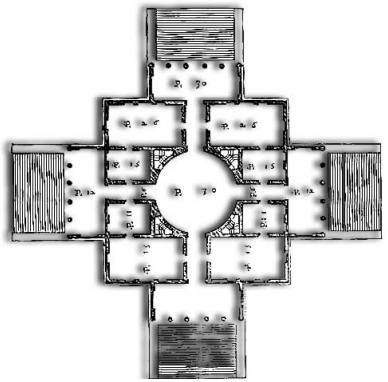
- Context independent specification
- Sub-models reduce complexity
- Arbitrary distribution functions

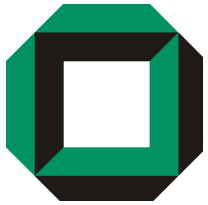
Multiple Analysis Methods

- Queuing network based simulation
- Stochastic process algebra

A Development Process

- Applicable with the PCM
- Explicit support of component ideas





References

- See the CoCoME book chapter