

Samuel Kounev, Fabian Brosig, Nikolaus Huber

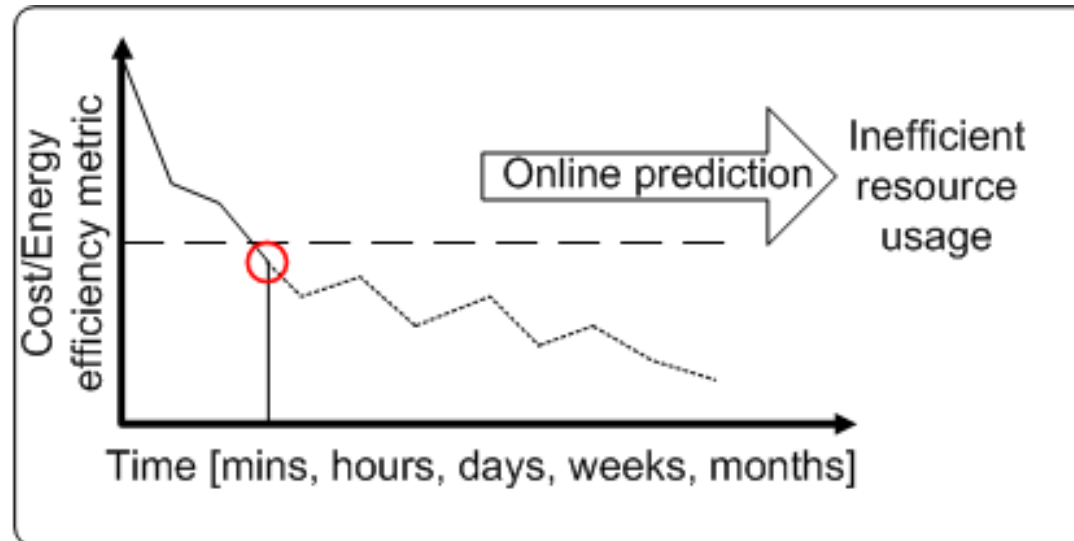
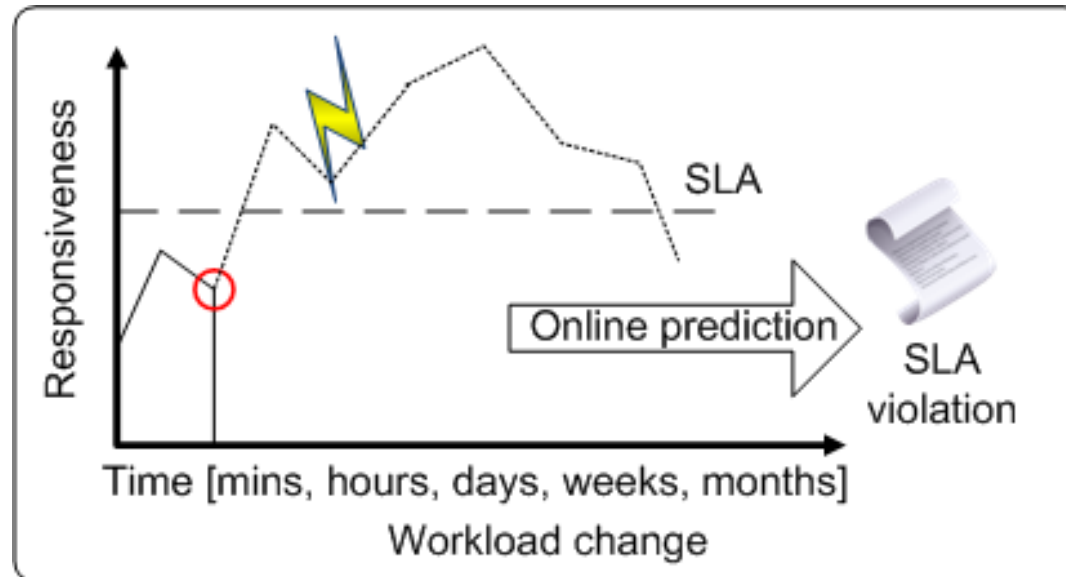
From Offline to Online Component Models for Run-Time Resource Management

Palladio Days, Karlsruhe, November 2011

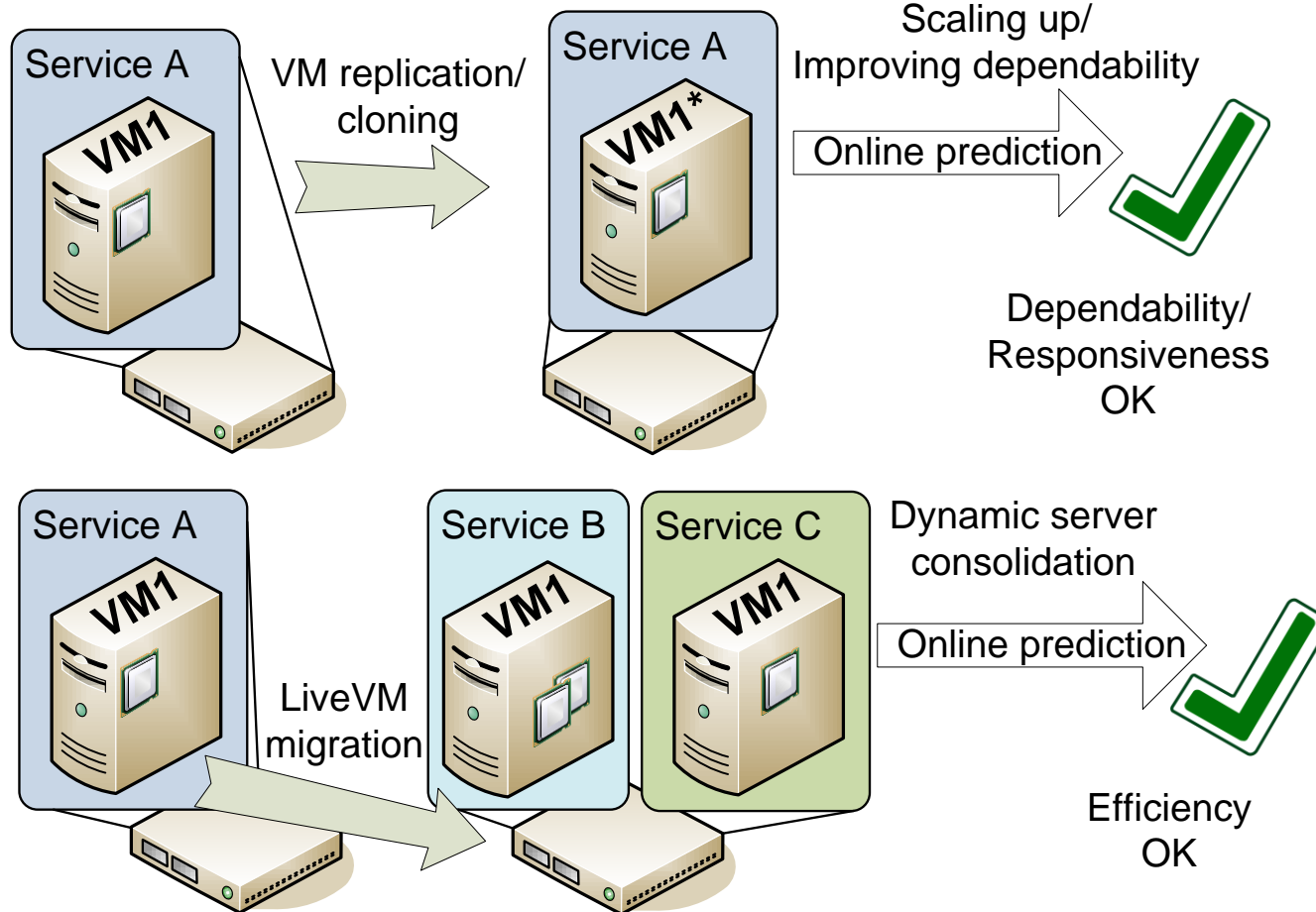
SOFTWARE DESIGN AND QUALITY GROUP
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION, FACULTY OF INFORMATICS



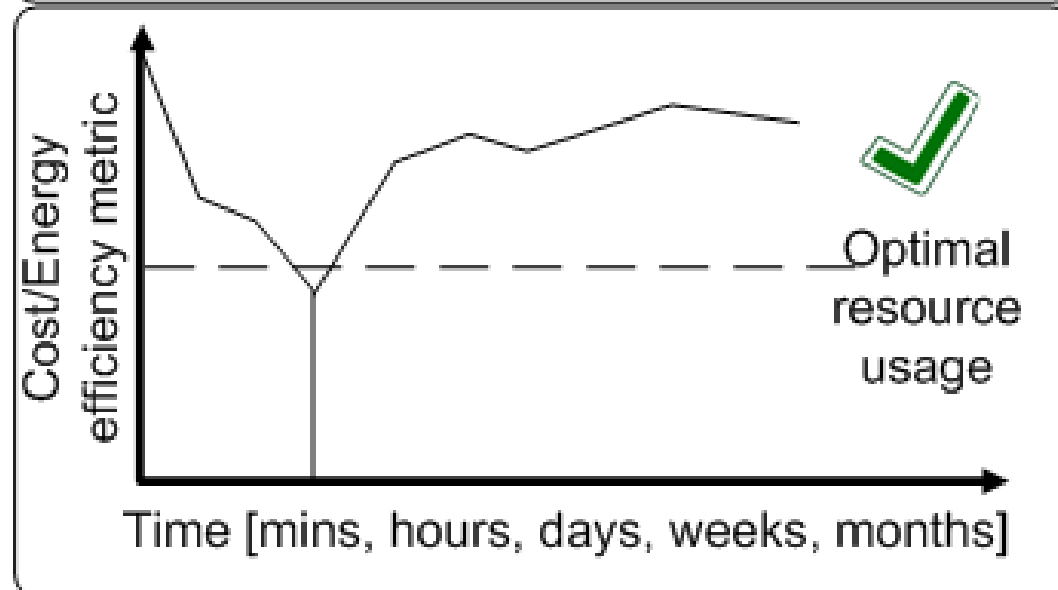
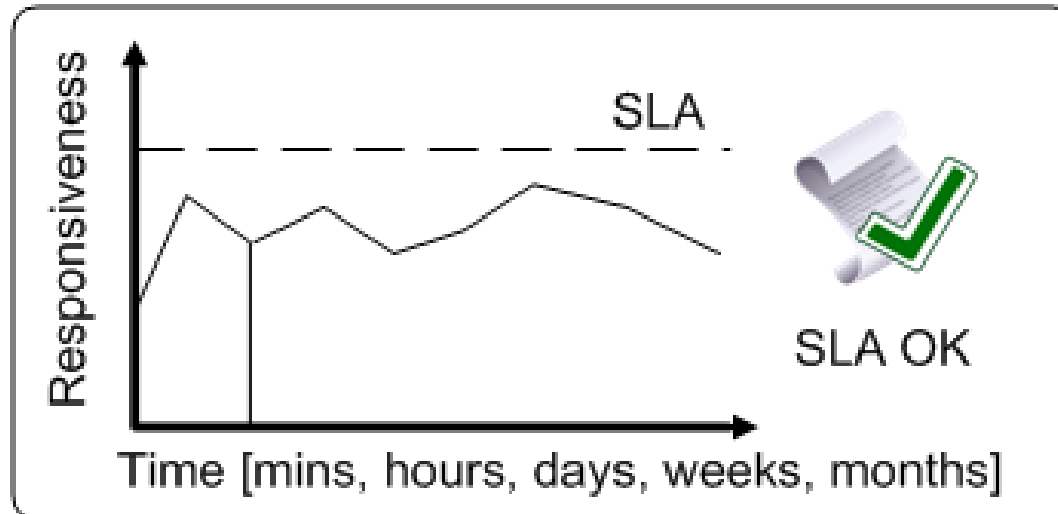
Proactive Run-time Resource Management



Online reconfiguration impact prediction for trade-off analysis

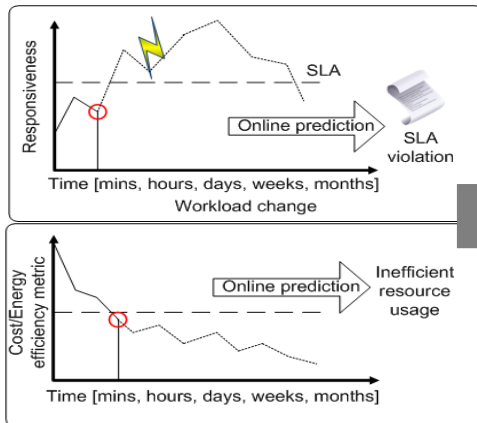


Proactive Run-time Resource Management



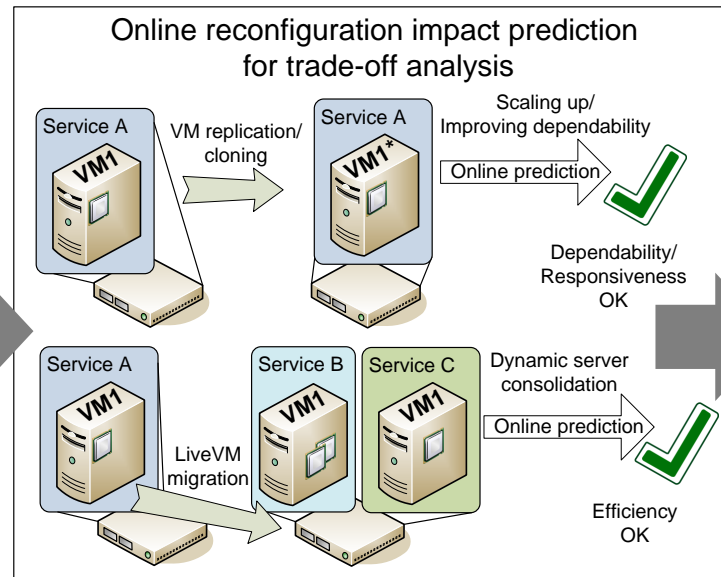
Proactive Run-time Resource Management

Online QoS prediction for problem anticipation



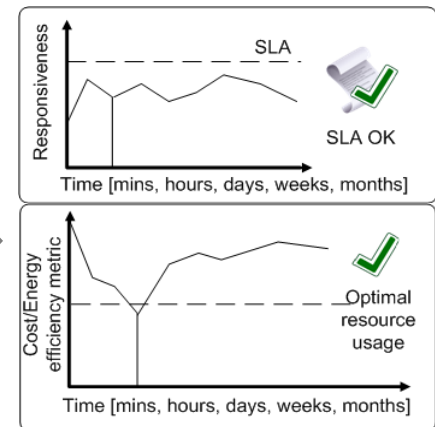
PART 1

Online QoS prediction for reconfiguration impact analysis



PART 2

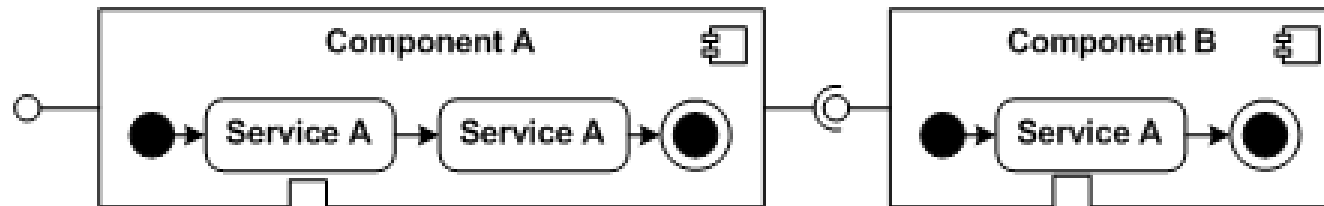
Autonomic system adaptation



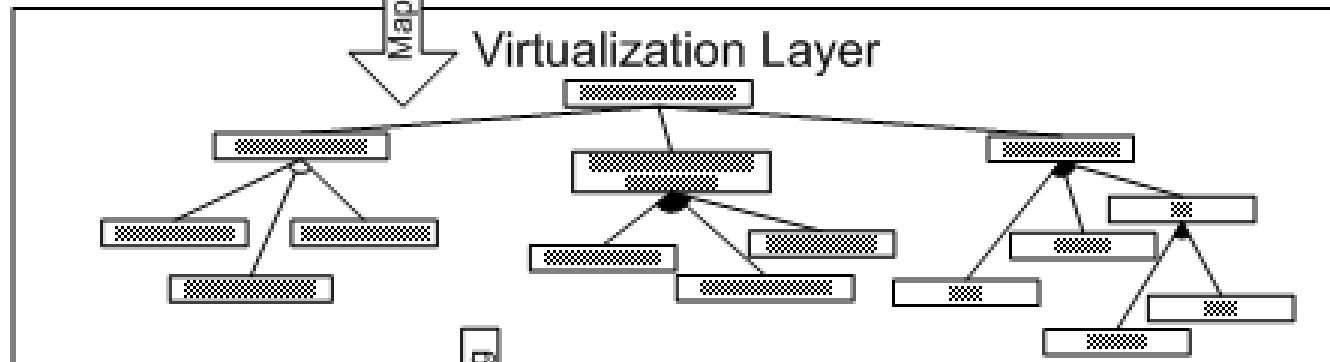
PART 3

Coming Soon: Descartes Meta-Model

Application Level



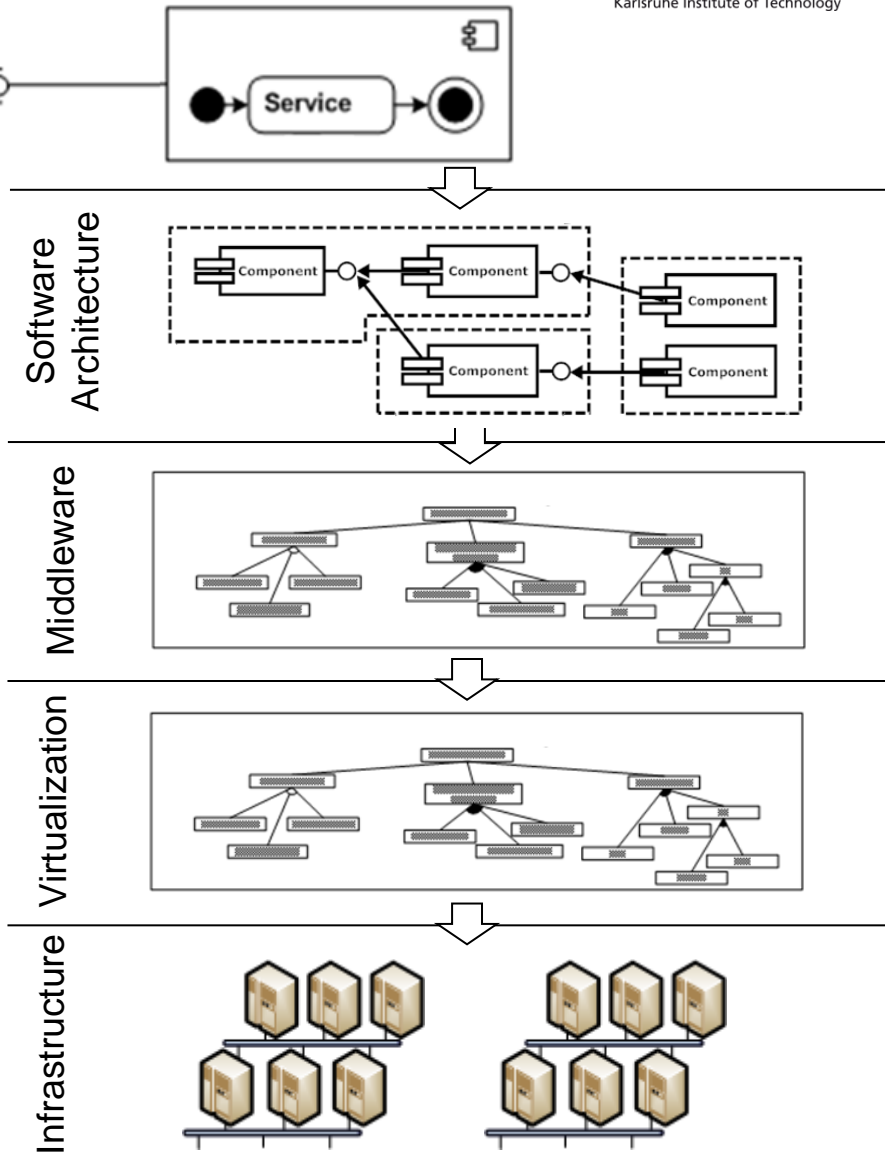
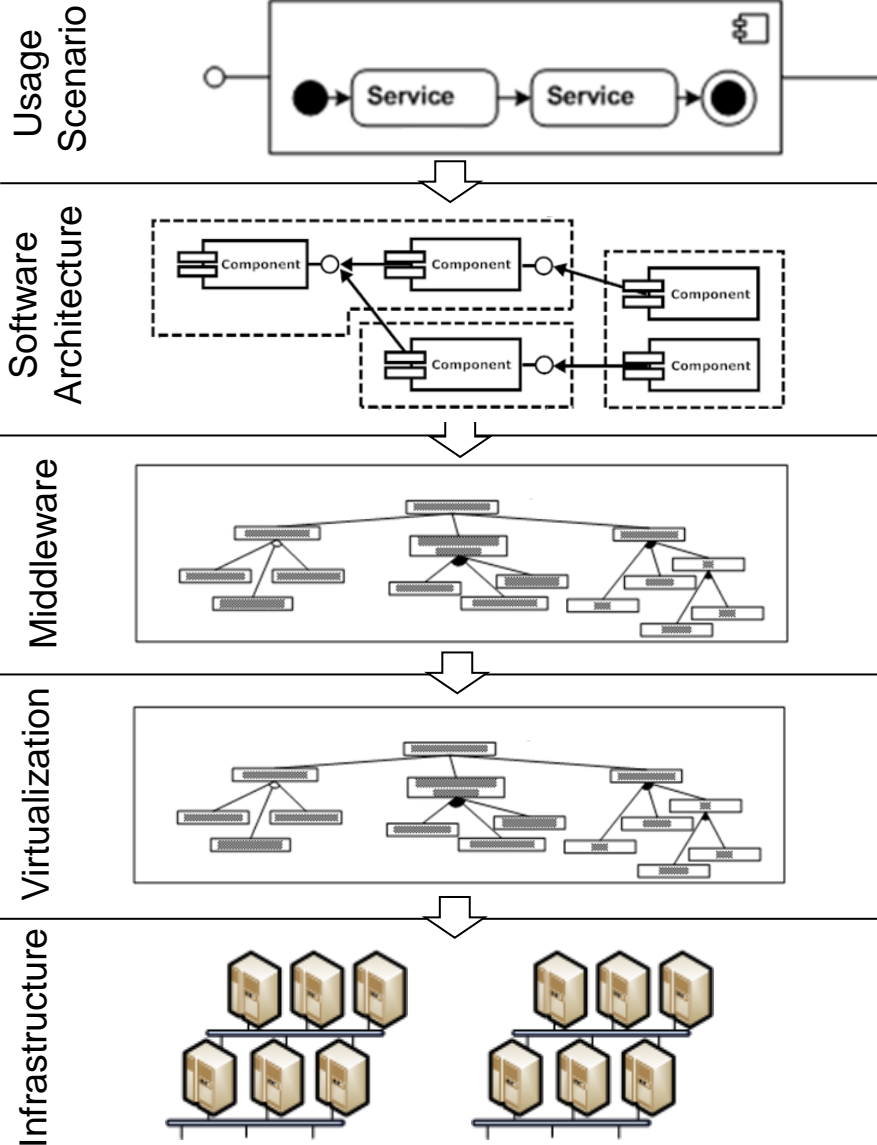
Platform Level



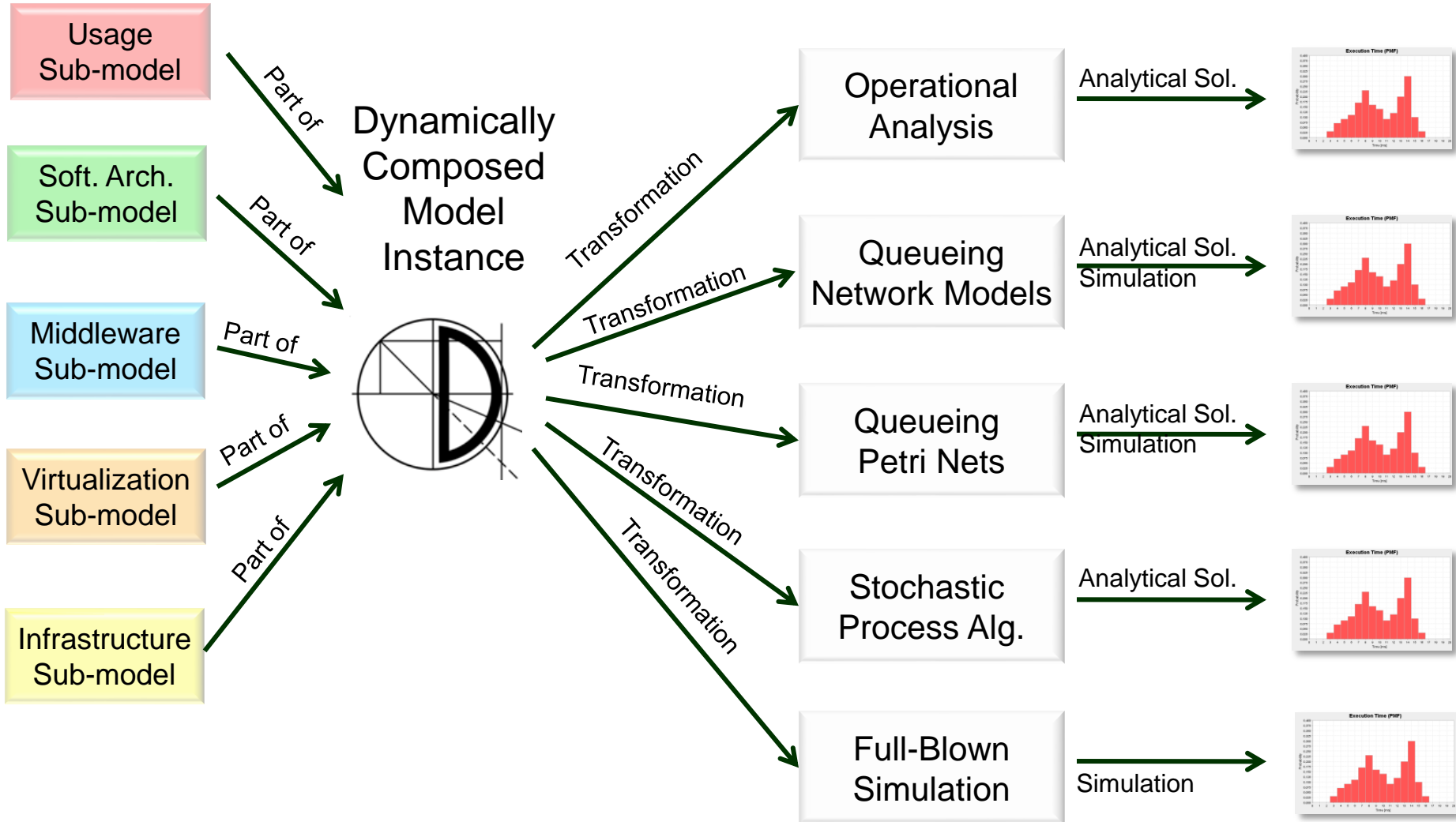
Infrastructure Level



Dynamic Model Instance Composition



Tailored Model-to-Model Transformations

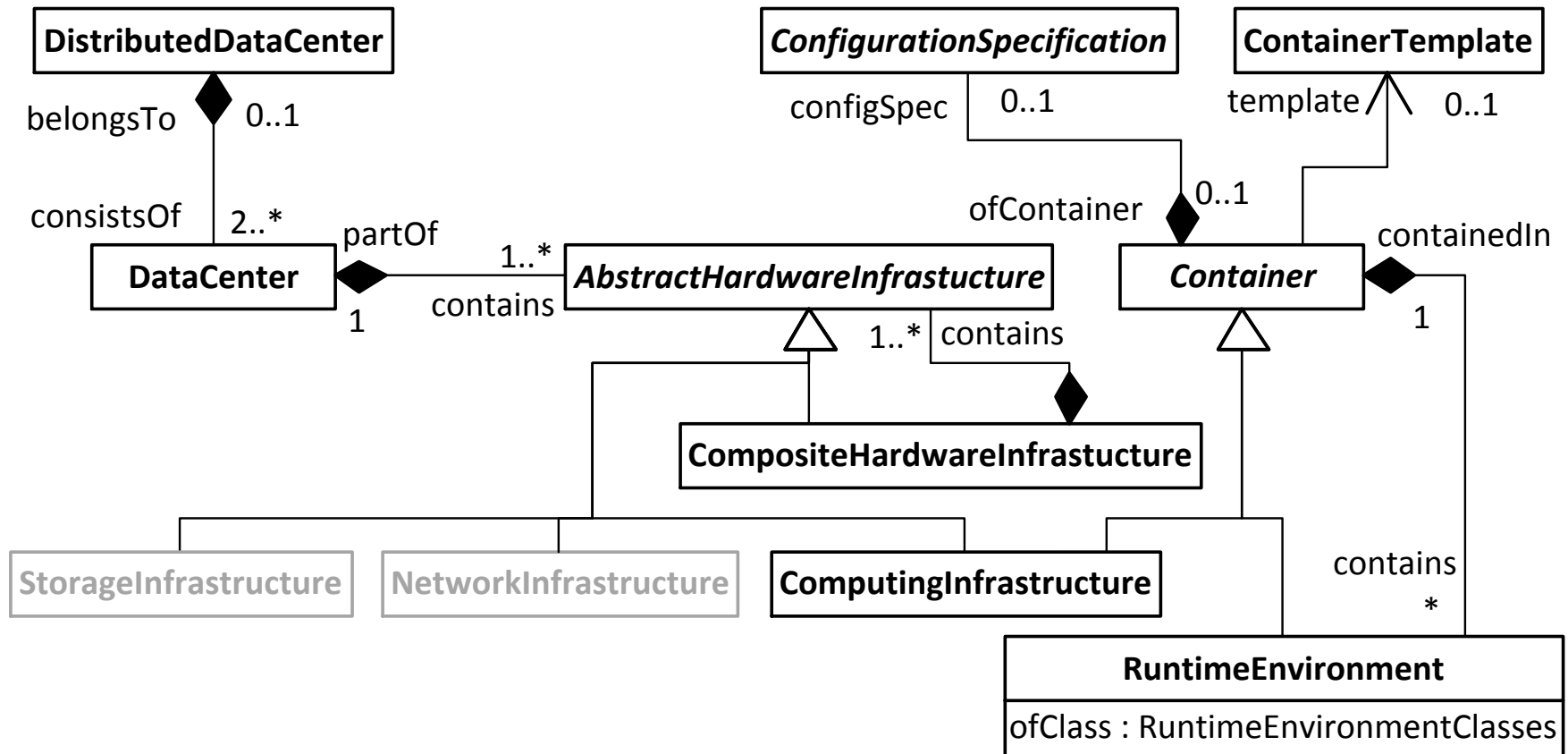


- Models should capture both static and dynamic aspects
- “Component” is a unit of composition *at run-time*
- Alignment with system layers as opposed to developer roles
- Dynamic model composition
 - Horizontally across components and vertically across layers
- Sub-models integrated into the system components
- Integration with monitoring data available at run-time
- Models intended for use by system as opposed to human
- Different abstraction levels & solution methods

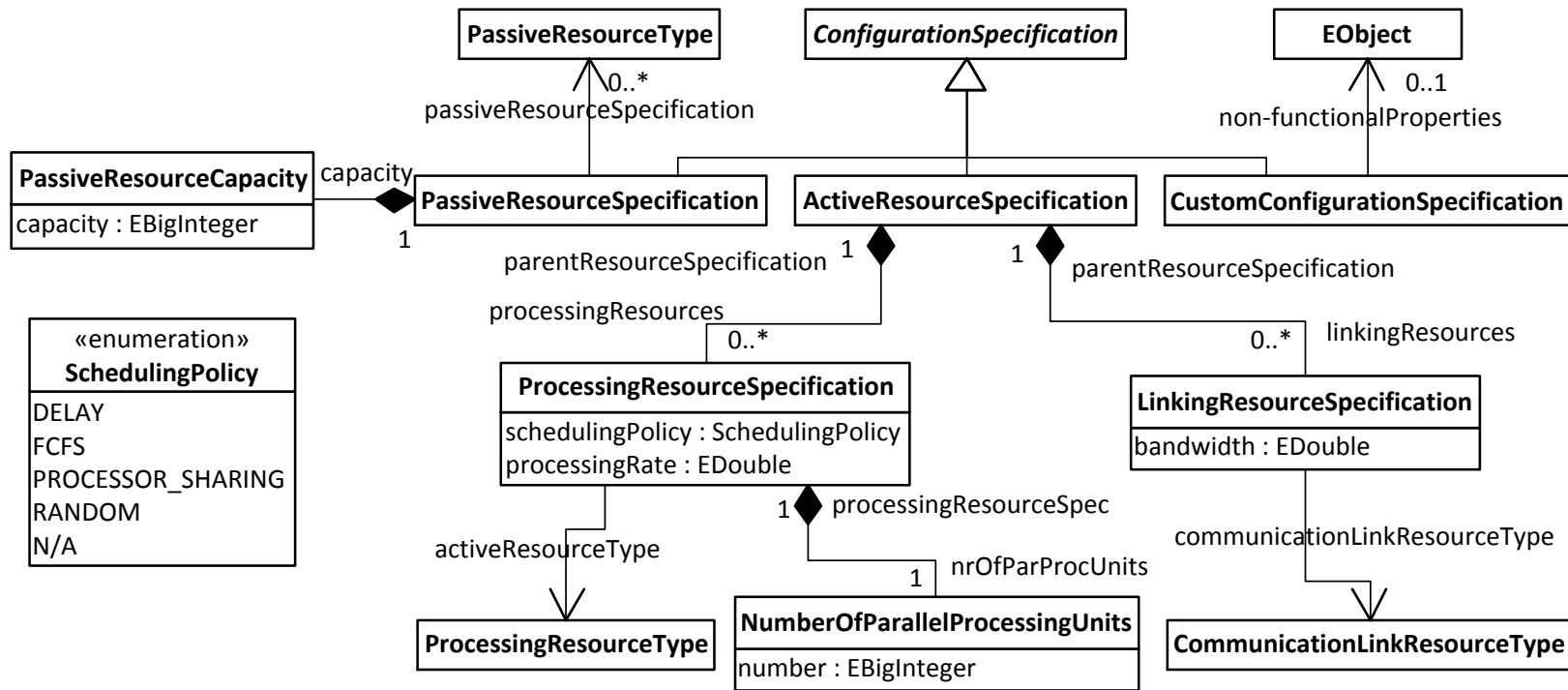
Models for Virtualized Data Centers

- Resource Landscape (static view)
 - Computing resources (physical, logical)
 - Layers of resources (Hardware, Virtualization, ...)
- Variability Points (dynamic view)
 - Configuration (#cores, #appServer, ...)
 - Landscape (Deployment, Users, ...)
- Ongoing Work: Reconfiguration Language
 - Description of the system reconfiguration
 - Example: SystemConfigA → SystemConfigB

Resource Landscape: Layering

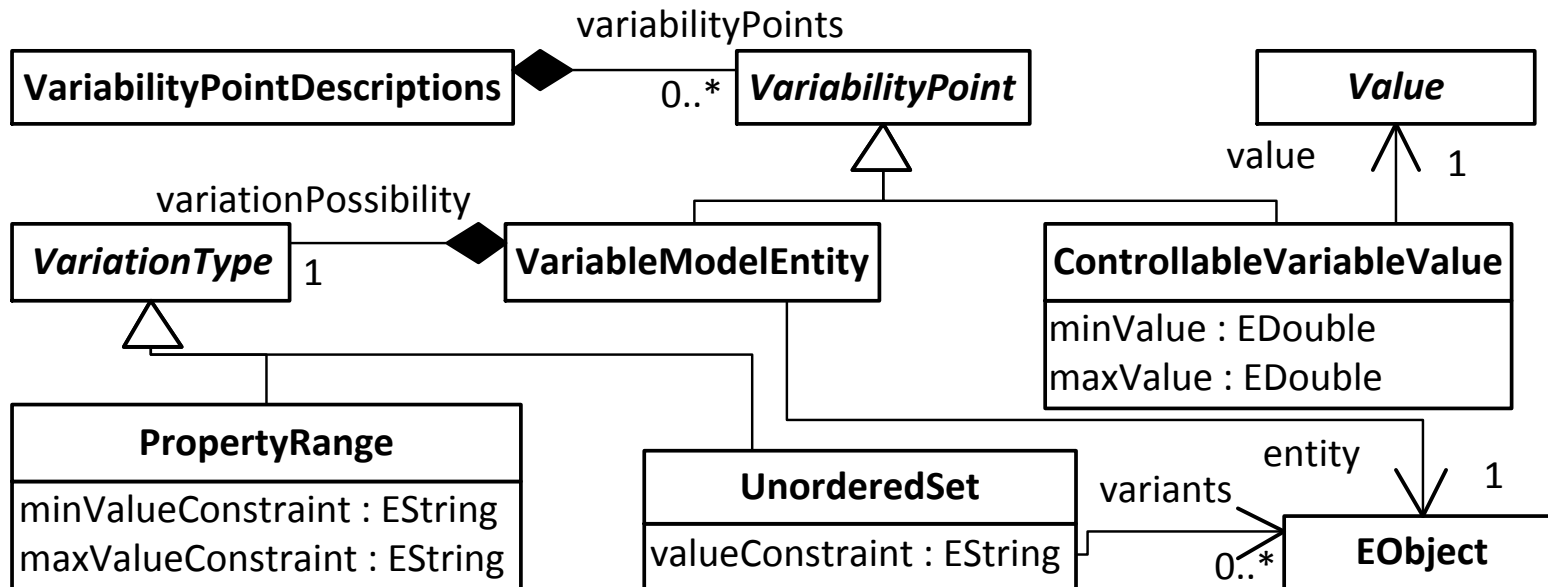


Resource Landscape: Config.Spec.



Variability Points

- “Decorator” model
- Dynamic view of the resource landscape



Modeling the Application Level

- Service Behavior Abstractions for Different Levels of Granularity
- Deployment-Specific Resource Demands / Response Times
- Probabilistic Parameter Dependencies

Service Behavior Abstractions

■ BlackBoxBehavior

- No information about resources, resource demands, control flow, call frequencies,...

■ CoarseGrainedBehavior

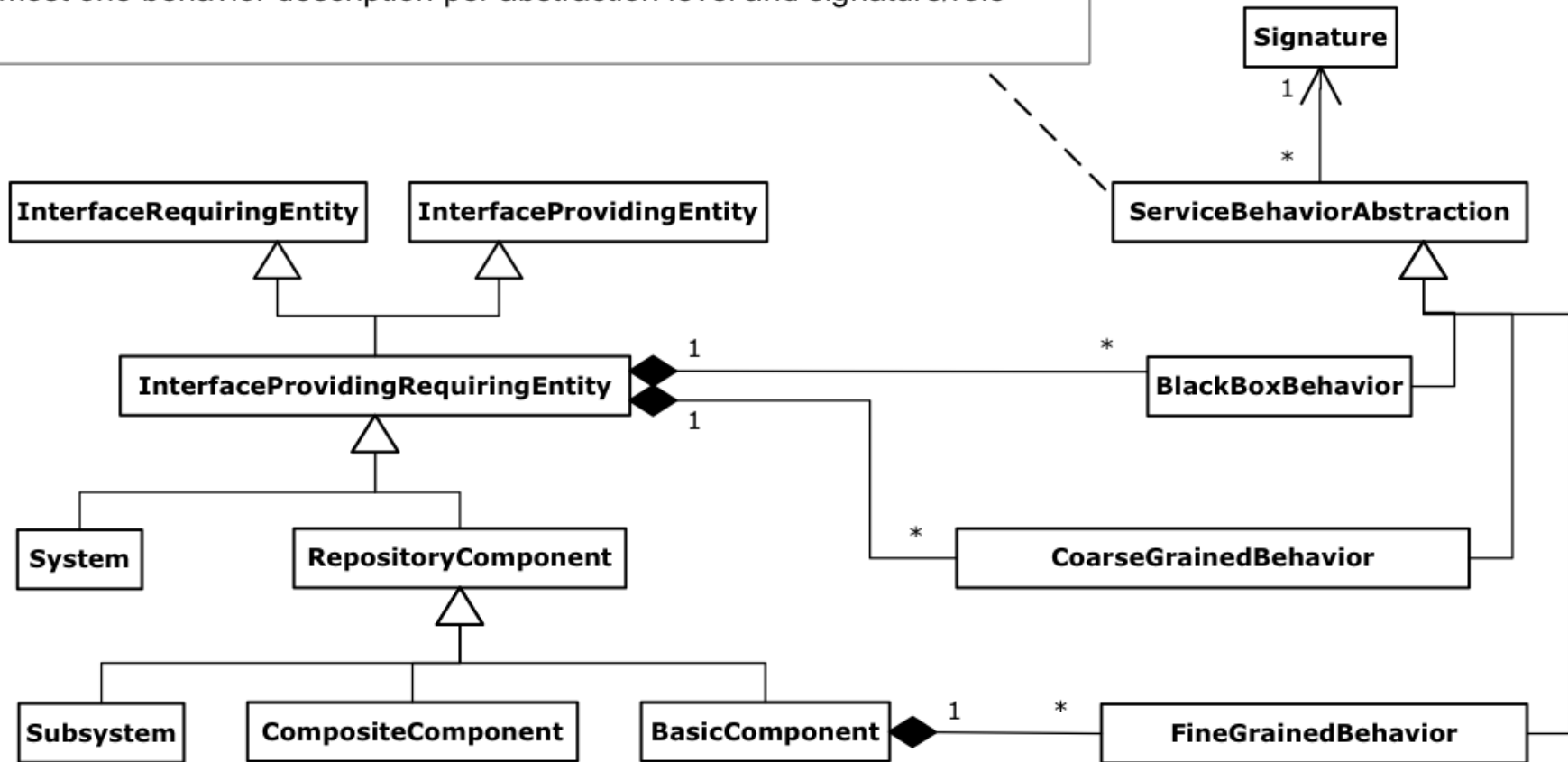
- Information at component boundary level (external services, resource consumption,...)

■ FineGrainedBehavior

- Information about component-internals (control flow, resource demands, parametric dependencies,...)

Service Behavior Abstractions

At most one behavior description per abstraction level and signature/role



Probabilistic Parameter Dependencies

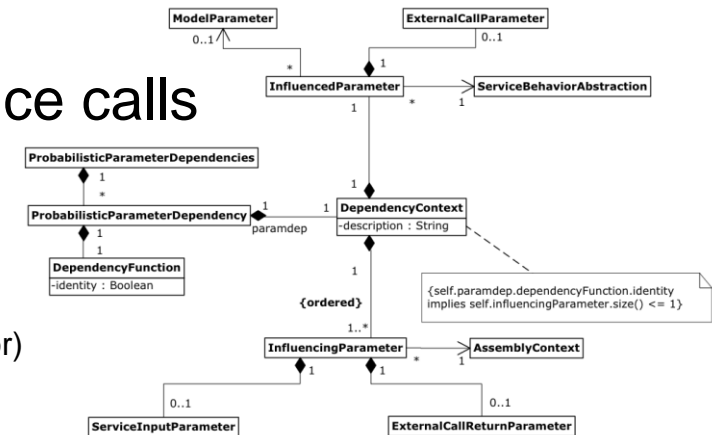
- Characterize dependencies statistically

- Influencing parameters

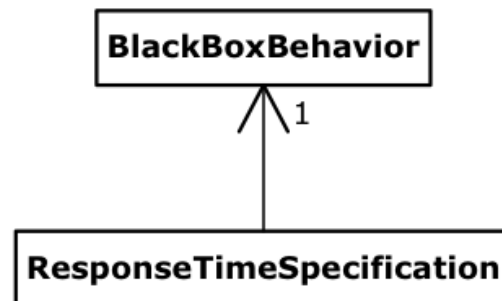
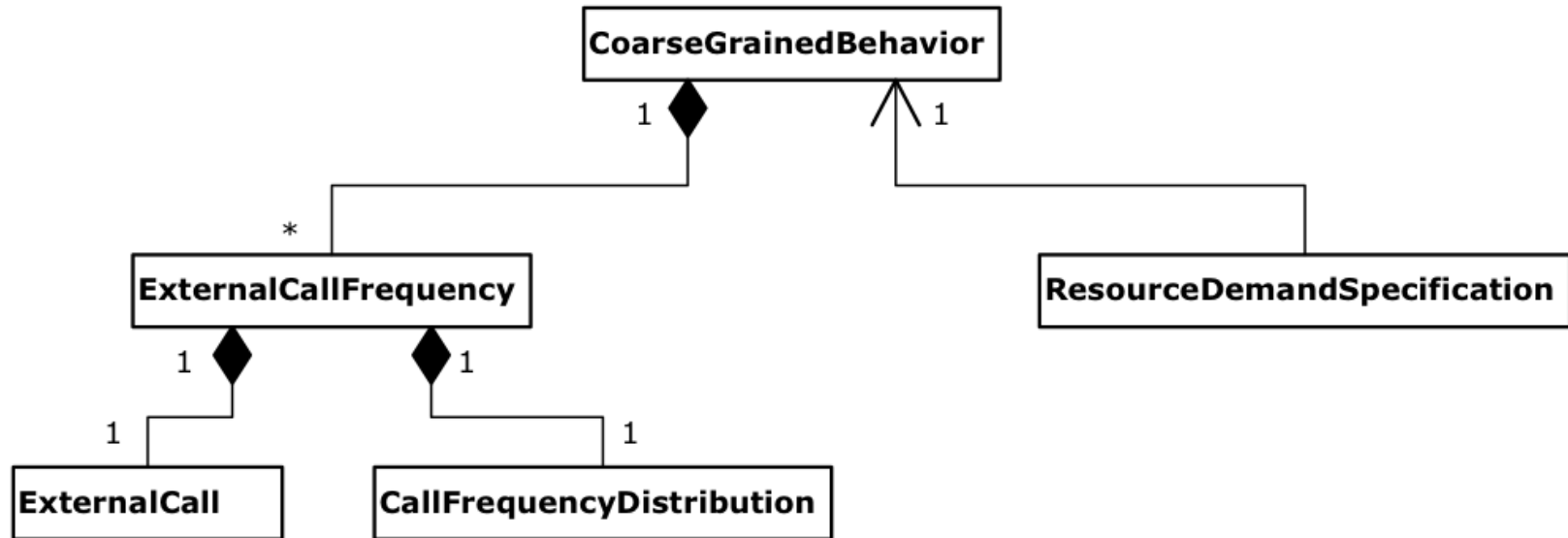
- Service input parameters
- Return parameters of external service calls

- Influenced quantities

- Loop iteration numbers (FineGrainedBehavior)
- Branch probabilities (FineGrainedBehavior)
- Call frequencies (CoarseGrainedBehavior)
- Resource demands (FineGrainedBehavior, CoarseGrainedBehavior)
- Response times (BlackBoxBehavior)
- Input parameter of ext. service call (FineGrainedBehavior, CoarseGrainedBehavior)



Service Behavior Abstractions



Probabilistic Parameter Dependencies

