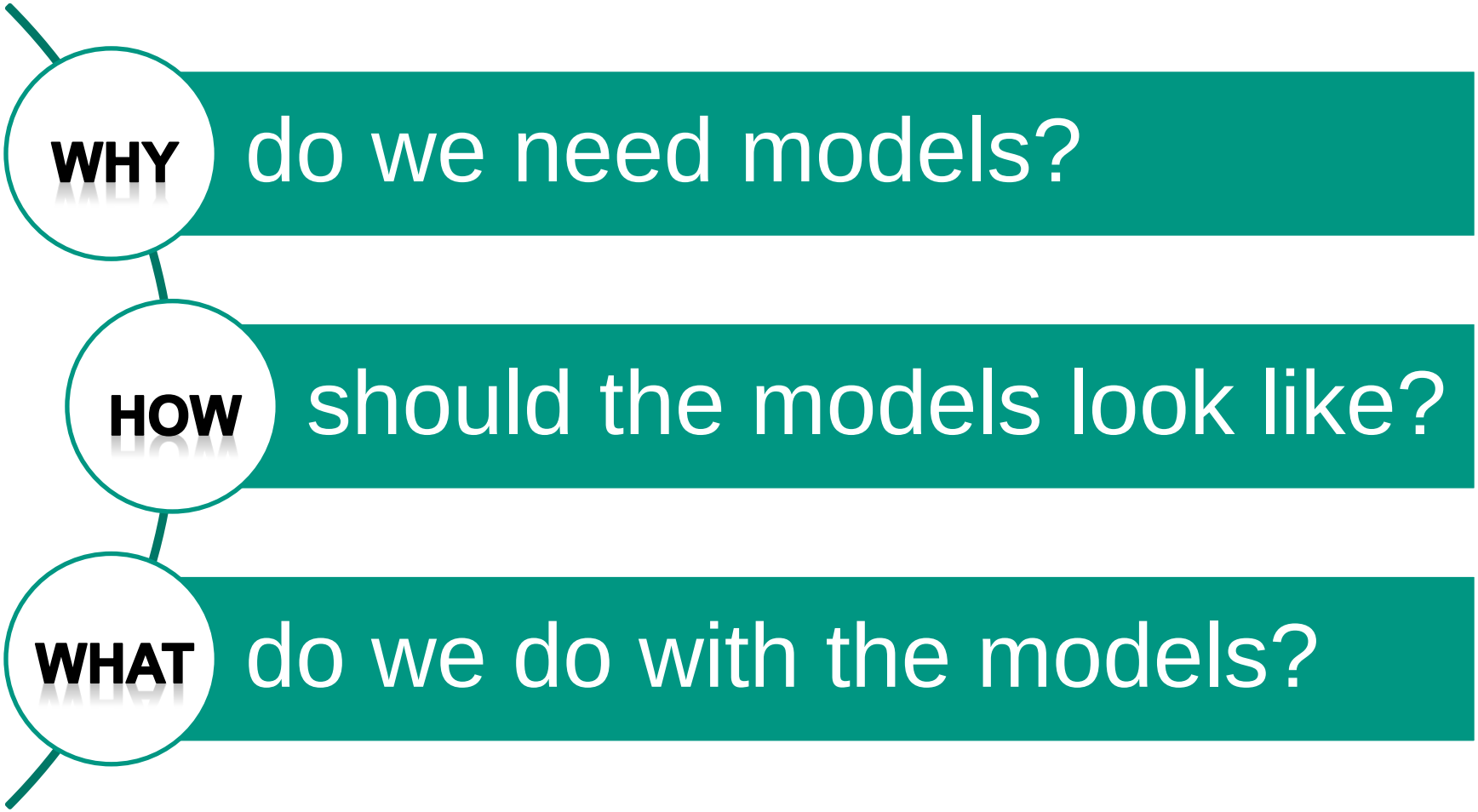
Model-based Run-Time Adaptation in Dynamic Virtualized Resource Landscapes

Open Excellence Workshop, Karlsruhe, December 4th, 2012

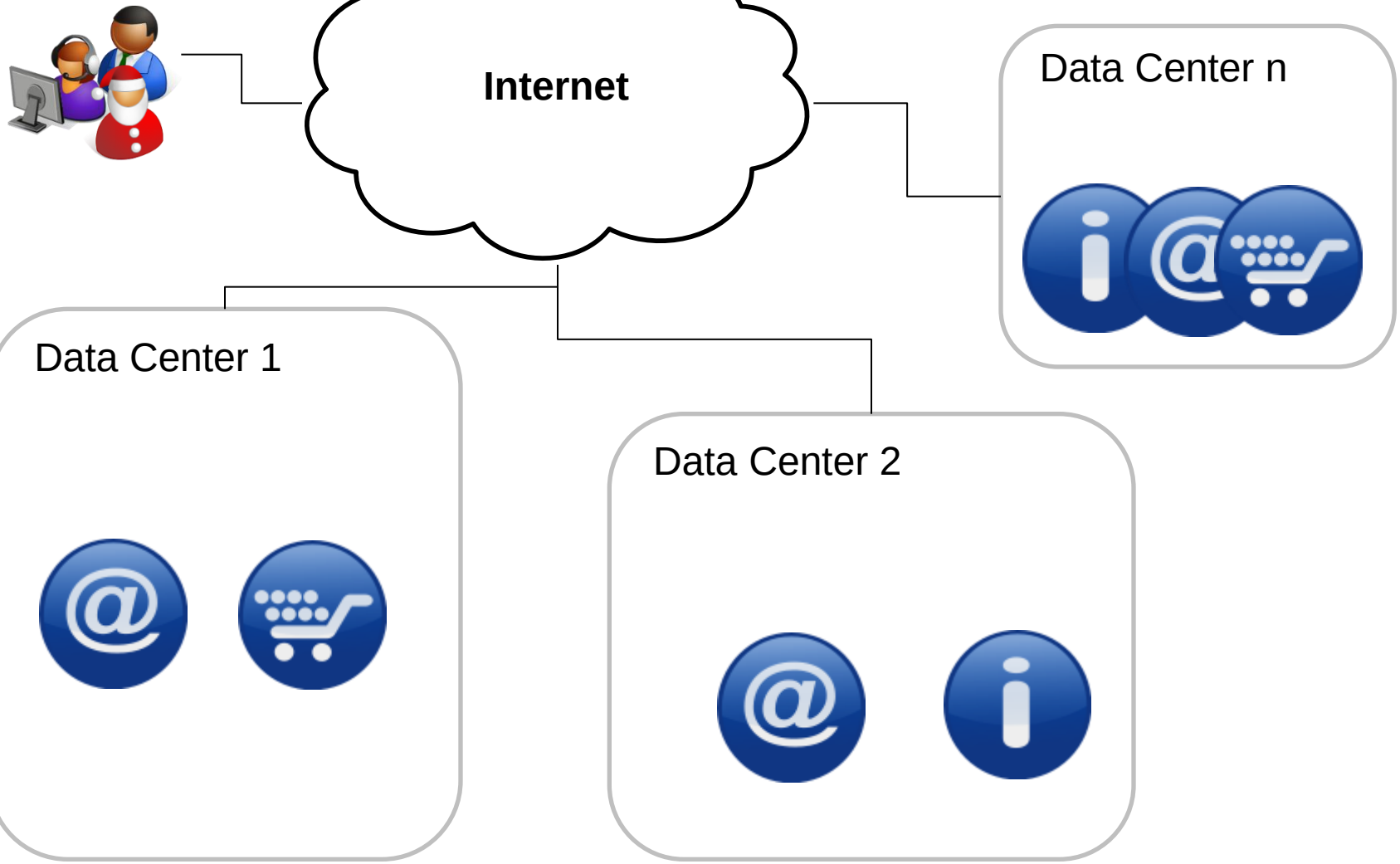
Nikolaus Huber (nikolaus.huber@kit.edu)

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

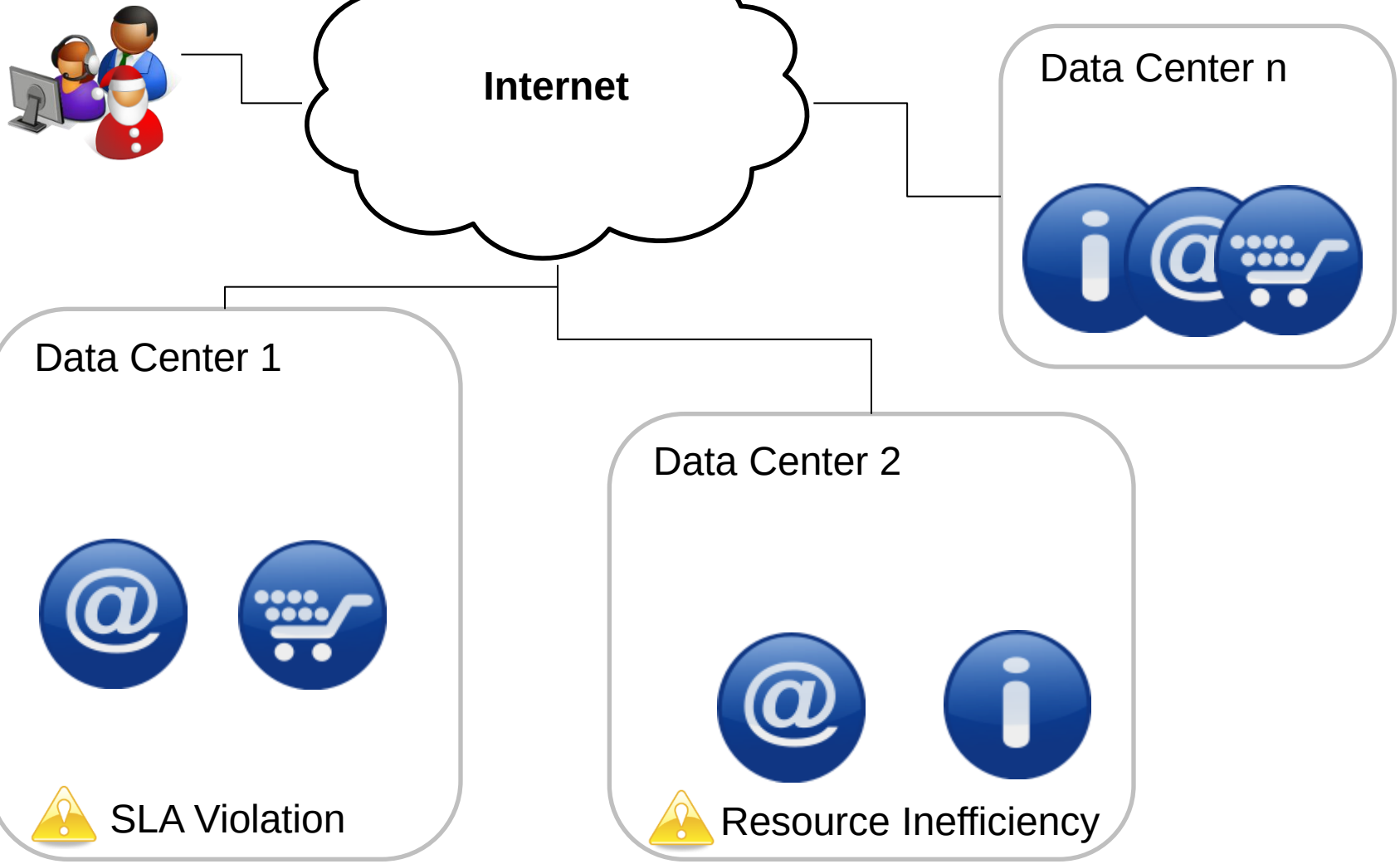
Overview


WHY
do we need models?
HOW
should the models look like?
WHAT
do we do with the models?

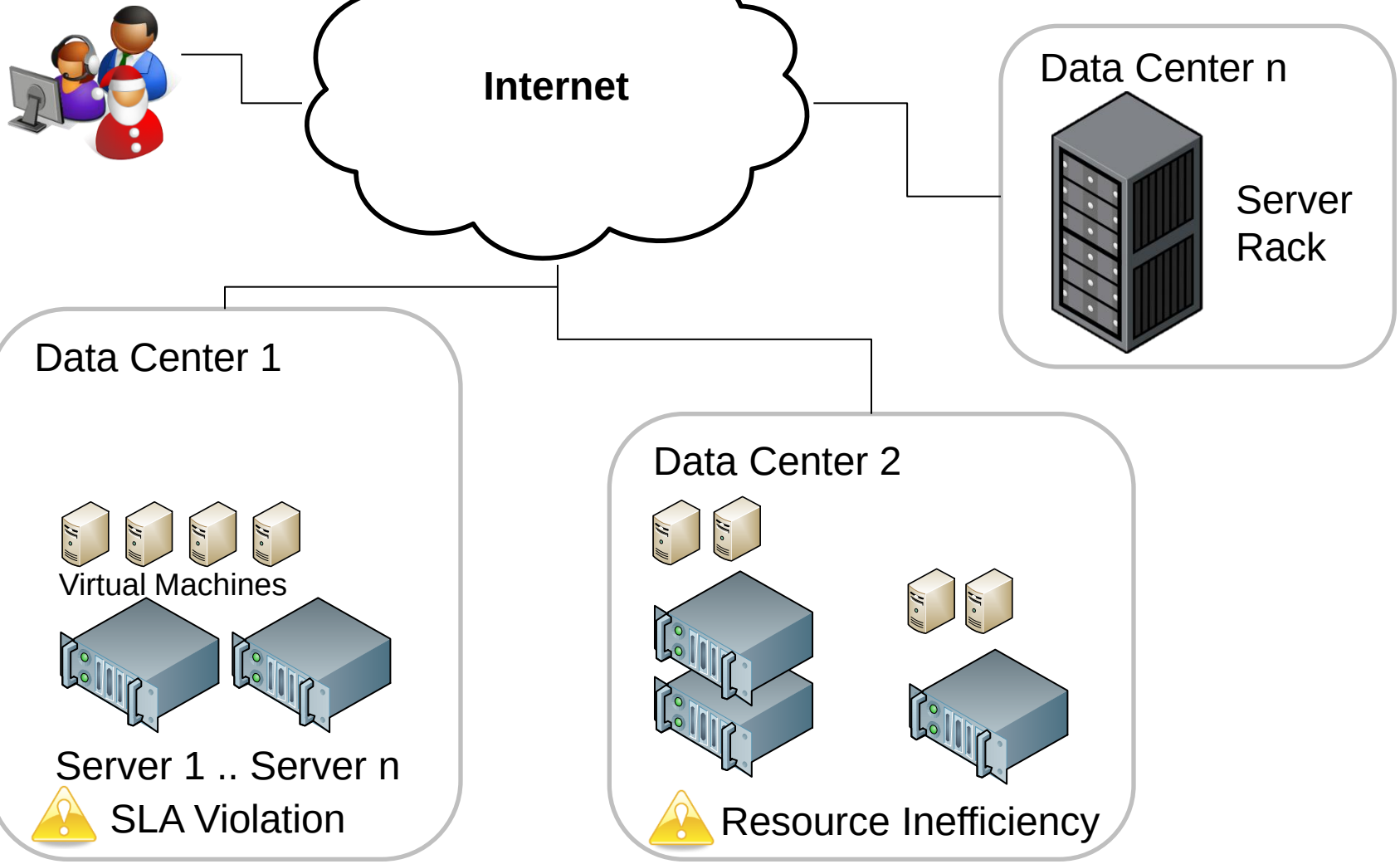
Motivation



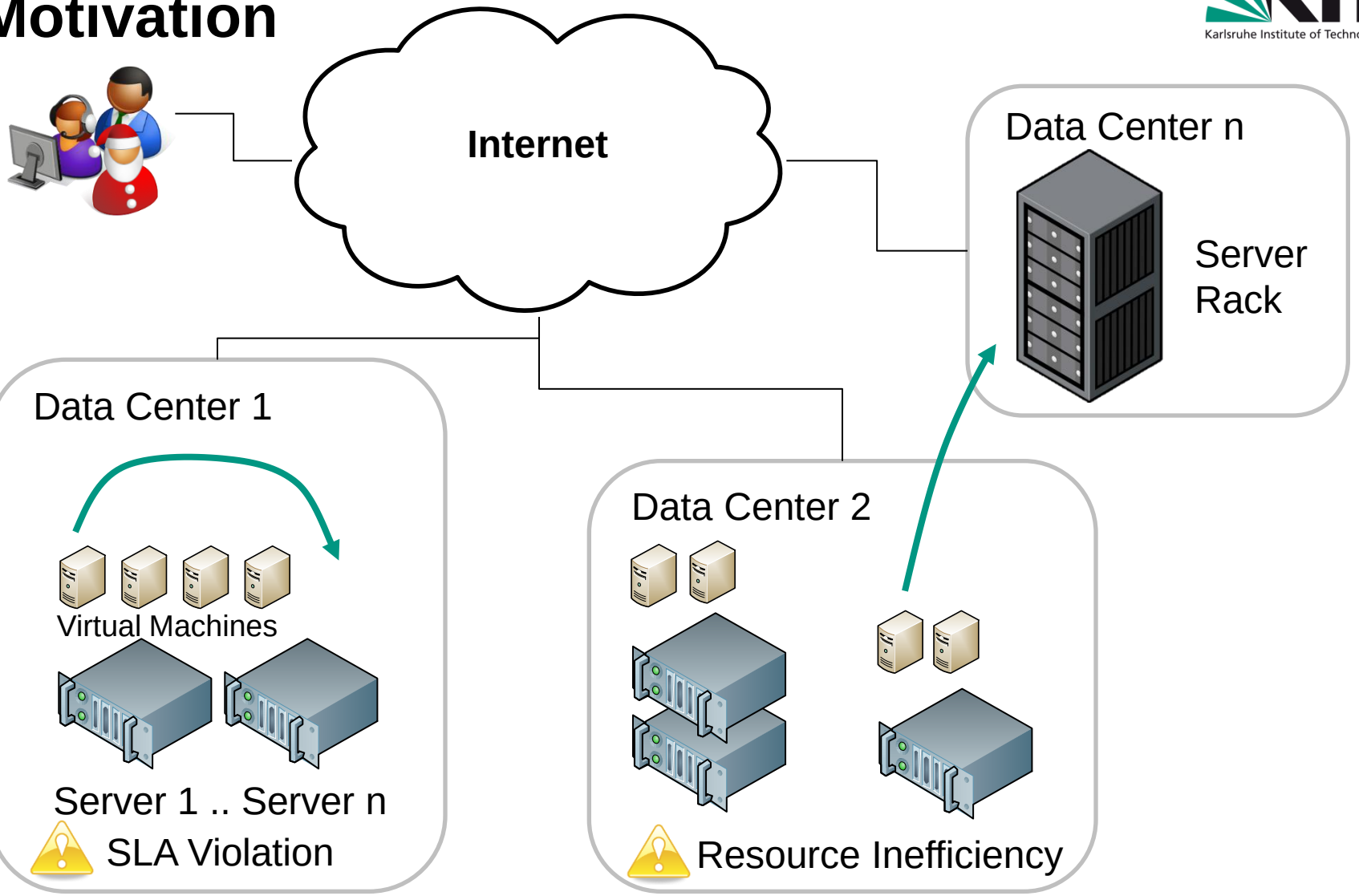
Motivation


Internet
Data Center n
Data Center 1
SLA Violation
Data Center 2
Resource Inefficiency

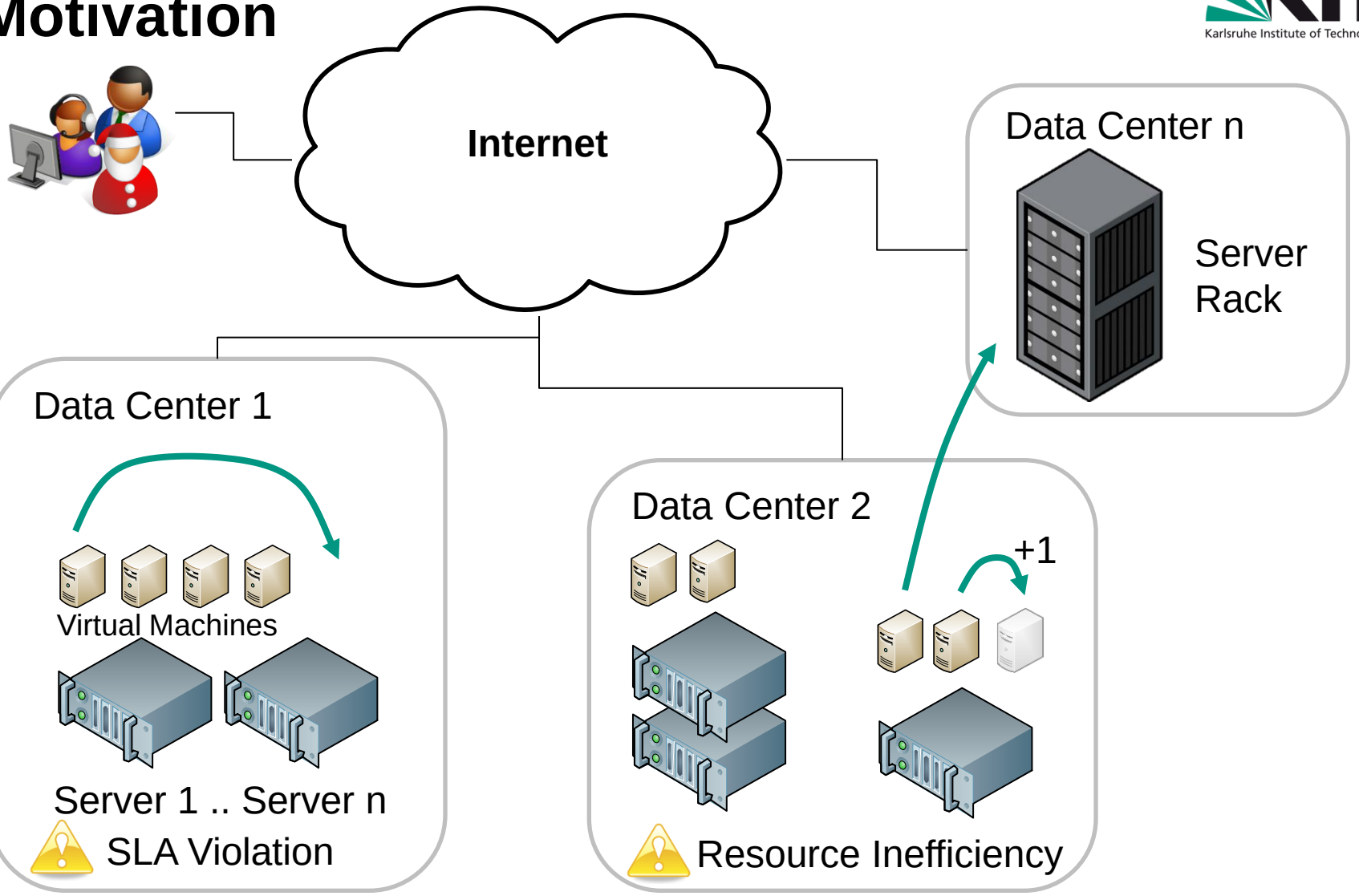
Motivation


Internet
Data Center n
Server
Rack
Data Center 1
Virtual Machines
Server 1 .. Server n
SLA Violation
Data Center 2
Resource Inefficiency

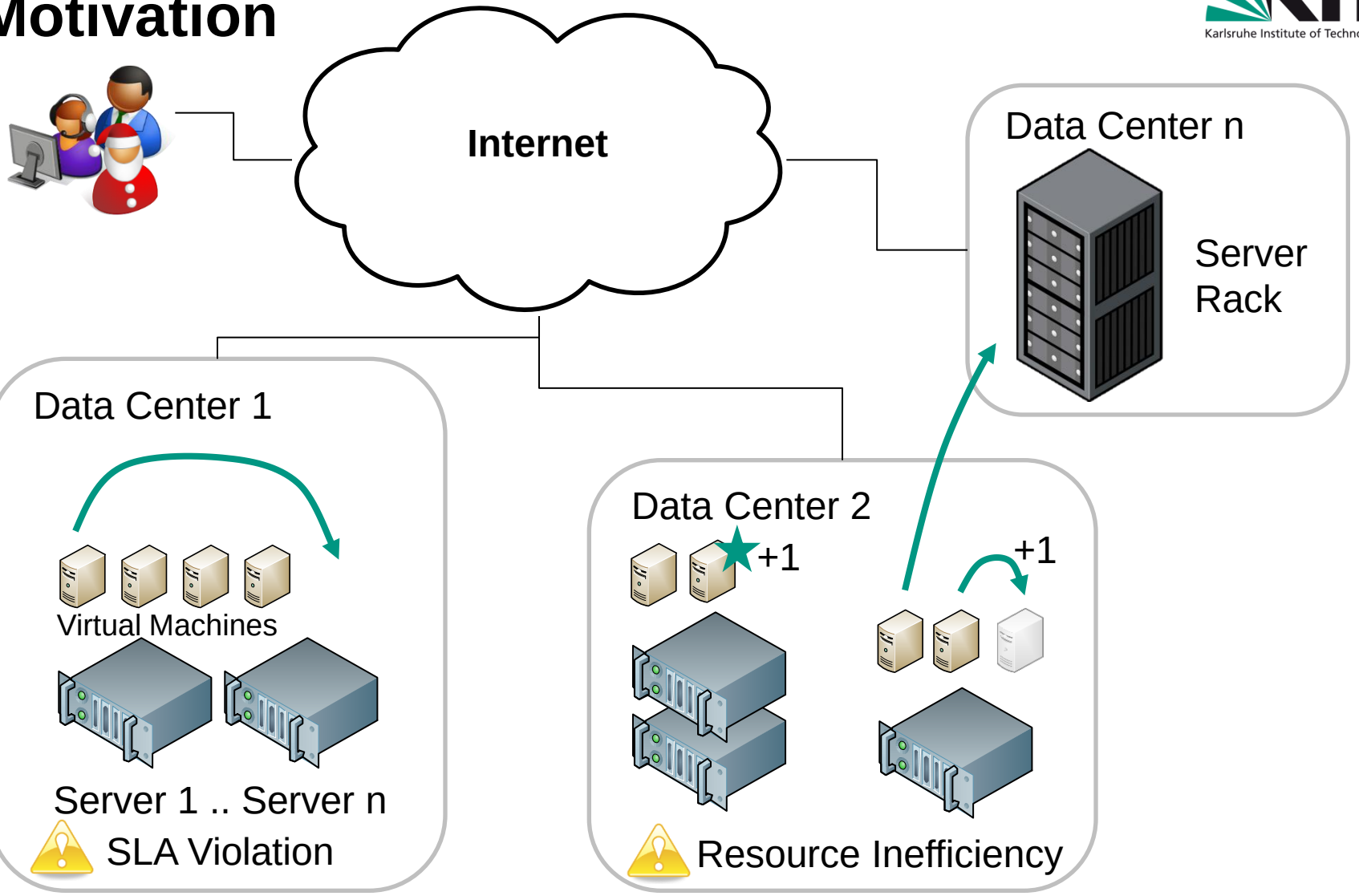
Motivation


Internet
Data Center n
Server
Rack
Data Center 1
Virtual Machines
Server 1 .. Server n
SLA Violation
Data Center 2
Resource Inefficiency

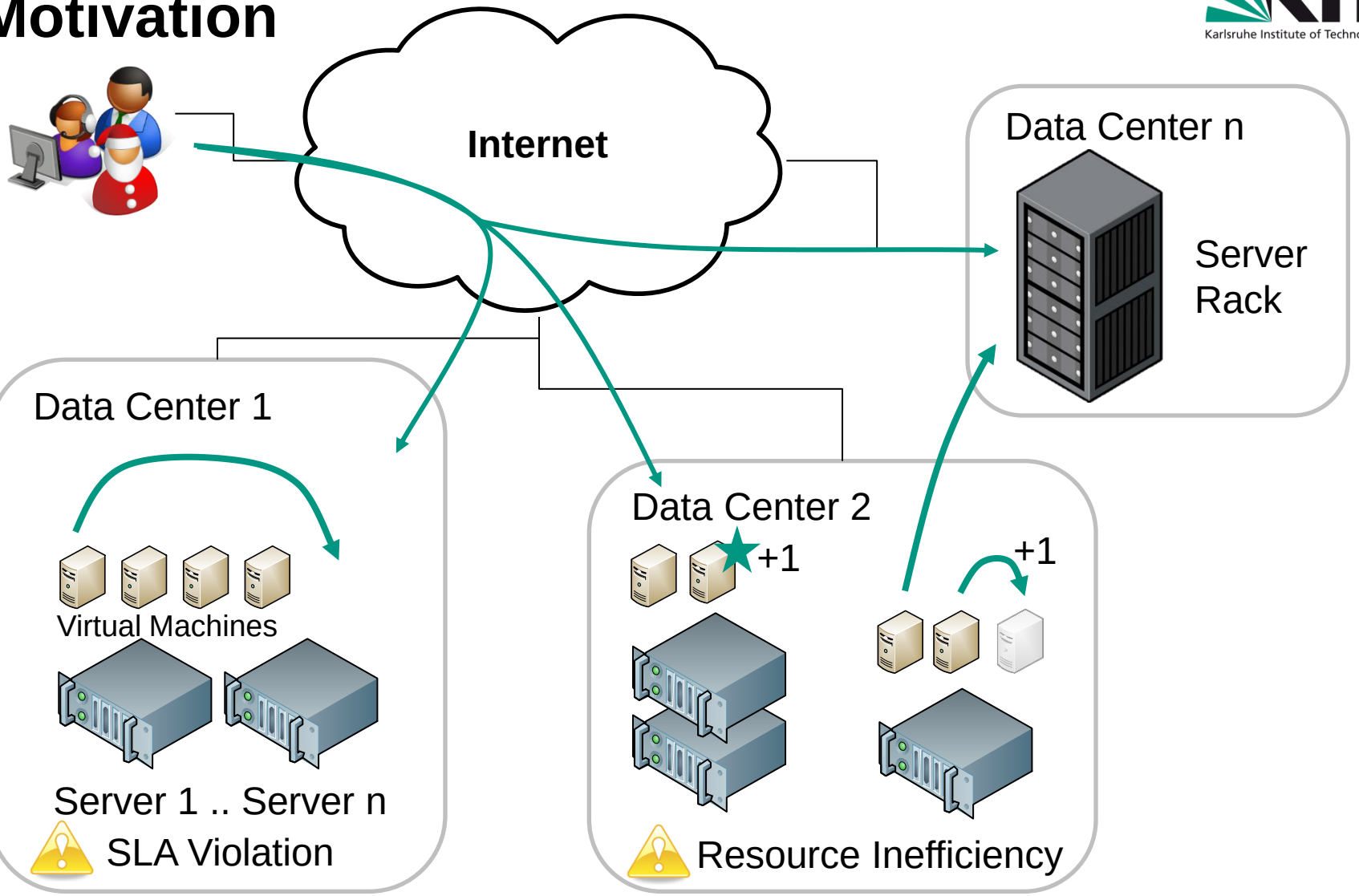
Motivation


Internet
Data Center n
Server
Rack
Data Center 1
Virtual Machines
Server 1 .. Server n
SLA Violation
Data Center 2
+1
Resource Inefficiency

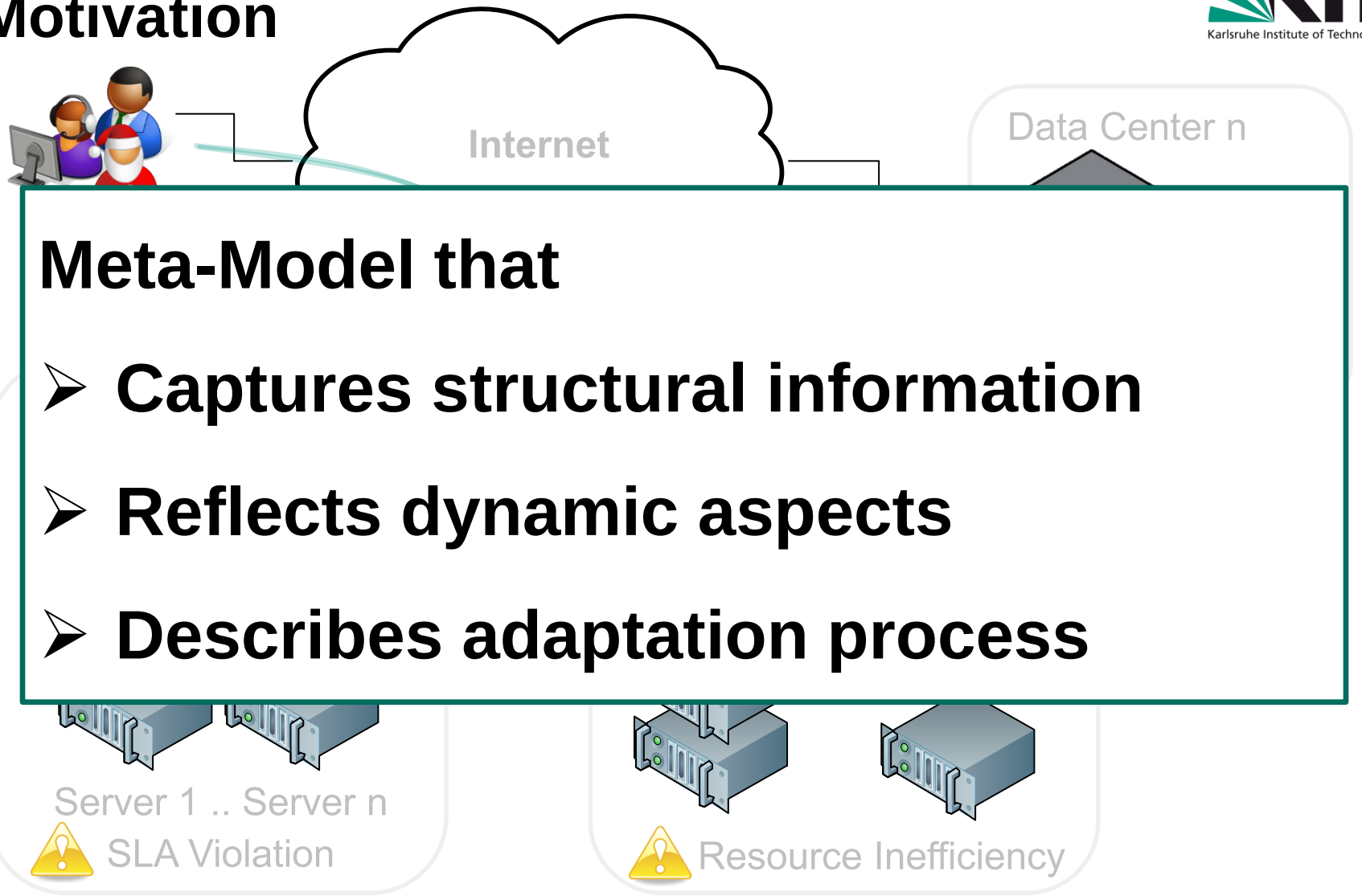
Motivation


Internet
Data Center n
Server
Rack
Data Center 1
Virtual Machines
Server 1 .. Server n
SLA Violation
Data Center 2
+1
+1
Resource Inefficiency

Motivation


Internet
Data Center n
Server
Rack
Data Center 1
Virtual Machines
Server 1 .. Server n
SLA Violation
Data Center 2
+1
+1
Resource Inefficiency

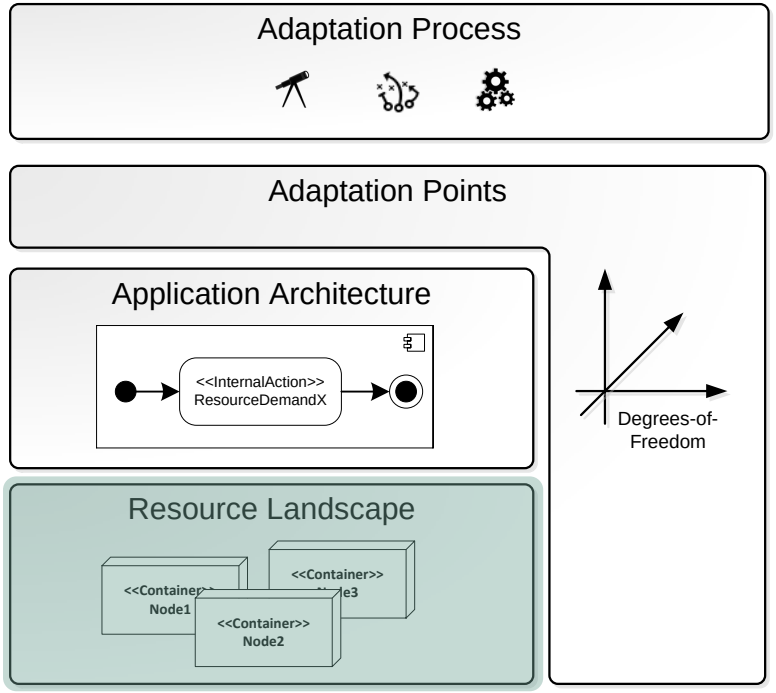
Motivation



Meta-Model that

- **Captures structural information**
- **Reflects dynamic aspects**
- **Describes adaptation process**

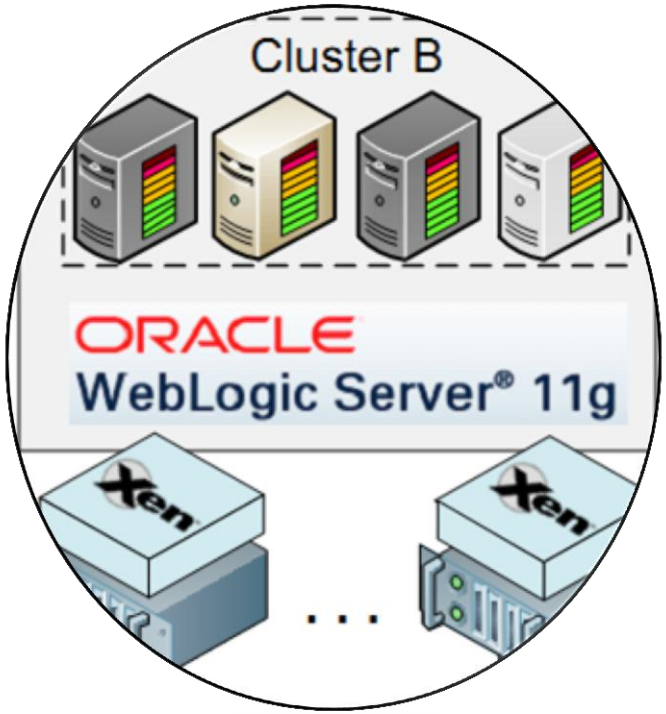
DMM: Resource Landscape

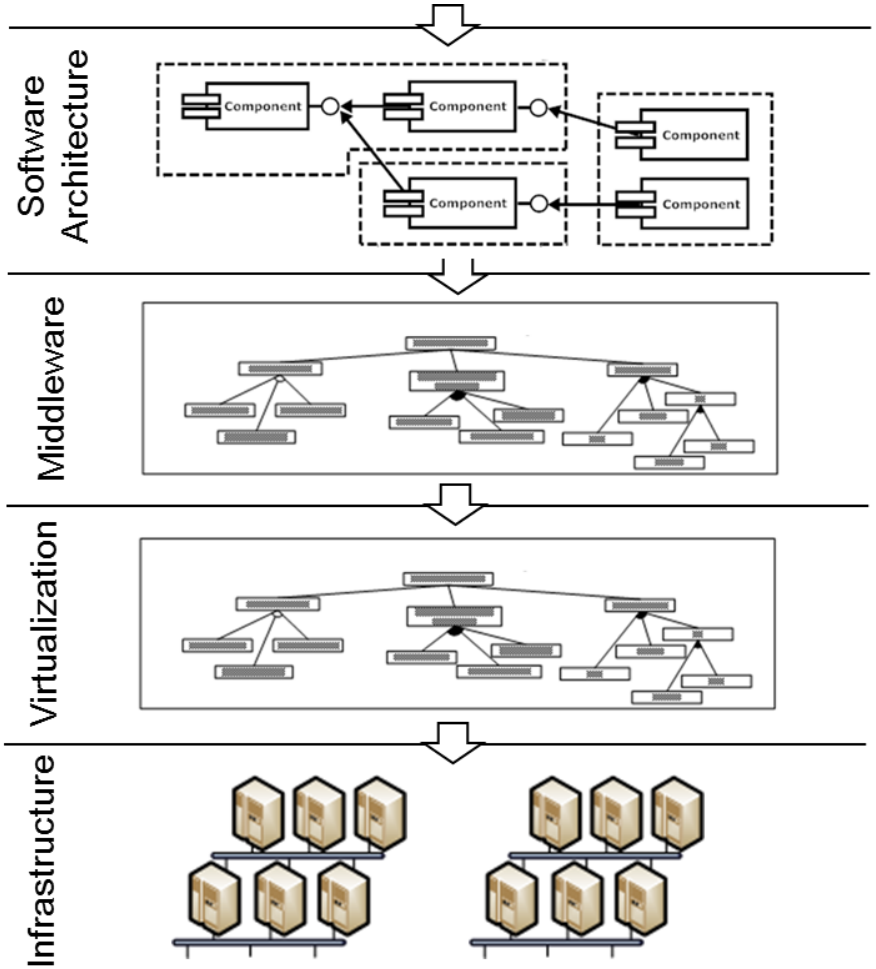


Further details in:

- N. Huber, F. Brosig and S. Kounev. **Modeling Dynamic Virtualized Resource Landscapes**. In Proceedings of the 8th ACM SIGSOFT International Conference on the Quality of Software Architectures (QoSA 2012), Bertinoro, Italy, June 25-28, 2012.

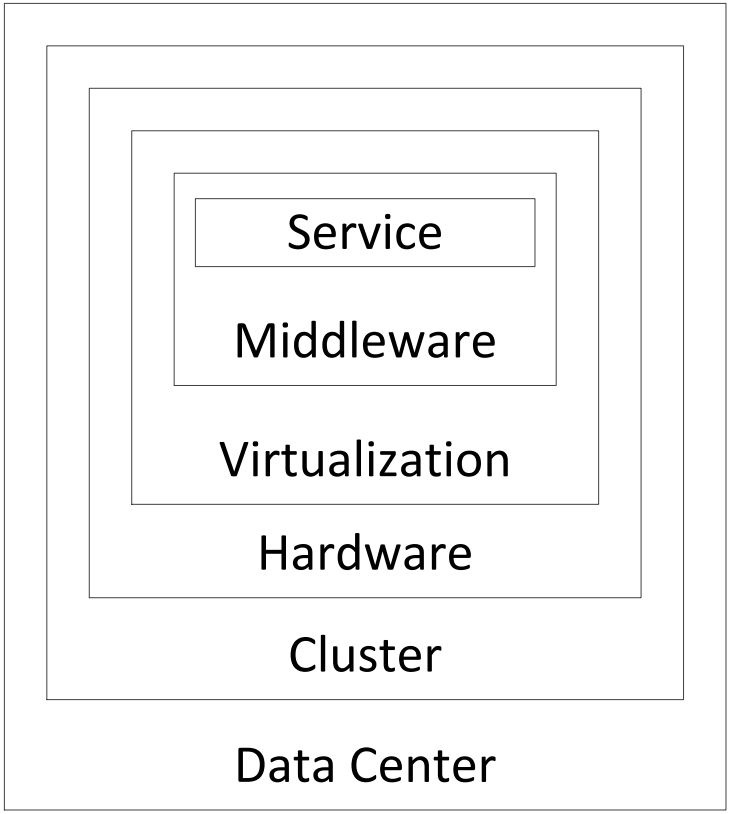
Resource Lanscape: Example



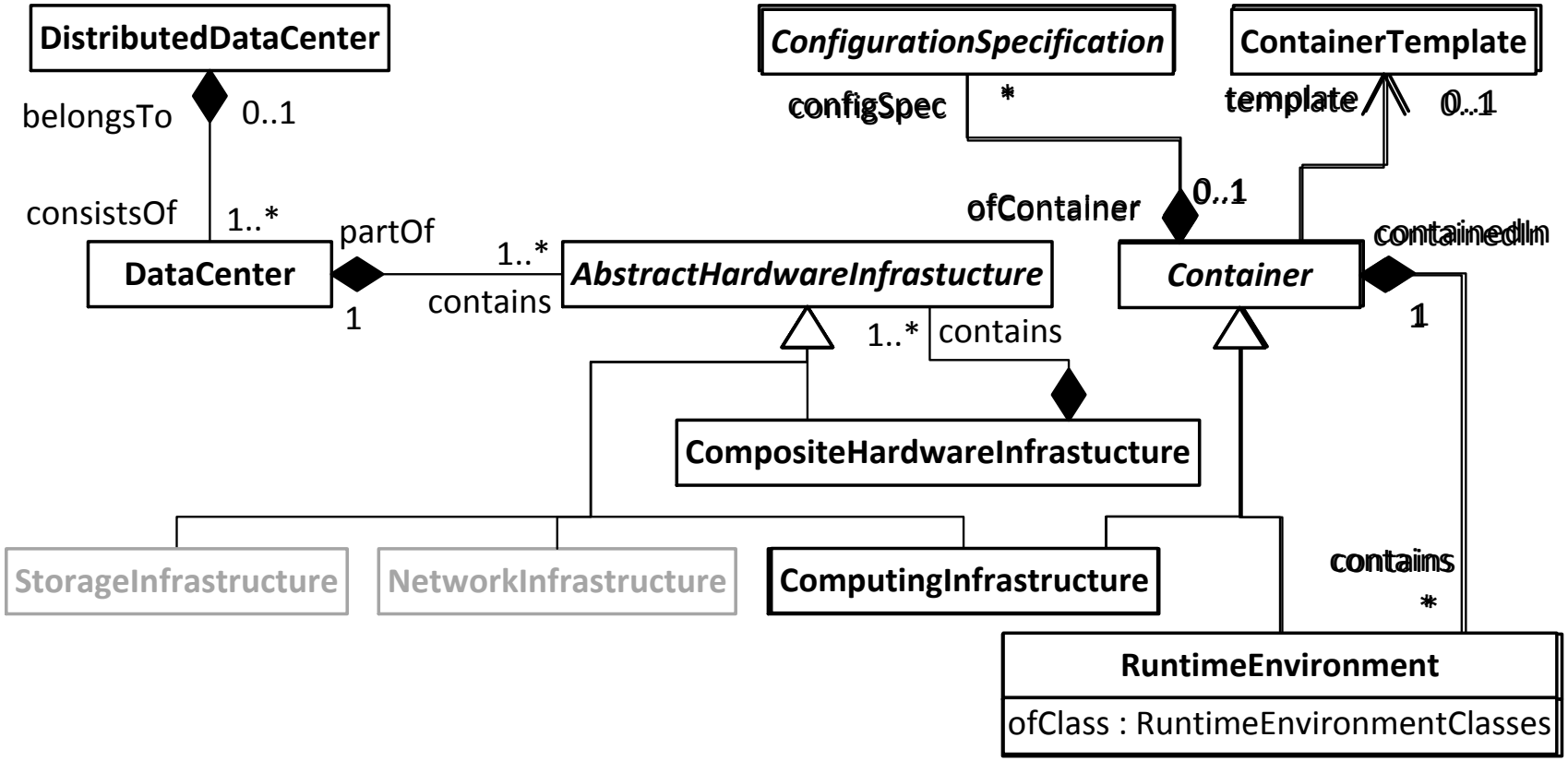


Resource Layers & Containers

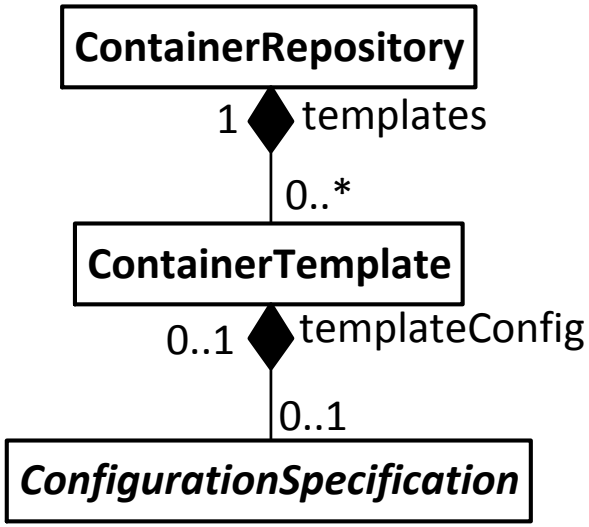
- Common concepts in data centers
  - Containers/Layering (physical and logical)
  - Abstraction of resources
  - Resource sharing
- Advantage:
  - Flexibility
  - Structural information



Resource Layers & Containers



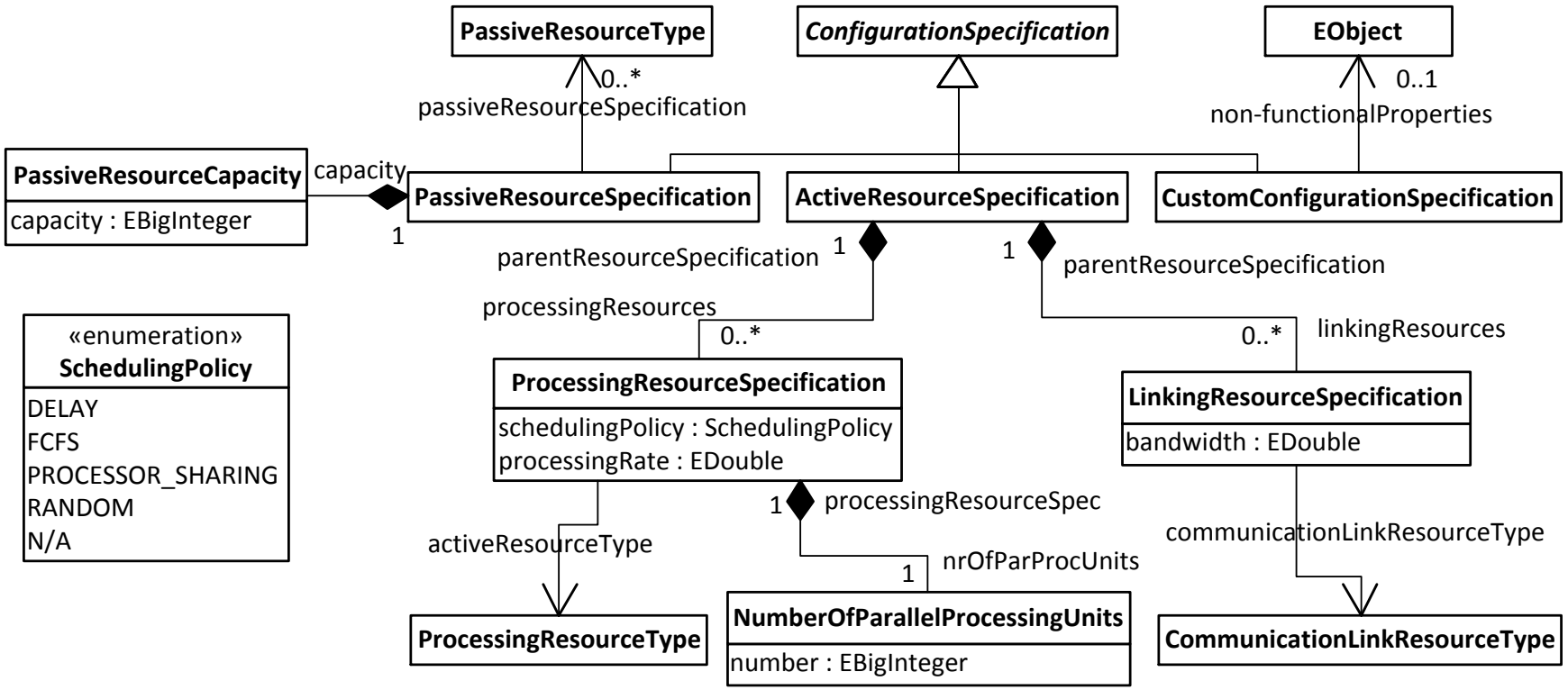
Resource Landscape: Container Templates and Runtime Environment Classes



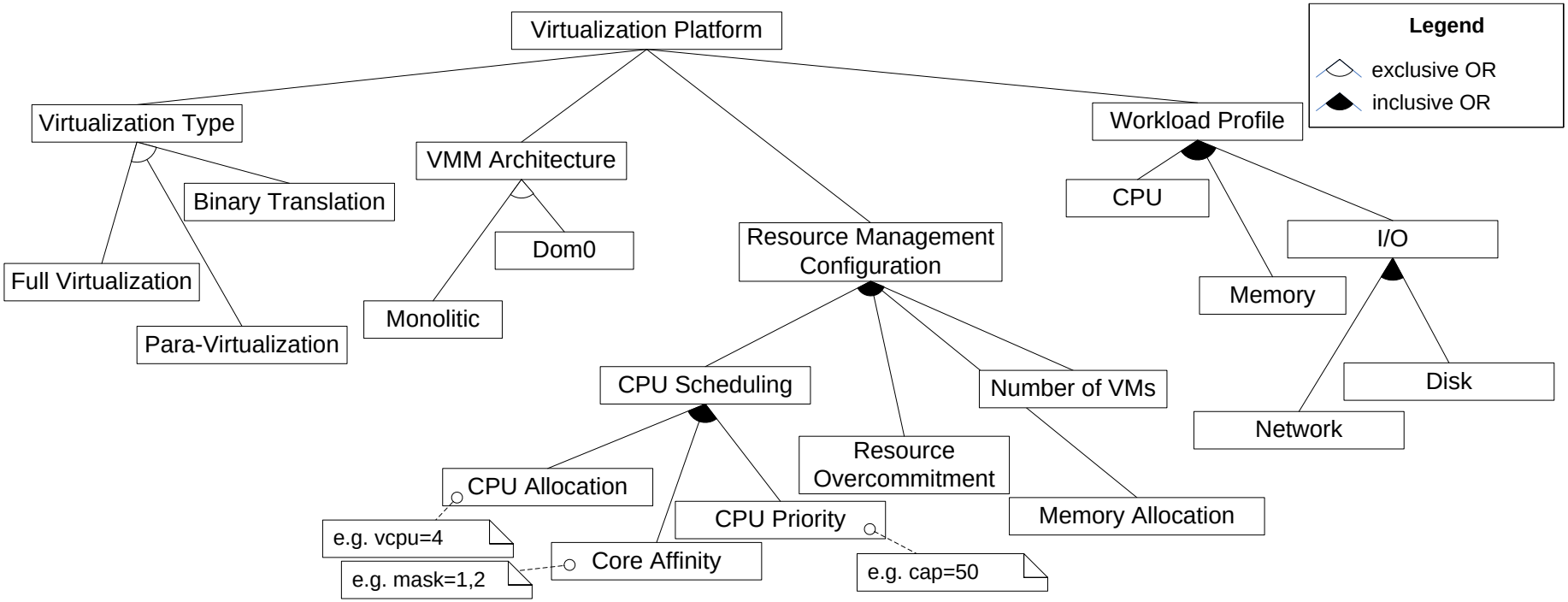
«enumeration»
RuntimeEnvironmentClasses

- HYPERVISOR
- OS
- OS_VM
- PROCESS_VM
- MIDDLEWARE
- OTHER

Resource Landscape: Configuration Specification



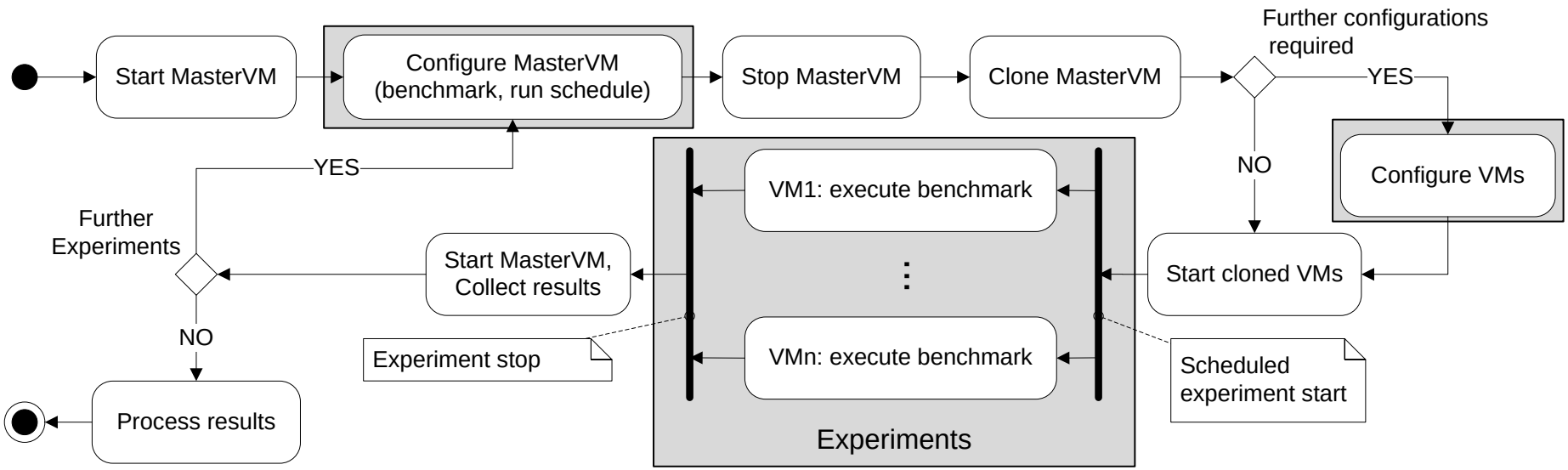
Modeling Virtualization Platforms



Further details in:

- N. Huber, M. Quast, M. Hauck, and S. Kounev. **Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments**. In *Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011), Noordwijkerhout, The Netherlands*, May 7-9 2011. Best Paper Award.

Automated Experimental Analysis



Further details in:

- N. Huber, M. von Quast, F. Brosig and S. Kounev. **Analysis of the Performance-Influencing Factors of Virtualization Platforms**. In 12th International Symposium on Distributed Objects, Middleware, and Applications (DOA 2010), Crete, Greece, October 2010. Springer Verlag.

Experiment Setup

- Virtualization Platforms
  - Citrix XenServer 5.5
  - VMware ESX 4.0

- Experimental environment
  - SunFire X4440 Server, AMD Opteron 24*2.4 GHz, 128 GB DDR2 RAM

- Different benchmark types
  - Passmark PerformanceTest v7 (CPU, Memory, HDD)
  - SPECcpu (CPU + Memory)
  - Iperf (Network)

Virtualization Overhead

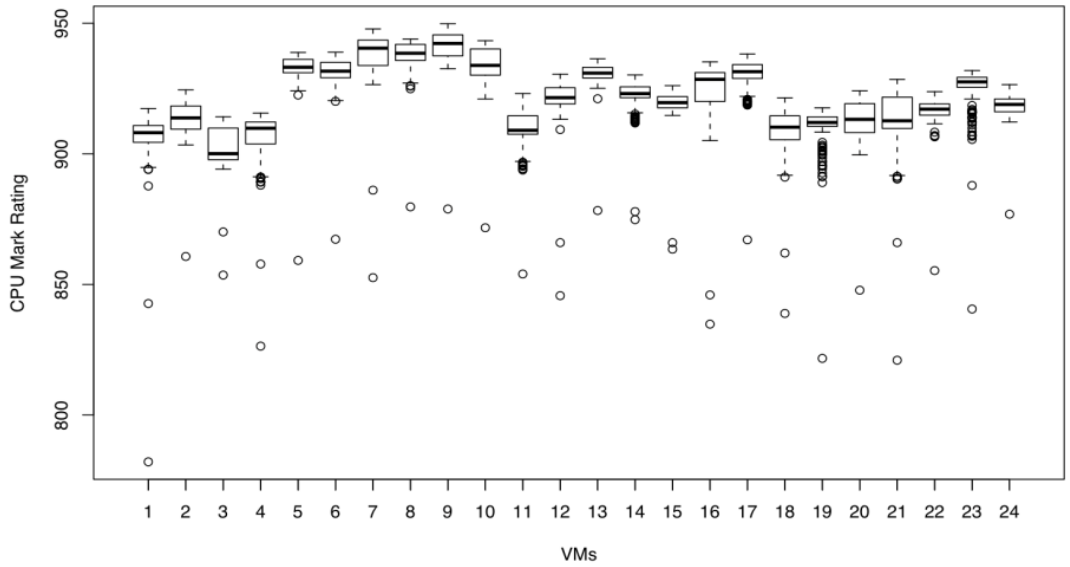
XenServer 5.5
Throughput metric: higher values are better

Benchmark	native	virtualized	Delta (abs.)	Delta (rel.)
Passmark CPU, 1 core	639.3	634.0	5.3	0,83%
Passmark CPU, 2 cores	1232.0	1223.0	9.0	0.97%
SPECint(R)_base2006				3.61%
SPECfb(R)_base2006				3.15%
Passmark Memory, 1 core	492.9	297.0	195.9	39.74%
Passmark Memory, 2 cores	501.7	317.5	184.2	36.72%
Iperf (send)	527.0	393.0	134.0	25,43%
Iperf (receive)	528.0	370.0	158.0	29,92%

Scalability

- Scaling CPU resource
- Performance impact of affinity

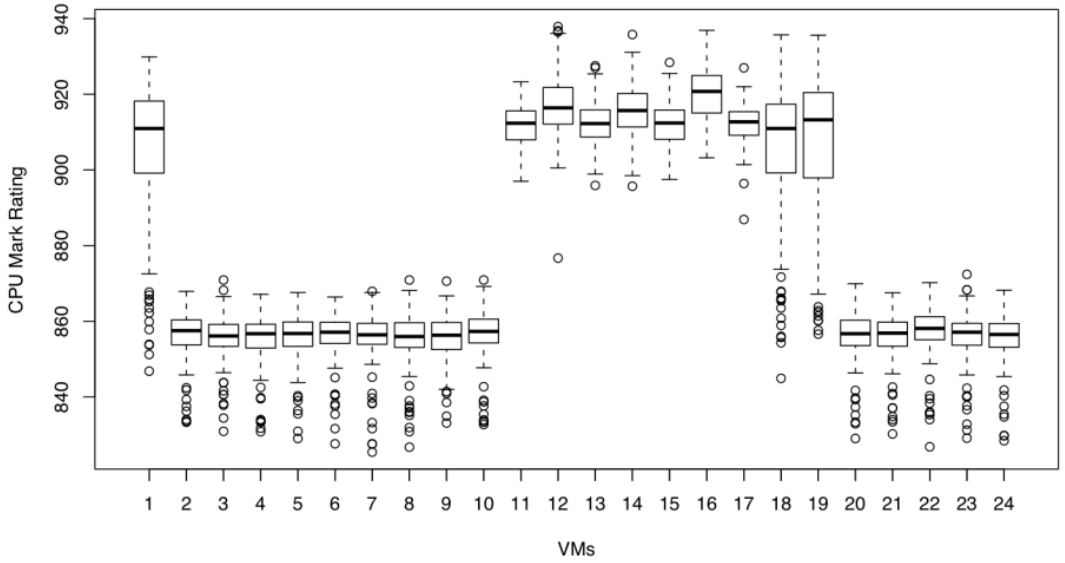
Affinity OFF



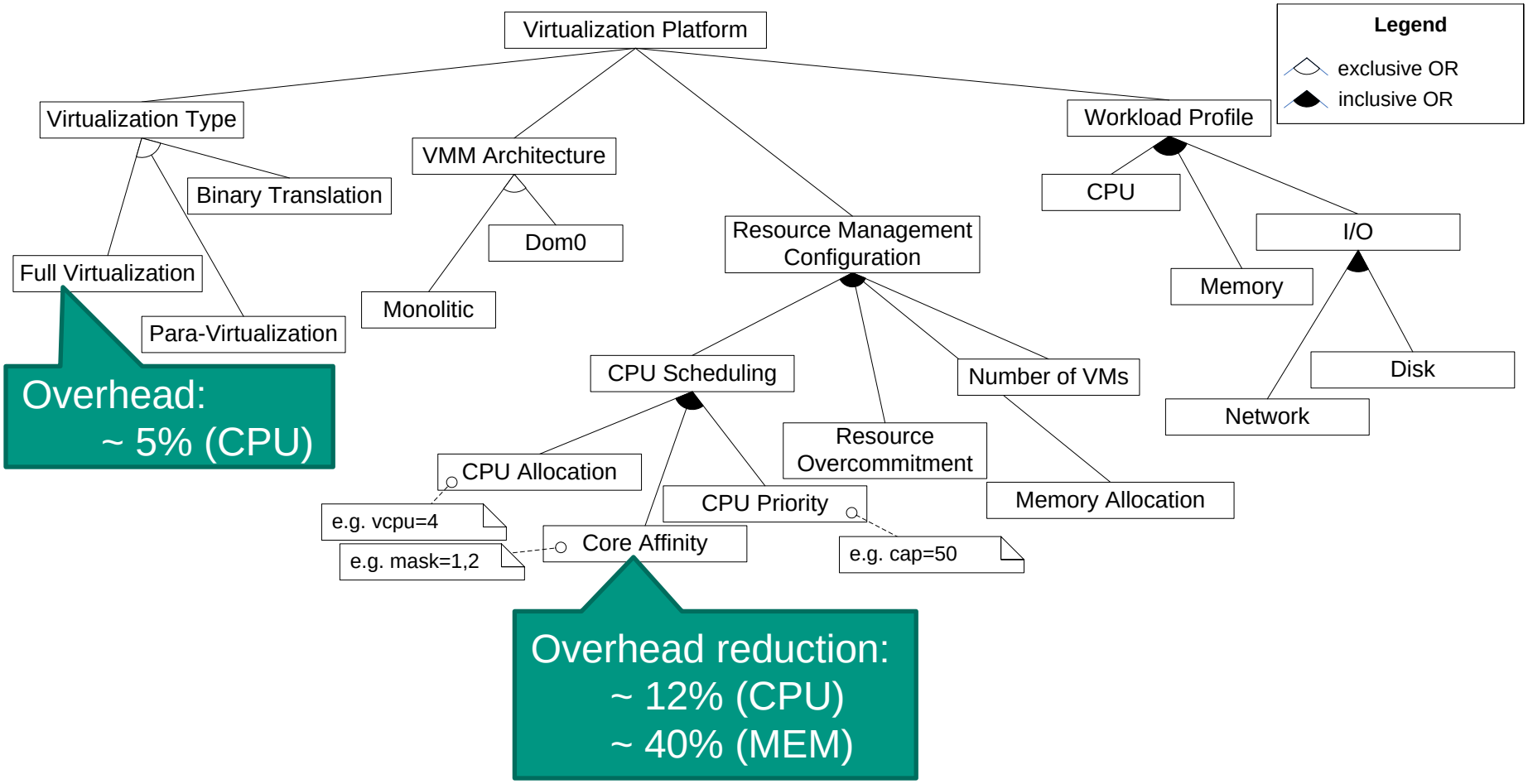
Scalability

- Scaling CPU resource
- Performance impact of affinity

Affinity ON

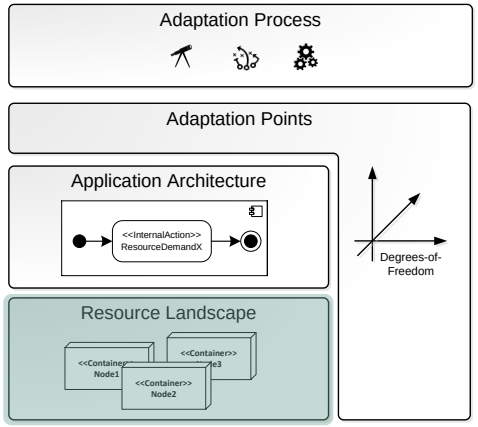


Modeling Virtualization Platforms - II

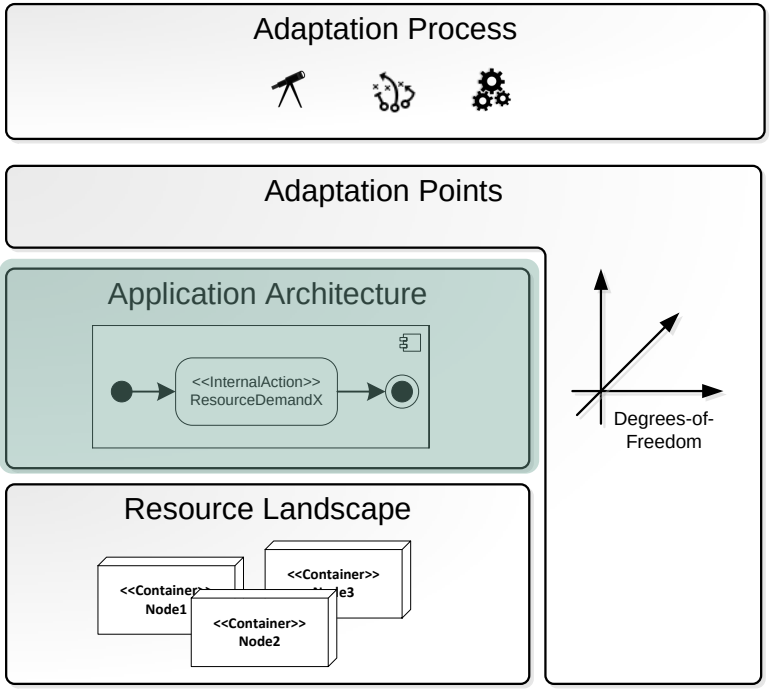


Resource Landscape - Summary

- Specification of containers
- Specification of resources
- Performance influence(s) of layers
  - Virtualization
  - Middleware
  - …



DMM: Application Architecture

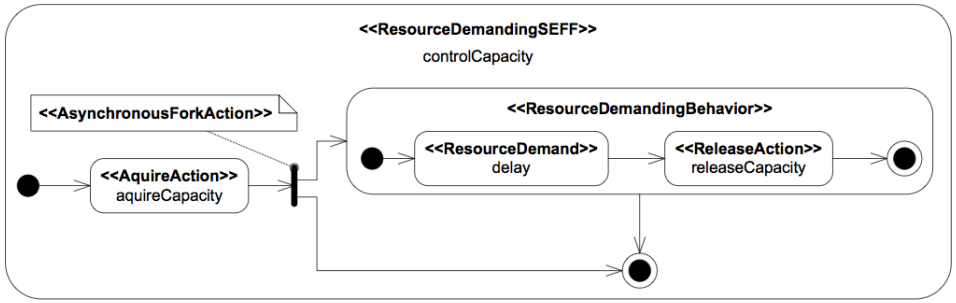


Further details in:

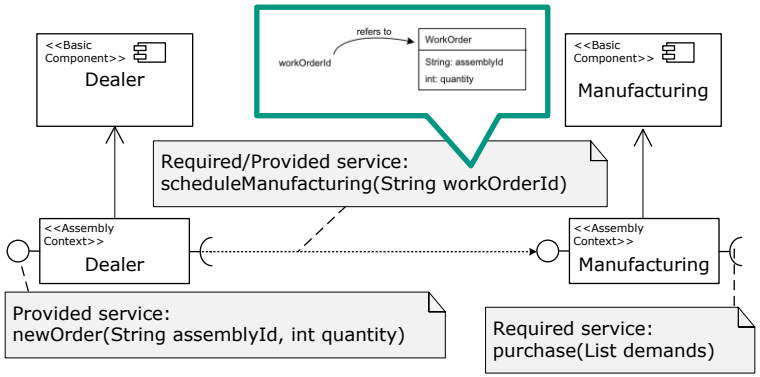
- F. Brosig, N. Huber, and S. Kounev. **Modeling Parameter and Context Dependencies in Online Architecture-Level Performance Models**. In *Proceedings of the 15th ACM SIGSOFT International Symposium on Component Based Software Engineering (CBSE 2012), Bertinoro, Italy*, June 26-28, 2012.

Modeling the Application Level

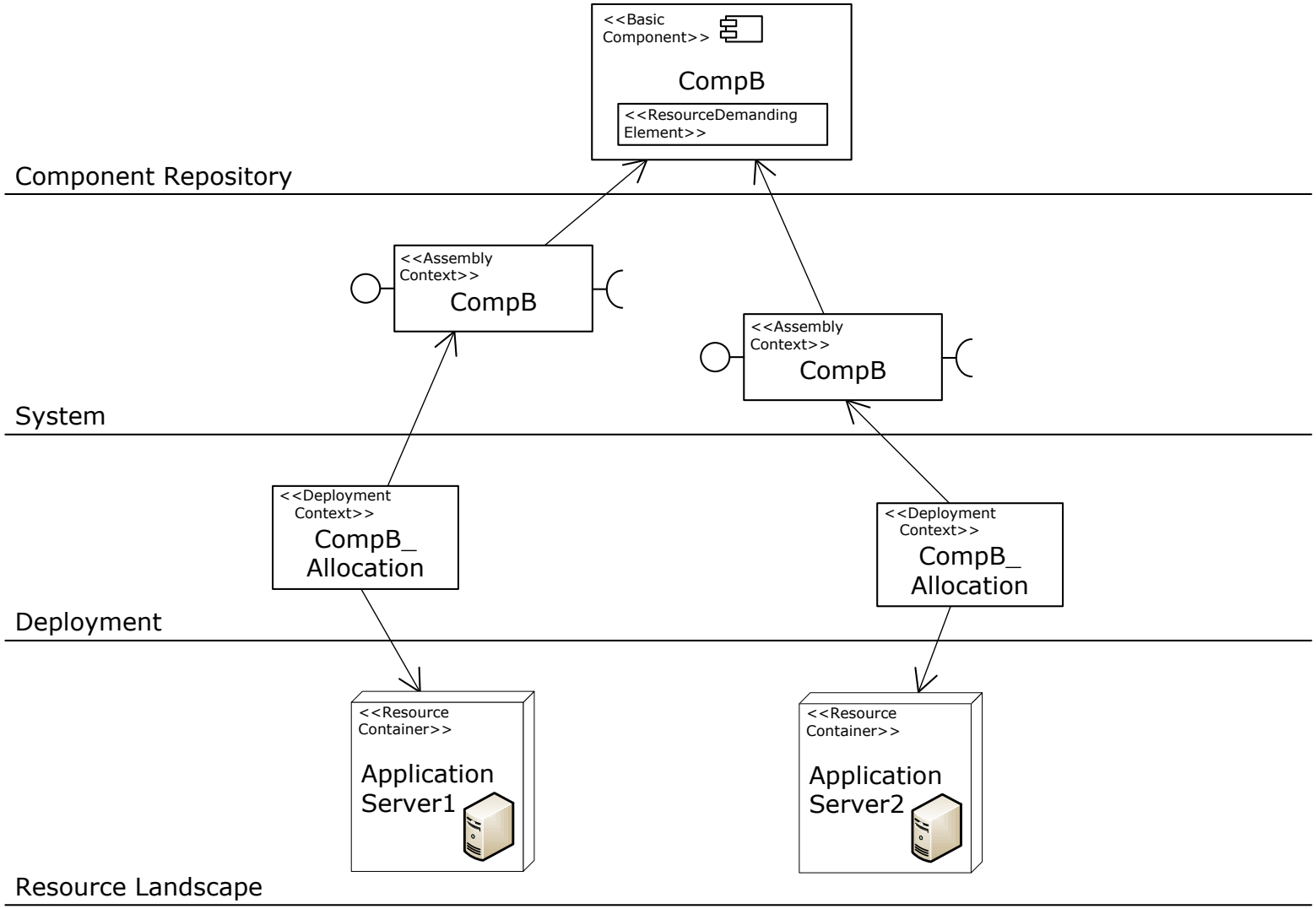
- Service Behavior Abstractions for Different Levels of Granularity



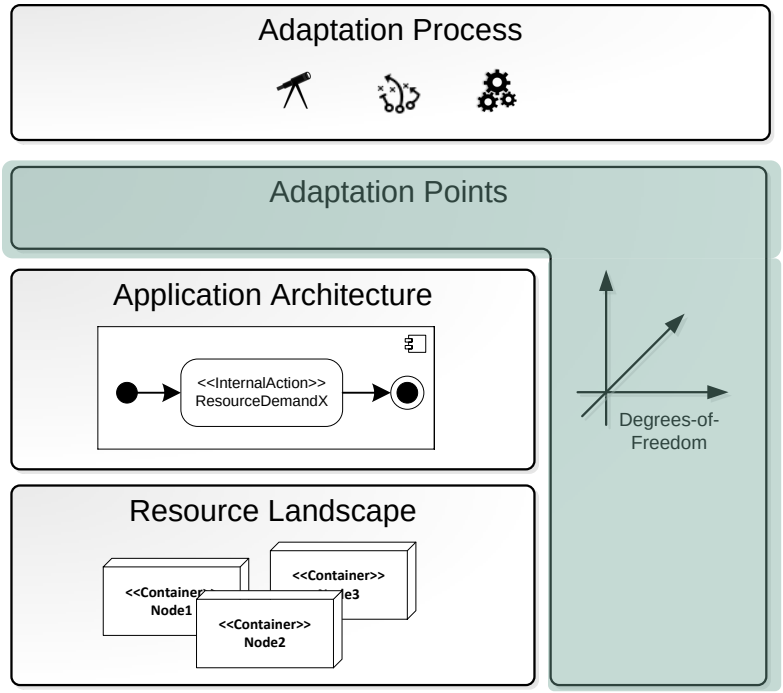
- Parameter and Context Dependencies



Deployment


<<Basic Component>>
CompB
<<ResourceDemanding Element>>
Component Repository
<<Assembly Context>>
CompB
<<Assembly Context>>
CompB
System
<<Deployment Context>>
CompB_ Allocation
<<Deployment Context>>
CompB_ Allocation
Deployment
<<Resource Container>>
Application Server1
<<Resource Container>>
Application Server2
Resource Landscape

DMM: Adaptation Points



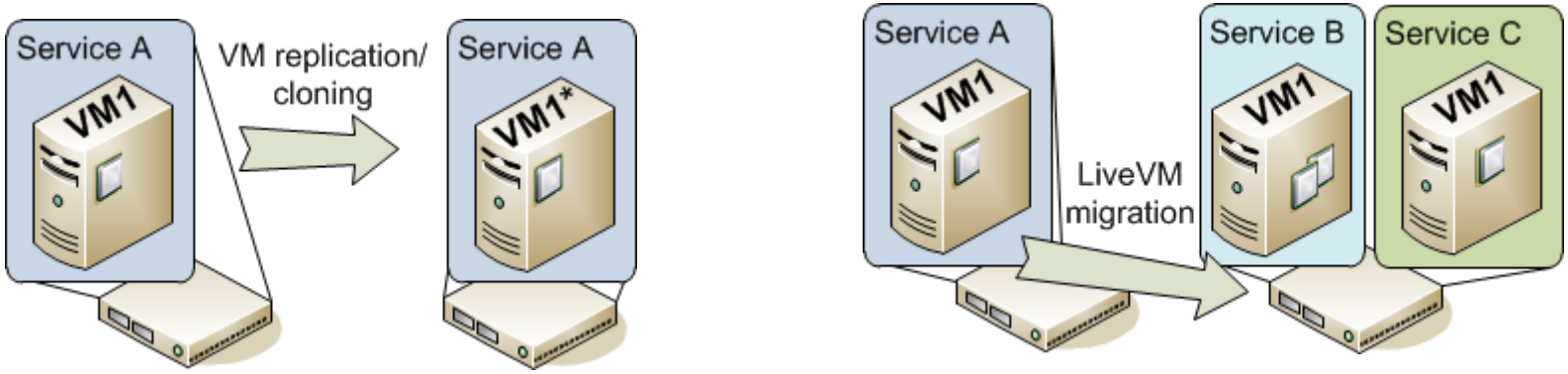
Further details in:

- N. Huber, F. Brosig and S. Kounev. **Modeling Dynamic Virtualized Resource Landscapes**. In Proceedings of the 8th ACM SIGSOFT International Conference on the Quality of Software Architectures (QoSA 2012), Bertinoro, Italy, June 25-28, 2012.

Adaptation Points: Examples

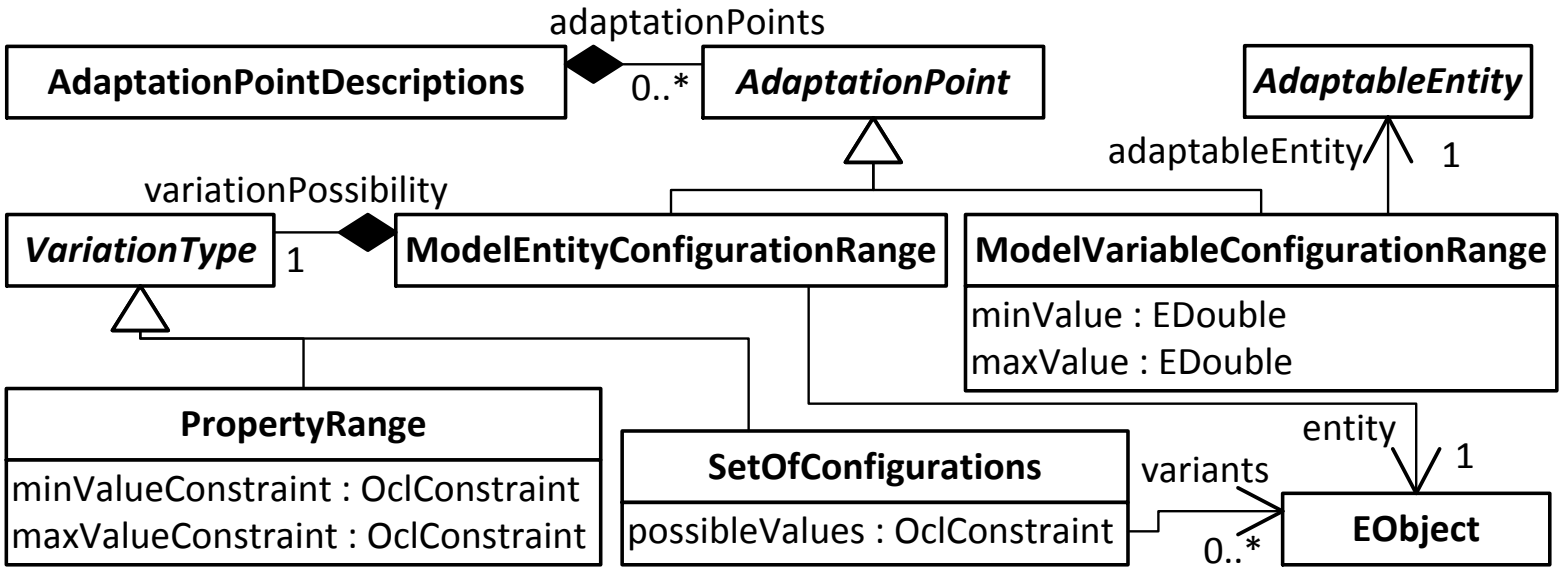
- Scaling Resources

- Replicating VMs, Migrating VMs

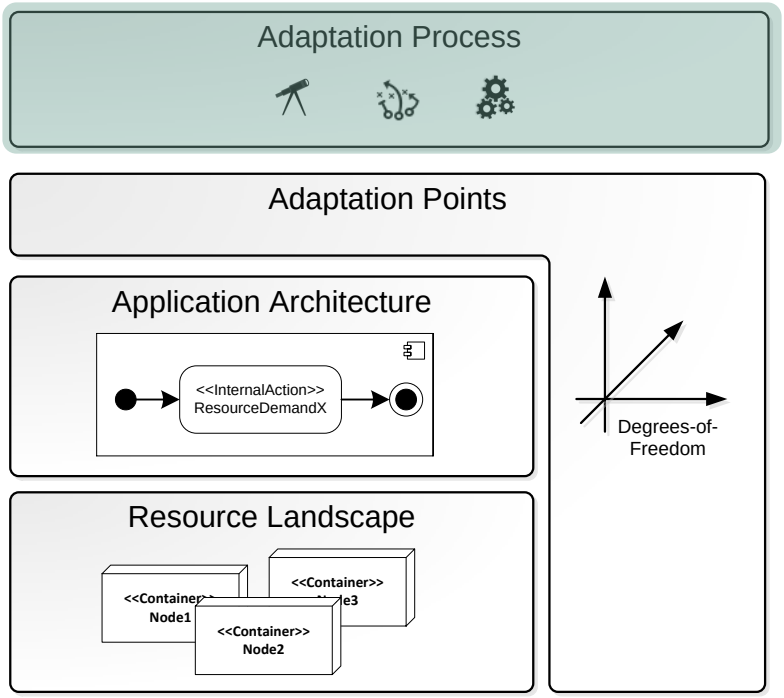


Adaptation Points

- Specification of valid system configurations
- “Decorator” model of static view



DMM: Adaptation Process



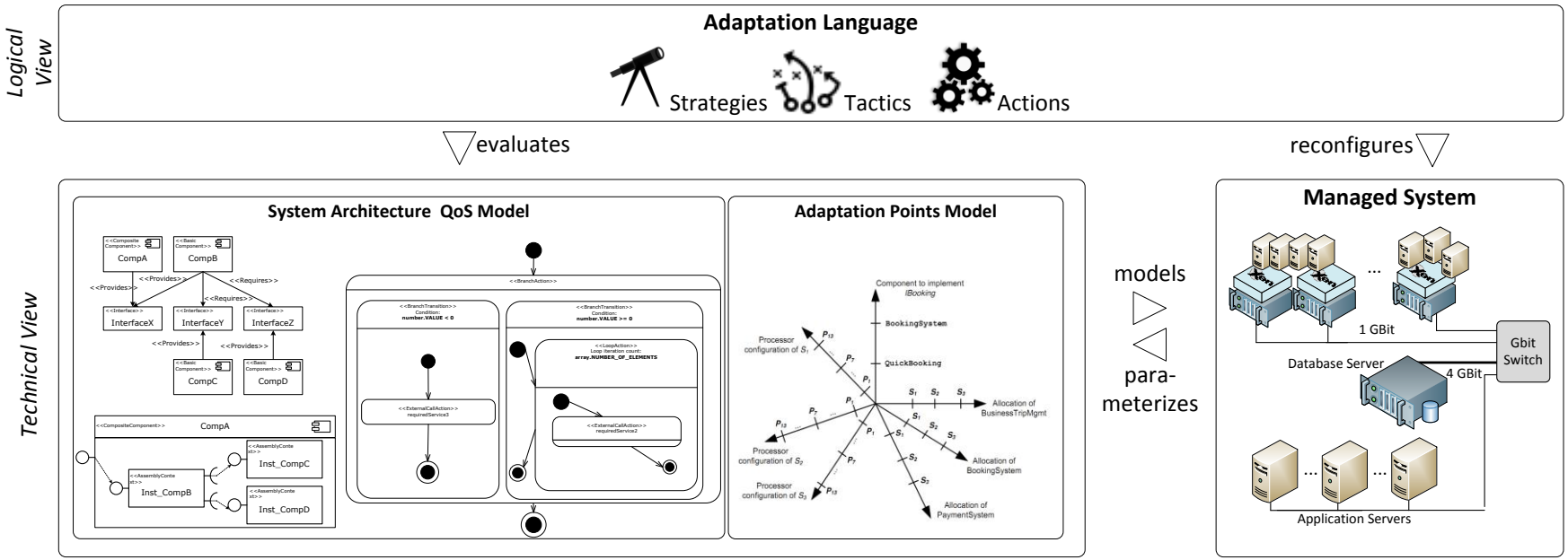
Details in:

- N. Huber, A. van Hoorn, A. Koziolek, F. Brosig, and S. Kounev. **S/T/A: Meta-modeling Run-time Reconfiguration in Component-based System Architectures**. In 9th IEEE Int. Conference on e-Business Engineering (ICEBE), Hangzhou, China, September 9-11 2012.

Motivation

- Rapid growth of autonomic computing and self-adaptive systems engineering
- Open challenges
  - Hard-coded or system-specific reconfiguration techniques
  - How to separate software design and implementation from system reconfiguration logic?
- Main issues
  - How to abstract from system-specific details?
  - How to enable the reuse of adaptation strategies?
- Vision: Holistic model-based Approach

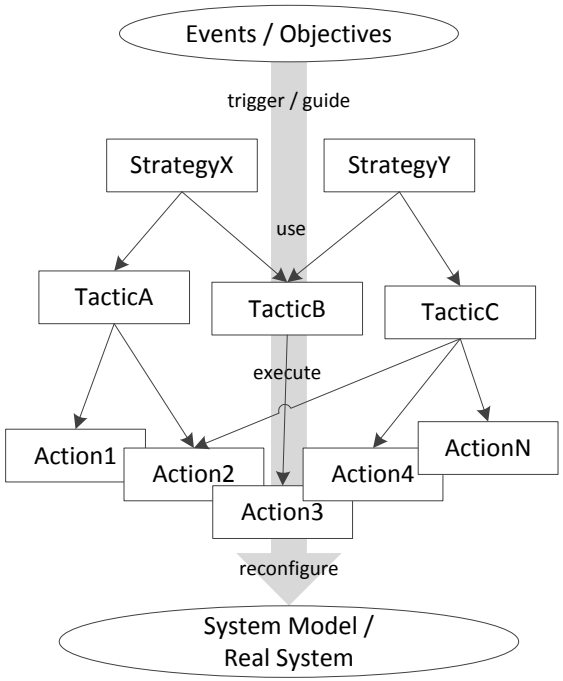
Model-based System Adaptation



Adaptation Language - Idea

- Describe system adaptation processes at the system architecture level
  - Distinguish high-level reconfiguration objectives from low-level implementation details
  - Explicitly separate technical from logical aspects
  - Capture reconfiguration logic in a generic, human-understandable, machine processable and reusable way
  - Provide intuitive modeling concepts that can be employed by system architects and software developers
  - Facilitate maintenance and reuse

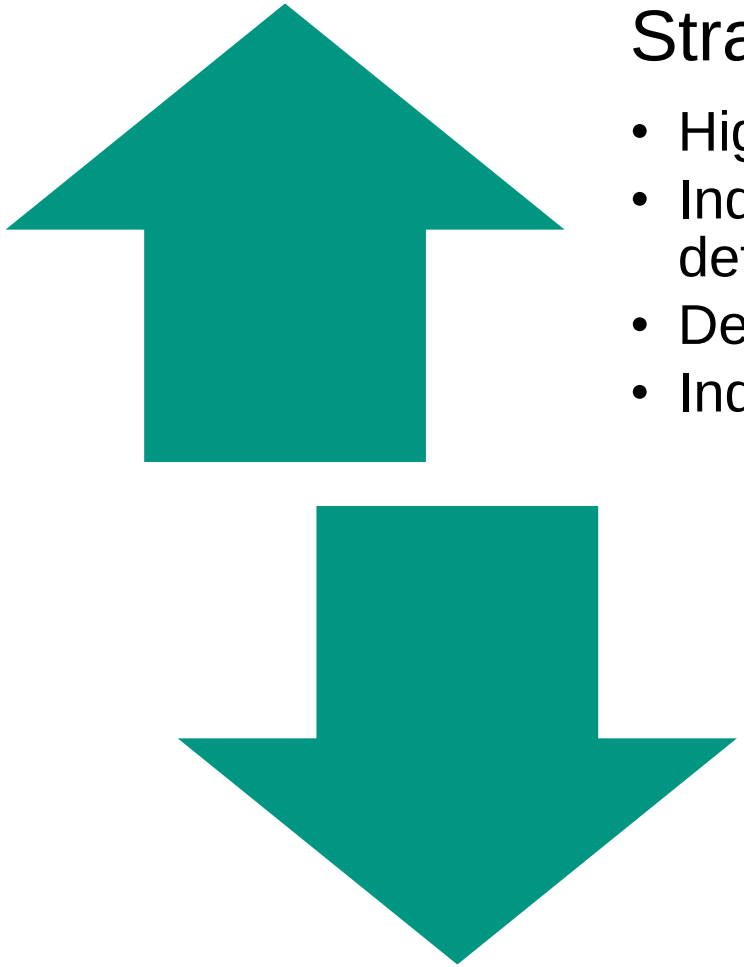
S/T/A Adaptation Language



Separate

- Logical view, high-level process
- Technical view, low-level operations

Separation of Concerns



Strategies

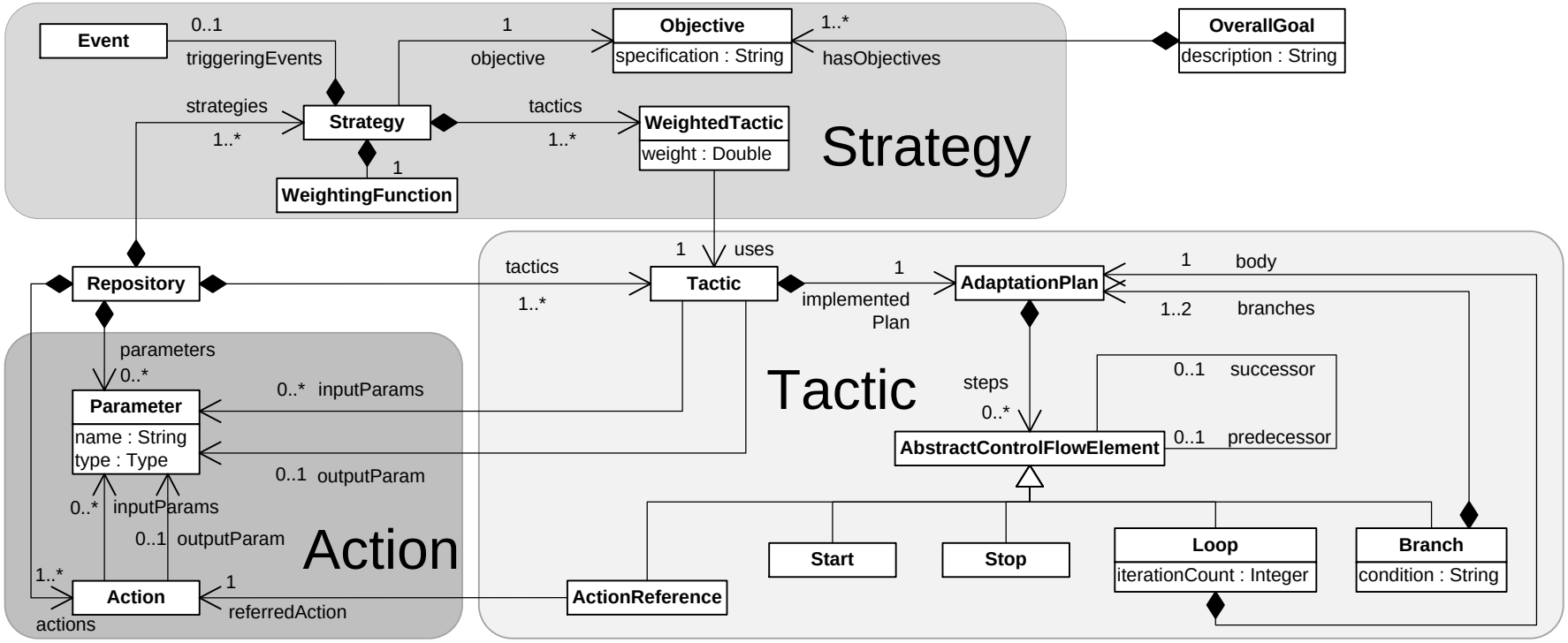
- High-level
- Independent of system specific details
- Describe process view
- Indeterminism

Tactics & Actions

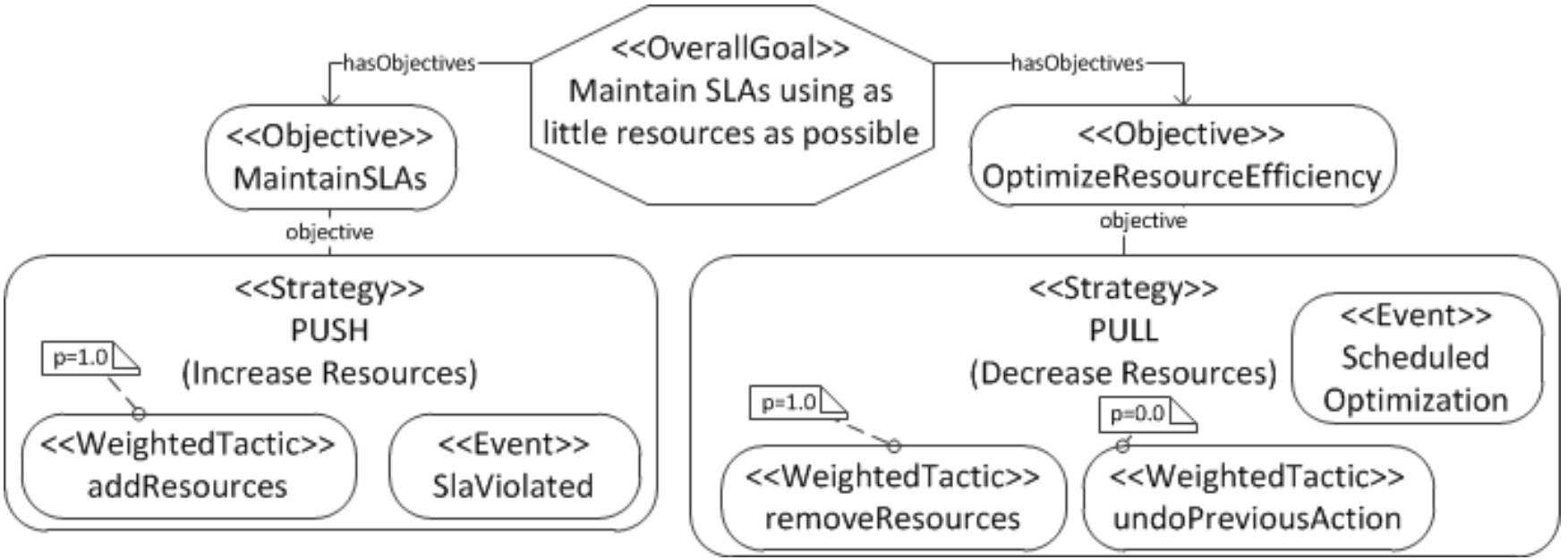
- Low-level
- System specific
- Reconfiguration operations
- Deterministic

S/T/A Meta-Model

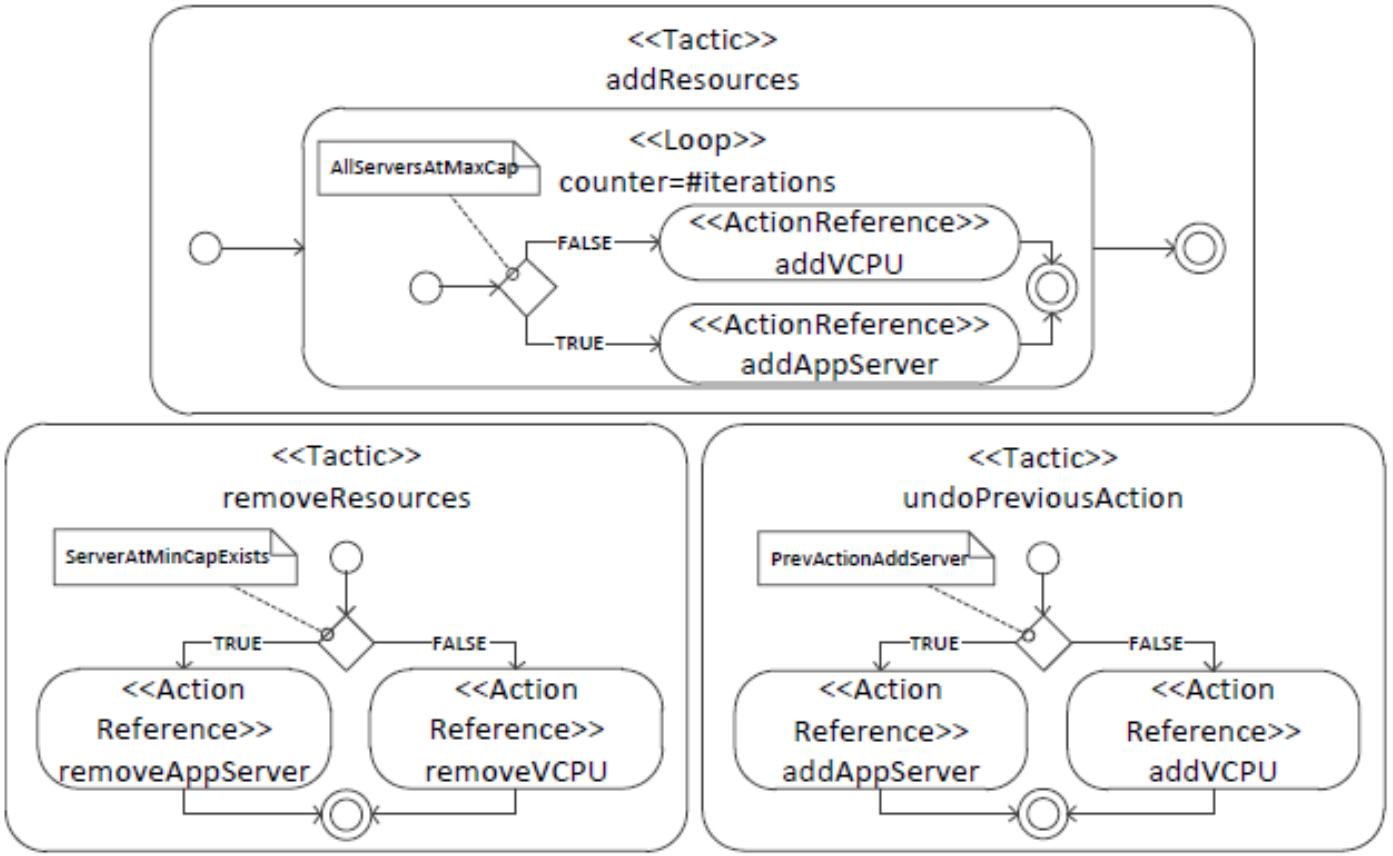
- Actions refer to adaptation points / DoF Model
- Tactics execute actions in adaptation plans
- Strategies use weighted tactics



Example Strategies

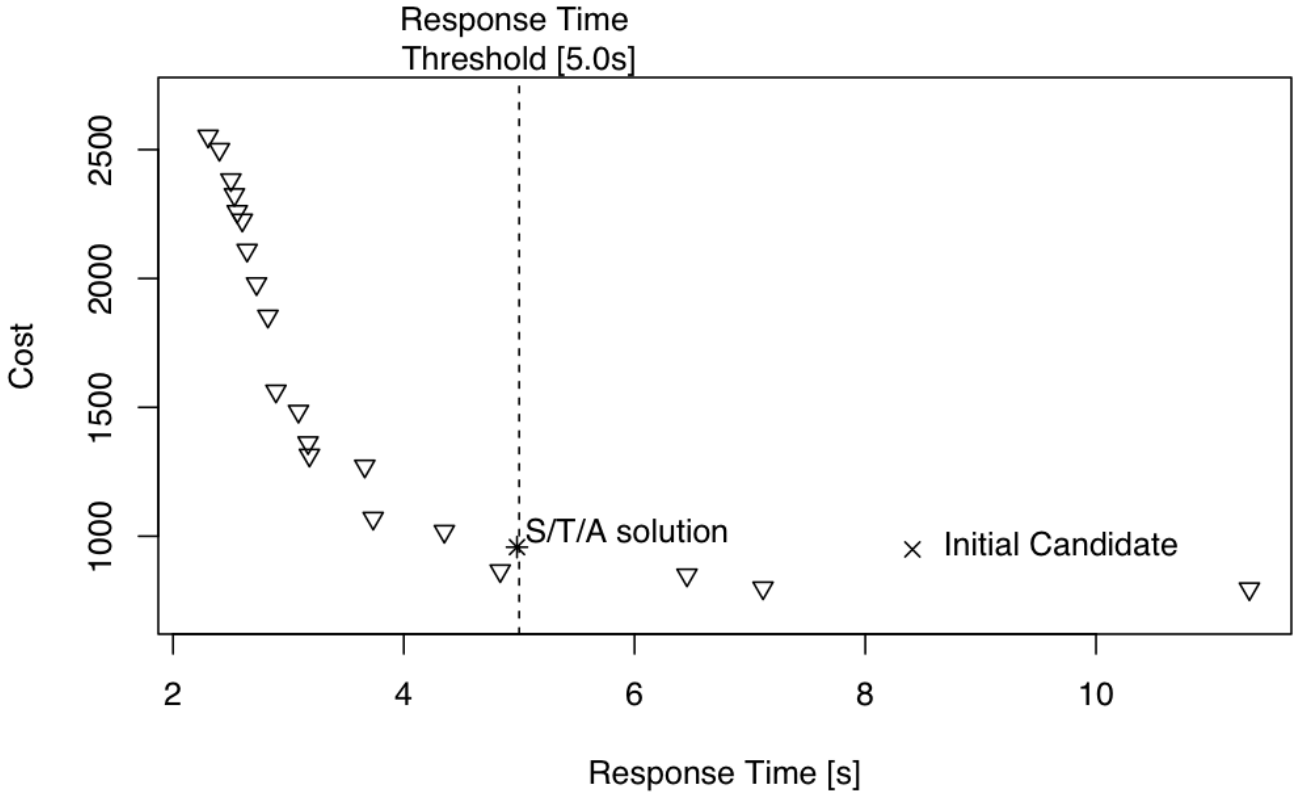


Example Tactics

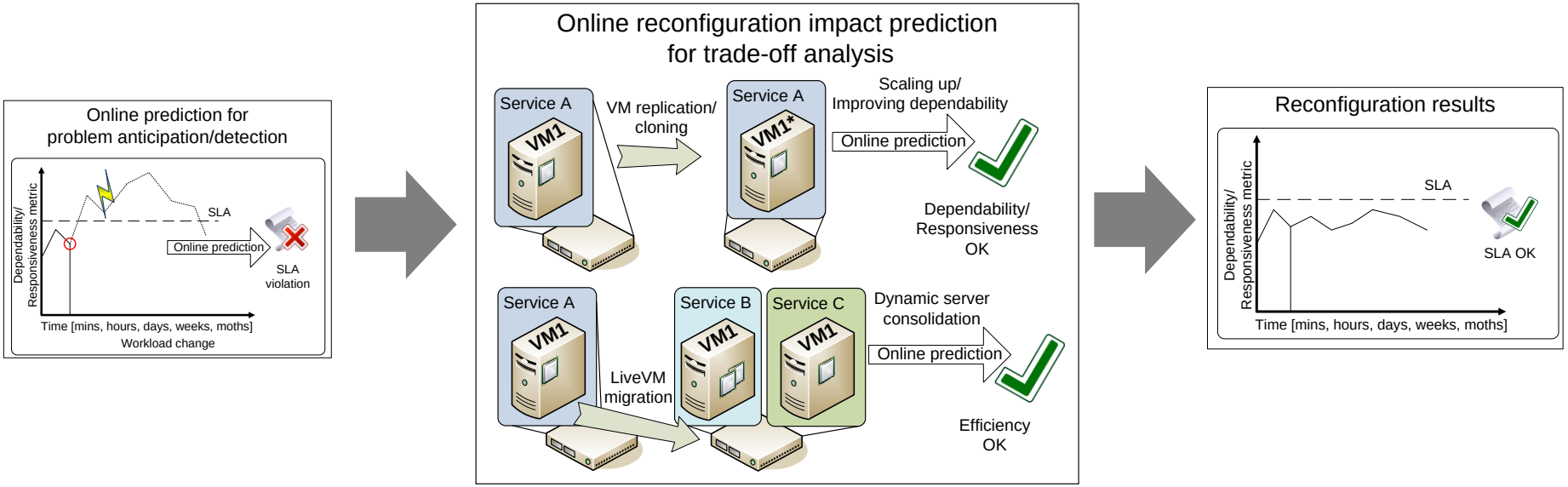


Evaluation

- S/T/A implemented in PerOpteryx



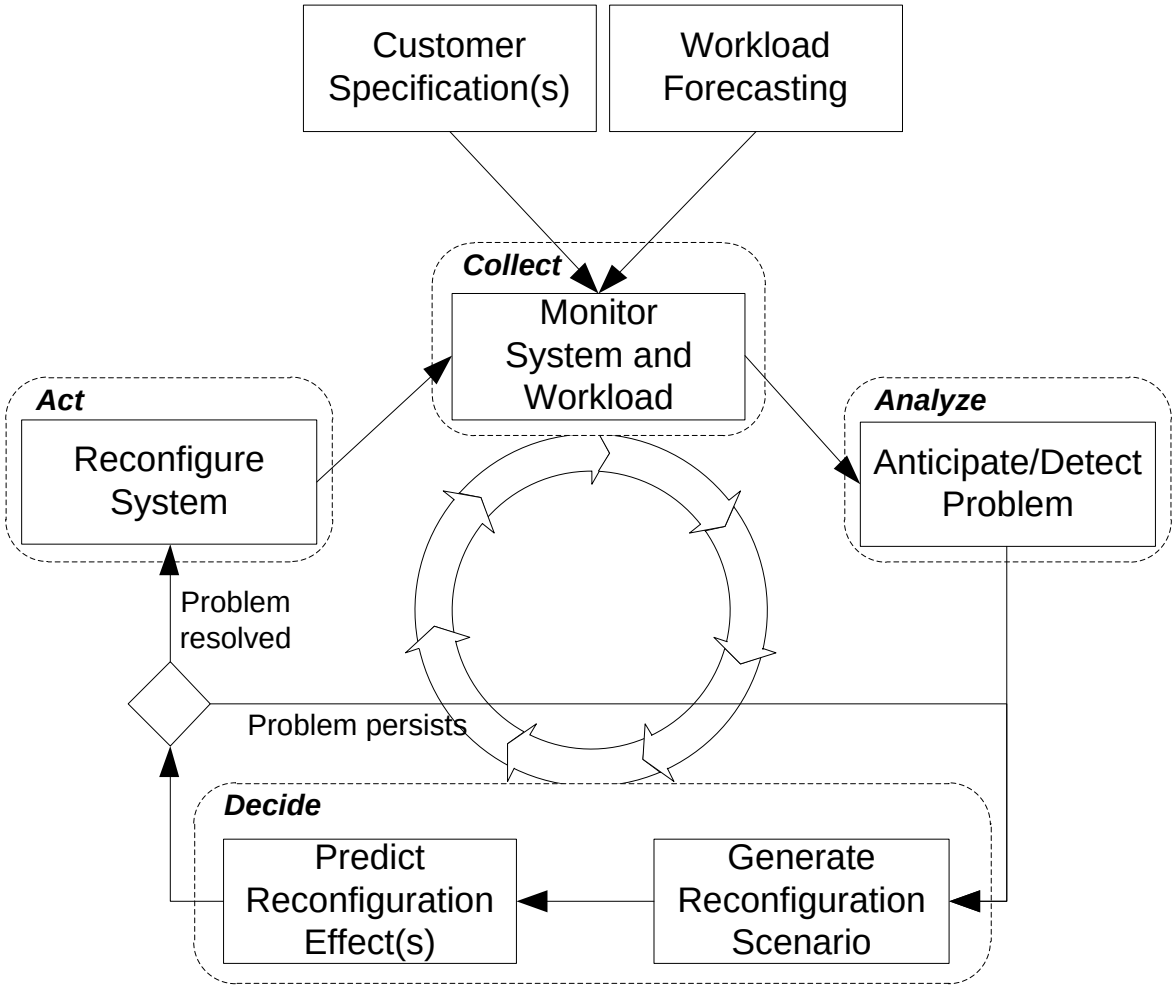
Self-Adaptive Resource Management



Further details in:

- N. Huber, F. Brosig, and S. Kounev. **Model-based Self-Adaptive Resource Allocation in Virtualized Environments**. In 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011), Honolulu, HI, USA, May 23-24, 2011.

Self-Adaptive Resource Allocation



Reconfiguration Algorithm

Decide

PUSH Phase

- Add resources
  - vCPUs (if available)
  - Application server nodes
  - until

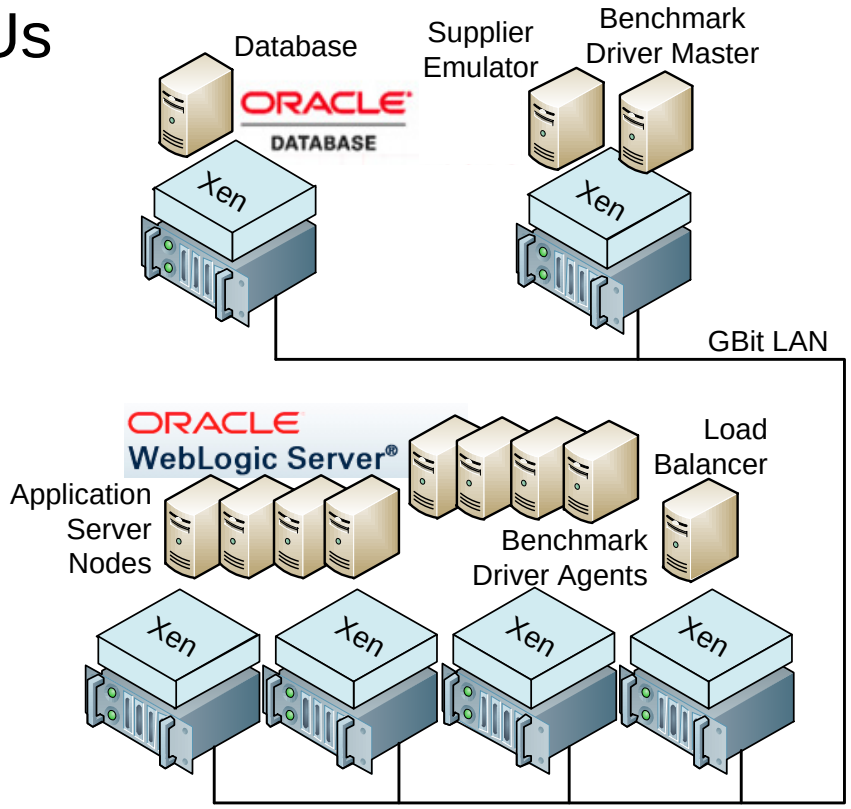
$$\overline{cap}(c,t) = \left\lceil \frac{\sum_{c \in \widetilde{C}} c[\lambda] \cdot D(c[s])}{\sum_{c \in C} c[\lambda] \cdot D(c[s])} \right\rceil \cdot cap(c,t)$$

PULL Phase

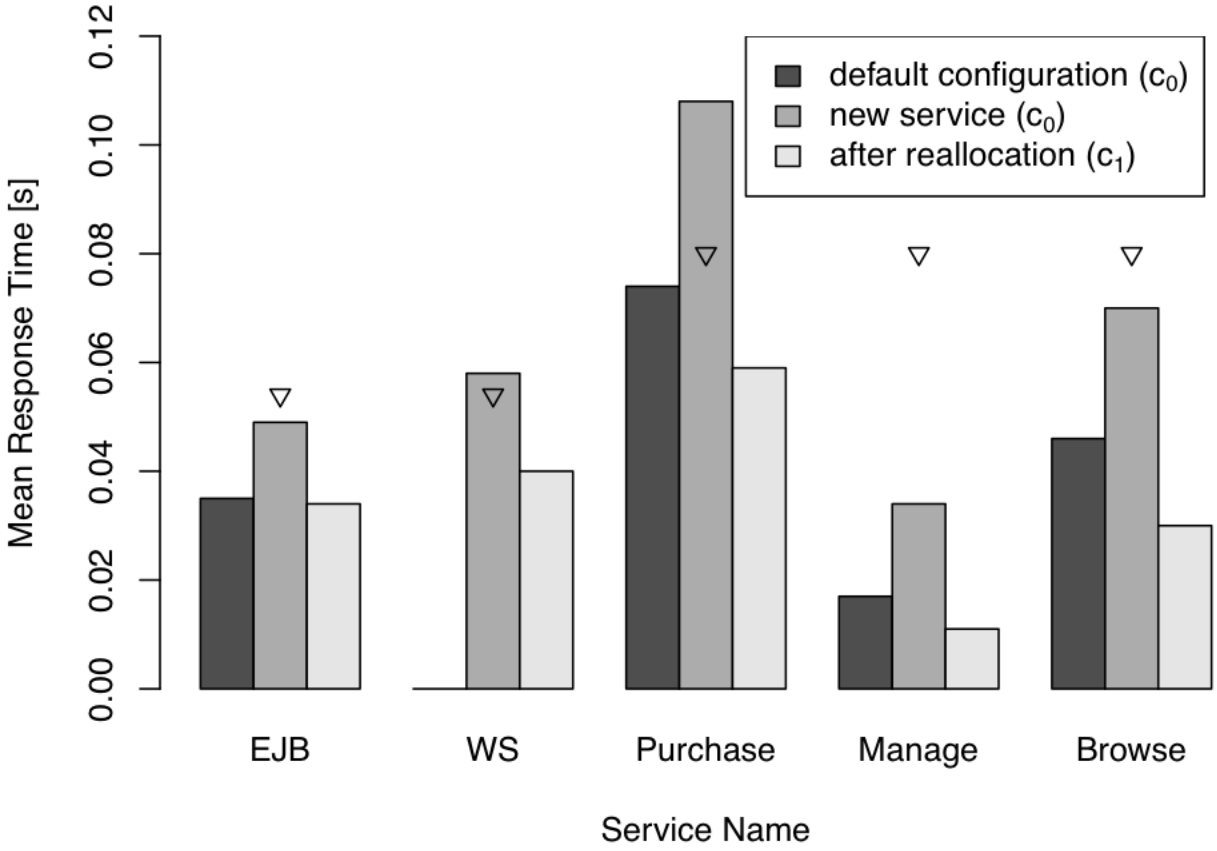
- Remove underutilized resources as long as no SLAs are violated

Experimental Setup

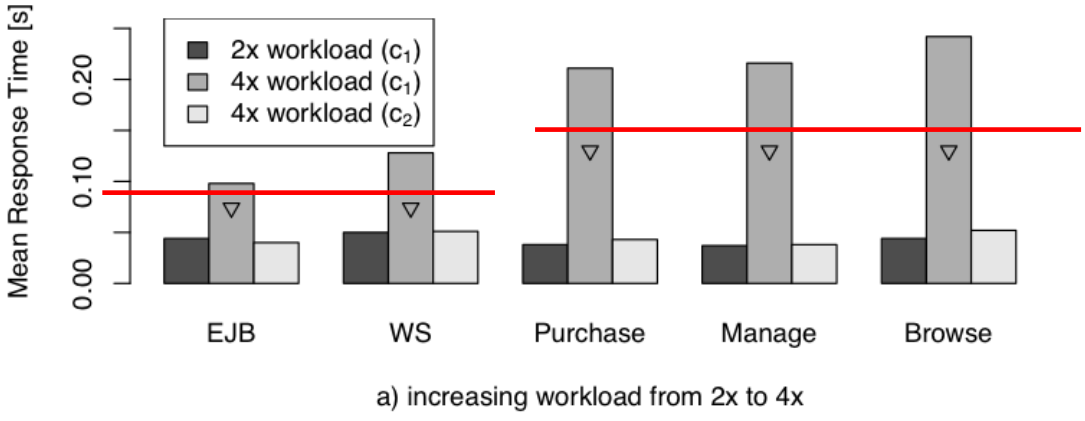
- Six blade servers
  - 2 Xeon E5430 4-core CPUs
  - 32 GB of main memory
  - Citrix XenServer 5.5
  - Oracle WebLogic 10.3.3
  - Oracle Database 11g



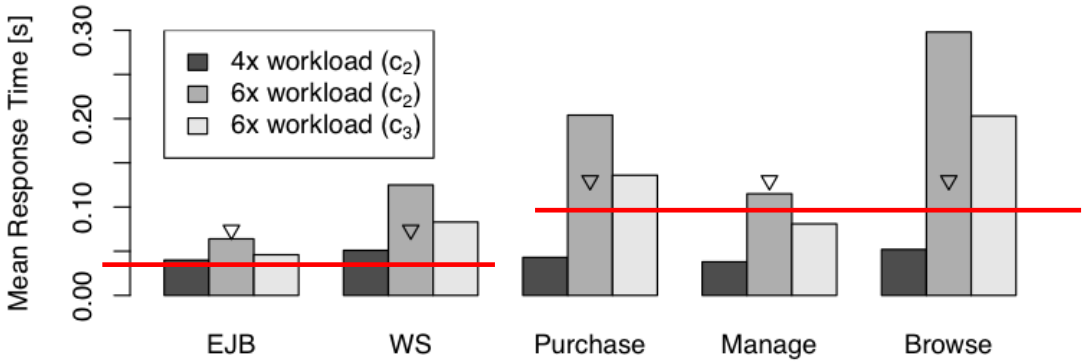
What If: New Service Added?



What If: Workload Changes?

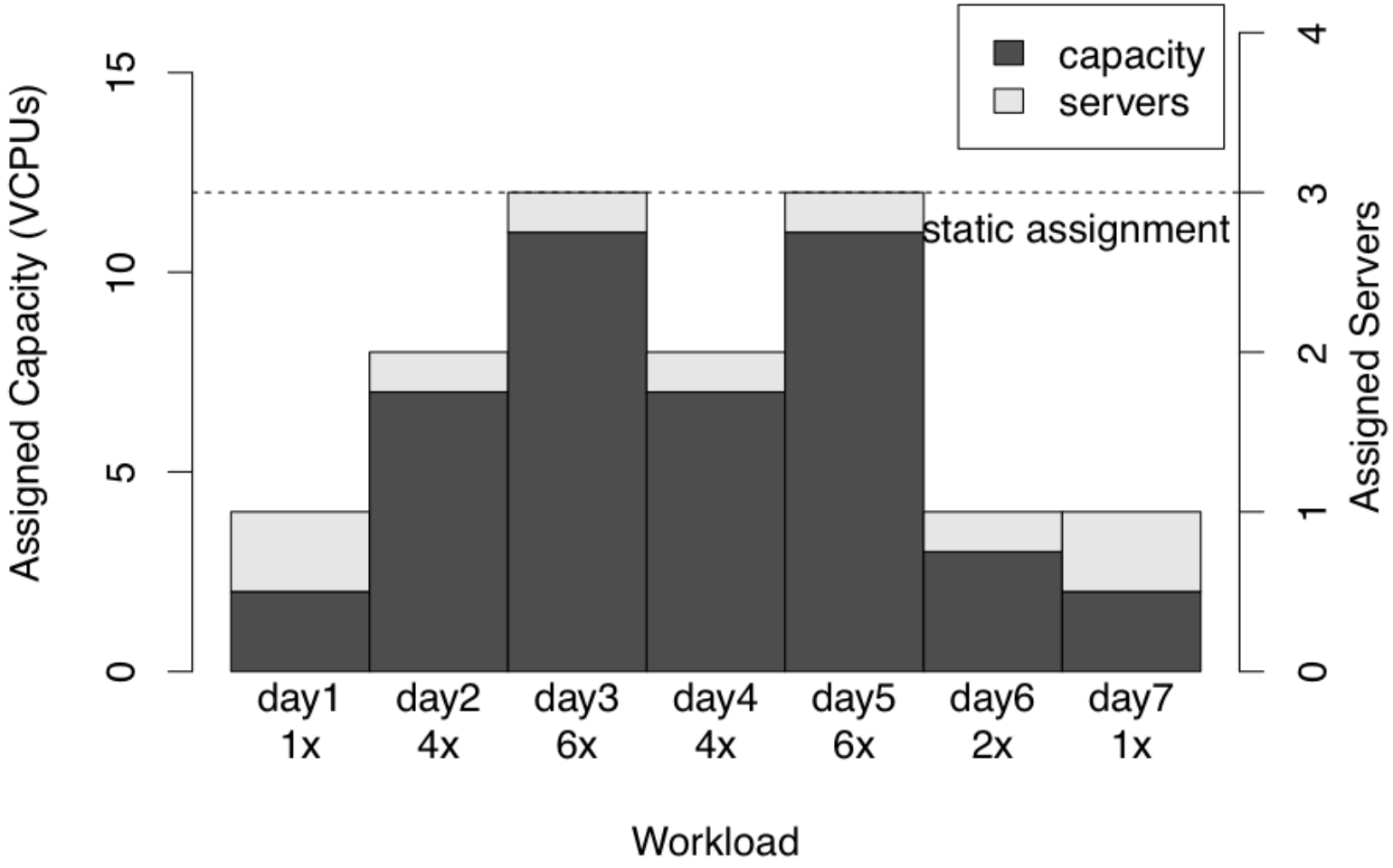


a) increasing workload from 2x to 4x

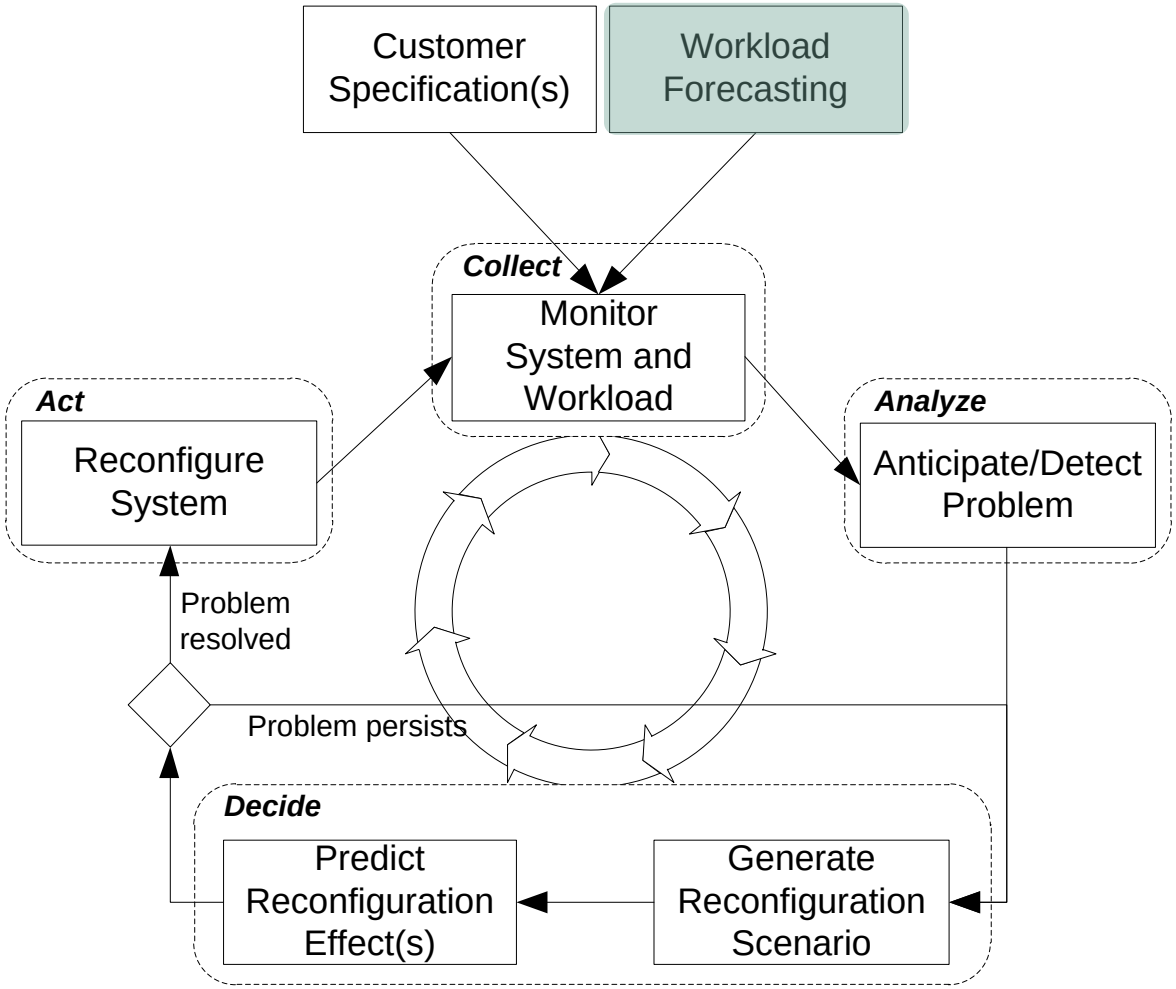


b) increasing workload from 4x to 6x

Benefits in Cost Savings



Self-Adaptive Resource Allocation

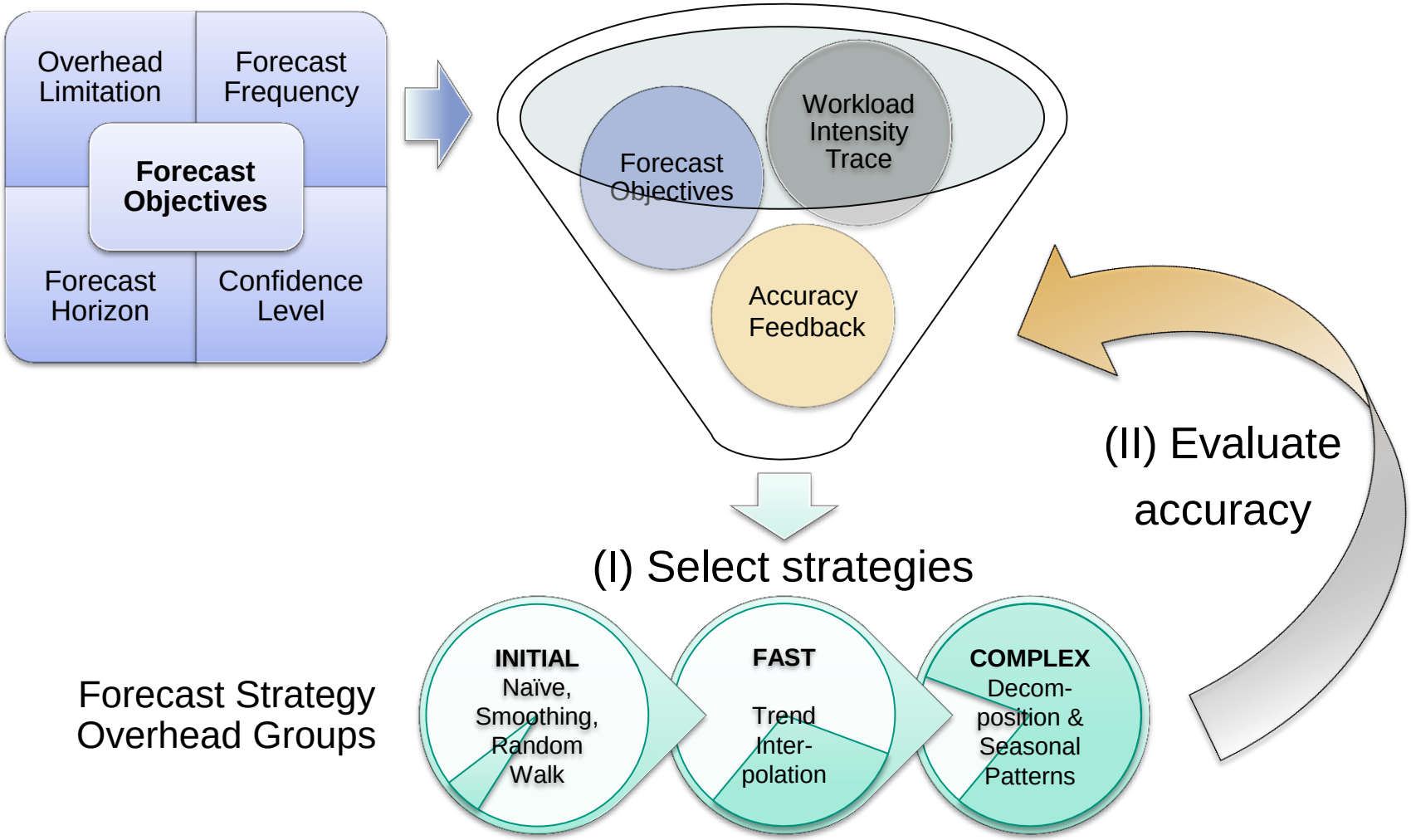


Karlsruhe Institute of Technology

Nikolas Herbst (Diploma thesis)

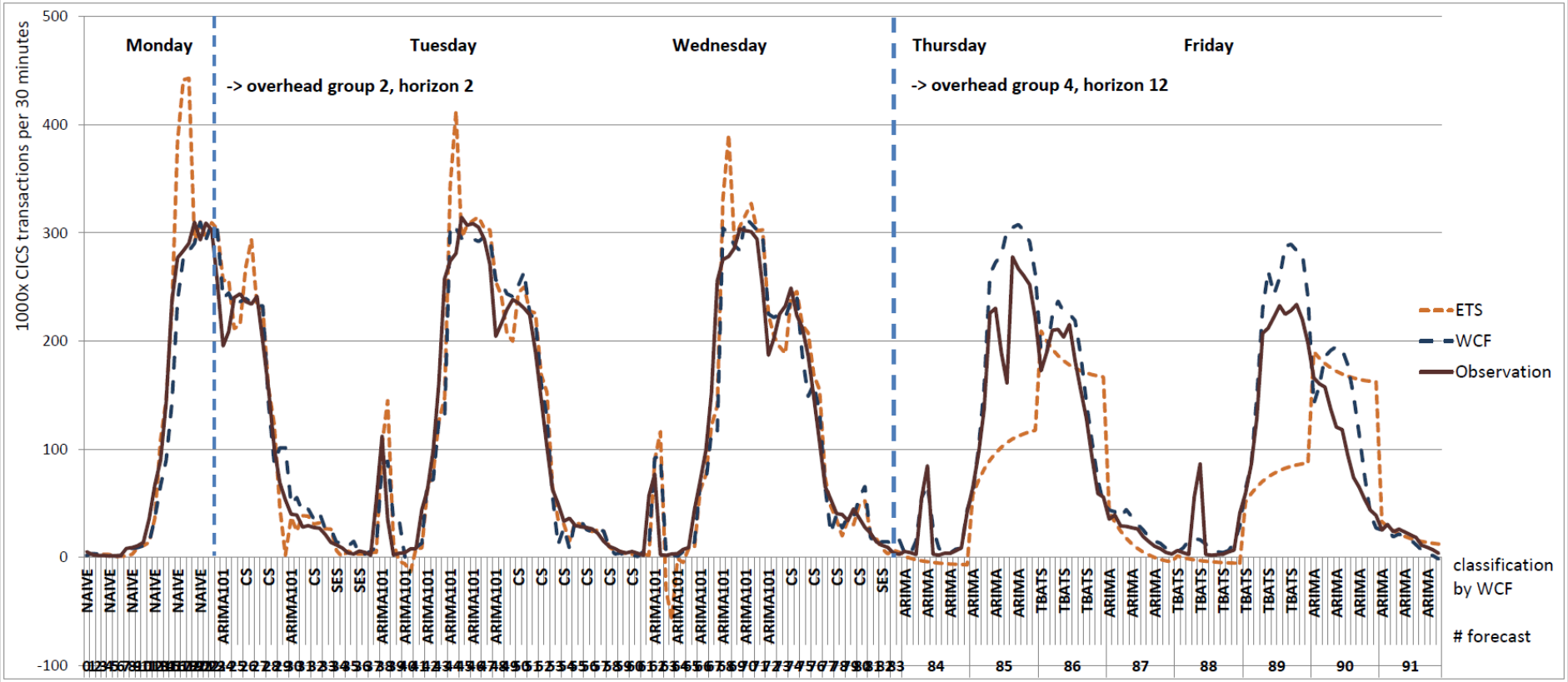
WORKLOAD CLASSIFICATION AND FORECASTING

Workload Classification & Forecasting

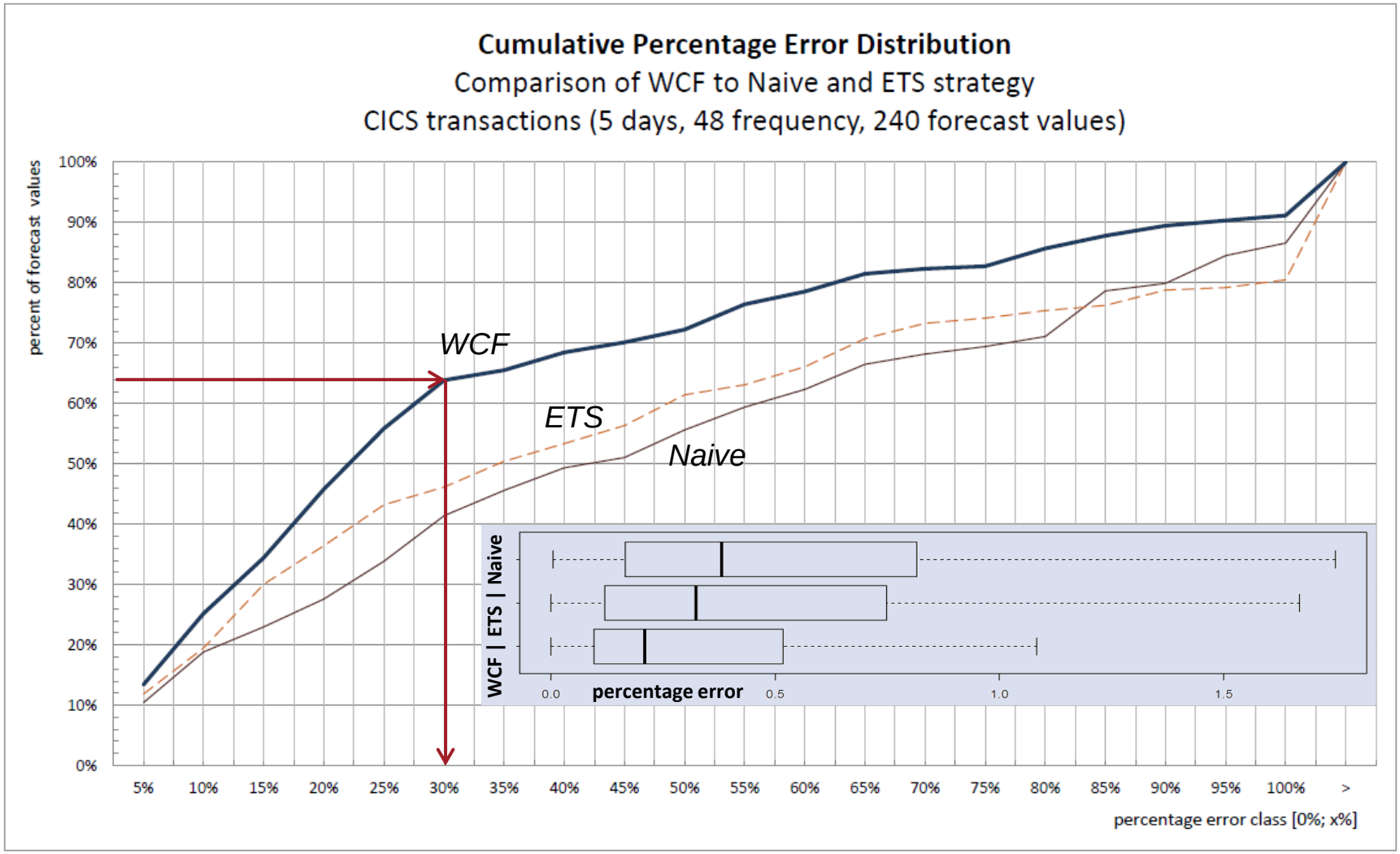


Experiment: Example for Forecast Accuracy Improvement

- Real-world workload intensity trace (IBM CICS transactions on System z)
- Comparison of **Workload Classification & Forecasting (WCF)** approach to **Extended Exponential Smoothing (ETS)** and **Naive** forecast

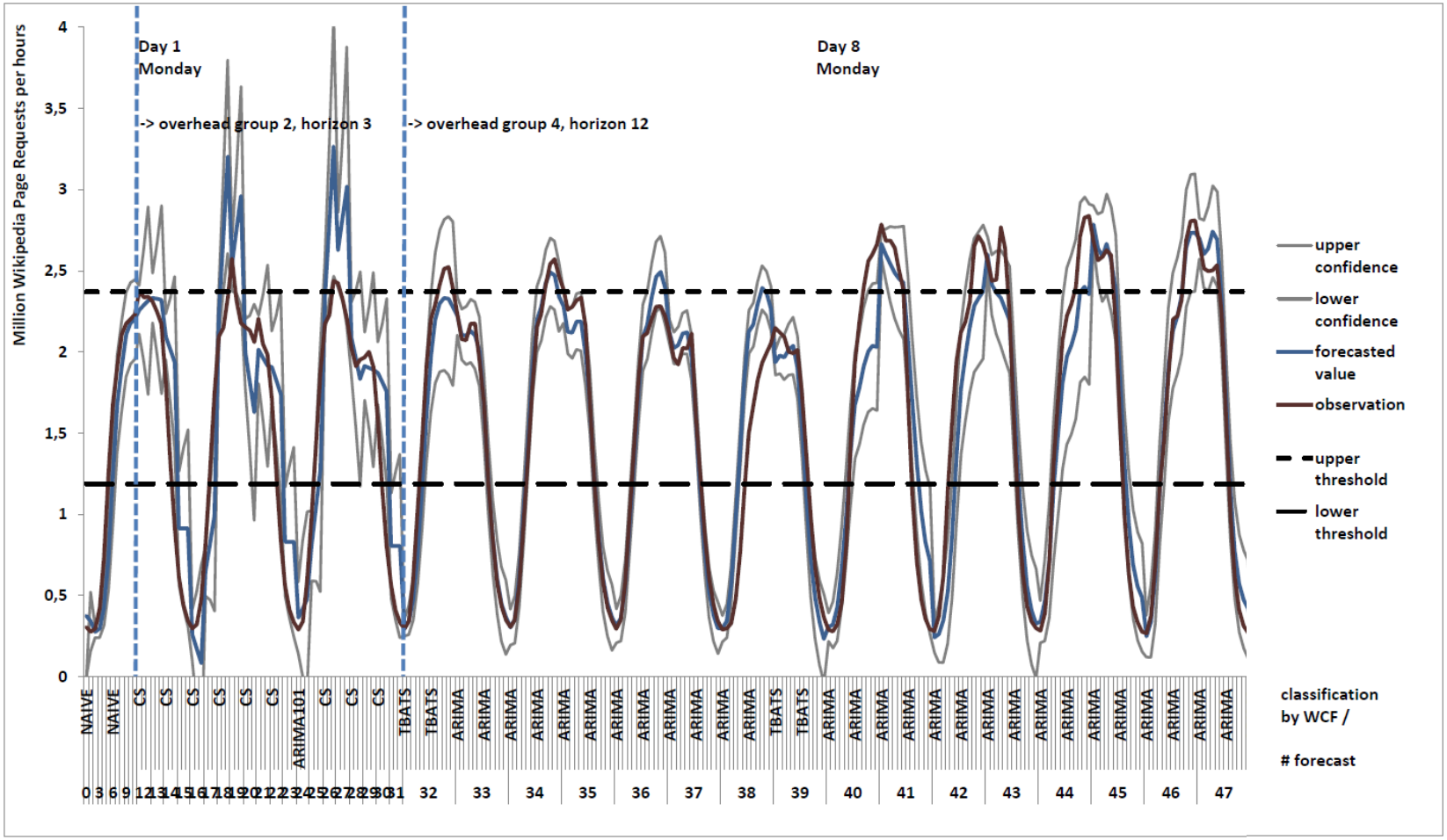


Experiment


Cumulative Percentage Error Distribution
Comparison of WCF to Naive and ETS strategy
CICS transactions (5 days, 48 frequency, 240 forecast values)
percent of forecast values
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
WCF
ETS
Naive
WCF | ETS | Naive
0.0
percentage error
0.5
1.0
1.5
5%
10%
15%
20%
25%
30%
35%
40%
45%
50%
55%
60%
65%
70%
75%
80%
85%
90%
95%
100%
>
percentage error class [0%; x%]

Case Study: Example for Using Forecast Results

- **Scenario:** Additional server instances at certain thresholds, 3 weeks
- Real-world workload intensity trace (**Wikipedia DE** page requests per hour)



Case Study

Resource provisioning:

(I) Without forecasting (solely reactive):

Resource provisioning actions triggered by

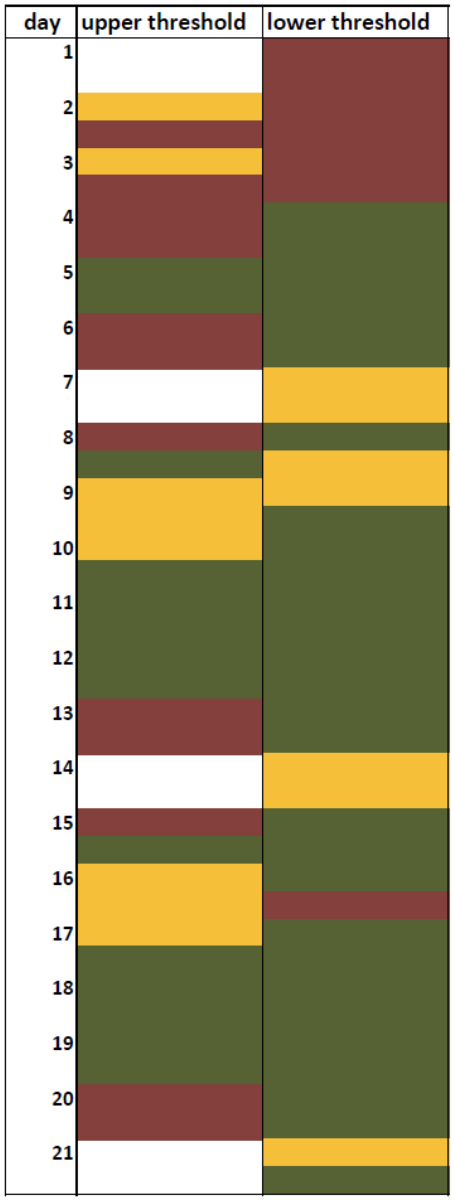
76 SLA violations

(II) Interpreting WCF forecast results (add. proactive):

Reduction to **34 or less SLA violations**

→ No significant change in resource usage observed

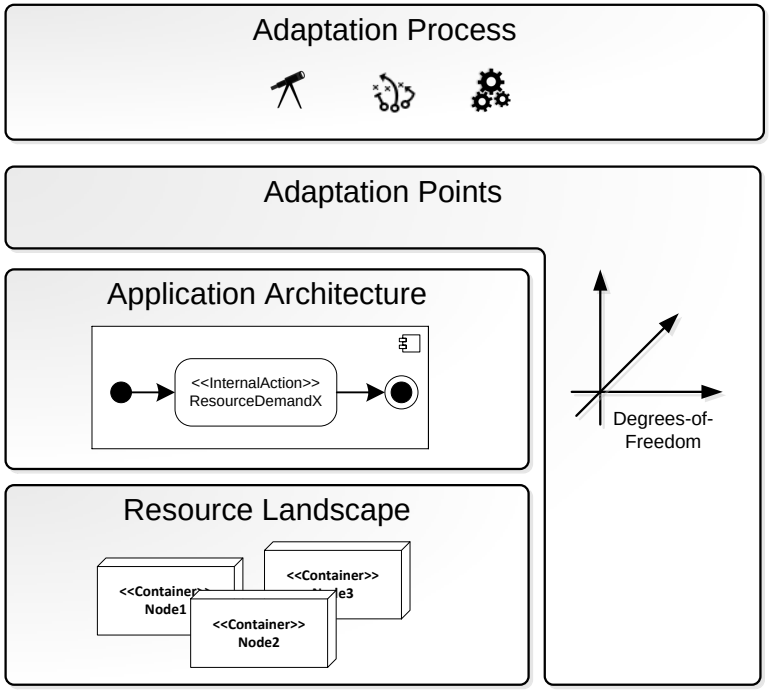
(server instances per hour)



8x	correct forecast:	server instance not needed
42 x	correct forecast:	server instance needed at time t
15 x	nearly correct forecast:	time t slightly too early or too late
19 x	incorrect forecast:	need not detected or false positive

Summary

- Meta-Model for
  - Resource Landscape
  - Performance Influences of Virtualization Layers
  - Adaptation Points
  - Adaptation Process
- Example: Model-based resource allocation
- Workload Forecasting



Thank you!
Any Questions?

http://www.descartes-research.net/