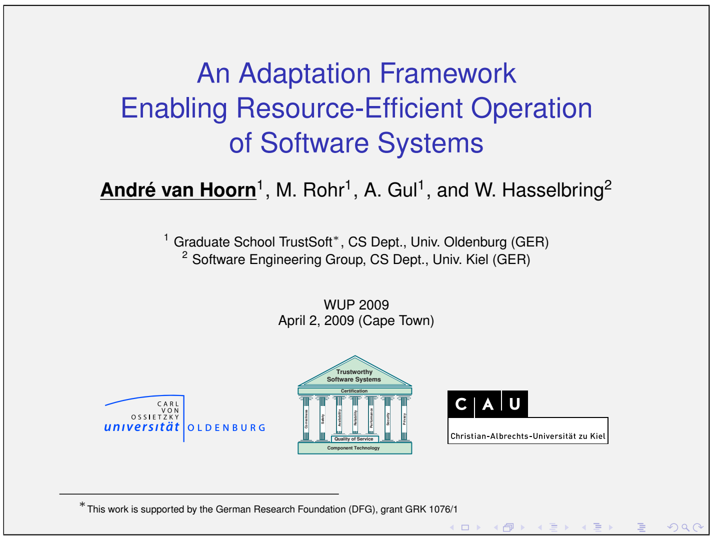


An Adaptation Framework Enabling Resource-Efficient Operation of Software Systems

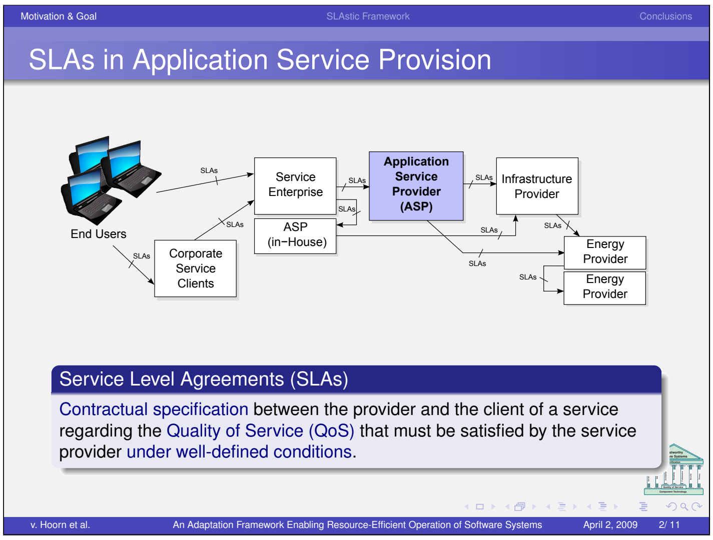
André van Hoorn[1], M. Rohr[1], A. Gul[1], and W. Hasselbring[2]

[1] Graduate School TrustSoft*, CS Dept., Univ. Oldenburg (GER)
[2] Software Engineering Group, CS Dept., Univ. Kiel (GER)

WUP 2009
April 2, 2009 (Cape Town)

CARL VON OSSIETZKY universität OLDENBURG
Trustworthy Software Systems
Certification
Quality of Service
Component Technology
C A U
Christian-Albrechts-Universität zu Kiel

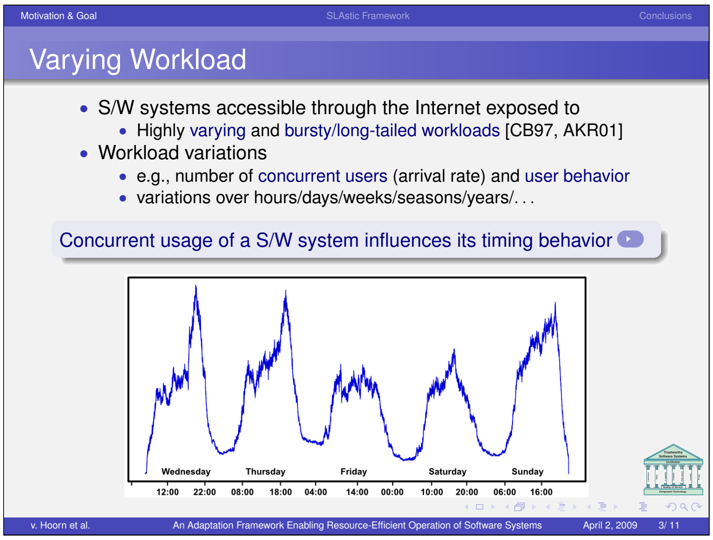
* This work is supported by the German Research Foundation (DFG), grant GRK 1076/1



SLAs in Application Service Provision

Service Level Agreements (SLAs)

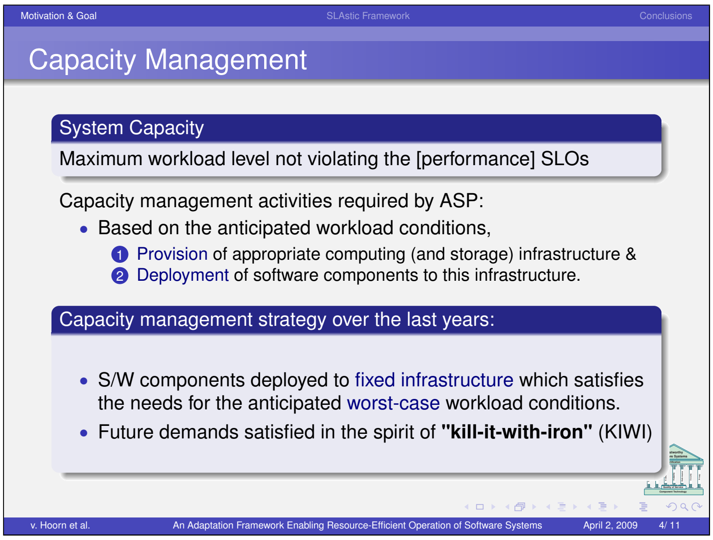
Contractual specification between the provider and the client of a service regarding the Quality of Service (QoS) that must be satisfied by the service provider under well-defined conditions.



Varying Workload

- S/W systems accessible through the Internet exposed to
  - Highly varying and bursty/long-tailed workloads [CB97, AKR01]
- Workload variations
  - e.g., number of concurrent users (arrival rate) and user behavior
  - variations over hours/days/weeks/seasons/years/. . .

Concurrent usage of a S/W system influences its timing behavior



Capacity Management

System Capacity

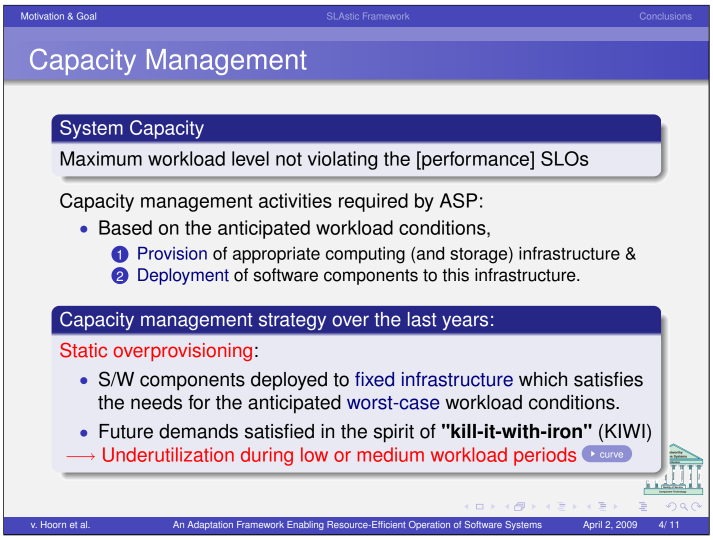
Maximum workload level not violating the [performance] SLOs

Capacity management activities required by ASP:

- Based on the anticipated workload conditions,
  1. Provision of appropriate computing (and storage) infrastructure &
  2. Deployment of software components to this infrastructure.

Capacity management strategy over the last years:

- S/W components deployed to fixed infrastructure which satisfies the needs for the anticipated worst-case workload conditions.
- Future demands satisfied in the spirit of **"kill-it-with-iron"** (KIWI)



Capacity Management

System Capacity

Maximum workload level not violating the [performance] SLOs

Capacity management activities required by ASP:

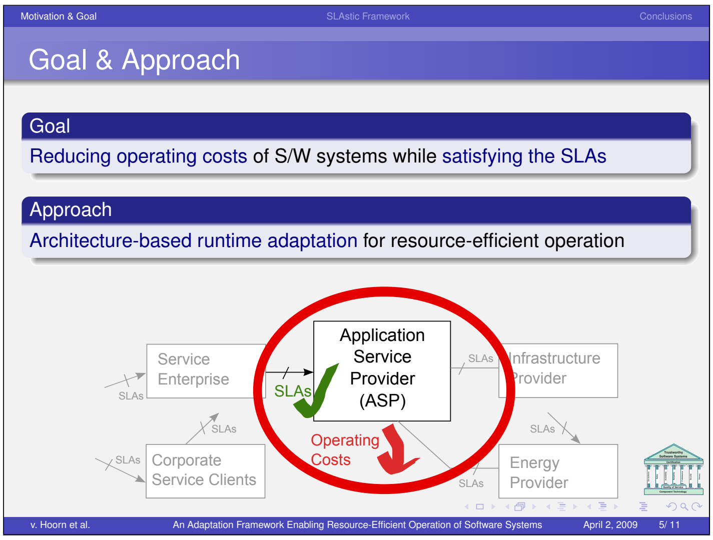
- Based on the anticipated workload conditions,
  1. Provision of appropriate computing (and storage) infrastructure &
  2. Deployment of software components to this infrastructure.

Capacity management strategy over the last years:

Static overprovisioning:

- S/W components deployed to fixed infrastructure which satisfies the needs for the anticipated worst-case workload conditions.
- Future demands satisfied in the spirit of **"kill-it-with-iron"** (KIWI)

⟶ Underutilization during low or medium workload periods curve



Goal & Approach

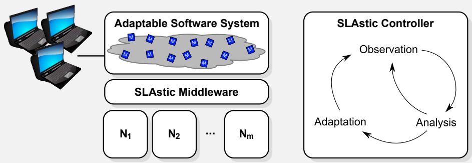
Goal

Reducing operating costs of S/W systems while satisfying the SLAs

Approach

Architecture-based runtime adaptation for resource-efficient operation

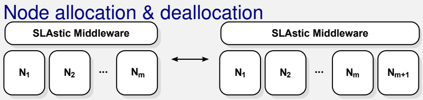
SLAstic (sɪˈlæstɪk) Adaptation Framework



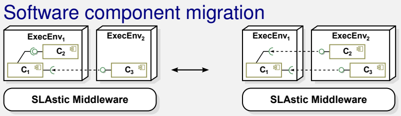
- Middleware
  - Provides instrumentation infrastructure (e.g., [RvHM$^+$08]) and
  - Executes adaptation operations (e.g., [Bun08])
- Controller
  - Executes the self-adaptation cycle
  - Maintains the runtime models
  - → Triggers adaptation operations

Considered Architectural Adaptation Operations

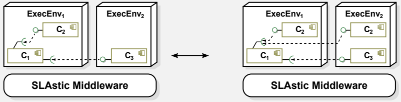
1. Node allocation & deallocation



2. Software component migration

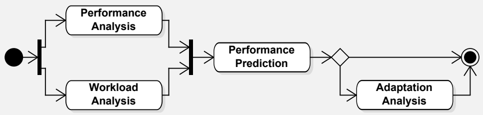


3. Component-level load-(un)balancing



Self-adaptation Cycle

1. Observation
   - Extract and pre-process measurement data of elapsed period (S/W system is continuously being monitored)
2. Analysis ▸ details



3. Adaptation
   - Trigger middleware to execute selected adaptation operations
   - Reflect changes in runtime models (after adaptation committed)

Architectural Modeling

- Approach requires explicit modeling of *relevant* aspects of the software system architecture at design time
- Architecture-level aspects to be modeled
  - Components (interfaces, behavior) and assembly
  - Deployment environment (available nodes and resources)
  - Component deployment (assignment of components to deployment environments)
  - (Performance) SLAs/SLOs
  - Adaptation constraints & policies
  - Adaptation costs (monetary and/or time)
- During runtime, (parts of) these models are refined, kept synchronized and used for the analyses

Summary & Related Work

Summary:

- Problem:
  Overprovisioning capacity management is cost/resource-inefficient
- Goal:
  QoS-aware reduction of operating costs (e.g., power consumption)
- Approach:
  Self-adaptive, architecture-based runtime capacity management

Related Work:

- Capacity Planning (e.g., [MA00, MA02, MAD04])
- Software performance prediction (e.g., [SG98, SW02, BKR09])
- Autonomic QoS management (e.g., [MBR05, NKJT08])
- Self-* software architectures (e.g., [OMT98, KM07, OMT08])

Current & Future Work

1. Confirmation of assumptions
   - Varying workload and resource utilization ▸ ongoing field studies
   - Analysis of potential cost savings ▸ power consumption
2. Specification of adaptation operations
3. Specification of modeling formalisms and runtime models
4. Adaptation framework (instantiation)
   - **Focus:** Development of adaptation analysis activity
     - Runtime performance prediction using performance models
     - Selection of adaptation operations (adaptation plans)
     - Updates to runtime models
   - Proof-of-concept implementation
5. Evaluation (simulation + lab study + field study) ▸ Methodology
   - Does the approach improve resource efficiency?
   - Is it applicable to realistic scenarios?

References I

Martin F. Arlitt, Diwakar Krishnamurthy, and Jerry Rolia.
Characterizing the scalability of a large web-based shopping system.
ACM Transactions on Internet Technology, 1(1):44–69, 2001.

Steffen Becker, Heiko Koziolek, and Ralf Reussner.
The Palladio Component Model for model-driven performance prediction.
Journal of Systems and Software, 82(1):3–22, 2009.
Special Issue: Software Performance - Modeling and Analysis.

Sven Bunge.
Transparent redeployment in component-based software systems (in German), December 2008.
Diploma Thesis, University of Oldenburg.

Mark E. Crovella and Azer Bestavros.
Self-similarity in World Wide Web traffic: evidence and possible causes.
IEEE /ACM Transactions on Networking, 5(6):835–846, 1997.

Jeff Kramer and Jeff Magee.
Self-managed systems: an architectural challenge.
In *2007 Future of Software Engineering (FOSE '07)*, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society.

Daniel A. Menascé and Virgilio A. F. Almeida.
Scaling for E Business: Technologies, Models, Performance, and Capacity Planning.
New Jersey: Prentice Hall, 2000.

Daniel A. Menascé and Virgilio A.F. Almeida.
Capacity Planning for Web Services: Metrics, Models, and Methods.
New Jersey: Prentice Hall, 2002.

References II

Daniel A. Menascé, Virgilio A. F. Almeida, and Lawrence W. Dowdy.
Performance by Design: Computer Capacity Planning By Example.
New Jersey: Prentice Hall, Upper Saddle River, NJ, USA, 2004.

Daniel A. Menascé, Mohamed N. Bennani, and Honglei Ruan.
On the use of online analytic performance models, in self-managing and self-organizing computer systems.
In Özalp Babaoglu, Márk Jelasity, Alberto Montresor, Christof Fetzer, Stefano Leonardi, Aad P. A. van Moorsel, and Maarten van Steen, editors, *Self-star Properties in Complex Information Systems*, volume 3460 of *Lecture Notes in Computer Science*, pages 128–142. Springer, 2005.

Ramon Nou, Samuel Kounev, Ferran Julia, and Jordi Torres.
Autonomic QoS control in enterprise Grid environments using online simulation.
Journal of Systems and Software, 2008.
To appear.

Peyman Oreizy, Nenad Medvidovic, and Richard N. Taylor.
Architecture-based runtime software evolution.
In *Proceedings of the 20th international conference on Software engineering (ICSE '98)*, pages 177–186. IEEE, 1998.

Peyman Oreizy, Nenad Medvidovic, and Richard N. Taylor.
Runtime software adaptation: framework, approaches, and styles.
In *ICSE Companion '08: Companion of the 30th international conference on Software engineering*, pages 899–910. ACM, 2008.

Matthias Rohr, André van Hoorn, Jasminka Matevska, Nils Sommer, Lena Stoever, Simon Giesecke, and Wilhelm Hasselbring.
Kieker: Continuous monitoring and on demand visualization of Java software behavior.
In Claus Pahl, editor, *Proceedings of the IASTED International Conference on Software Engineering 2008 (SE 2008)*, pages 80–85, Anaheim, CA, USA, February 2008. ACTA Press.

References III

Bridget Spitznagel and David Garlan.
Architecture-based performance analysis.
In *Proceedings of the 1998 Conference on Software Engineering and Knowledge Engineering*, pages 146–151, 1998.

Connie U. Smith and Lloyd G. Williams.
Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software.
Amsterdam: Addison-Wesley, 2002.