Microservices for Scalability

Keynote at ICPE 2016, Delft, NL

Prof. Dr. Wilhelm (Willi) Hasselbring

Software Engineering Group, Kiel University, Germany
http://se.informatik.uni-kiel.de/
Competence Cluster Software Systems Engineering
http://kosse-sh.de/

KoSSE
KOMPETENZVERBUND
SOFTWARE SYSTEMS ENGINEERING
SCHLESWIG-HOLSTEIN

C A U
Christian-Albrechts-Universität zu Kiel

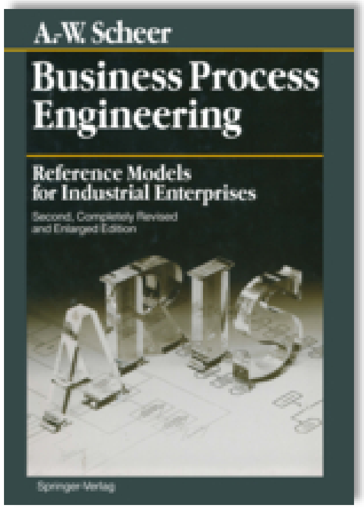
Agenda

1. **Integrated Information Systems**
   - **Including its Limits to Scalability**
2. Information Systems Integration
   - Including its Anti-Patterns to Scalability
3. Microservice Architectures for Scalability
   - Performance and Elasticity
   - Software Development Scalability
4. Takeaways

Integrated Information Systems ?

Why not employing an integrated information system?

Example: ARIS Architecture of Integrated Information Systems



A.-W. Scheer

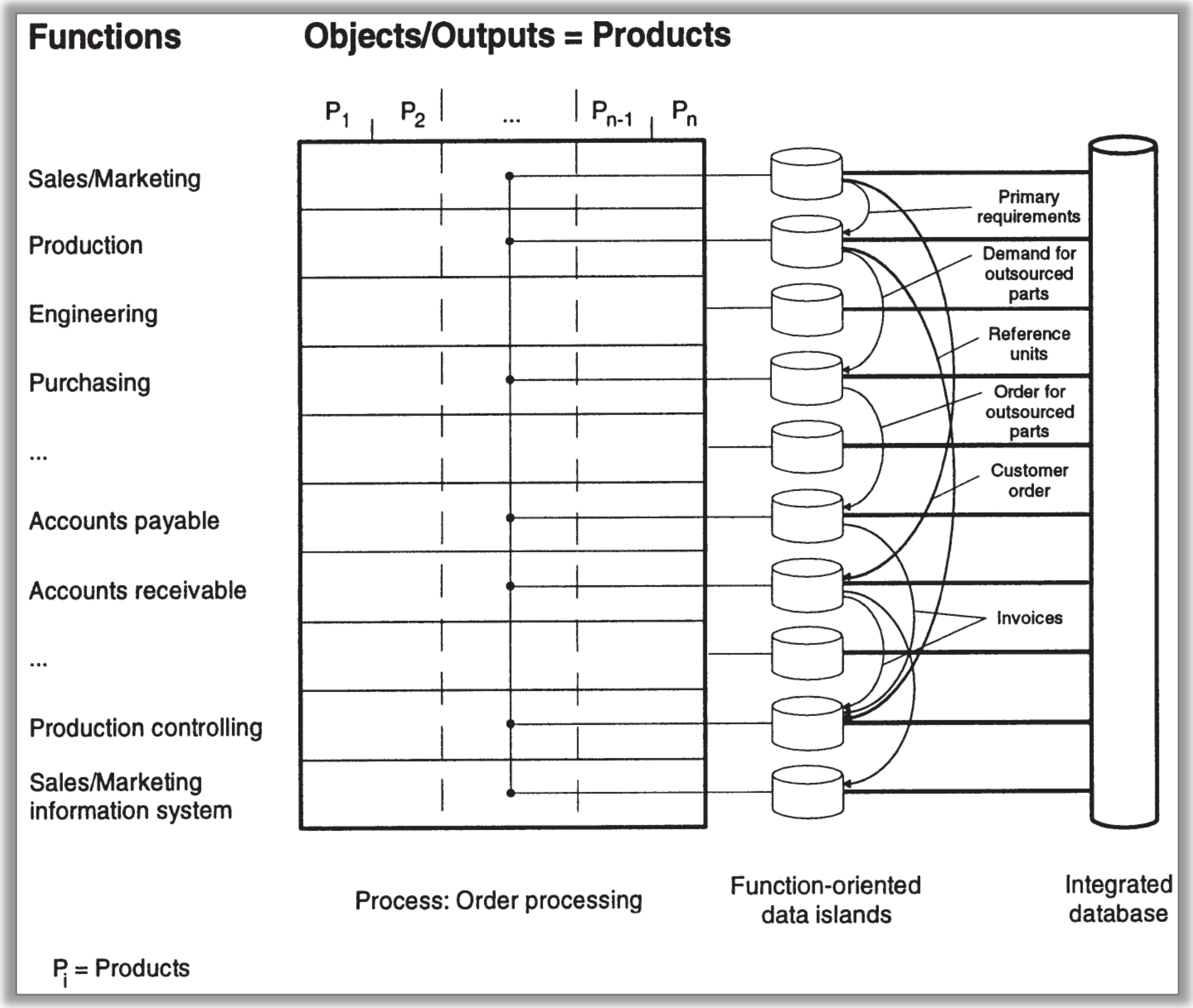
Business Process Engineering

Reference Models for Industrial Enterprises

Second, Completely Revised and Enlarged Edition

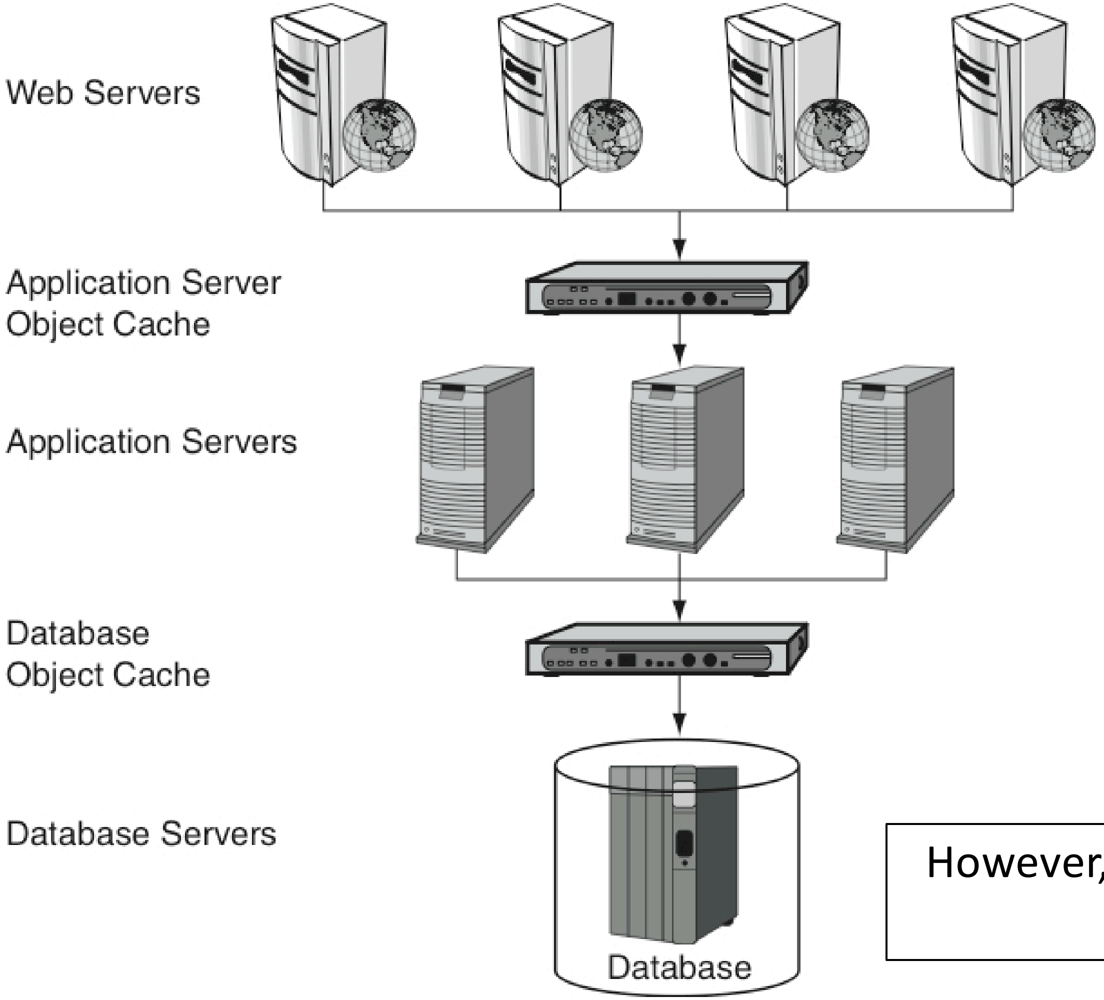
Springer-Verlag

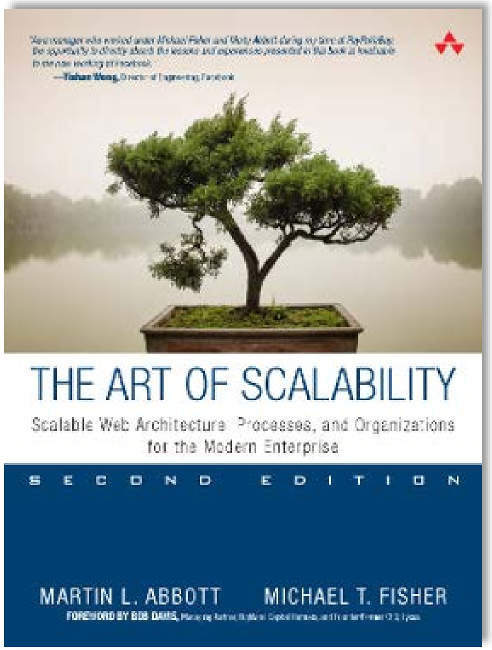
Source: [Scheer 1994]



Web Information Systems
Cache Architecture

Source: [Abbott & Fisher 2015]





Approaches to Scalability on the database layer:

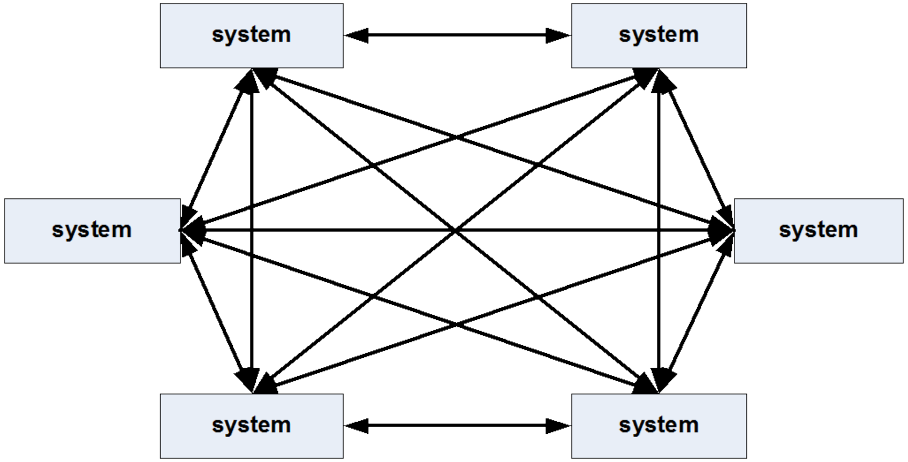
- Big enterprise server
- Database replication
- Database sharding

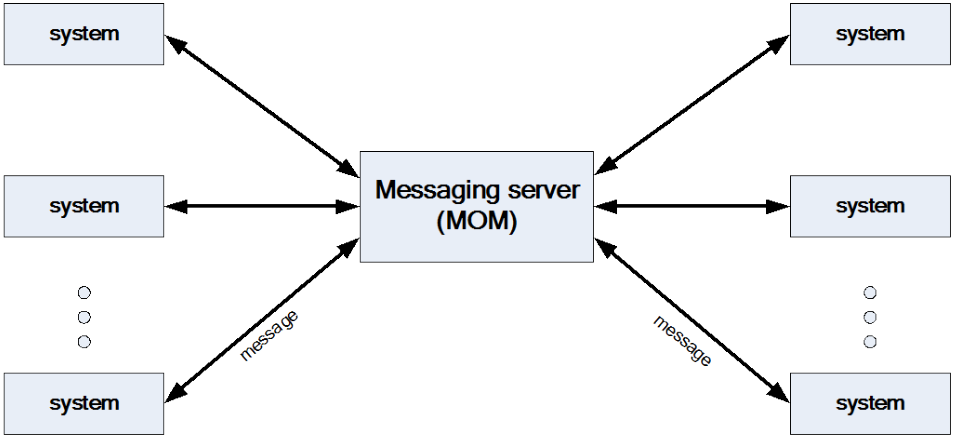
However, you have to scale everything to scale anything!

Agenda

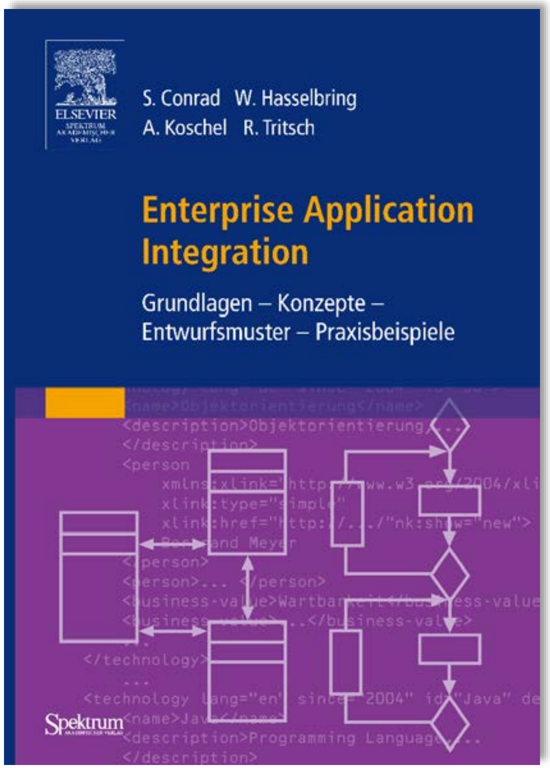
1. Integrated Information Systems
   – Including its Limits to Scalability
2. **Information Systems Integration**
   – **Including its Anti-Patterns to Scalability**
3. Microservice Architectures for Scalability
   – Performance and Elasticity
   – Software Development Scalability
4. Takeaways

Information Systems Integration ?





Source: [Conrad et al. 2005]



ELSEVIER SPEKTRUM AKADEMISCHER VERLAG

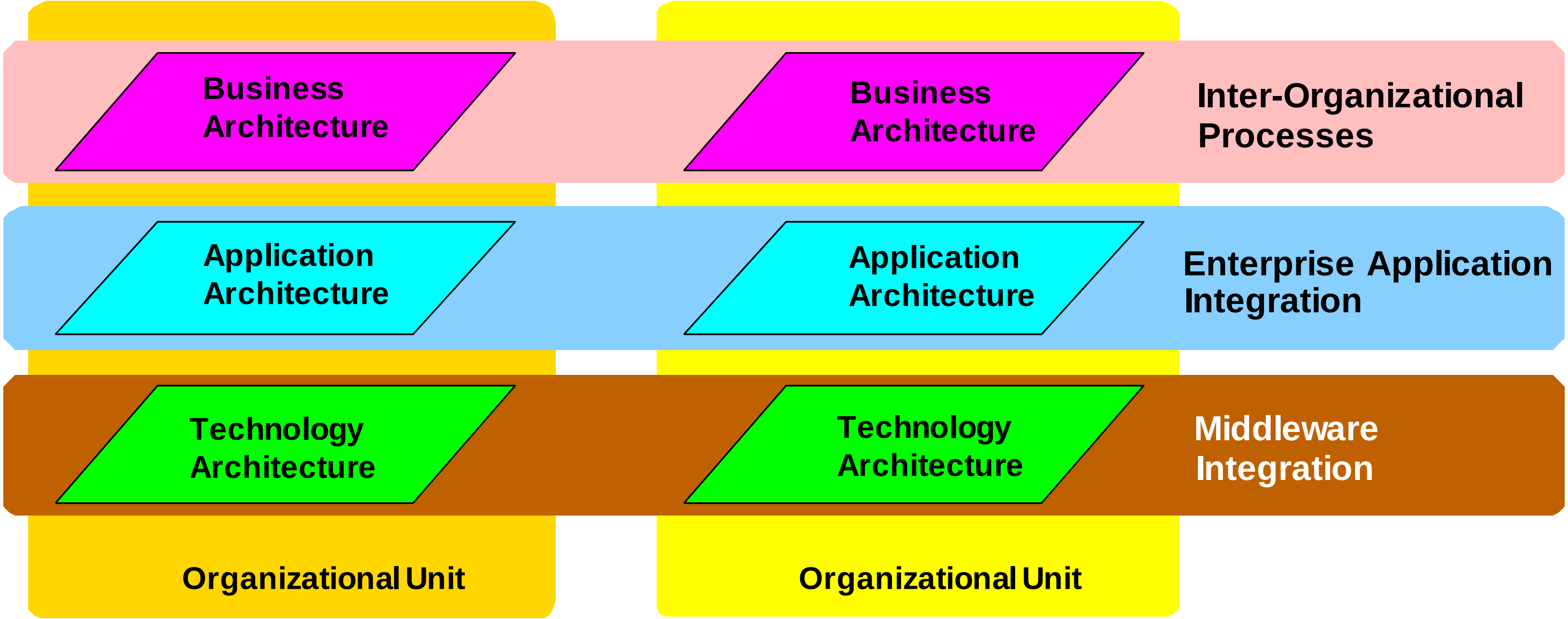
S. Conrad W. Hasselbring
A. Koschel R. Tritsch

Enterprise Application Integration

Grundlagen – Konzepte –
Entwurfsmuster – Praxisbeispiele

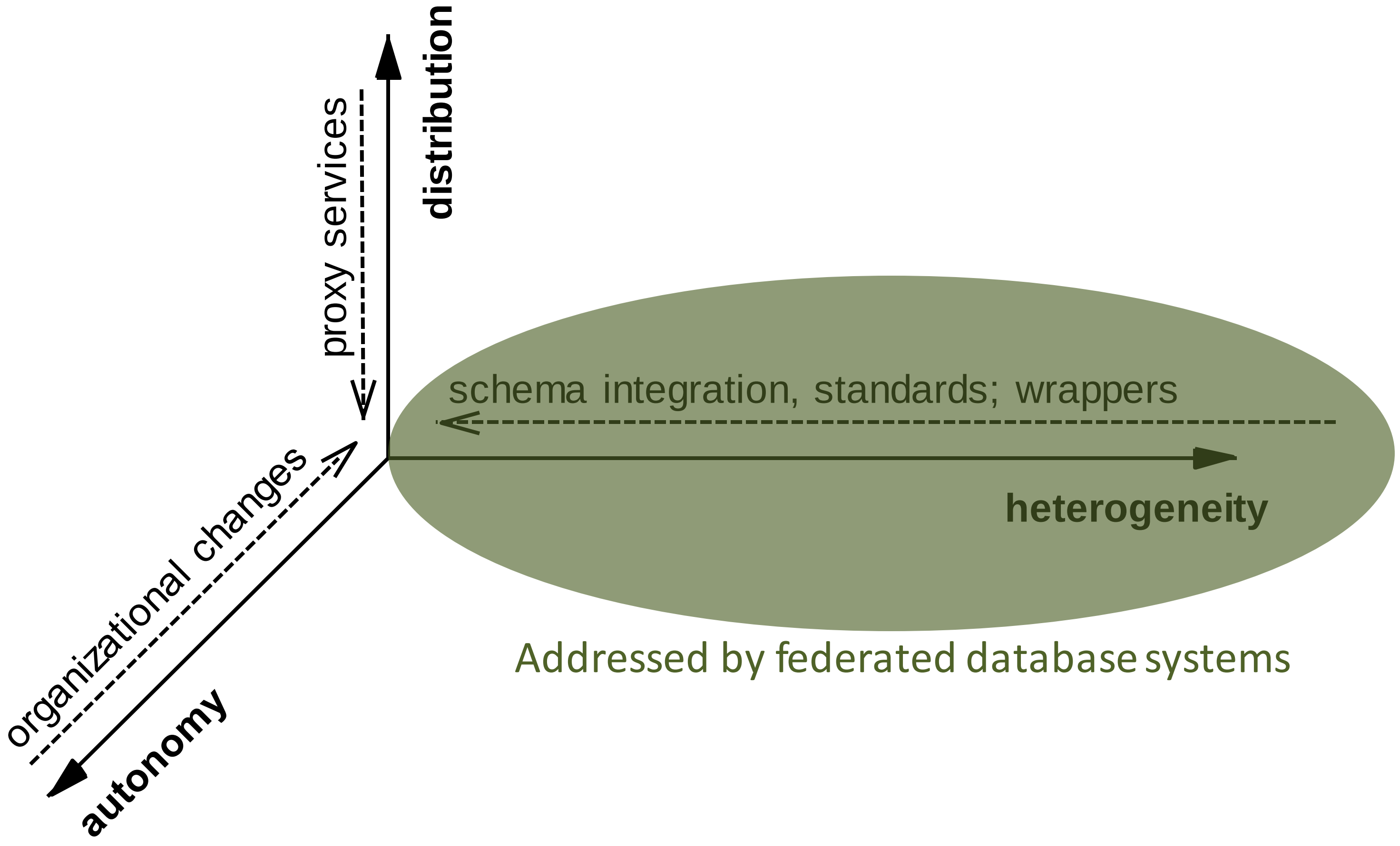
Spektrum

Architecture and Integration Layers for Business Information Systems



Source: [Hasselbring 2000]

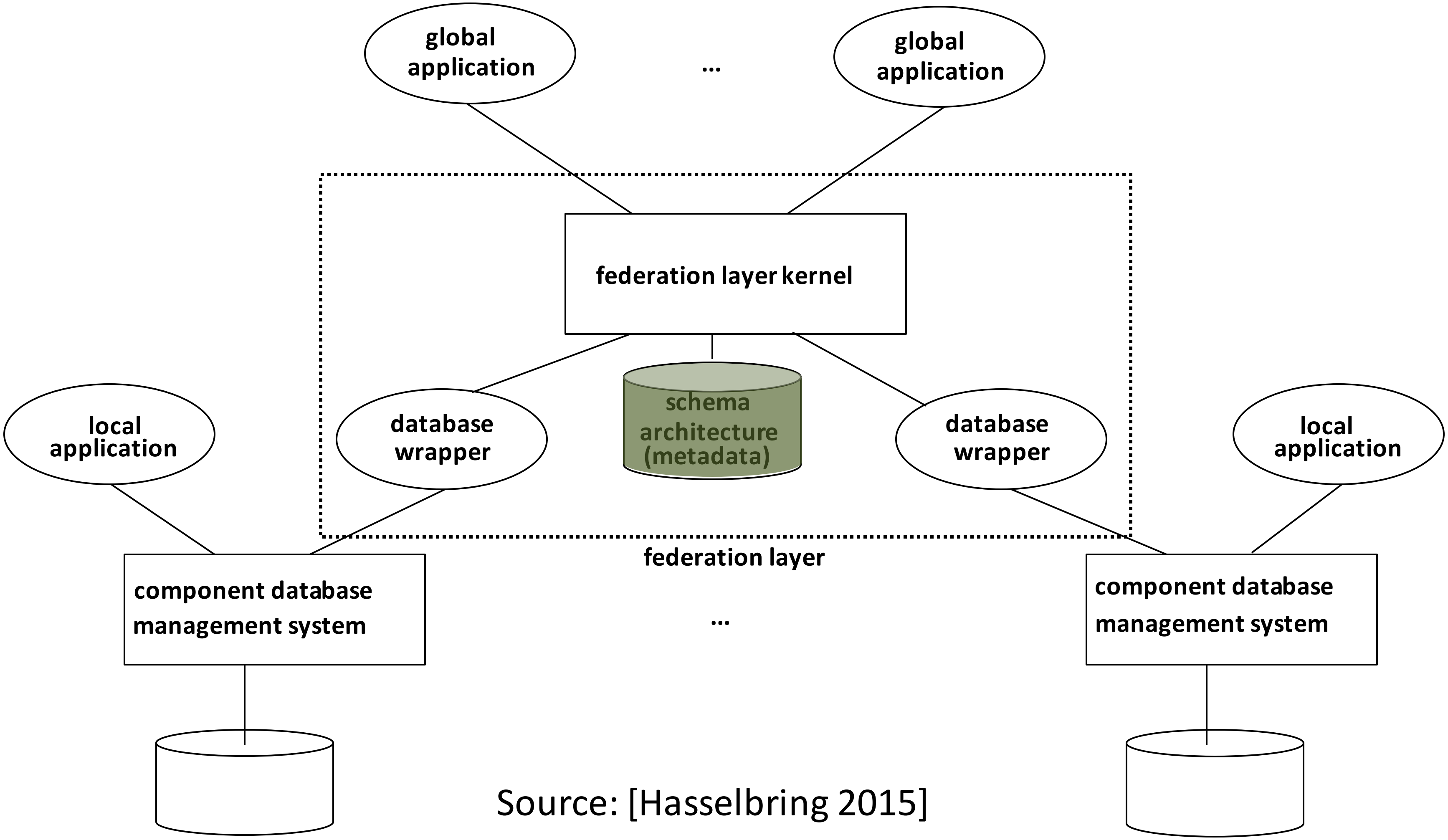
Integration Dimensions



Addressed by federated database systems

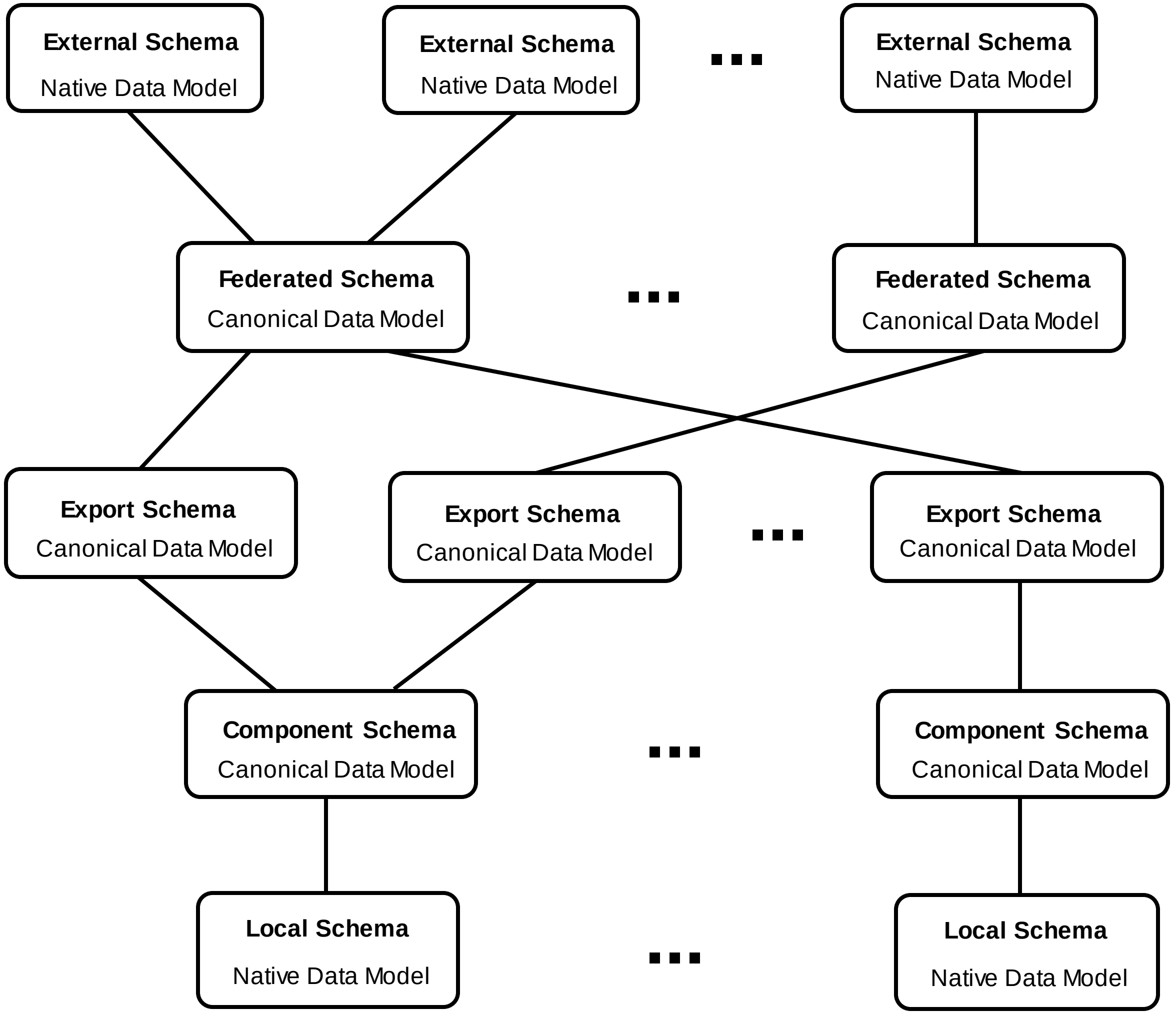
Source: [Hasselbring 2000]

General System Architecture of Federated Database Systems



Source: [Hasselbring 2015]

Five-level schema architecture for federated database systems



Source:
[Sheth & Larson 1990, Hasselbring 2015]

Result:
Tight coupling between integrated databases!

Some Anti-Patterns to Scalability of Information Systems

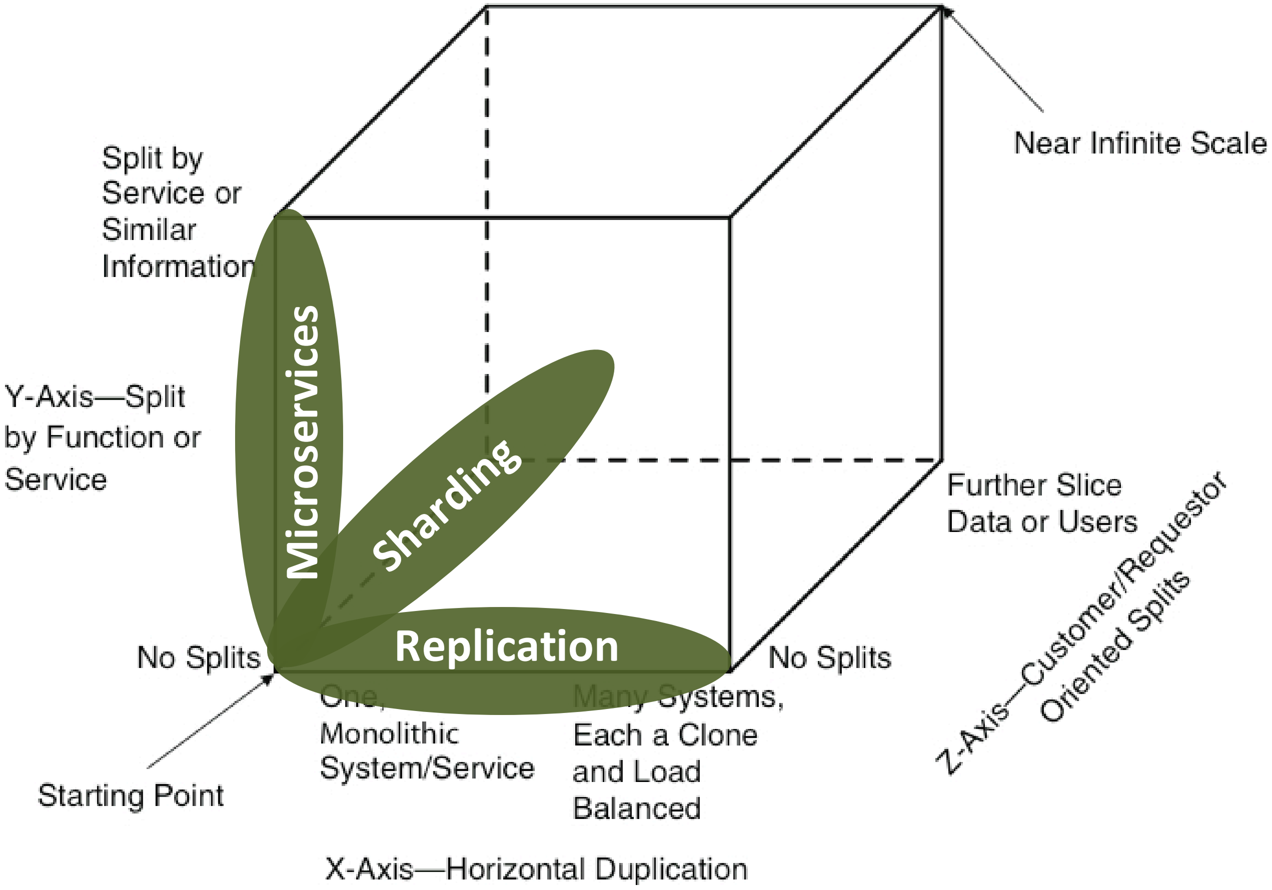
1. One central database
2. Distributed transactions
3. Schema-based integration
4. Limited capacity
5. Shared code

Not meant to be exhaustive,
but discussed in this talk.

Agenda

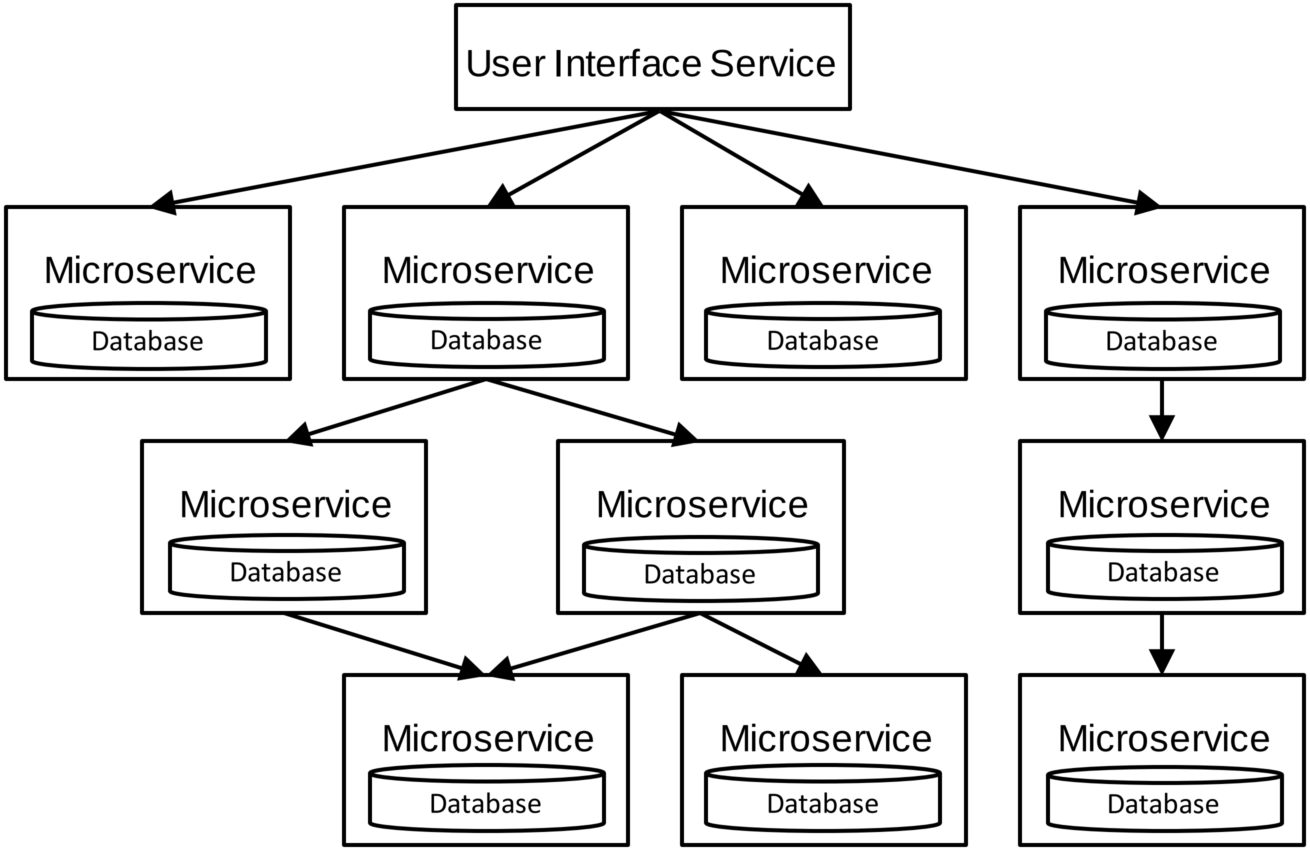
1. Integrated Information Systems
   - Including its Limits to Scalability
2. Information Systems Integration
   - Including its Anti-Patterns to Scalability
3. **Microservice Architectures for Scalability**
   - **Performance and Elasticity**
   - Software Development Scalability
4. Takeaways

The Scale Cube


Near Infinite Scale
Split by Service or Similar Information
Y-Axis—Split by Function or Service
Microservices
Sharding
Replication
Further Slice Data or Users
Z-Axis—Customer/Requestor Oriented Splits
No Splits
No Splits
One, Monolithic System/Service
Many Systems, Each a Clone and Load Balanced
Starting Point
X-Axis—Horizontal Duplication

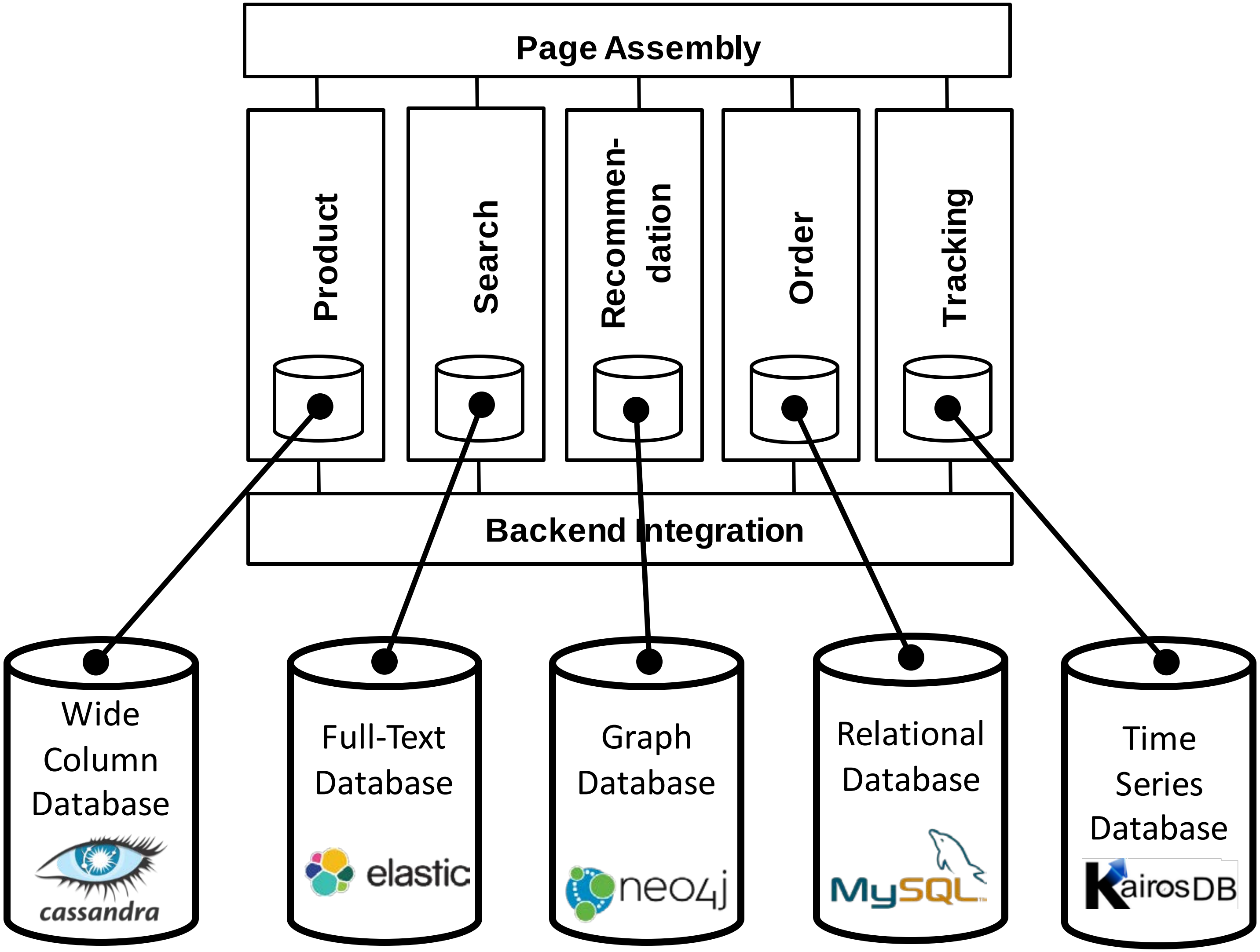
Based on [Abbott & Fisher 2015]

Y-Axis Scaling via Independently Deployable Microservices

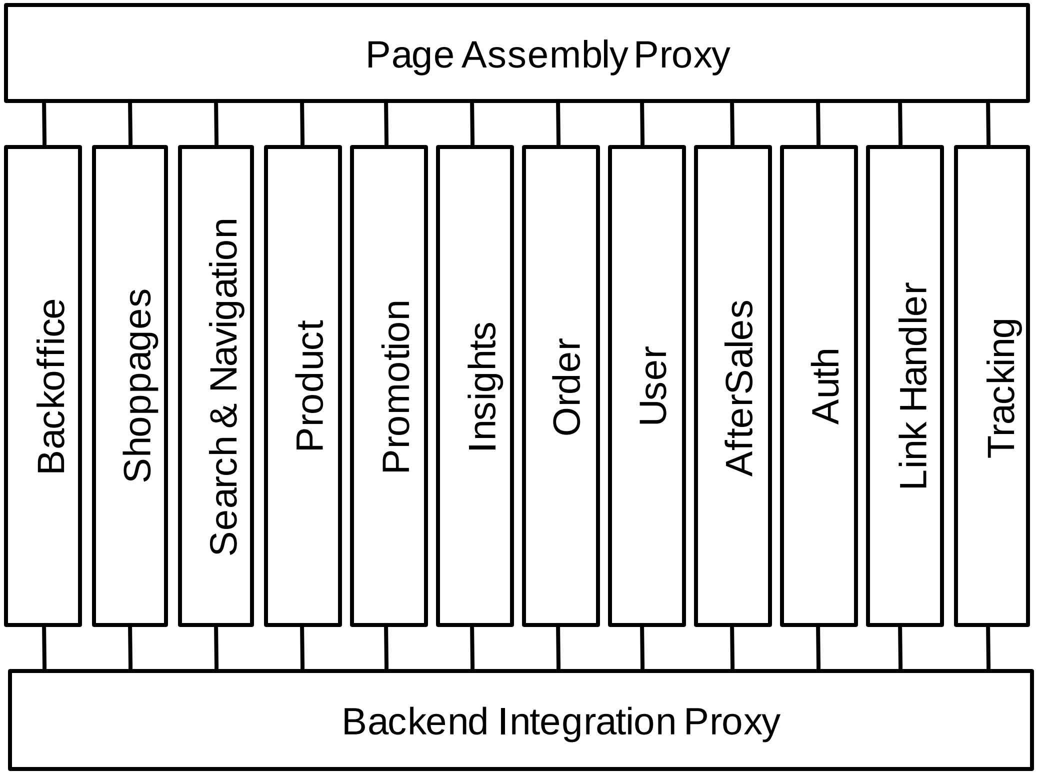


Based on [Bas et al. 2015].

Polyglot Persistence

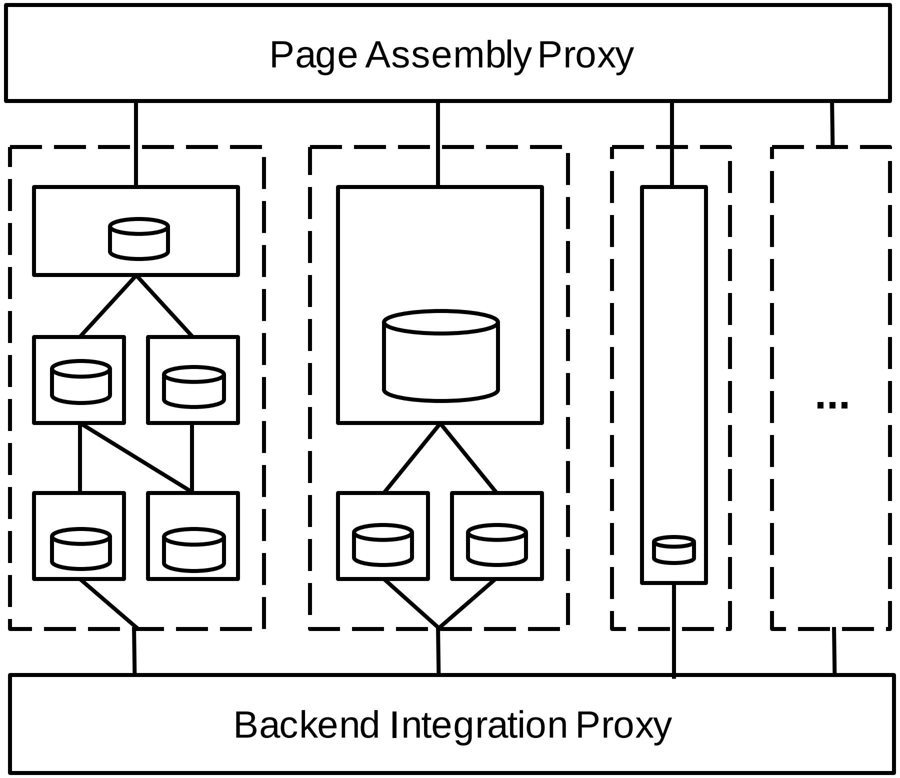


Verticals for Business Functions Example: otto.de



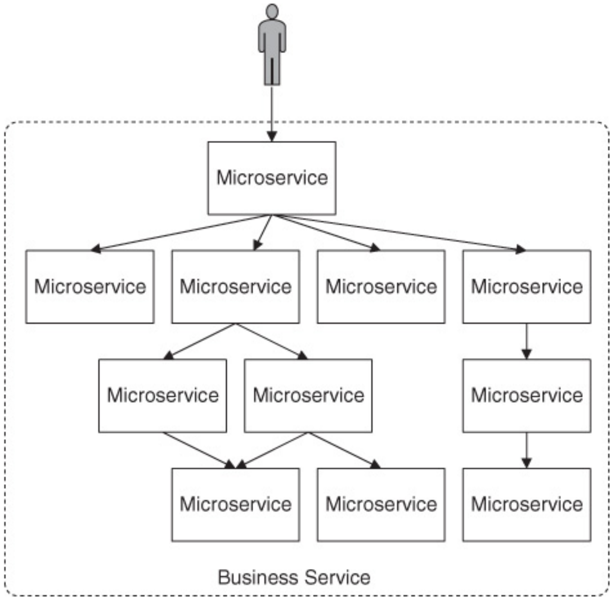
Based on [Kraus et al. 2013 Steinacker 2014]

Verticals and Microservices

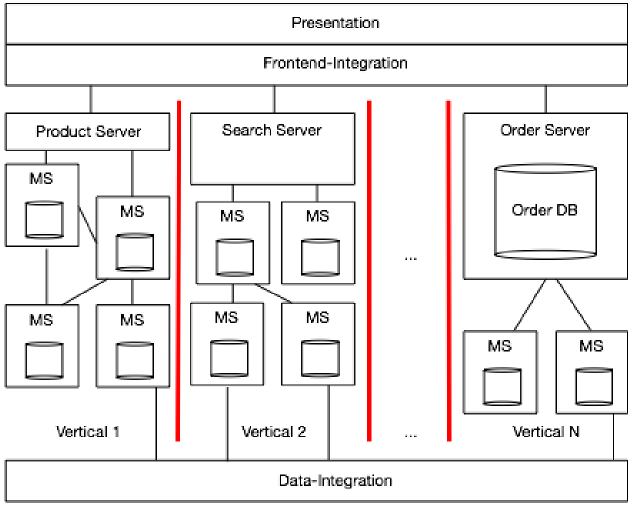


Based on [Steinacker 2014]

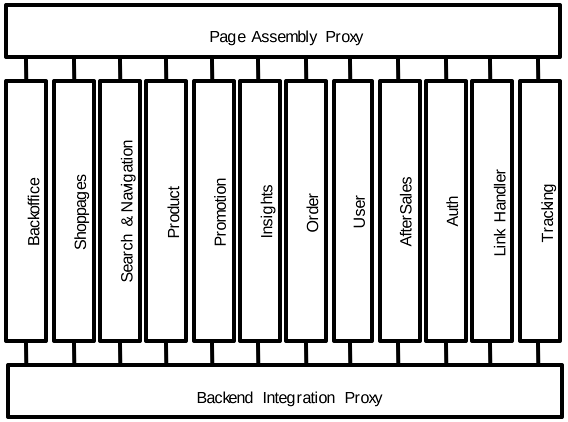
Microservice Architecture Variations



[Bas et al. 2015]



[Steinacker 2014]



[Kraus et al. 2013]

"Scalability is managed by each service individually and is included in its SLA in the form of a guaranteed response time given a particular load."

[Bas et al. 2015, Chapter 4]

"The trade-off between many small components and a few large components must be considered in component and system design."

[Hasselbring 2002]

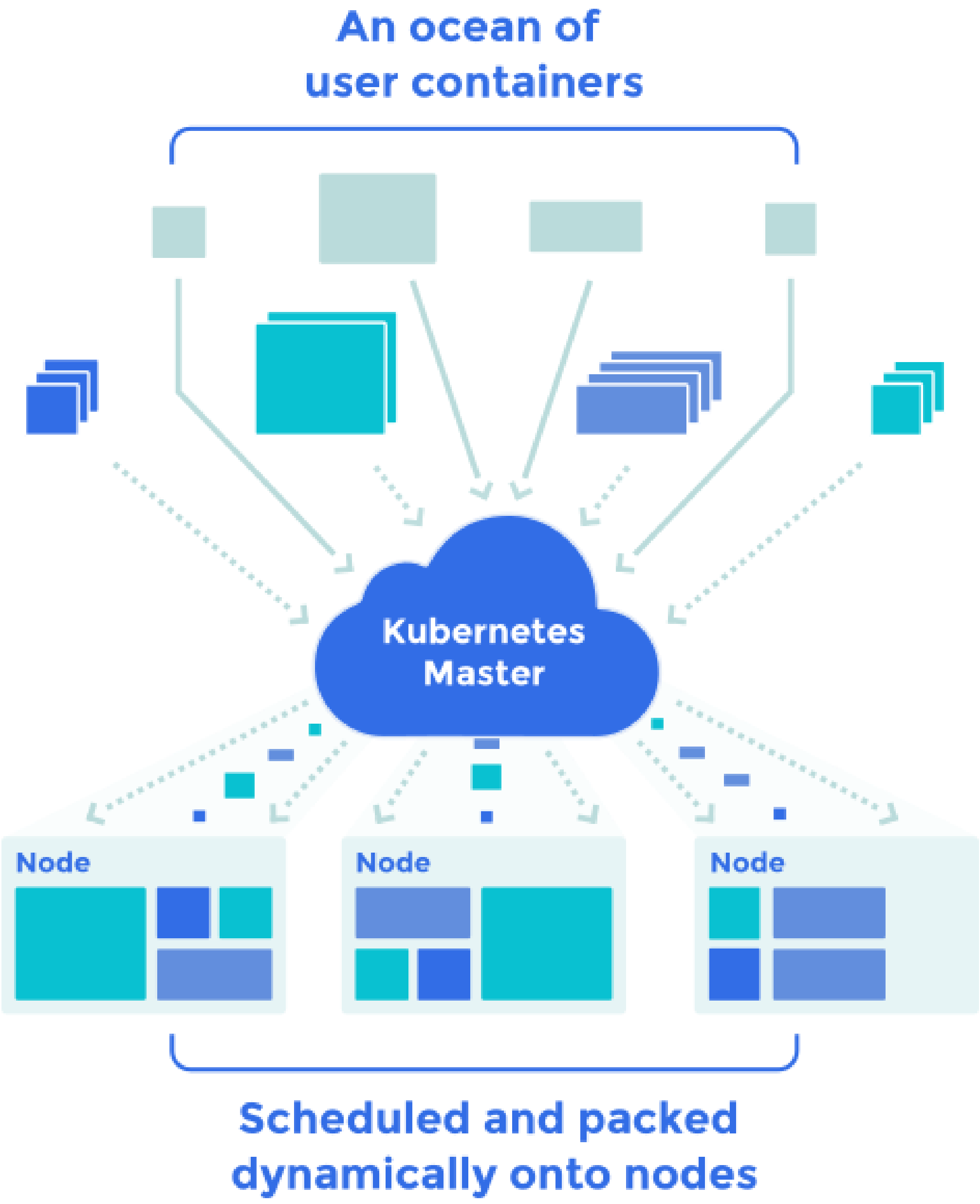
Vertical and Horizontal Scalability

There are two primary approaches to scaling:

- Vertical scaling is also known as scaling up, which means to
  - increase the overall application capacity of individual nodes through hardware improvements, e.g., change to other nodes with higher memory, or increase the number of CPU cores.
- Horizontal scaling is also called scaling out, which means to
  - increase the overall application capacity by adding more nodes, each additional node typically has the equivalent capacity, such as the same amount of memory and the same CPU.

  → **Elasticity required**

Manage a cluster of containers for horizontal scalability


An ocean of user containers

Kubernetes Master

Node

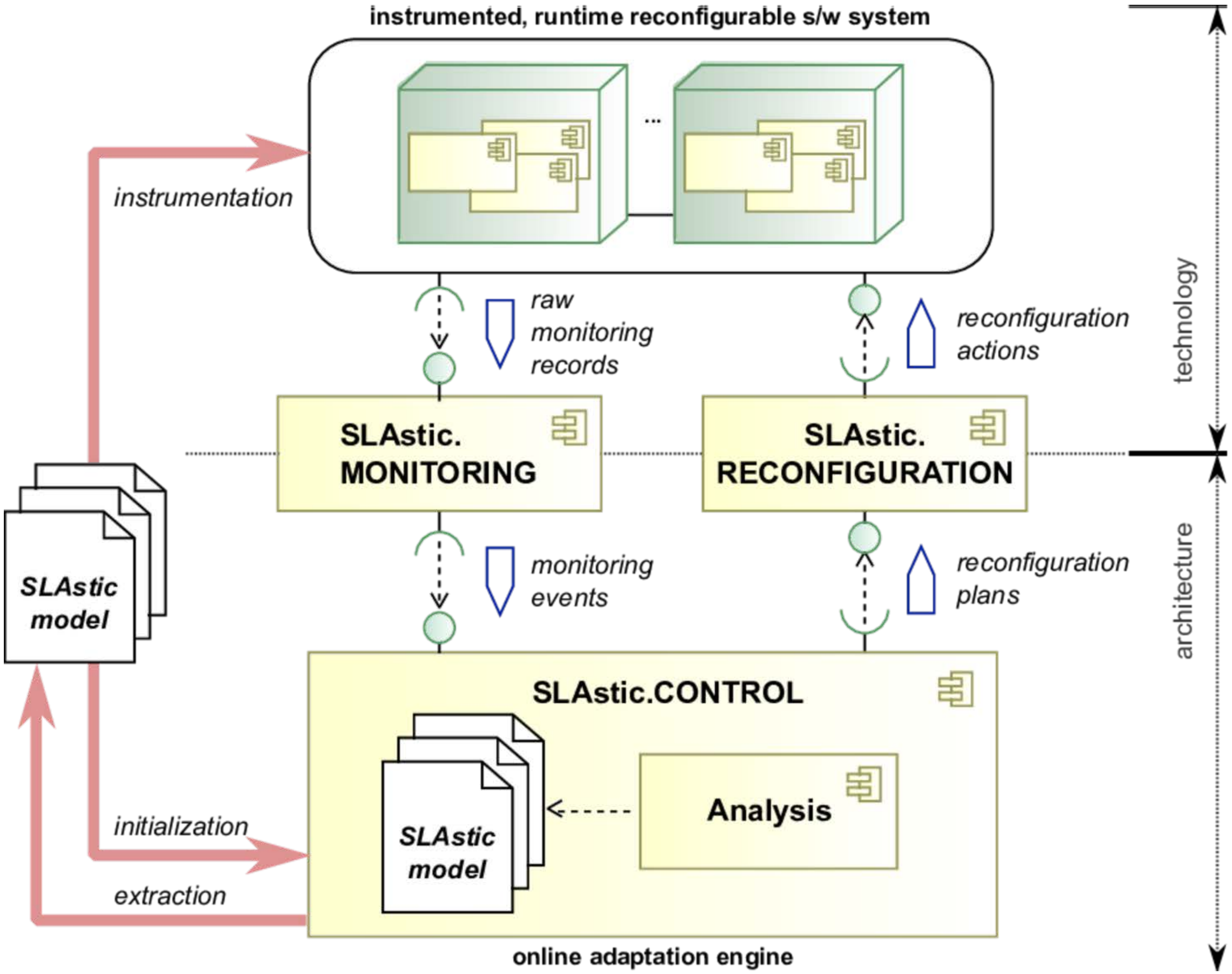
Node

Node

Scheduled and packed dynamically onto nodes

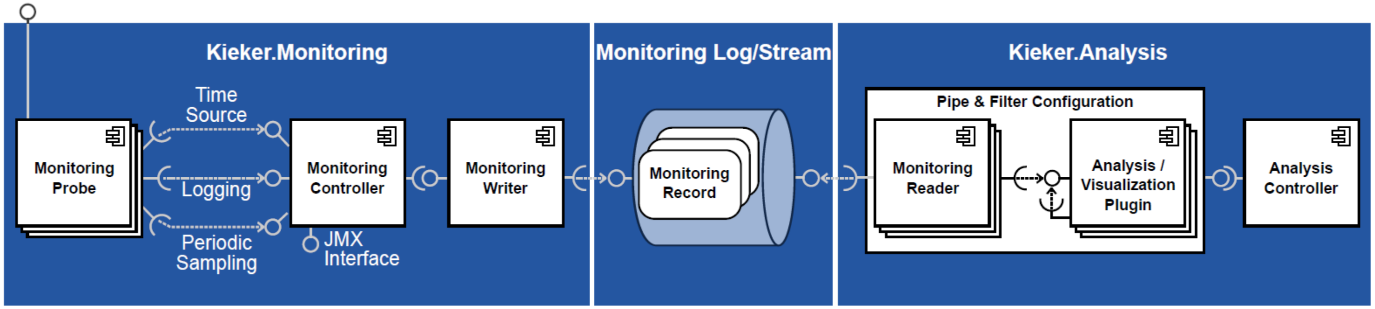
http://kubernetes.io/

SLAstic: Online Capacity Management



[van Hoorn et al. 2009, van Hoorn 2014]

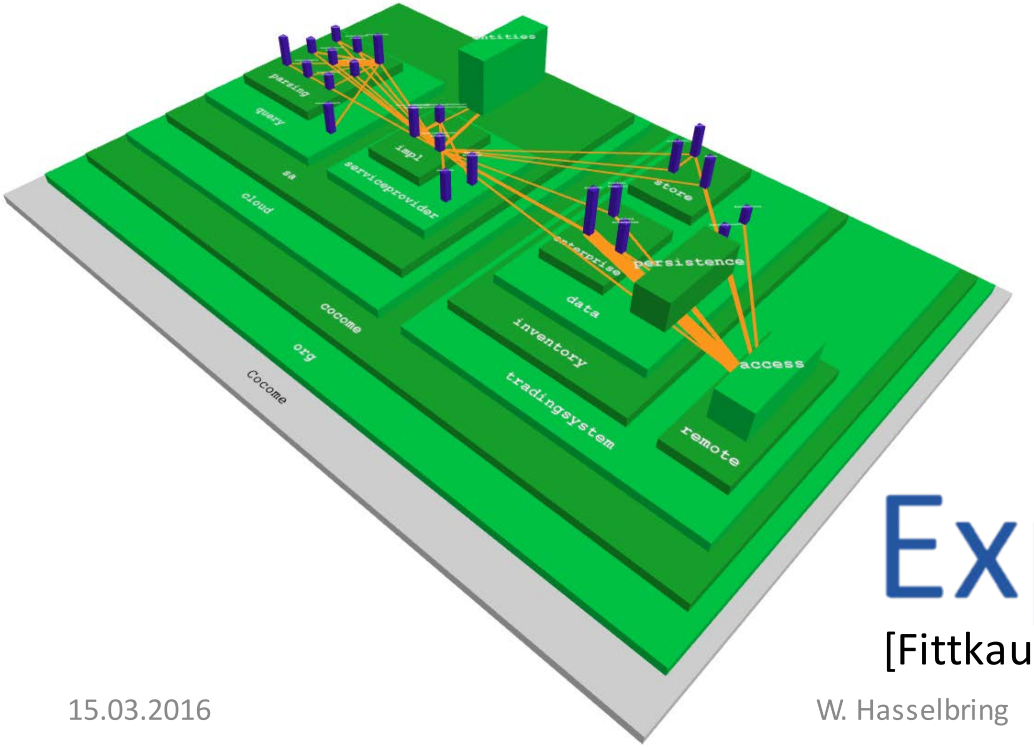
Essential in this Context: Continuous Monitoring



Kieker

[van Hoorn et al. 2012]

spec® Research

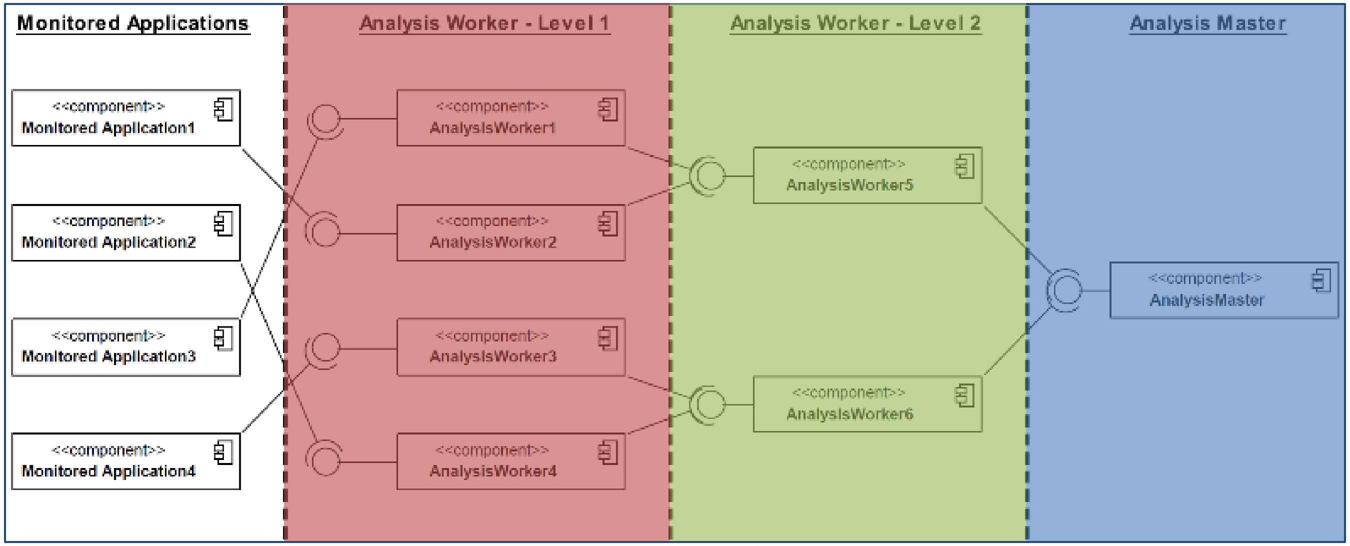


ExplorViz

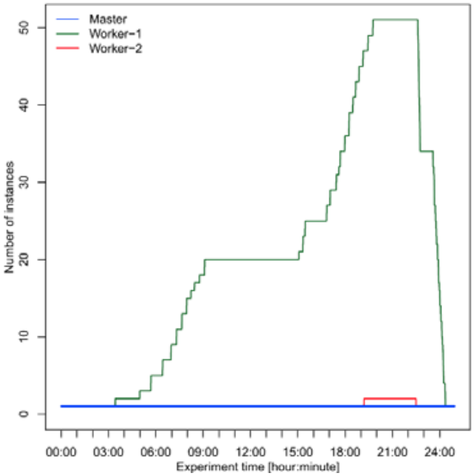
[Fittkau et al. 2013, 2015a]

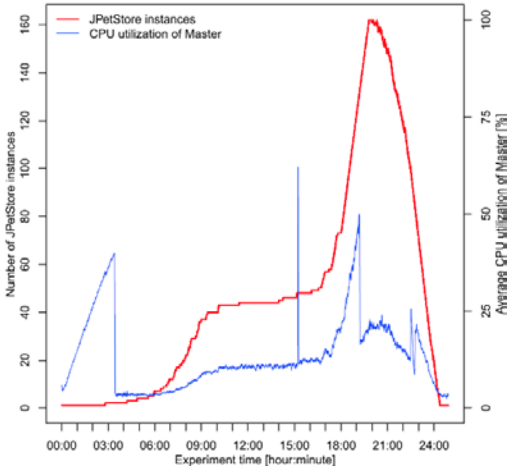
Monitoring for Online Capacity Management
But also Scalable Monitoring Trace Processing

ExplorViz



Monitored Applications
Analysis Worker - Level 1
Analysis Worker - Level 2
Analysis Master
<<component>> Monitored Application1
<<component>> Monitored Application2
<<component>> Monitored Application3
<<component>> Monitored Application4
<<component>> AnalysisWorker1
<<component>> AnalysisWorker2
<<component>> AnalysisWorker3
<<component>> AnalysisWorker4
<<component>> AnalysisWorker5
<<component>> AnalysisWorker6
<<component>> AnalysisMaster



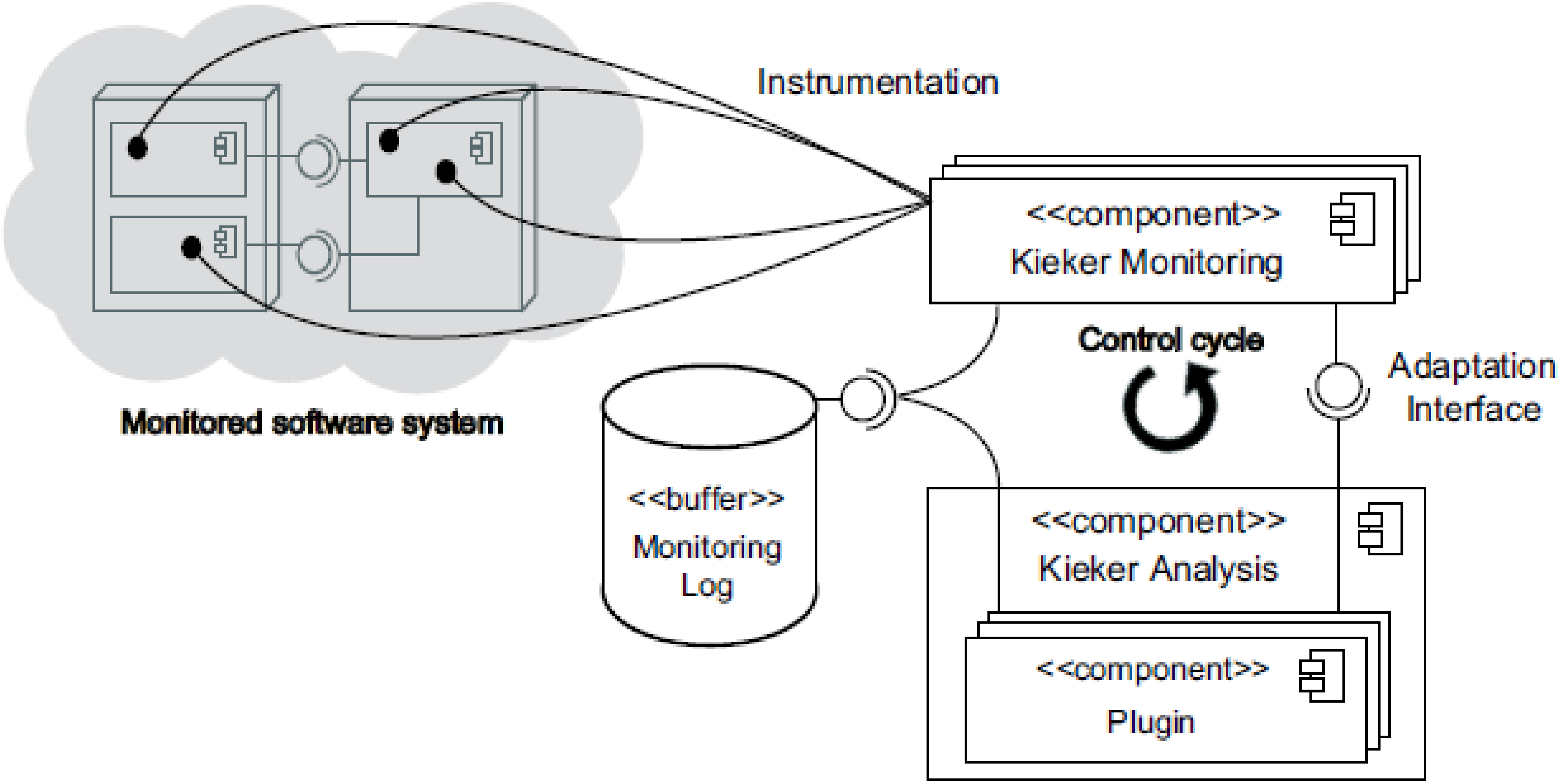


Processing Capabilities:

- ✓ Cost efficient
- ✓ Scalable to millions of monitored methods per second

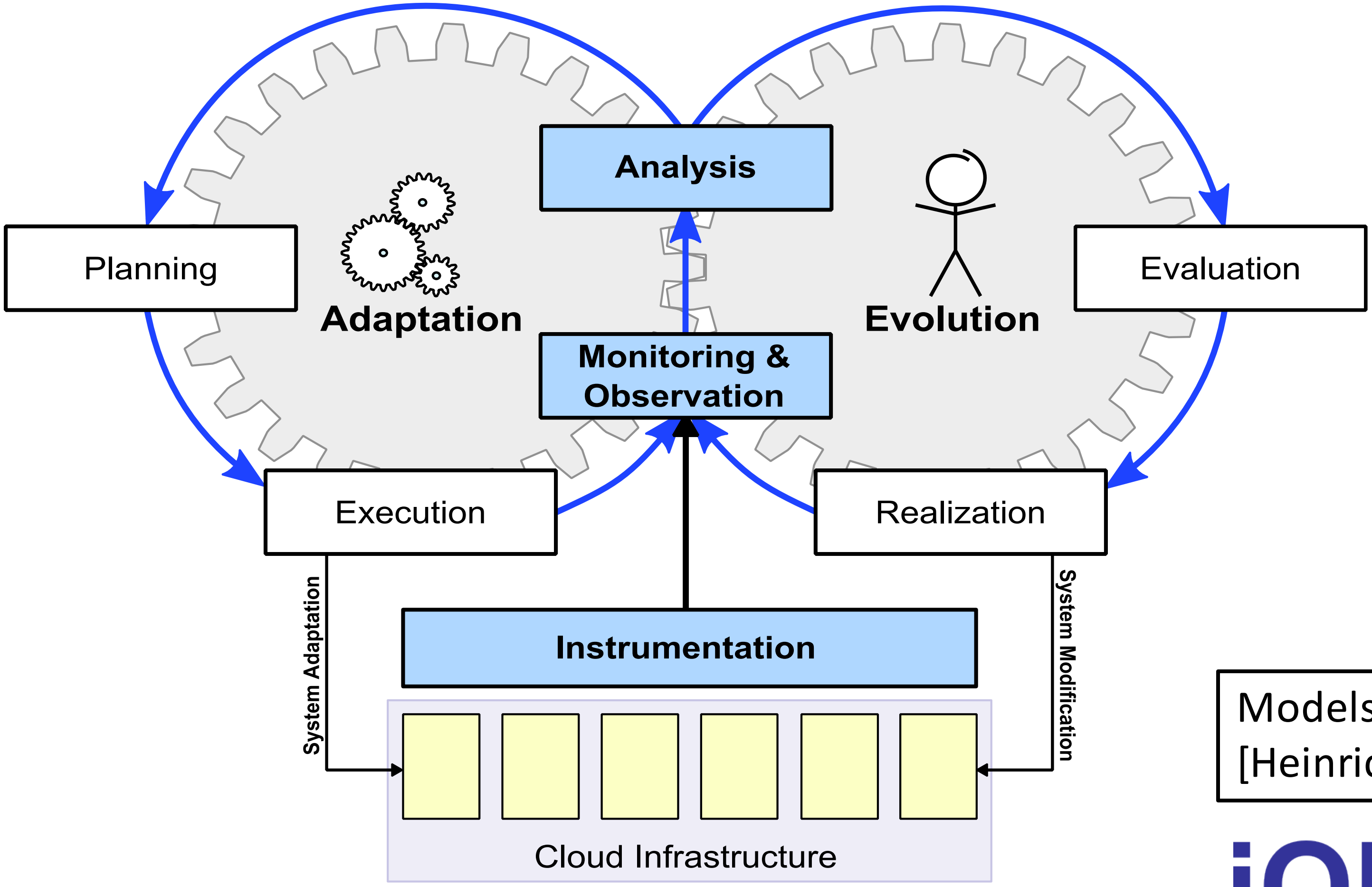
[Fittkau et al. 2015b]

Adaptive Monitoring:
Adjust Instrumentation Coverage at Runtime



Adaptation based on anomaly detection [Marwede et al. 2009, Ehlers et al. 2011]

Integration of Adaptation and Evolution



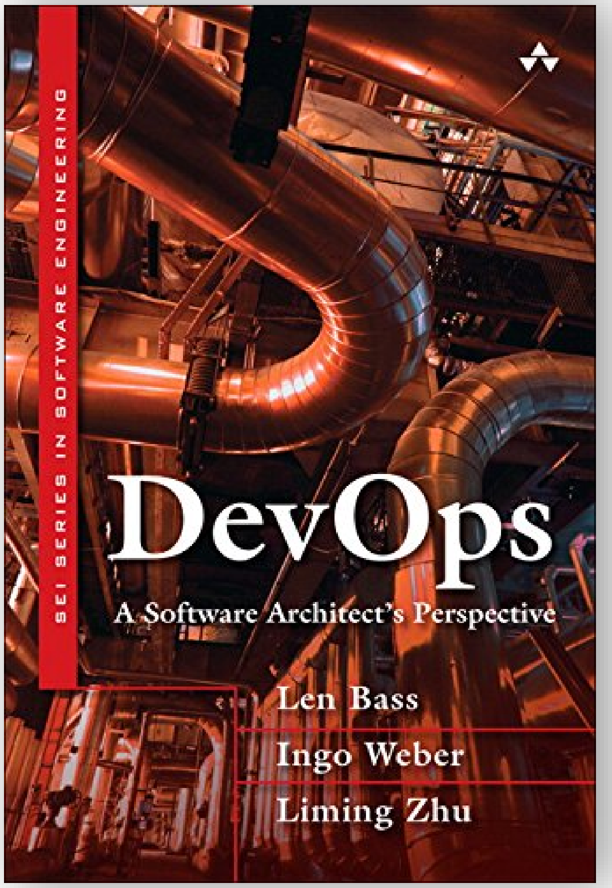
Models @ Runtime
[Heinrich et al. 2014, 2015]

iObserve

Agenda

1. Integrated Information Systems
   – Including its Limits to Scalability
2. Information Systems Integration
   – Including its Anti-Patterns to Scalability
3. Microservice Architectures for Scalability
   – Performance and Elasticity
   – **Software Development Scalability**
4. Takeaways

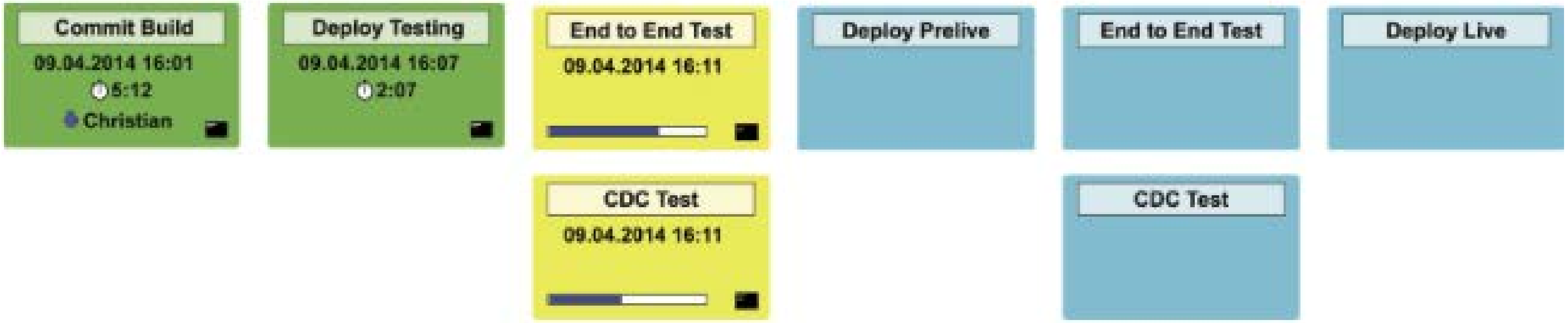
DevOps & Software Architecture


SEI SERIES IN SOFTWARE ENGINEERING
DevOps
A Software Architect's Perspective
Len Bass
Ingo Weber
Liming Zhu

"The **deployment pipeline** is the place where the **architectural** aspects and the process aspects of DevOps intersect."

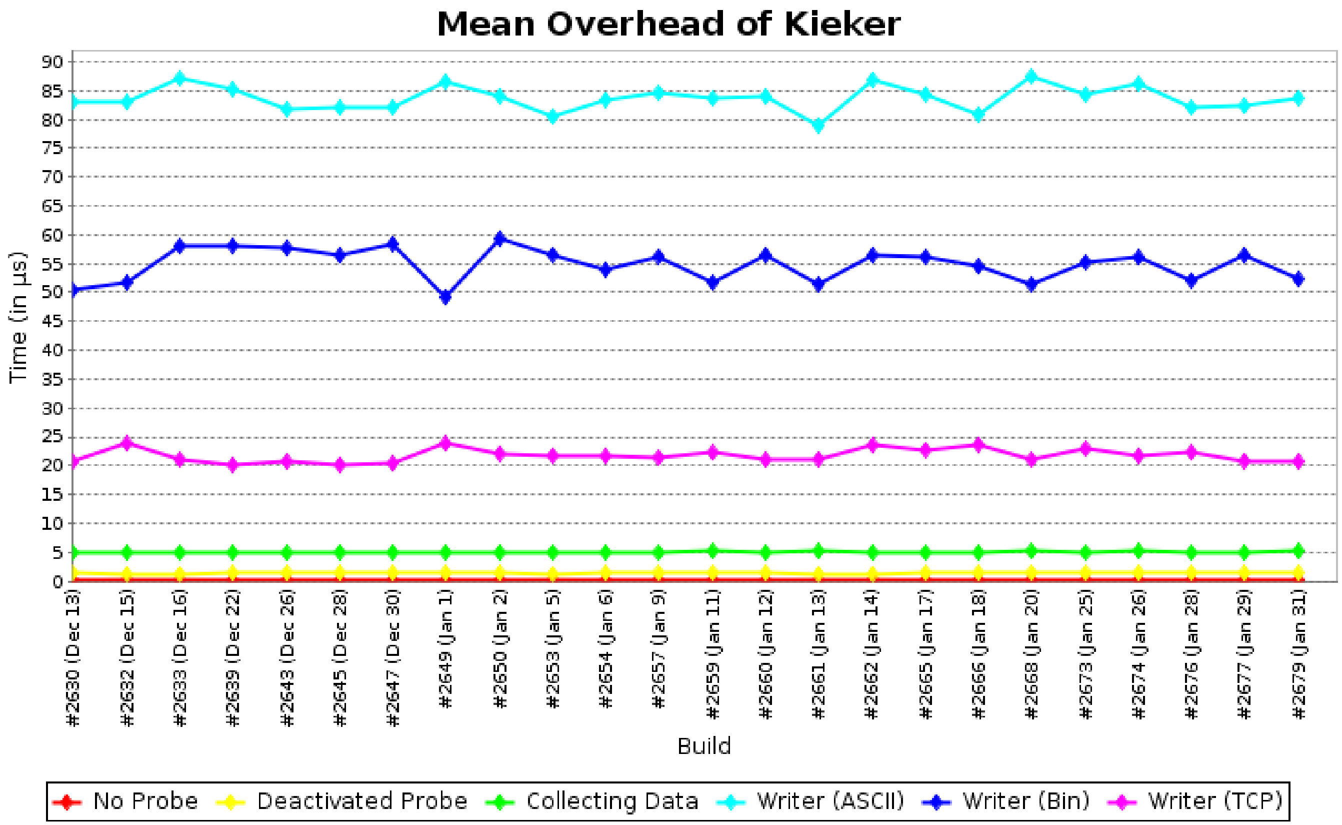
[Bas et al. 2015]

Deployment Pipelines for Continuous Deployment
Example Deployment Pipeline @ Otto.de



Source: [Breetzmann et al. 2014]

Automated Quality Assurance Example: Regression Benchmarking



Integrated into Continuous Integration Setup [Waller et al. 2015]

Should include automated anomaly detection [Marwede et al. 2009, Ehlers et al. 2011]

https://build.se.informatik.uni-kiel.de/jenkins/job/kieker-nightly-release/plot/

Conway’s Law

“The basic thesis of this article is that organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations”

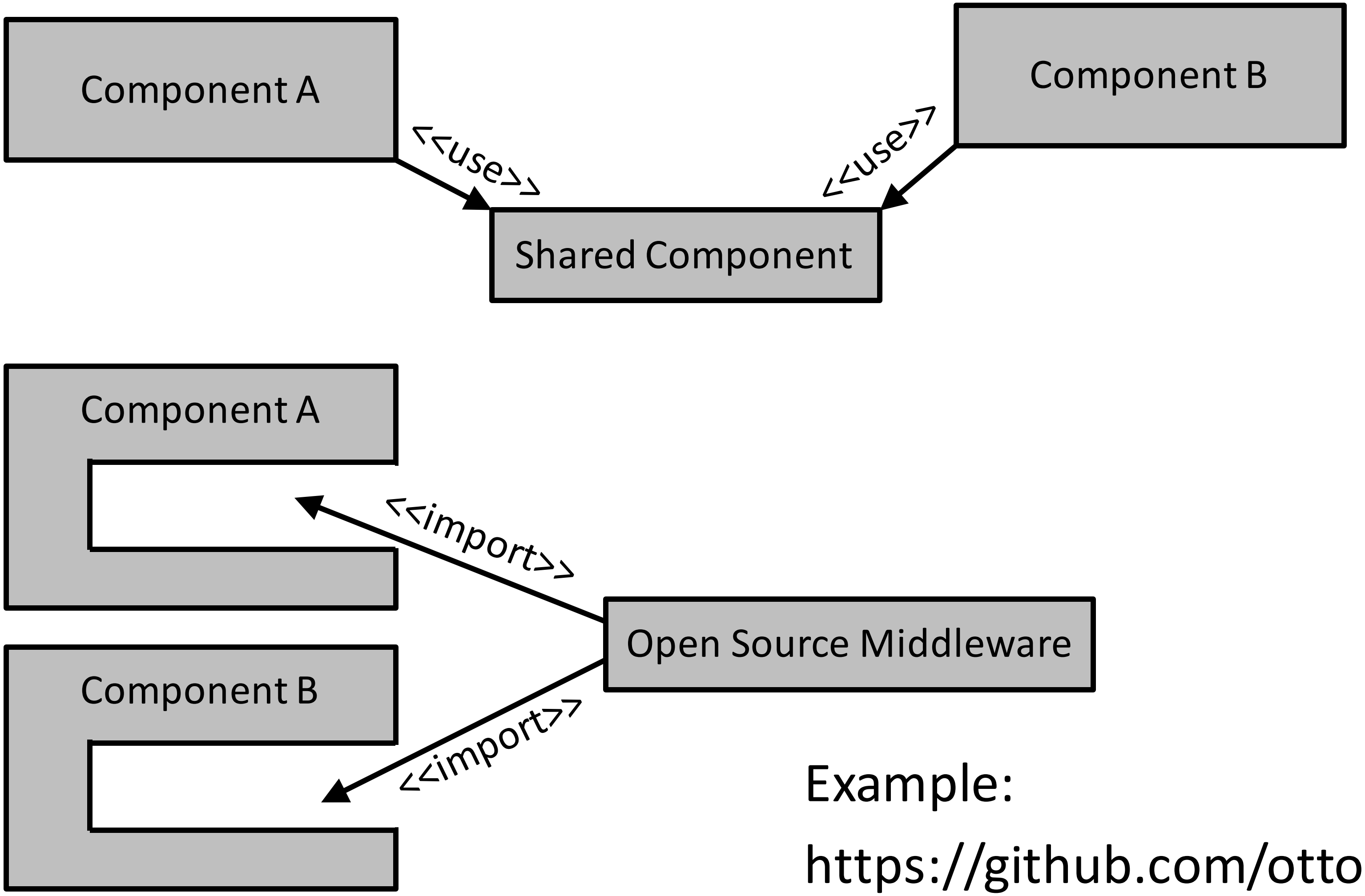
[Conway 1968]

If the organizational structure is decomposed vertically and according to the microservices structure into cross-functional feature teams,

- **scaling** development capacities according to changing business requirements is enabled.
- The **feature teams** should be highly independent, having members of all roles and skills that are required to build and maintain their microservice.

→ Decoupling teams as relevant as decoupling software modules

Component vs. Middleware Reuse



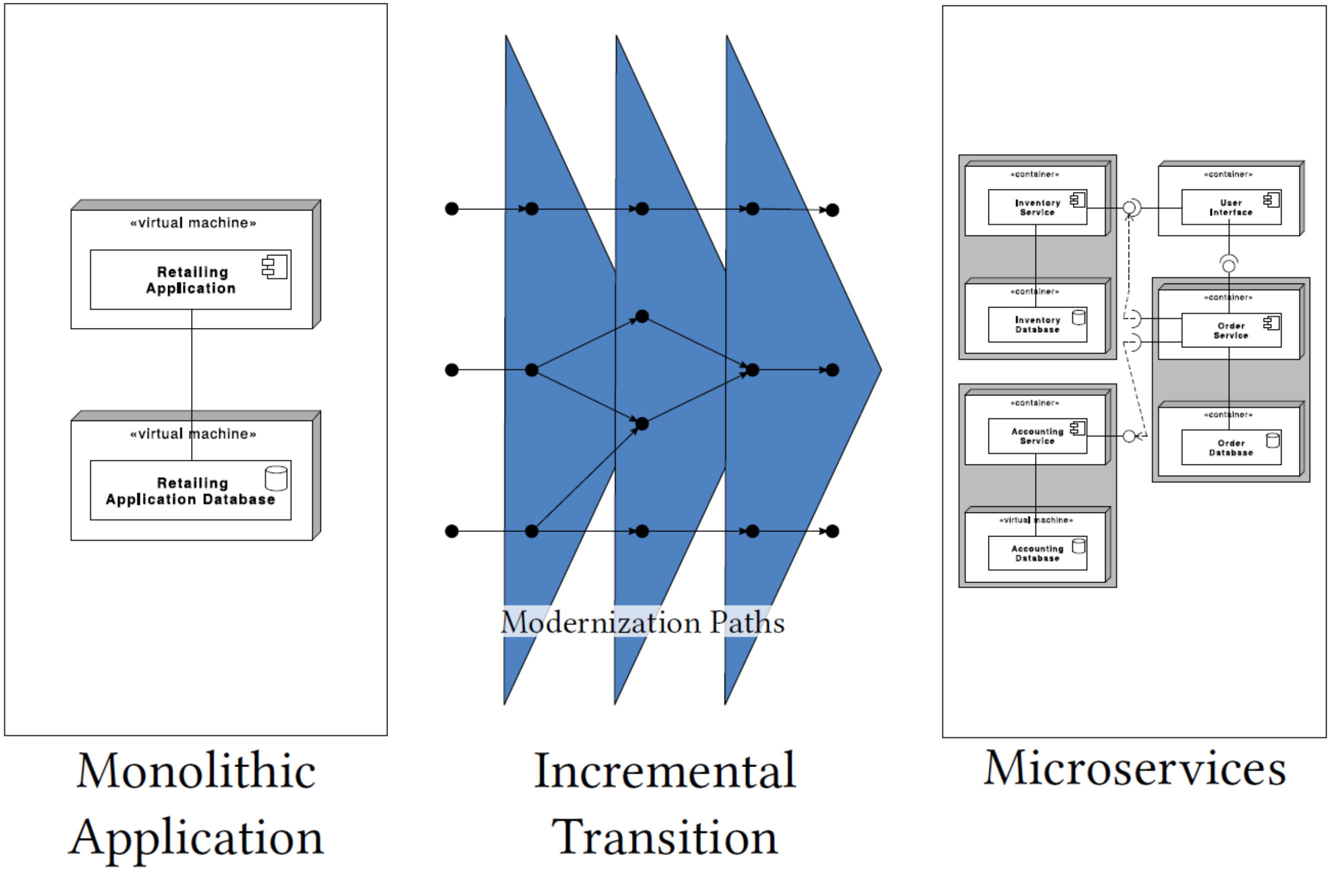
Example:

https://github.com/otto-de/

From Monoliths towards Microservices

Yesterday, at the ICPE 2016 Doctoral Symposium

- *Holger Knoche:* “Sustaining Runtime Performance while Incrementally Modernizing Transactional Monolithic Software towards Microservices”



Anti-Patterns and Solutions to Scalability of Information Systems

1. One central database
   → Polyglott persistence
2. Distributed transactions
   → Eventual consistency
3. Schema-based integration
   → Loose coupling via asynchronous messaging
4. Limited capacity
   → Continuous monitoring for elastic capacity management
5. Shared code
   → Open source frameworks

Microservices offer such solutions.

Scalability for both, runtime performance and development performance (DevOps).

However,
be aware of the imposed costs!

Advertisements

- Softwareforen Leipzig, April 12-13, 2016
  Microservice Architectures and Continuous Delivery
  http://www.softwareforen.de/goto/sar
- DevOpsDays Kiel, May 12-13, 2016
  http://www.devopsdays.org/events/2016-kiel/
- KoSSE Day on DevOps, June 1, 2016
  http://kosse-sh.de/
- Symposium on Software Performance
  November 08–09, 2016 in Kiel
  (Descartes/Kieker/Palladio Days 2016)
  http://www.performance-symposium.org/

References

[Abbott & Fisher 2015] M.L. Abbott, M.T. Fisher: The Art of Scalability. Addison-Wesley, 2nd Edition, 2015.

[Bas et al. 2015] Len Bass, Ingo Weber, Liming Zhu: “DevOps: A Software Architect’s Perspective”, Addison-Wesley 2015.

[Breetzmann et al. 2014] R Breetzmann, S. Kraus, C. Stamm: “Null Toleranz für Fehler: Wie wir auf otto.de die Qualität hoch halten“, OBJEKTspektrum 4/2014, 18-23.

[Conrad et al. 2005] S. Conrad, W. Hasselbring, A. Koschel, R. Tritsch: Enterprise Application Integration. Spektrum Akademischer Verlag, 2005.

[Conway 1968] M.E. Conway: How do committees invent? Datamation, 14(4):28-31, April 1968.

[Ehlers et al. 2011] J. Ehlers, A. van Hoorn, J. Waller, W. Hasselbring: Self-Adaptive Software System Monitoring for Performance Anomaly Localization. In: 8th IEEE/ACM International Conference on Autonomic Computing (ICAC 2011).

[Fittkau et al. 2013] Florian Fittkau, Jan Waller, Christian Wulf, Wilhelm Hasselbring: “Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach“, In: 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013).

[Fittkau et al. 2015a] Florian Fittkau, Sascha Roth, Wilhelm Hasselbring: “ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes“, In: 23rd European Conference on Information Systems (ECIS 2015).

[Fittkau et al. 2015b] F. Fittkau, W. Hasselbring: Elastic Application-Level Monitoring for Large Software Landscapes in the Cloud.” In: Proceedings of the 4th European Conference on Service-Oriented and Cloud Computing (ESOCC 2015). September 2015

[Hasselbring 2000] W. Hasselbring: Information System Integration. Communications of the ACM, 43(6): 32-36, June 2000.

[Hasselbring 2002] W. Hasselbring: Component-Based Software Engineering. In: Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing, Singapore, pp. 289-305, 2002.

[Hasselbring 2006] W. Hasselbring: Software-Architektur - Das aktuelle Schlagwort. Informatik-Spektrum 29(1): 48-52, February 2006.

[Hasselbring 2015] W. Hasselbring. Formalization of federated schema architectural style variability. Journal of Software Engineering and Applications, 8(2):72-92, Feb. 2015.

[Heinrich et al. 2014] R. Heinrich, E. Schmieders, R. Jung, K. Rostami, A. Metzger, W. Hasselbring, R. Reussner, K. Pohl: “Integrating Run-Time Observations and Design Component Models for Cloud System Analysis“, In: 9th Workshop on Models@run.time 2014.

[Heinrich et al. 2015] R. Heinrich, R. Jung, E. Schmieders, A. Metzger, W. Hasselbring, R. Reussner, K. Pohl: “Architectural Run-Time Models for Operator-in-the-Loop Adaptation of Cloud Applications”. In: 9th IEEE Symposium on the Maintenance and Evolution of Service-Oriented Systems and Cloud-Based Environments (MESOCA 2015),

[Kraus et al. 2013] S. Kraus, G. Steinacker, O. Wegner: “Teile und Herrsche – Kleine Systeme für große Architekturen“, OBJEKTspektrum 5/2013, 8-13.

[Marwede et al. 2009] N. Marwede, M. Rohr, A. van Hoorn, W. Hasselbring: “Automatic Failure Diagnosis in Distributed Large-Scale Software Systems based on Timing Behavior Anomaly Correlation“, In: 13th European Conference on Software Maintenance and Reengineering (CSMR 2009).

[Reussner & Hasselbring 2008] R. Reussner, W. Hasselbriung: Handbuch der Software-Architektur. dpunkt.verlag, 2nd edition, 2008.

[Scheer 1994] A.-W. Scheer: Business Process Engineering, Springer-Verlag, 1994.

[Sheth & Larson 1990] A. Sheth and J. Larson: “Federated database systems for managing distributed, heterogeneous, and autonomous databases”. ACM Computing Surveys, 22(3):183-236, 1990.

[Steinacker 2014] G. Steinacker: Scaling with Microservices and Vertical Decomposition. http://dev.otto.de/2014/07/29/scaling-with-microservices-and-vertical-decomposition/, 2014.

[van Hoorn 2014] A. van Hoorn: “Model-Driven Online Capacity Management for Component-Based Software Systems”., PhD Thesis, Faculty of Engineering, Kiel University, 2014.

[van Hoorn et al. 2009] A. van Hoorn, M. Rohr, I.A. Gul, W. Hasselbring: “An Adaptation Framework Enabling Resource-efficient Operation of Software Systems” In: 2nd Warm Up Workshop for ACM/IEEE ICSE 2010 (WUP 2009) .

[van Hoorn et al. 2012] A. van Hoorn, J. Waller, W. Hasselbring: “Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis”, In: 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE 2012).

[Waller et al. 2015] Jan Waller, Nils Ehmke, Wilhelm Hasselbring: “Including Performance Benchmarks into Continuous Integration to Enable DevOps“, In: ACM SIGSOFT Software Engineering Notes, 40(2).