Microservice Architectures for Scalability, Agility and Reliability in E-Commerce

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/

Guido Steinacker
Otto GmbH & Co KG
Werner-Otto-Straße 1–7
D-22179 Hamburg, Germany
https://dev.otto.de/
## Facts and figures about OTTO

<table>
<thead>
<tr>
<th>Profile</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Founded</td>
<td>August 17th, 1949</td>
</tr>
<tr>
<td>No. of employees</td>
<td>4,350</td>
</tr>
<tr>
<td>Revenue in 2015/16 FY</td>
<td>2.563 billion Euros</td>
</tr>
<tr>
<td>Online revenue share</td>
<td>round 90 percent</td>
</tr>
</tbody>
</table>

http://www.ottogroup.com/de/presse/material.php
Web Shop until 2013

https://support.intershop.com/kb/index.php/Display/276B90
Scaling such Web Information Systems

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. Context
2. Microservice Architectures for Scalability
3. Microservice Architectures for Agility
4. Microservice Architectures for Reliability
5. Scalable Development of Microservices
6. Takeaways
Polyglot Persistence

Page Assembly

Product

Search

Recommendation

Order

Tracking

Backend Integration

Wide Column Database

cassandra

Full-Text Database

Graph Database

Relational Database

Time Series Database

[Hasselbring 2016]
Project Lhotse 2011-2013

• In 2011, Otto started a complete re-implementation of their ecommerce software from scratch.
• The drivers for this decision were diverse, but had mostly to do with non-functional requirements like scalability, performance and fault tolerance.
  – Regarding scalability, they were not only thinking about technical scalability in terms of load or data.
  – They needed a solution that was scaling with respect to the number of teams and/or developers working on the software at a given time.
  – In addition to that, they planned to practice DevOps including continuous deployment, in order to deliver features quickly to the customer.

https://www.otto.de/unternehmen/de/newsroom/dossiers/lhotse.php
Modernization Strategy

• What they have found was in the first place a little bit unusual, but in the end highly successful:
  – Instead of setting up a single development team to create a new platform for the shop, they were actively employing Conway’s Law by starting development with initially four separate teams with four loosely coupled applications (a.k.a. microservices):
    • **Product**, being responsible for products and their presentation.
    • **Order** for shopping carts and the order process.
    • **Promotion**, serving product recommendations and promotions for assortments, brands, and so on.
    • **Search and Navigation** for search and navigation in the shop.

• In the following years, they founded several more teams and systems.
Verticals at otto.de
Verticals and Microservices at otto.de

This architecture allows **elasticity** for horizontal **scalability**

Agenda

1. Context
2. Microservice Architectures for Scalability
3. **Microservice Architectures for Agility**
4. Microservice Architectures for Reliability
5. Scalable Development of Microservices
6. Takeaways
“The deployment pipeline is the place where the architectural aspects and the process aspects of DevOps intersect.”

[Bas et al. 2015]
Goals:
• Set up new pipelines quickly and automatically using a templating mechanism.
• Being able to build pipelines just like the microservices from tested code maintained in a Git repository.
• Because most of the time, pipelines are simply waiting to be triggered, they aim at a solution that does not waste hardware resources while not in use.
Solution: LambdaCD (internal domain-specific language written in Clojure)
Life Deployments @ Otto.de

Live-Deployments per Week 2014-2017

#Deployments/Week

2014 2015 2016 2017

Week

#Deployments
Agenda

1. Context
2. Microservice Architectures for Scalability
3. Microservice Architectures for Agility
4. Microservice Architectures for Reliability
5. Scalable Development of Microservices
6. Takeaways
Deployment Pipeline with Quality Gates

[Farley 2007]  [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/
Life Deployments & Incidents @ Otto.de
Agenda

1. Context
2. Microservice Architectures for Scalability
3. Microservice Architectures for Agility
4. Microservice Architectures for Reliability
5. Scalable Development of Microservices
6. Takeaways
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/
Microservice Architectures for Scalability, Agility and Reliability

**Scalability** for both, runtime performance and development performance.

---

![Graph showing live deployments and Prio 1 incidents per week from 2014 to 2017.](image-url)
Backup
Microservice Architecture Variations

"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."

[Steinacker 2014]

[Kraus et al. 2013]

[ICSA, 05.04.2017] W. Hasselbring & G. Steinacker
References


