Technology
Arts Sciences
TH Köln

Invitation for Guest Lectures on Microservice-related Topics

Prof. Dr. Stefan Bente (stefan.bente@th-koeln.de)
B.Sc. Marco Reitano (marco.reitano@th-koeln.de)
B.Sc. Jann Deterling (jann.deterling@th-koeln.de)

Version 2018-10-06

Summary

For a Master course on Microservice architecture, we invite IT practitioners and experts from the field to share their insights, knowledge, and experience with our students by delivering guest lectures. The lectures can be held in German or English.

If you are interested, this document provides you with information about the course, organizational framework, and a (non-exclusive) list of preferred topics.

Für eine Vorlesung im Informatik Master über Microservice-Architekturen möchten wir IT-Praktiker und Experten einladen, Ideen, Wissen und Erfahrungen in Form eines Gastvortrags mit unseren Studierenden zu teilen. Die Vorträge können auf Deutsch oder Englisch gehalten werden.

Für diejenigen, die interessiert sind, einen solchen Vortrag zu halten, beschreibt dieses Dokument die Veranstaltung, die organisatorischen Randbedingungen und enthält eine (nicht ausschließlich gemeinte) Liste von Themenvorschlägen.

Guest Lecture as Part of a Computer Science Master Course

„Domain-Specific Architecture“ (in German: „Fachspezifischer Architekturentwurf”, abbreviated (FAE)) is a mandatory course in the 2nd semester of the Computer Science Master Study at TH Köln. The course is rated as 6 Credit Points, which translates into 180h of overall workload per student. A more detailed description of the course can be found **here**.

In this course, the students learn to design the architecture of complex software systems. We decided to focus the course solely on *Microservice*-based architectures, because Microservices have evolved a predominant architecture style for large and loosely coupled systems, implemented with an Agile approach. In addition, Microservices lean towards the Domain-Driven Design methodology by Eric Evans, allowing for a domain-centric design of the system.

The course is structured to guide students through the various stages of software system design. Based on a real-life case study, the students will design a system. In subteams, they will also implement a very basic partial prototype of the system. The intention is that the students don’t end up with “just paper”, but also with some (to an extent) working software.

Each step of the design and implementation process includes a **lecture** on relevant concepts, methods, patterns, technologies etc. For these lectures, we invite **practitioners and experts from the field** to share their experience with the students.

These guest lectures are explicitly not about pure textbook knowledge. Instead, they should focus on the **benefits**, **challenges, pitfalls,**… of technologies and methodologies when applied in **real-life projects** (with all their ambiguities, contradictions, and day-to-day difficulties).

What is the Benefit For You And Your Company?

By giving a guest lecture, you can inspire young IT professionals with your enthusiasm for your field of expertise. As noted on many occasions, students are eager to listen to experienced professionals who talk about what it's like to work with software systems in real life. So, delivering a guest lecture is usually a positive and inspiring experience for you as a lecturer.

Your company has the chance to present itself to potential future employees. Students highly appreciate competence and enthusiasm. By allocating the time and money to send an expert to deliver a guest lecture at a university, your company shows commitment to the values of expertise and education.

Frequently Asked Questions - Organizational Framework

How many students are in the course?

- The course will consist of approx. **15 students** (Computer Science Master, 2nd semester). Most of them have already gathered work experience as software developers (or in a related role).
- However, lecturers should not expect the students to be deeply familiar with relevant technologies and the challenges of professional, commercial software development in real life.
- Should you have doubts about what knowledge level you can presume, please contact the supervisors (Bente / Reitano / Deterling).

Does the lecture have to be held in English?

- Guest lectures can be held in **German or English**.
- This document was written in English in order to accommodate for non-German-speaking external experts.
- The students are reasonably fluent in business English. However, there is no problem at all to hold the lecture in German.

How long should a lecture be?

- The lecture should be **between 45 and 90 min**.
- After the lecture, there will be group work (see the attached course schedule). As a guest lecturer, you are invited to stay and engage in this group work (e.g. by answering project-related questions, handing out small exercises, etc.)
- However, there is no obligation to do so – you can also come for the lecture alone.

What are possible timeslots for lectures?

- Unfortunately, our flexibility in terms of time is limited due to the course structure.
- Timeslot for guest lectures should be on a **Friday,** somewhere between **13:00 – 16:00** (there is some flexibility here).
- Other timeslots will be difficult to arrange, as most Master students work half-time, in most cases with packed schedules.

Where should the lecture take place?

- The course runs on the TH Köln computer science campus in Gummersbach, about 40 km from Köln / Cologne.
- The building is LC6 (a side building, not the main building). **The room is 0501**.
- You will find a description how to get there here: http://blogs.gm.fh-koeln.de/bente/files/2018/06/Anfahrt-Prof.-Dr.-Bente-TH-K%C3%B6ln-Campus-Gummersbach.pdf

Can I deliver the lecture via web conference / web cast?

- This might be a viable alternative if the location and/or timeslot proves to be a problem.
- In any case, we can technically arrange for a remote delivery of the lecture.
- However, the best way to interact with the students is of course face to face.

What topics should I lecture on?

- You find a **list of preferred topics** further down in this document.
- However, don't feel restricted to this list. If you have a topic about which you can share practical field experience with the students, please contact us to discuss.
- We have a number of "Advanced Topic" wildcard slots, where we can accommodate virtually any interesting subject – as long as it is not textbook-only, and related to Microservices.

What if I cannot cover the complete range of aspects planned in the "preferred topics" list?

- This is not really mandatory.
- If there are aspects that we as supervisors think the students should learn about, we will discuss them with you.
- If you cannot / don't want to cover these aspects in your lecture, you just do your part, and we will add ours.

Preferred Topics

This is a list of preferred topics that we have either planned for the course, or have considered worth covering in an in-depth course on Microservices. If you would like to cover one of these topics, please contact us.

However, don't feel restricted to this list. If you have a topic that missing in the list, but on which you can share practical field experience with the students, please contact us. We have a number of "Advanced Topic" wildcard slots, where we can accommodate virtually any interesting subject – as long as it is not textbook-only, and related to Microservices.

The same applies for the "tailoring" of the lectures. If your preferred topic(s) don't map exactly to one of the list items (but e.g. to a combination of several topics), please let us also know. In addition, we are especially interested in practical case studies (green- or brownfield). So, a 360° case study report (covering many of the below aspects in lesser detail) would also be very interesting for our students.

Topics that in our opinion should be dealt with in the Master course are tagged "**mandatory**". This doesn't mean that the other "non-mandatory" topics are any less appreciated. A practitioner's view on them will be of great interest to the students.

1. Basic Concepts

1.1. DDD & Organization Structure ***(mandatory)***

- Introduction to DDD core concepts
- "What is this all about"? Relationship between domain-driven design, agility and organization structure
- Practical advice for domain and bounded context analysis (good practices, rules of thumb for size, …)

1.2. Microservice Concepts ***(mandatory)***

- Motivation: Why Microservices? Advantages and disadvantages, comparison with monolithic architectures and with SOA
- Core Microservice principles (loose coupling, you build it / you run it, freedom of technology choice, …)
- Approach when modelling services (e.g. aggregate root = service as a design starting point)
- Ideal size for a Service (developer anarchy vs. self-contained system)

1.3. CAP-Theorem und Eventual Consistency

- Explanation of CAP theorem and its significance for MS architectures
- Principle of Eventual Consistency
- Practical examples from real-live projects
- Assessment of de-facto consequences and compromises in real life, when Eventual Consistency paradigm is applied – where does it hurt, where is it easy to tolerate

2. APIs

2.1. API Patterns ***(mandatory)***

- Rules for REST APIs
- Maturity model according to Richardson, esp. level 2 vs. level 3
- GraphQL as alternative to REST
- Good practices: what-to-use-when
- API First vs. Code First

2.2. Advanced Aspects of REST API

- Connection between REST and Messaging
- Approach for designing APIs
- n:m relations in REST
- Local vs. global IDs

3. Service Integration

3.1. Complex Functions in a Rich Domain Model ***(mandatory)***

- Distinction Anemic / Rich Domain Model
- Why is it sensible to use REST level 3 with a RDM?
- How to implement complex interactions between services using Level 3

3.2. Transactions between Microservices ***(mandatory)***
- Transaction patterns (event sourcing, Saga pattern, interaction between REST and messaging)
- Introduction to messaging and frequently used technologies

3.3. Large transactions and business processes in a Microservice landscape
- Typical challenges in real-life projects wrt. transactions and business processes, depending on the application domain (eCommerce, insurance, …)
- Conceptual level: orchestration vs. choreography, hybrid concepts
- Do's and don'ts

3.4. Resilience in large systems
- Real-life challenges – what are typical, frequently occurring problems?
- Patterns for resilient services (circuit breaker, …)
- Tools and technologies

4. UIs in a Microservice Landscape

4.1. UIs in a Microservice Landscape ***(mandatory)***
- Popular MS patterns to connect UIs: API Gateway, Backend for Frontend, Sidecar, …, …
- Do's and Don'ts when connecting clients
- UI integration concepts (HTML links, monolithic UIs, client / service side composition, Micro Frontends, ...)

5. Infrastructure Architecture

5.1. Container & Execution Environment ***(mandatory)***
- DevOps principles
- Docker as a concept
- Useful patterns for distributing containers
- Comparison of tried and trusted technologies and tools
- Service Discovery

5.2. Test Automation
- Concepts CI / CD
- Patterns, technologies
- A practical approach to overall test design (how to navigate the thin line between carelessness and gold-plating)
- Practical do's and don'ts

5.3. Monitoring
- Practical advice: what should be supervised, and how?
- Log Aggregation
- Tried and trusted tools and technologies

5.4. XYZ as Code
- Overview: what can be done in terms of automation
- Experience from real-life projects: To what extent does automation pay off? What are the trade-offs?
- Pragmatic approach „how to get started"

5.5. Load Balancing
- Overview: approaches to load balancing (in general, and in a Microservice landscape)
- How to assess load requirements in practice – real-life project experience appreciated
- Patterns and tools for Load Balancing

5.6. Authentication und authorization
- Practical advice on suitable technologies
- Integration with enterprise-wide systems for user management (LDAP, Active Directory, …)

6. Microservice Development

6.1. Debugging in distributed environments
- Practical tips for bug analysis in complex distributed applications
- Good practices for tools and IDEs

6.2. Documentation
- MS = just code, no documentation? What and how to document in MS development?
- Principles for light-weight architecture documentation
- Good practices: what tools work best? Advice for team rules on documentation?

7. Real-Life Experience

7.1. Real-life case studies
- 360° experience report on real-life projects
- What was easy – what went wrong – where were the pitfalls …

7.2. Brownfield vs. Greenfield
- Real-life experience: How to deal with domain analysis on existing monoliths? Top-down or bottom-up analysis? What if the existing domain structure is completely messed up – How to start with a clean slate?
- Politische aspects: What if the organizational structure *has* to be changed? How (and to what extent) is it advisable and possible to combine Technical Consulting and Organisational Change Management?
- Approach – „low hanging fruit“ first, …

Zeitplan für Informatik-Master-Veranstaltung "Fachspezifischer Architekturentwurf (FAE)" WS 18/19

Schedule for Computer Science Master Course "Domain Specific Architectures" WS 18/19

Date	Time	Lecture / Presentation Part	by …	Group Work Part	Result(s) To be presented next course meeting
Fr 12.10.	12:00 - 16:00	**Kickoff** - Course structure, time schedule, further organizational details, grading criteria - Rules for result documentation (Github)	TH Köln	- Discussion of the domain at hand: what subdomains do we have? - Definition of 3-4 subteams, along the subdomain borders	- Subdomains identified - Each student mapped to a subdomain team
Fr 19.10.	13:00 - 17:00	**1.1. DDD & Organization Structure** - Introduction to DDD core concepts - "What is this all about"? Relationship between domain-driven design, agility and organization structure - Practical advice for domain and bounded context analysis (good practices, rules of thumb for size, …)	TH Köln	- Subteams work on their own (sub-)domain model, using DDD concepts	- Initial sub-domain model "on paper"
Fr 26.10.	13:00 - 17:00	**1.2. Microservice Concepts** - Motivation: Why Microservices? Advantages and disadvantages, comparison with monolithic architectures and with SOA - Core Microservice principles (loose coupling, you build it / you run it, freedom of technology choice, …) - Approach when modelling services (e.g. aggregate root = service as a design starting point) - Ideal size for a Service (developer anarchy vs. self-contained system)	external or TH Köln	- Teams implement simple CRUD services	- Aggregates defined - JPA definitions ready
Fr 02.11.	13:00 - 17:00	**2.1 API Patterns** - Rules for REST APIs - Maturity model according to Richardson, esp. level 2 vs. level 3 - GraphQL as alternative to REST - Good practices: what-to-use-when - API First vs. Code First	external or TH Köln	- Teams design and implement CRUD REST API - Constraint: Spring Boot + Spring Web MVC must be used (not Sprint Data REST)	- REST API specified & documented - REST API implementation can be tested using Postman
Fr 09.11.	13:00 - 16:00	**3.1. Rich Domain Model** - Distinction Anemic / Rich Domain Model - Why is it sensible to use REST level 3 with a RDM? - How to implement complex interactions between services	TH Köln	- Teams define examples for complex functions on services, implement them as dummies, and implement interaktion between services - Constrain: first use HateOAS package (to feel the pain), then migrate to Spring Data REST (which gives you level 3 "for free")	- API based on Spring Data REST specified & implemented (testable by Postman) - complex interaction scenario can be demonstrated (Postman)
Fr 16.11.	13:00 - 17:00	**4.1. UIs in a Microservice Landscape** - Popular MS patterns to connect UIs: API Gateway, Backend for Frontend - Do's and Don'ts when connecting clients - UI integration concepts (HTML links, monolithic UIs, client / service side composition, Micro Frontends, ...)	external or TH Köln	- Teams build simple UI, connect them via REST and integrate them using a chosen integration paradigm	- Simple UIs in place - Basic UI integration - Connected via REST API
Fr 23.11.	*No Lecture due to Project Week*				
Fr 30.11.	13:00 - 17:00	**5.1. Container & Execution Environment** - Docker as a concept - Service Discovery - DevOps principles	external or TH Köln	- Teams set up execution environment	- "All on Docker" - Service Discovery in place - Basic integration between MS available (REST calls)
Fr 07.12.	13:00 - 17:00	**3.2 Transactions between Microservices** - Transaction patterns (event sourcing, Saga pattern, interaction between REST and messaging) - Introduction to messaging and frequently used technologies	external or TH Köln	- Teams define events (provider) and connect to message broker - Teams select events to be consumed and implement a listener	- events specified for exemplary services - some services connected to message broker (provider & consumer)
Fr 14.12.	tbd	**Joint Workshop with Social Workers (Prof. Dr. Isabel Zorn)**	TH Köln	*See workshop concept, agenda, and goals*	
Fr 21.12.	13:00 - 17:00	**Advanced Architecture Topic 1** - *see list of advanced topics*	external	*to be defined*	
Fr 28.12.	*Xmas Break*				
Fr 04.01.					
Fr 11.01.	13:00 - 17:00	**Advanced Architecture Topic 2** - *see list of advanced topics*	external	*to be defined*	
Fr 18.01.		**Advanced Architecture Topic 3** - *see list of advanced topics*	external		
Fr 25.01.		**Advanced Architecture Topic 4** - *see list of advanced topics*	external		
Fr 01.02.	13:00 - 17:00	**Wrapup** - Summary - Retrospective: Conclusions, Lessons Learnt - Organizational details for final architecture documentation and final presentation	TH Köln	- Teams prepare their own conclusions, to be shared with the larger group later	-

Green: Invitations for external guest lectures