Microservices in Higher Education

– Migrating a Legacy Insurance Core Application –

Moritz Lange[1], Arne Koschel[1], and Andreas Hausotter[1]

University of Applied Sciences and Arts in Hannover, Hannover 30459, Germany,
(moritz.lange@stud. | arne.koschel@ | andreas.hausotter@) hs-hannover.de

1 Introduction

As part of the bachelor's program in Applied Computer Science of the Univ. of Applied Sciences and Arts, Hannover (HsH), the practical project *Potential and Challenges of Microservices in the Insurance Industry* was carried out with two partner companies. During the winter semester 2017, the theoretical foundations of the microservices approach have been studied, technologies were selected and applied to a demo scenario. Results have been presented to and discussed with the partners. The following semester focused on the migration of a monolithic mainframe-based core application, namely the `Partner Management System`, whereby only functional requirements were specified by the companies. The students task was to design the functional and technical concept and the implementation of a prototype based on the selected technology stack. The aim was to prove the suitability of the microservices approach for the insurance industry. In addition, known benefits of the microservices architecture should be highlighted and challenges and limitations addressed.

The remainder of this article is organized as follows: In sec. II we focus on the educational asspects of the project. In sec. III we show the core application system and address issues with the monolithic approach. Sec. IV introduces the architecture of the migrated system. In Sec. V we evaluate the outcomes of our project. Sec. VI summarizes our results and draws a project conclusion.

2 Studying Applied Computer Science at the HsH

Our bachelor's program in Applied Computer Science is an academic course of study with a focus on practical application to optimally prepare students for a successful career in the area of IT [1]. The overarching goal is the development of professional competence, methodical expertise, social skills, and self-competence. To achive this, a fair degree of the curriculum is devoted to practical work, such as lab exercises, seminars, bachelor theses, and practical projects. Within the department of computer science we operate our own private cloud to teach, for example, practical excercises for service-based as well as data-intensive applications (cf. [4]).

With regard to projects, students have to apply the competencies acquired in previous courses. They work in a project group for two semesters on a specific problem, often in coop-

eration with industry partners. All project management tasks are in the responsibility of the students. The lecturers are coaches and the steering committee, but they are not members of the project team.

From our point of view the practical project *Potential and Challenges of Microservices in the Insurance Industry* was very suitiable to develop the aforementioned competencies:

- Professional competence. The students had to work out independently the concepts of the microservices approach and the underlying technologies. The migration of the `Partner Management System` towards a microservices architecture was a big challenge, from a functional and technical point of view. The students had to become aquainted with the business use cases and the technologies to realize the migration process.
- Methodical expertise. As project management tasks were completely in the hands of the students, they had to make the choice for the process model, being aware of all consequences of the decision. After careful deliberation, they opted for Scrum.
- Social skills, and self-competence. Agile methods and small development teams require a big amount of communication skills, ability to work in a team, critical faculties, and a large amount of self-discipline.

3 Core Application: `Partner Management System`

As introduced, the task was to design the `Partner Management System`, a system for managing partners of an insurance company. Here partners are defined as natural and legal persons who have a relation with the insurance company (clients, brokers, lawyers etc.). Fig. 1.a shows the domain model, which was developed with the insurance companies involved in the project. The model is based on the reference architecture for German insurance companies (VAA) [3].

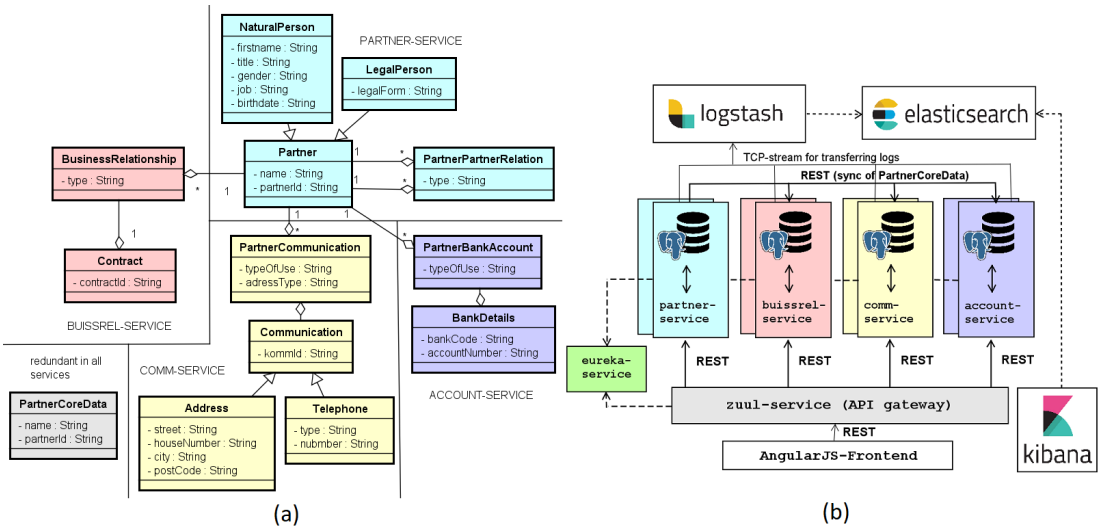

Figure 1: (a) domain model of the system, (b) infrastructure of the system

The `Partner Management System`'s core is a CRUD application that manages an entity partner and its properties. Usually this functionality is implemented in a single SOA service.

Modularization through SOA services provides significant benefits to the overall system (e.g., scalability and resilience), but the `Partner Management System` is still an atomic deployment unit. Since the `Partner Management System` is a central service of an insurance company, different parts of this system are subject to varying levels of stress. Especially at night, the load profile differs. While only minor changes are made to existing datasets during the day, low occupancy during the night is used to slowly persist all new datasets collected on the day so as not to overburden the mainframe-based application. However, in practice this approach often leads to crashes at night. The central problem of the existing system is obviously the poor flexibility, scalability and fault tolerance. This leads to a microservices approach.

4 Microservice Architecture: `Partner Management System`

The architecture developed divides the application into four independent services (Fig. 1.b). To determine this separation, the original domain was divided into subdomains and specific requirements of each subdomain were analyzed. It had to be ensured that the subdomains are as independent as possible and that rational use cases exist in which a subdomain can be used without any other. For this, the architecture keeps parts of the partner (PartnerCoreData) redundant in all subdomains. This corresponds to bounded contexts, as described by Fowler [2]. Technically, the resulting subdomains were implemented as Java REST web services. Following Fowler [2], each service has its own data management (dedicated PostgreSQL database). REST calls keep the PartnerCoreData synchronized across all services. Parts of the Netflix OSS stack were used for the system infrastructure. Netflix Eureka (`eureka-service`) as a service discovery and Netflix Zuul (`zuul-service`) as an API gateway. Zuul also provides an AngularJS frontend for the application. The ELK stack was used for logging/monitoring. Components (Fig. 1.b) of the architecture are deployed in Docker containers and connected by a virtual network.

5 Evaluation of the Outcomes

In this section we evaluate the outcomes of our project from a business and technical point of view as well as from a teaching perspective. We start with the latter.

At the beginning of the project the students had only basic knowledge about distributed systems, software architectures, team development, and agile project management. Over time, they became familiar with the microservices approach, developing the relevant concepts and technologies largely autonomously. The project outcome has shown that the students knew how to use the enormous increase in professional competence in a purposeful way.

Without any doubt, applying Scrum was a particular challenge. The team was most successful in the adoption of agile principles and living the 'Scrum values' (commitment, focus, openness, respect, and courage). Thus, the team has benefited from the increase of independence and self-reliance, the open communication and the trust among each other. On the other side, initially the consequent application of the Scrum framework caused some trouble.

From the business and technical point of view, the project results are also a success. Although the architecture developed does not seem very surprising, it can eliminate many of the problems of the existing implementation. In particular, the microservice architecture can guarantee the required scalability and fault tolerance, so that this approach can certainly be considered suitable for practical use. Moreover, two bachelor theses – both in the microservices space – were jointly with the insurance partners successfully written by former members of the bachelor project.

6 Conclusion and Future Work

In this article we presented a microservices student project, developed jointly with two partner companies. As part of their studies, bachelor students explored microservices conceptually and then applied their insights to a technical demo and to a core legacy application from the insurance domain. Students and partners were quite satisfied with the results. The students learned a lot about microservices, to value conceptual research first before developing prototypes, and also to work for a long time as a team. For the partners helped the project to evaluate microservices pros and cons for their organization. In future work, we will examine consistency mechanisms for microservices in the `Partner Management System` context.

References

[1] University of Applied Sciences Department of Computer Science and Arts Hannover. Study Guide Faculty IV, Dept. of Computer Science. https://f4.hs-hannover.de/fileadmin/media/doc/f4/Studium/Bachelor_Studiengaenge/BIN/Study_Guide_Course_Catalogue_HsH_Computer_Science_2014.pdf, 2014. Acc: 11.11.2018.

[2] Martin Fowler and James Lewis. Microservices a definition of this new architectural term. https://martinfowler.com/articles/microservices.html, March 2014.

[3] GDV. Die Anwendungsarchitektur der Versicherungswirtschaft - Grundlagen und Prinzipien, 1999.

[4] Dominik Schöner, Arne Koschel, and Felix Heine. Teaching microservices in the private cloud by example of the edudscloud. In *Proc. 10th International Conferences on Advanced Service Computing (SERVICE COMPUTATION 2018)*, pages 36–39, Barcelona, Spain, 2018. IARIA / ThinkMind.