Design and Implementation of a Remote Care Application Based on Microservice Architecture

Philip Nils Wizenty, Florian Rademacher, Jonas Sorgalla, and Sabine Sachweh

University of Applied Sciences and Arts Dortmund,
Institute for the Digital Transformation of Application and Living Domains,
Otto-Hahn-Straße 23, 44227 Dortmund, Germany,
{philipnils.wizenty,florian.rademacher,jonas.sorgalla,sabine.sachweh}@fh-dortmund.de

Abstract. Microservice Architecture (MSA) is an architectural style for service-based software systems. MSA puts a strong emphasis on high cohesion and loose coupling of the services that provide systems’ functionalities. As a result of this, MSA-based software architectures exhibit increased scalability and extensibility, and facilitate the application of continuous integration techniques. This paper presents a case study of an MSA-based Remote Care Application (RCA) that allows caregivers to remotely access smart home devices. The goal of the RCA is to assist persons being cared in Activities of Daily Living. Employing MSA for the realization of the RCA yielded several lessons learned, e.g., (i) direct transferability of domain models based on Domain-driven Design; (ii) more efficient integration of features; (iii) speedup of feature delivery due to MSA facilitating automated deployment.

Keywords: Microservice Architecture · Smart Home · Remote Care

1 Introduction

In the upcoming years, the number of people aged 60 and older will steadily increase and is predicted to worldwide grow from 901 million in 2015 to approximately more than 1.4 billions in 2030 [13]. This will result in an increasing demand for caregivers which can not be covered by the labor market [10]. Hence, new solutions are needed to cope with the resulting care gap. The IT sector is one of the central domains from which such solutions are expected. That is, because it is perceived of being able to provide additional support for both persons being cared and caregivers by developing new supportive technologies [10,3]. Next to mainly hardware-based solutions like wearables, Internet of Things based distributed software systems could aid in coping with the broadening care gap [3].

In this paper we present a distributed software system that addresses the domain of ambulant care. More specifically, the system denotes a Remote Care

⋆ This research is partially funded by the German Federal Ministry of Education and Research in the project “QuartiersNETZ” (grant number 02K12B061.)
Application (RCA) [8], which leverages Microservice Architecture (MSA) [5] as its underlying architectural style. Its main purpose is to provide a platform for professional caregivers and nursing relatives to remotely interact with smart home devices in households of persons being cared and hence support them in their Activities of Daily Living (ADL) [9], e.g., housekeeping, multimedia, taking medicine, or personal hygiene. We present the RCA in the form of a case study with the following objectives in mind: (i) provide MSA researchers with a practice-related, well-documented research object; (ii) present our experiences in MSA development; and (iii) elucidate our lessons learned in practical MSA development. Therefore, we describe the design, implementation and deployment of the RCA with a strong focus on architectural challenges and requirements.

The remainder of the paper is organized as follows. Section 2 identifies functional and non-functional requirements for the RCA. Section 3 describes the RCA’s design and implementation based on the requirement elicitation. Section 4 discusses the realized MSA solution for the RCA and the lessons learned during its implementation. Section 5 concludes the paper.

2 Functional and Non-Functional Requirements of the Remote Care Application

This section describes the identified functional and non-functional requirements of the RCA. The requirements elicitation process was part of a holistic, iterative participatory design methodology, which we developed and proved within the research project of which the RCA was one result [12]. Among others, the methodology comprises application-specific phases for stakeholder identification, as well as selection and application of the requirements elicitation process model being most appropriate for the stakeholders and the application to develop participatory. Employing the methodology resulted in (i) persons being cared, professional caregivers and nursing relatives being the relevant RCA stakeholders; (ii) the future workshop method [11] in combination with goal-oriented requirements elicitation [4] being the most sensible process model for requirements elicitation.

Tables 1 and 2 show the functional and non-functional requirement goals resulting from applying the process model. However, due to space constraints, we only list the tier1- and tier2-top-level goals and omit more fine-grained sub-goals.

The goals T1-FG-1 and T1-FG-2 in Table 1 describe the general purpose of the RCA. Accordingly, the RCA provides means to remotely assist household residents in human-technology interaction scenarios, e.g., programming the washing machine. Additionally, the RCA enables ambulant caregivers to dynamically prepare for their service remotely, e.g., heating up the bathroom in advance of their arrival. Based on tier 1, tier 2 Goal T2-FG-1 addresses the need to read data from smart home devices with the RCA to, e.g., display current and historical device states. Next to basic read access, selected stakeholders may have to control devices, as expressed in Goals T2-FG-2 and T2-FG-3.

Table 2 shows non-functional requirement goals related to the RCA. They were actually the main drivers for implementing the RCA on the basis of MSA.
Table 1. Tier 1 and tier 2 functional requirement goals of the Remote Care Application

| ID       | Goal              | Description                                                                 |
|----------|-------------------|-----------------------------------------------------------------------------|
| T1-FG-1  | Remote Support    | The RCA must enable caregivers to remotely assist persons being cared        |
|          |                   | in human-technology interaction scenarios.                                  |
| T1-FG-2  | Preparing         | The RCA must enable caregivers to remotely prepare a household in advance    |
|          | Household         | of an ambulant care visit.                                                  |
| T2-FG-1  | Smart Home        | The RCA must be connected with various smart homes and be able to read device|
|          |                   | states.                                                                     |
| T2-FG-2  | Remote Control    | Devices in connected smart homes must be controllable.                      |
| T2-FG-3  | Access Rights     | Device control must be explicitly granted by admins.                        |

Table 2. Non-functional requirement goals of the Remote Care Application

| ID   | Goal        | Description                                                                 |
|------|-------------|-----------------------------------------------------------------------------|
| NG-1 | Scalability | The RCA must be able to handle thousands of households.                     |
| NG-2 | Security    | Security must be high due to personal data being involved.                  |
| NG-3 | Availability| High availability on the basis of increased resilience, robustness and       |
|      |             | functional independence.                                                    |
| NG-4 | Extensibility| New functionalities need to be flexibly integrable and provicable.            |

(cf. Subsection 3.1). Goal NG-1 expresses the need for high scalability of the RCA being particularly relevant for professional caregivers having a potential customer base of thousands of households. Additionally, because of the high sensitivity of data related to the persons being cared and in case of misuse the potential of burglary, data and communication security must follow a high state of practice (goal NG-2). The RCA must also exhibit a high degree of availability to enable quick reaction by professional caregivers and nursing relatives in cases of emergency (goal NG-3). Another relevant concern is extensibility (goal NG-4) to allow flexible integration of new smart home technologies and devices as well as new functionalities, e.g., means for data analysis, at runtime.

3 Design and Implementation of the Remote Care Application

This section describes the design and implementation of the RCA according to the identified requirements (cf. Section 2). First, Subsection 3.1 elucidates our decision to apply MSA to the RCA w.r.t the requirements. Next, Subsection 3.2 presents the application design. Finally, Subsection 3.3 describes the RCA’s implementation.

3.1 Motivation for Microservice Architecture

We chose MSA as the RCA’s underlying architectural style mainly because of the high requirements for scalability, availability and extensibility, i.e., goals NG-1, NG-3 and NG-4 in Table 2. The scalability requirement is satisfied because
microservices are scalable independently due to functional isolation and technical self-containment [5]. Availability can be achieved because MSA fosters the definition of well-defined, functional service boundaries. Eventually, compared to monolithic applications, this results in an increased resilience as the faulty service (i) fails instead of the whole application; (ii) can be identified more effectively than in tightly coupled monoliths [5]. Extensibility is an inherent characteristic of service-based architectural styles [1] and comes with well-partitioned service boundaries, which is one central characteristic of MSA [5].

3.2 Model-based Design of the Application

The RCA’s design process employed various types of models applicable to MSA for different design-specific concerns, i.e., capturing of domain concepts and microservice identification, interface modeling, and deployment modeling [7].

We applied the Domain-driven Design (DDD) methodology [2] to iteratively capture relevant domain concepts and their relationships. This resulted in the domain model shown in Figure 1, which we created in collaboration with domain experts, i.e., representatives of the RCA stakeholders (cf. Section 1). Its notation and elements’ semantics rely on a UML profile for DDD-based domain models [6], to prospectively enable semi-automatic model validation and code generation. Due to lack of space, however, we only present the result of the model creation process’s final iteration. That is, a domain model with some technical information relevant to the RCA’s implementation (cf. Subsection 3.3).

The domain model is decomposed into three Bounded Contexts, of which each denotes a candidate for a functional microservice [5,7]. The HistoryContext bundles domain-specific concepts that model data of connected smart homes, devices’ components (called items) and states. This corresponds to requirement goal T2-FG-1 (cf. Table 1), whereby the current state of a given DeviceItem is the DeviceState with the highest timestamp. A concrete SmartHome instance may then, for instance, contain a DeviceItem “ParlorLight_Color” with DeviceState value “(210,0.25,1)”, which corresponds to light blue in the HSV color model. Furthermore, the context encapsulates a DDD Repository that models storage and retrieval of SmartHome instances and a Service to retrieve smart home and their device instances. The RemoteControlContext expresses characteristics for remote control of devices (cf. T2-FG-2). The stakeholders regarded
smart home devices as objects that may receive descriptive commands like “switch off palor light”. Therefore, the context’s technical Service realizes communication of commands to a concrete smart home. The AccessControlContext models regulation of access rights (cf. T2-FG-3). Within the collaborative domain modeling process we discussed several remote care control scenarios, that were identified in the future workshop with the stakeholders (see Section 2). Together with the representative domain experts, we came to the conclusion that simple read and write access rules are sufficient. Central to the context is the AccessControlService, which is used by the other contexts and deals with checking the access rights of a given user. Hence, they are organized in AccessItems to subsume concrete, regulated domain concepts like Command or DeviceItem.

![Diagram](image-url)

**Fig. 2.** Interface model of the Remote Care Application in MSA-adapted notation [7]

From the domain model, the interface model [7] depicted in Figure 2 was deduced by mapping each Bounded Context in Figure 1 to a microservice. As the interface model represents the technical implementation of the microservices, it needs to exhibit concrete technical information not present in the domain model, i.e., (i) interfaces with methods (deduced from the Bounded Contexts’ Services of the domain model); (ii) types for exchanged data structures, e.g., Command, and their fields, e.g., value; (iii) service interaction relationships, e.g., the HistoryService consumes the AccessControlService (deduced from the use relationships in the domain model); (iv) additional elements necessary for the implementation, e.g., additional add methods in the AccessControl interface.

### 3.3 Implementation

Based on the interface model, we started to implement the RCA. applied an MSA-specific, technically motivated Architectural Design (MSA-AD) [14]. Next to the functional microservices as business-related components, it comprises common infrastructure components of MSA.

Load Balancer and Circuit Breaker denote service-specific infrastructure components [14]. Load Balancers may cope with increased amounts of requests by measuring incoming network traffic and distributing requests to different service instances. Circuit Breakers on the other hand increase the resilience of microservices by blocking requests that continuously result in errors or communication faults. Satisfying requirement goals NG-1 and NG-3 (see Table 2), both
Another key infrastructure component of the MSA-AD relevant to the RCA is the Discovery Service [14]. It allows functional and infrastructure microservices to expose their own interfaces, discover exposed interfaces of other services and establish a communication relationship with those services. The Discovery Services addresses requirement goal NG-4 (cf. Table 2) as it facilitates the flexible integration of new functionalities provided by microservices. The demand for security related to the RCA (goal NG-2 in Table 2) is satisfied by a dedicated Security Service [14]. It acts as an identity provider for client authorization and authentication. The services employ token-based security by populating each request’s data with an access token. The necessary user data is managed by the User Management Service. Hence, the caller can be clearly identified, even when the request is transitively delegated from service to service. In combination with an additional API Gateway Service, the Security Service realizes a Single Sign-On Gateway, i.e., a central point for authentication. Besides that, the API Gateway Service denotes the entry point to the RCA for external callers.

The implementation of the RCA is based on Spring Cloud\textsuperscript{1}. Consequently, each microservice is a standalone Java archive built with Spring Boot. The Discovery and API Gateway Service employ Eureka and Zuul, respectively. For architecture-internal communication synchronous RESTful HTTP is used. Furthermore, the API Gateway Service provides a REST endpoint for external requests. However, to increase the RCA’s scalability (goal NG-1 in Table 2), asynchronous message-based communication via MQTT is applied with HiveMQ\textsuperscript{2} as message broker for receiving data from connected smart homes. Eventually, the Security Service was based on OAuth2\textsuperscript{3}.

Figure 3 presents the deployment overview of the RCA and connected software components. The RCA communicates with various external clients. Admin-

---

\textsuperscript{1} http://projects.spring.io/spring-cloud
\textsuperscript{2} https://www.hivemq.com
\textsuperscript{3} https://oauth.net/2
Administrators can configure access rights and control a connected smart home with the ManagementPlatform (cf. Subsection 3.2). It is mainly used on the office computers of professional caregivers. The RemoteAssistanceApp is the counterpart of the platform for mobile devices. Both control applications, i.e., platform and app, enable device control independent of a certain vendor. Additionally, the RCA acts as a mediator between smart homes and the control applications. Therefore, each SmartHome needs to execute an instance of Eclipse Smart Home⁴, because both HistoryConnector and ControlConnector are based on this framework to abstract from vendor-specific protocols. While the HistoryConnector transmits status changes to the RCA, the ControlConnector receives and executes remote control commands (cf. Subsection 3.2).

To provide MSA researchers with an operating case study, whose creation was mainly driven by practical requirements, we made the RCA’s code available as open source on GitHub⁵.

⁴ https://www.eclipse.org/smarthome
⁵ https://github.com/SeelabFhdo/RemoteCareApplication
Fourth, the model-based design process (cf. Subsection 3.2) eased the implementation of the RCA. However, it also introduced an additional amount of effort as we deduced the interface and deployment model manually from the domain model [7]. We expect that this extra work may be reduced by (semi-) automatically transforming the domain model into MSA-specific design models.

5 Conclusion

In this paper, we presented a case study on how to apply MSA in the remote care domain. We identified requirements for the RCA leveraging participatory design techniques and employed DDD to create an appropriate domain model. From this, we deduced further design models which enabled us to derive the RCA’s implementation consisting of several functional and infrastructure microservices. Hence, the case study may not only shed light on MSA’s applicability for remote care, but also on how model-based design may aid in microservice development.

References

1. Erl, T.: Service-Oriented Architecture (SOA) Concepts, Technology and Design. Prentice Hall (2005)
2. Evans, E.: Domain-Driven Design. Addison-Wesley (2004)
3. Islam, S.M.R., Kwak, D., Kabir, M.H., Hossain, M., Kwak, K.S.: The internet of things for health care: A comprehensive survey. IEEE Access 3, 678–708 (2015)
4. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proc. of the Fifth Int. Symp. on Requirements Engineering, pp. 249–262 (2001)
5. Newmann, S.: Building Microservices. O’Reilly Media (2016)
6. Rademacher, F., Sachweh, S., Zündorf, A.: Towards a uml profile for domain-driven design of microservice architectures. In: Software Engineering and Formal Methods. LNCS, vol. 10729, pp. 230–245. Springer (2018)
7. Rademacher, F., Sorgalla, J., Sachweh, S.: Challenges of domain-driven microservice design: A model-driven perspective. IEEE Software (May–June 2018), in press
8. Rashid Bashshur, G.S.: History of telemedicine: Evolution, context, and transformation. New Rochelle (2009)
9. Rashidi, P., Mihailidis, A.: A survey on ambient-assisted living tools for older adults. IEEE Journal of Biomedical and Health Informatics 17(3), 579–590 (2013)
10. Redfoot, D., Feinberg, L., Houser, A.: The aging of the baby boom and the growing care gap: a look at future declines in the availability of family caregivers. AARP Public Policy Institute (2013)
11. Robert Jungk, N.M.: Future Workshops: How to Create Desirable Futures. Institute for Social Inventions (1996)
12. Sorgalla, J., Schabsky, P., Sachweh, S., Grates, M., Heite, E.: Improving representativeness in participatory design processes with elderly. In: Proc. of the 2017 CHI Conf. Extended Abstracts on Human Factors in Computing Systems, pp. 2107–2114. ACM (2017)
13. United Nations: World Population Ageing 2015. No. ST/ESA/SER.A/390 (2015)
14. Wizenty, P., Sorgalla, J., Rademacher, F., Sachweh, S.: Magma: build management-based generation of microservice infrastructures. In: Proc. of the 11th Europ. Conf. on Software Architecture (ECSA). pp. 61–65. ACM (2017)