# Model-driven Development of Microservice Architecture: An Experiment on the Quality in Use of a UML- and a DSL-based Approach

Jonas Sorgalla
University of Applied Sciences and Arts Dortmund
IDiAL Institute
Dortmund, Germany
jonas.sorgalla@fh-dortmund.de

Florian Rademacher
University of Applied Sciences and Arts Dortmund
IDiAL Institute
Dortmund, Germany
florian.rademacher@fh-dortmund.de

Sabine Sachweh
University of Applied Sciences and Arts Dortmund
IDiAL Institute
Dortmund, Germany
sabine.sachweh@fh-dortmund.de

Albert Zündorf
University of Kassel
Department of Computer Science and Electrical Engineering
Kassel, Germany
zuendorf@uni-kassel.de

## ABSTRACT

Microservice Architecture denotes a software architectural style for service-based software systems whereby business capabilities are encapsulated in autonomously developable and deployable services. To foster a more efficient development it is considered feasible to apply the means of Model-driven Development in order to handle the complexity of such a distributed system and avoid the manual creation of boilerplate code using code generation. In this paper, we present and evaluate two modeling approaches for microservice systems. The first approach is based on a set of domain-specific modeling languages and the second approach on the Unified Modeling Language. We evaluate both approaches in the context of an experiment conducted during a master's course in informatics with 32 participants. We compare the modeling approaches based upon their achieved effectiveness and perceived efficiency, satisfaction, and accessibility of the students. We do this by evaluating the correctness of created models during an assignment and through a questionnaire. Our results indicate that both approaches are generally suited for modeling microservices. However, the students did commit fewer modeling errors while using the set of domain-specific modeling languages.

## CCS CONCEPTS

• **Software and its engineering** → **Model-driven software engineering**; **Software notations and tools**; *Abstraction, modeling and modularity*; *Unified Modeling Language (UML)*; • **Human-centered computing** → *Empirical studies in HCI*; • **Computer systems organization** → *Cloud computing*.

## KEYWORDS

Software Architecture, Microservices, Domain-Specific Languages, Modeling Language Evaluation

*Experiment Report, April 15, 2020,*
2020. https://doi.org/doi:10.17170/kobra-202010302034

## 1 INTRODUCTION

Microservice Architecture (MSA) [22] is an emerging software architectural style for developing and deploying service-based software systems [32]. In an MSA, a business capability is encapsulated in a corresponding service. Multiple of these services are loosely coupled in a distributed system minimizing dependencies to other components. Hereby, services agree on contracts as predefined specifications of communication relationships which are realized through interfaces for the actual interactions [13, 22]. Besides the services realizing a capability, an MSA also consists of infrastructural services that enable the operation of the system, e.g., storing configurations or providing means to discovery other services [1]. From an organizational point of view, each microservice is owned by exactly one team that accounts for the service's design, development, deployment, and maintenance [21].

Although the application of MSA is considered beneficial for a system's scalability, adaptability, and resilience [22], the application also increases the complexity and initial development cost of a system [7, 29]. A promising approach to address these challenges is Model-driven Development (MDD) [10]. Hereby, MDD aims to reduce complexity by utilizing models as means for abstraction and first-class citizens in the development process. In particular, service-based software systems such as MSAs are considered to benefit from such abstraction and the sophisticated techniques, e.g., code generation, it enables [2].

In previous research, we focused on providing the conceptual and practical foundation to exploit the benefits of MDD for MSA by creating a dedicated set of textual modeling languages called *Language Ecosystem for Modeling Microservice Architecture* (LEMMA)[1] [3, 26]. LEMMA addresses the different team roles which are typically involved in the creation of an MSA by including various *modeling viewpoints* [23]. However, while our approach takes the unique characteristics of MSA into account, there are also alternative means which can be applied to its modeling, e.g., the Unified Modeling Language (UML) [24].

[1] https://github.com/EASE2020auxmat/material/

At present, such different modeling approaches for MSA have not yet been compared to each other. We are therefore lacking clear evidence in which form a modeling language for the MDD of MSAs has to be designed for users in order to make the development process more effective and efficient. Therefore, in this paper we present an empirical pilot study with the purpose to compare the *Quality in Use* [5] of LEMMA compared to UML. Our results enable insights for the future development of modeling approaches for MSA and in particular contribute to the future direction of LEMMA's development. The experiment was conducted with 32 participants during a master course. The students worked on a complex modeling task involving LEMMA as well as UML and then completed a questionnaire.

The remainder of this paper is organized following the scheme for experiment reporting by Wohlin et al. [31]. Therefore, Section 2 gives an overview of related work. Section 3 presents the compared modeling approaches in a nutshell as background information. Section 4 describes the experimental design including the scoping and design of our research. Section 5 elaborates on execution of the experiment. In Section 6 we analysis and interpret the gathered data. Section 7 discusses the results and their implications. Section 8 describes threats to validity. Finally, Section 9 concludes the paper and gives an outlook on future work.

## 2 RELATED WORK

The development of modeling and programming languages for microservices in particular and service-based systems in general is an active field of research. For example, Düllmann and van Hoorn present a metamodel for benchmarking of MSA-based software systems [12], Montesi et al. introduce Jolie as a programming language for MSAs [20], and Terzic et al. present MicroBuilder for modeling REST-based microservice architectures [30]. However, up to the best of our knowledge all of these approaches have not been empirically evaluated which is the scope of this paper.

Haselböck et al. conducted an experiment broadly related to ours in which they studied the impact of decision models for decision documentation when building MSAs [17]. Although the experiment design is similar to ours, the purpose and, therefore, the gained insights concerning decision modeling are not related to structural modeling approaches for MSAs like the approaches which we compare (cf. Section 3).

Although not related to the domain of MSA, there are several empirical studies on modeling approaches. For example, Meliá et al. [19] evaluated the advantages and disadvantages of textual and graphical modeling notations. Barišić et al. evaluate a domain-specific visual query language in [5]. We built upon the results of Meliá et al. when constructing our questionnaire and used the definition of *Quality in Use* given by Barišić et al. for scoping our research purpose.

## 3 MODELING APPROACHES

UML is a widely used approach for the modeling of service-based software systems [2] and may also be employed to model the specifics of MSA-based software systems [27]. However, as a general-purpose modeling language, UML lacks specialized modeling constructs for microservices, which may result in ambiguous models as generic modeling elements of UML need to the adapted in order to capture the specifics of MSA. For this reason, we created LEMMA as a specialized modeling ecosystem for MSA that allows for concisely and efficiently expressing the relevant viewpoints of an MSA-based software system (cf. 1).

In the following, we briefly describe LEMMA and a UML-based approach which we both evaluate through comparison in our experiment.

### 3.1 LEMMA

LEMMA comprises a set of textual languages for model-driven MSA development. Each modeling language targets a specific *viewpoint* on MSA engineering. That is, each language provides semantic and syntactic elements in order to enable a certain stakeholder group to concisely express its concerns regarding an MSA-based software system [23]. Specifically, LEMMA implements languages that aim to support domain experts, service developers, and operators in the design, implementation, and operation of MSA-based software systems. LEMMA's modeling languages allow for leveraging various benefits of MDD when dealing with distributed software systems, e.g., fostering of stakeholder communication based on abstracted implementation details, static analysis, simulation prior to implementation, and code generation [10]. Furthermore, as LEMMA's languages are implemented with the Eclipse Modeling Framework[2], they integrate with the Eclipse IDE, hence increasing the usability for modelers with features such as syntax highlighting, code completion, and cross-referencing.

Specifically, LEMMA comprises the following modeling languages tailored to certain concerns of MSA engineering:

- *Domain Data Modeling Language*: This language enables domain experts and service developers to model domain-specific types, such as data structures and lists. It also integrates modeling elements for Domain-driven Design (DDD) patterns [14]. The language targets the *domain data viewpoint* [28] on MSA engineering.
- *Service Modeling Language*: This language allows service developers for modeling microservices, their interfaces, and operations. Moreover, it provides modeling elements for specifying endpoints, protocols, and data formats used for service interaction. It accounts for the *service viewpoint* [28] on MSA engineering.
- *Operation Modeling Language*: This language aims to support service operators in defining (i) microservices' deployment on containers, which is an established approach towards achieving self-containment of services [18]; (ii) infrastructure nodes being used by services, e.g., service discoveries and API gateways [4]; and (iii) configuration parameters of containers and infrastructure nodes. By implementing two distinct modeling languages for service developers and operators, LEMMA implicitly supports DevOps-based MSA engineering [21]. The Operation Modeling Language focuses on the *operation viewpoint* [28] on MSA engineering.

Next to these languages, the *Technology Modeling Language* is a means to capture technologies, e.g., for service implementation or deployment, in technology models and manage them separately

[2] https://www.eclipse.org/modeling/emf

from domain, service, and operation models [26]. However, participants of our experiment did not need to create a technology model themselves. Instead we provided them with a pre-built for the service implementation technology "Java" including Spring[3] and the service deployment technology Docker[4]. That is, in the following we only describe the syntax and semantics of those languages in detail, with which the participants were requested to create models on their own, i.e., the Domain Data, Service, and Operation Modeling Languages.

*3.1.1 Domain Data Modeling Language.* Listing 1 shows an example model created with LEMMA's Domain Data Modeling Language.

**Listing 1: Example LEMMA domain model.**

```
context TrailDomain {
structure Trail {
long id,
string name,
Points pois}
structure PointOfInterest {
long id,
string name,
string desc,
double latitude,
double longitude}
list Points {PointOfInterest poi}
}
```

Line 1 defines a context called `TrailDomain`. The Context modeling concept enables domain modelers to cluster coherent domain-specific types in separated namespaces. It is semantically equivalent to the Bounded Context pattern of DDD [14]. Lines 2 to 5 and lines 6 to 11 define two data structures, i.e., `Trail` and `PointOfInterest`. A data structure consists of typed and named data fields, e.g., `id` or `name`. LEMMA comprises the same primitive types as Java, and also treats strings and dates as primitive types for convenience reasons. Next to data structures, the Domain Data Modeling Language also implements a List concept (cf. line 12 in Listing 1) for expressing sequences of primitive or structured type instances.

*3.1.2 Service Modeling Language.* Listing 2 shows an example model created with LEMMA's Service Modeling Language.

**Listing 2: Example LEMMA service model.**

```
1  import datatypes from "trail.data" as trail
2  import technology from "java.technology" as java
3  @technology(java)
4  functional microservice org.example.TrailService {
5  @endpoints(java::_protocols.rest : "/trail";)
6  interface TrailInterface{
7  @endpoints(java::_protocols.rest : "/readTrail";)
8  readTrail(
9  sync in trailId : long,
10 sync out trail : trail::TrailDomain.Trail);
11 @endpoints(java::_protocols.rest : "/addPoi";)
12 addPointOfInterest(
13 sync in trailId : long,
14 sync in point: trail::TrailDomain.PointOfInterest);
15 } }
```

Line 1 of the service model imports the domain model shown in Listing 1. LEMMA integrates an import mechanism in order to make models reusable across MSA viewpoints (cf. Subsection 3.1). By importing the domain model, the service model can refer to the specified domain-specific types by their fully-qualified name, e.g., for typing service operation parameters. Line 2 of Listing 2 imports the `java` technology model (cf. Subsection 3.1), which clusters definitions related to microservice implementation with Java and Spring, e.g., technology-specific types and communication protocols. The technology model is then assigned in line 3 via the built-in `@technology` annotation to the `TrailService` microservice, which is modeled in lines 4 to 15 of Listing 2.

The microservice comprises an interface called `TrailInterface` (cf. line 6 in Listing 2), which defines two operations, i.e., `readTrail` and `addPointOfInterest` (cf. lines 8 and 12). Both operations take the `trailId` as input (cf. lines 9 and 13) as indicated by the `in` keyword of the Service Modeling Language for specifying operation parameters' directions. `readTrail` then returns the instance of the `Trail` data structure (cf. Listing 1) for the given trail ID. To this end, the outgoing `trail` parameter (cf. line 10) is typed by the data structure via its fully-qualified name, i.e., `TrailDomain.Trail`, which consists of the structure's name preceded by its surrounding context's name. To prevent name collisions, the fully-qualified data structure name also has to be preceded by the import alias of the defining domain model, i.e., `trail`, as specified in line 1 of Listing 2. `addPointOfInterest`, on the other hand, takes an instance of the `PointOfInterest` data structure as input (cf. line 14) and assigns it to the `Trail` instance identified by the passed trail ID.

Endpoints are assigned to the interface and its operations using LEMMA's built-in `@endpoints` annotation (cf. lines 5, 7, and 11 of Listing 2). An endpoint specification assigns an endpoint address to a protocol being specified within a technology model. All three endpoints of the example model rely on REST-based communication and the `java` technology model defines a corresponding protocol called `rest` together with a data format called `json`. Since the data format is the default format for the `rest` protocol, it can be omitted in the model. The protocol is referred in the endpoint specifications by its fully-qualified name, which consists of the protocol's name, the built-in namespace `_protocols`, and the alias of the imported technology model, i.e., `java` (cf. line 2 of Listing 2).

*3.1.3 Operation Modeling Language.* Listing 3 shows an example model created with LEMMA's Operation Modeling Language.

**Listing 3: Example LEMMA operation model.**

```
1  import microservices from "trail.services" as trailServ
2  import technology from "java.technology" as java
3  @technology(java)
4  container TrailServContainer
5  deployment technology java::_deployment.docker
6  deploys trailServ::org.example.TrailService {
7  default values { basic endpoints {
8  java::_protocols.rest: "http://example.com:8088";
9  } } }
10 @technology(java)
11 discovery is java::_infrastructure.serviceDiscovery
12 used by trailServ::org.example.TrailService {
13 endpoints {
14 java::_protocols.rest: "http://example.com:8089";
15 } }
```

In line 1 of Listing 3 the service model shown in Listing 2 is imported in order to model deployment-related information for the `TrailService`. Furthermore, the `java` technology model is imported in line 2, because it also defines the deployment technologies

[3] https://www.spring.io
[4] https://www.docker.com

being used during the experiment. In lines 3 and 10 the technology model is assigned to the subsequently declared container and infrastructure nodes.

Lines 4 to 9 specify the container `TrailServContainer`. It employs Docker as deployment technology (cf. line 5) and is responsible for deploying the imported `TrailService` (cf. line 6). In lines 7 to 9, the *basic endpoint* of the container is specified. That is, it denotes the physical network address and deployed microservices are reachable by preceding their logical endpoint addresses (cf. Subsubsection 3.1.2) with the container's basic endpoint address.

Additionally, lines 10 to 15 specify an infrastructure node called `discovery`. While containers deploy microservices, infrastructure nodes provide the architecture with infrastructure-related capabilities, and may be used by a variety of services, containers, and other infrastructure nodes. For instance, the `discovery` node in Listing 3 is a `serviceDiscovery` in the sense of the `java` technology model. That is, it provides microservices with the possibility to discover and interact with each other [4]. The `used by` directive in line 12 determines that the `TrailService` uses the discovery node for discovery purposes. Lines 13 to 15 of Listing 3 then model the physical endpoint of the service discovery.

## 3.2 UML

As general purpose language UML has the means to model an MSA in various ways and thus can be used as an alternative to LEMMA. In order to address the same viewpoints as previously described with LEMMA (cf. Subsection 3.1), we propose the usage of three different diagram types for each viewpoint in alignment with a previous work [27]. The viewpoints of LEMMA translate to UML diagrams as follows:

**Domain Viewpoint** is modeled using the *class diagram* type.
**Service Viewpoint** relies on the *component diagram* type.
**Operation Viewpoint** utilizes the *deployment diagram* type.

Figure 1 shows the same microservice architecture modeled with LEMMA (cf. Subsection 3.1) using the proposed diagram types.

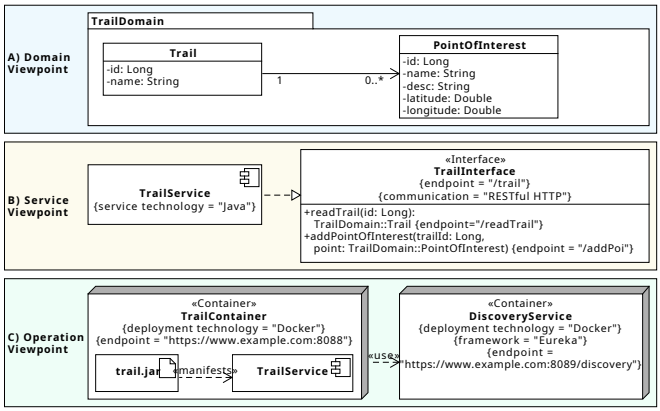


**Figure 1: Exemplary representation of the UML approach with the three diagram types representing the viewpoints.**

Although the UML approach includes the same information, it differs in comparison to modeling with LEMMA through its graphical notation. In order to address MSA's specific characteristics the approach relies on *tagged values*, e.g., for endpoint addresses or deployment technologies. Also, we introduced the custom stereotype *container* because we found that the default *device* type is misleading in the context of MSA, since a microservice is usually not deployed on a specific device but through containerization [21].

# 4 EXPERIMENTAL DESIGN

In order to compare the LEMMA and UML modeling approaches, we define the research scope as well as the design of the experiment using the guideline given by Wohlin et al. [31] as the following.

## 4.1 Research Scoping & Goals

*Object of Study.* The object of study is the participants, i.e., their experience and results during the modeling of MSA.

*Purpose.* The main purpose of the study is to evaluate the *Quality in Use* of LEMMA for modeling MSA in comparison with the UML-based approach (cf. Section 3).

*Quality Focus.* The quality focus is given by the *Quality in Use* definition provided by Barišić et al. [5]. It can be divided into the subcategories *effectiveness*, *efficiency*, *satisfaction*, and *accessibility*.

*Perspective.* The perspective is from the point of view of the researchers, i.e., the researchers want to know if there is a significant difference between the Quality in Use of both modeling languages as well as the general experience of modeling of MSA.

*Context.* The experiment is run as an assignment during a master course in computer science given in 2019 over the period of one semester at the *omitted per double blind reviewing*

. The course focuses on the general concepts of microservices, their implementation with the Java Spring framework[5], and their modeling. Both modeling approaches (cf. Section 3) were presented and exercised in multiple sessions, so that it can be assumed that the students are familiar with both approaches and are able to apply them. As modeling means for LEMMA, the participants mandatory used *Eclipse IDE for Java and DSL Developers*[6] in version 2019-03. For modeling with UML *Visual Paradigm*[7] in version 15.0 was mandatory. The students are partially familiar with Visual Paradigm because it is regularly used in undergraduate courses at our university. The assignment was placed at the end of the course, whereby the students were able to achieve partial points for the course exam based on their performance as incentive.

*Goal Summary.* According to the GQM template originally provided by Basili & Rombach [6] the goal of our student experiment can be summarized as the following:

Analyze *the modeling experience* for the purpose of *evaluating LEMMA in comparison with UML* with respect to their *Quality in Use* from the point of view of the *researchers* in the context of *an assignment during a master course in computer science.*

[5] https://spring.io/
[6] https://www.eclipse.org/downloads/packages/release/2019-03/r/eclipse-ide-java-and-dsl-developers
[7] https://www.visual-paradigm.com/

## 4.2 Research Design

*Hypothesis.* Based on the goal definition, we derive the general hypothesis that LEMMA *Quality in Use* is higher when modeling an MSA compared to UML. This informal hypothesis statement can be formalized in accordance with the scope of our experiment into four formal hypotheses each addressing one component of the *Quality in Use* as follows.

(1) Null hypothesis, $H_0$ : The effectiveness of the master students modeling an MSA using LEMMA is less or equal to the UML approach.
$H_0$ : effc(LEMMA) ≤ effc(UML).
Alternative hypothesis, $H_1$ : effc(LEMMA) > effc(UML).
Measures needed: Used modeling approach (LEMMA or UML) and effectiveness.
(2) Null hypothesis, $H_0$ : The efficiency of the master students modeling an MSA using LEMMA is less or equal to the UML approach.
$H_0$ : effi(LEMMA) ≤ effi(UML).
Alternative hypothesis, $H_1$ : effi(LEMMA) > effi(UML).
Measures needed: Used modeling approach (LEMMA or UML) and efficiency.
(3) Null hypothesis, $H_0$ : The satisfaction of the master students using LEMMA as modeling means is less or equal compared to the UML approach.
$H_0$ : sat(LEMMA) ≤ sat(UML).
Alternative hypothesis, $H_1$ : sat(LEMMA) > sat(UML).
Measures needed: Used modeling approach (LEMMA or UML) and satisfaction (cf. Paragraph 4.1).
(4) Null hypothesis, $H_0$ : The accessibility of LEMMA is less or equal compared to the UML approach for the students.
$H_0$ : acs(LEMMA) ≤ acs(UML).
Alternative hypothesis, $H_1$ : acs(LEMMA) > acs(UML).
Measures needed: Used modeling approach (LEMMA or UML) and accessibility (cf. Paragraph 4.1).

This means that the applied modeling approach (LEMMA or UML) and the *Quality in Use*'s components need to be measured. For the latter's evaluation, we asked the participants regarding efficiency, satisfaction, and accessibility using a questionnaire. Furthermore, we analyzed each model artifact created during the assignment and used the number of modeling errors as measurement for effectiveness.

*Variable Selection.* The independent variable is the applied modeling approach. The dependent variables are effectiveness, efficiency, satisfaction, and accessibility.

*Subjects.* The subjects are chosen based on convenience, i.e. the subjects are students taking the master program's microservice course. They are a sample from all students at the master program, but no random sample.

*Design.* Based on the defined hypotheses, measures, and variables we identified one factor (MSA modeling) and two treatments (LEMMA and UML) as key factors for our research design. Since the number of subjects of the experiment is limited to the number of course participants, we used a **paired comparison design** [31] to increase measurements for each treatment. Hereby, each student uses both treatments, i.e., LEMMA and UML, on the same object, i.e., the given assignment. This allows us to have as many measurements for each treatment as we have participants. However, this design faces the challenge of possible learning effects during the exposure to the treatments, i.e., participants who first used one modeling approach could learn from that experience and achieve better results using the second approach not because of the approach itself but because of learning effects. We addressed these issue following the general design principles [31]. We divided the subjects into two *balanced* groups, each starting with a different treatment. Therefore, we achieve a **balanced design**. We applied *randomization* during the assignment to determine with which treatment the participants would start. As we addressed the possibility of learning effects with applying the balanced design, we did not apply any further *blocking*.

*Instrumentation.* All data is gathered based upon a complex modeling assignment during the end of the course. It comprises a textual description containing a scenario with multiple information which need to be addressed in an MSA model. The scenario description is based upon a real world project of the tourism department of the City of Dortmund *omitted due to double blind reviewing* managing city trails which connect multiple points of interest for hiking. We provide a translated version of the assignment on GitHub[8].

As instruments to measure the *Quality in Use* components as well as the applied modeling approach, a questionnaire (*subject-completed instrument*) and a review of the created models (researcher-completed instrument) is used. The questionnaire (cf. Appendix A) was designed using closed raiting questions with the objective to collect data focusing on efficiency, satisfaction, and accessibility of the modeling approaches. In total, the questionnaire comprises three sections. First we gather general information. In the following section we assess the *Quality in Use* components by asking the student's degree of agreement to 14 statements. We ask this for UML and LEMMA, respectively. In the final section we ask the students about there preferred approach using forced and unforced binary questions [11]. In total the questionnaire comprises 43 items. A translated version of the questionnaire can be found in Appendix A. For reviewing the resulting model we rigorously cross-evaluated each modeling artifact of the assignment per student.

The resulting data is analyzed with descriptive statistics (bar plots and box plots). Hypothesis testing is done for $H_0$ and $H_1$ of all four hypotheses.

## 5 EXECUTION

The experiment was executed in the last week of the course whereby the necessary knowledge regarding both modeling approached was taught in the previous three lectures. In this preparation weeks, we provided the students with two exercises which were similar in structure to the actual task in the experiment. By doing this, we wanted to ensure that all students knew the scope of the task and that they were familiar with the hardware environment of the software laboratory in which we held the lectures of the course and performed the experiment. In the lecture before the experiment, we informed the students that we intended to gather their feedback in

[8] https://github.com/EASE2020auxmat/material/

form of a questionnaire in order to compare the taught approaches with each other and that completing the questionnaire is voluntary and does not affect their exam scores. They were not informed about our hypotheses, the analytic procedure, or the design of the questionnaire. During the experiment the students received the textual scenario description as well as the questionnaire with the request to fill it after they had finished their assignment. In accordance to the balanced design decision we prepared the textual task in two versions. Version A started with the instruction to first create a model of the described system using UML and afterwards create a model of the same system using LEMMA. Version B instructed the reverse case. We stressed the students that it would be necessary to perform the tasks in the given order and ensured this by observation while the students were working on the task. In total the students had 80 minutes to finish the assignment and upload the created models to our university's e-learning platform.

The experiment material comprising the scenario description, the questionnaire, and the lecture material was created in multiple planning sessions and pre-tested to ensure a reasonable timeframe for the students and prevent flaws due to incorrect wording.

Overall 32 students participated in the experiment of which 31 handed in a questionnaire. One student was not able to use the infrastructure properly because he did not attend any of the previous lectures and did no exercises. Because he did not met the experiment requirement of being familiar with the use of both approaches we removed his data from the set leaving 31 observations for the analysis of modeling artifacts and 30 for the questionnaire. 3 of these 30 did not complete the repetitive section for both modeling approaches of the questionnaire, therefore we performed the analysis and interpretation concerning items in the repetitive section only with data from 27 students. If no item of the repetitive section was concerned, we analyzed all 30 questionnaires.

## 6 ANALYSIS & INTERPRETATION

This section presents the analysis of the gathered data and its interpretation. The data is available on GitHub[9].

### 6.1 Descriptive Statistics

We analyze the data using descriptive statistics with respect to our hypotheses that the *Quality in Use* for modeling MSA is higher using LEMMA compared to UML. As data sources we leverage the created modeling artifacts of each student as well as their questionnaire which comprises the repetitive (items 12 to 25 for LEMMA and 26 to 39 for UML) and the comparison section (items 40 to 43). Figure 4 depicts the data of the repetitive sections as box-and-whisker plots [9] constructed for each statement and grouped by similarity, i.e., the items are grouped in such a way that the values of agreement regarding the same statement for LEMMA and UML are next to each other. The degree of agreement uses a scale from: 1 (completely agree) to 6 (completely disagree) (cf. Appendix A). Due to the ordinal nature of most items we rely on the median value (*Mdn*) instead of the mean average (*M*). In the following, we discuss each *Quality in Use* component in a single paragraph.

[9]https://github.com/EASE2020auxmat/material/

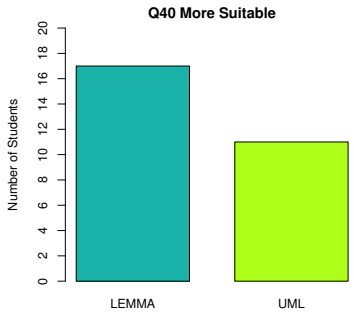


**Figure 2: Number of students preferring a certain approach overall (forced choice, $n = 28$).**

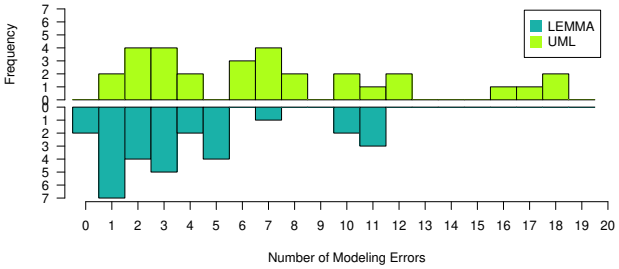


**Figure 3: Mirrored histograms of made modeling errors in comparison ($n = 30$).**

*Effectiveness.* We rely on correctness of created modeling artifacts as measurement for the effectiveness of both modeling approaches and therefore diligently evaluated each of the students' uploaded modeling artifact (cf. Section 5). Figure 3 shows the distribution of the total number of errors identified during the evaluation process as a mirrored histogram for LEMMA and UML. The median amount of modeling errors for UML is 6.5 ($M = 7.2$, $SD = 5.14$) whereby LEMMA's is 3.0 ($M = 3.93$, $SD = 3.46$). We further categorized the errors by whether a student made a similar error in both diagrams and how many of the errors were of a syntactical nature. Thus, we are able to identify errors committed exclusively in only one of the two modeling approaches per student. The median value for exclusive errors done by students is 1.0 for LEMMA ($M = 1.10$, $SD = 1.56$) and 4.0 for UML ($M = 4.37$, $SD = 3.34$). The difference can be partially explained looking at the syntactic errors in particular. Students did no syntactical errors using LEMMA at all ($Mdn = 0.0$, $M = 0.00$, $SD = 0.00$). In comparison, more syntactical errors occurred with the UML approach ($Mdn = 1.00$, $M = 0.94$, $SD = 1.36$). The data indicates that students make fewer errors when modeling with LEMMA, i.e., they create models with a higher effectiveness, using LEMMA compared to the UML approach. When the students make a modeling mistake using the LEMMA approach they probably do the same mistake in UML but not vice versa. Students make mistakes that are exclusive to modeling with the UML approach. Probably a portion of these errors can be classified as syntactical errors. This is additionally supported using the answers from the questionnaire's comparison section. Figure 5 shows in the left bar chart that while most students

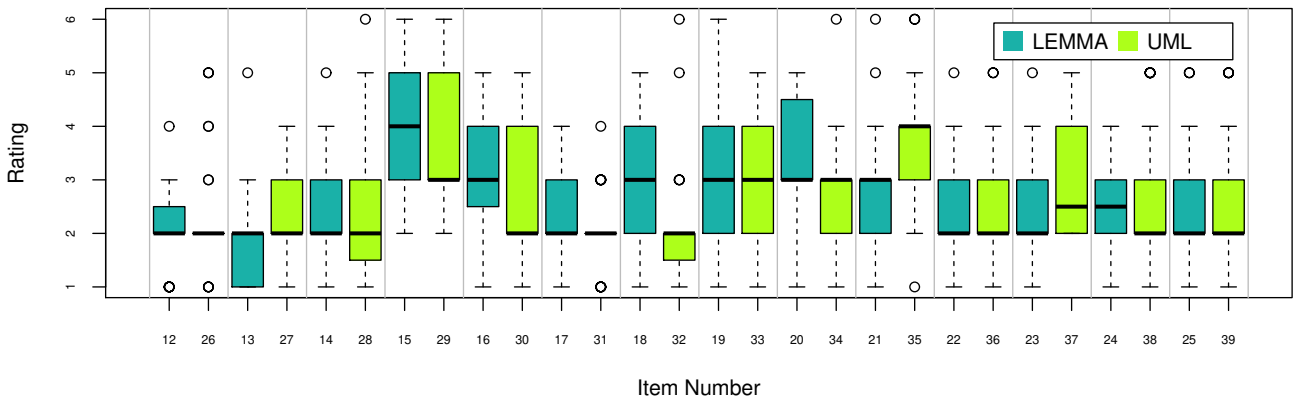


**Figure 4: Boxplots of items in the repetitive section of the questionnaire grouped by similar questions ($n = 28$).**

($n = 14$) think both approaches are equally precise, a third of the students exclusively prefer LEMMA ($n = 12$) over UML ($n = 4$) in terms of precision.

*Efficiency.* As means to measure efficiency we rely on self assessment using the questionnaire. As depicted in the right bar chart in Figure 5, in direct comparison most of the students state the LEMMA approach as being more efficient ($n = 14$) followed by UML ($n = 9$). A quarter of the students think both are equally efficient ($n = 7$). Another measurement for the efficiency becomes visible by comparing the degree of agreement to the statement whether the required effort is in proportion to the benefits for LEMMA (Item 23) and for UML (Item 37). As the box plot in Figure 4 shows, the median degree of agreement for LEMMA is 2.0 ($M = 2.46$) while the median for UML is 2.50 ($M = 3.00$). The direct comparison as well as compared answers of items 23 and 37 indicate that the students think of LEMMA being slightly more efficient than the UML approach for modeling MSA.

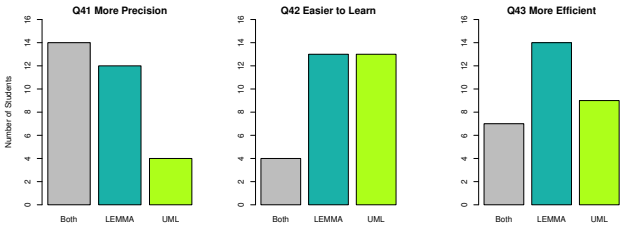


**Figure 5: Bar charts showing the number of students on the y-axis regarding their preference of modeling approaches ($n = 30$).**

*Satisfaction.* Satisfaction is measured in both the comparative and repetitive sections of the questionnaire. As shown in Figure 2 60 percent of the students think of LEMMA being the generally more suitable approach for modeling MSAs compared to UML ($n = 28, LEMMA = 17, UML = 11$). In the repetitive section, a comparably large number of items deal with the positive attitude and freedom of inconvenience which are both parts of the general satisfaction regarding a modeling approach [5]. As shown in Figure 4, the median agreement for the approach being suitable for modeling MSAs (Item 12/26) as well as being able to describe them well (Item 13/27) are for both approaches 2.0 (Item 12/26: $M_{LEMMA} = 2.11$, $M_{UML} = 2.28$, Item 13/27: $M_{LEMMA} = 1.93$, $M_{UML} = 2.25$). The similarity in satisfaction in the usage of both approaches is further backed by other items, for example Item 20/34. Here, the students slightly agree to the negative statement that it is easy to make modeling mistakes for both approaches ($Mdn = 3.0$). A difference can be seen for the agreement that it is easy to spot errors (Item 21/35) to which students slightly agree for LEMMA ($Mdn = 3.0$, $M = 2.79$) and slightly disagree for UML ($Mdn = 4.0$, $M = 3.64$). Overall the data indicates that the students are satisfied with both approaches, although spotting errors in LEMMA seems to be slightly easier compared to UML.

*Accessibility.* Four items of the repetitive section and one compare question are used to access the accessibility of the modeling approaches in the questionnaire. When asked which approach is easier to learn the respondents nearly equally divide in two halves as depicted in the center bar chart of Figure 5. 13 students think LEMMA is easier to learn while 13 prefer UML. Only 4 people think both approaches are equally easy to learn. In the repetitive section of the questionnaire most students agree that UML and LEMMA are easy to memorize (Item 14/28: $Mdn_{LEMMA} = Mdn_{UML} = 2.0$, $M_{LEMMA} = M_{UML} = 2.32$). However, students agree more to UML being intuitive than to LEMMA being intuitive (Item 16/30: $Mdn_{LEMMA} = 3.0$, $M_{LEMMA} = 3.00$, $Mdn_{UML} = 2.0$, $M_{UML} = 2.68$). They agree that the approach is easy to understand for both approaches (Item 17/31: $Mdn_{LEMMA} = 2.0$, $M_{LEMMA} = 2.32\, Mdn_{UML} = 2.0$, $M_{UML} = 2.07$). Overall the results and the impression of the box plots for the mentioned items in Figure 4 suggest that the students attest a good accessibility to both approaches.

## 6.2 Data Set Reduction

In order to perform hypotheses testing using the means of inferential statistics we reduce the number of variables in the data set

**Table 1: Combined variables for hypotheses testing.**

| Component | Included Measurements |
|---|---|
| **Effectiveness** | Number of modeling errors |
| **Efficiency** | Items 22/36,23/37, 25/39 |
| **Satisfaction** | Items 12/26, 13/27, 18/32, 19/33, 24/38 |
| **Accessibility** | Items 14/28, 16/30, 17/31, 21/35 |

because several variables have a high correlation. This is due to multiple items aim at the similar components of the *Quality in Use*, e.g., the statements "the usage of «modeling approach» is intuitive" (Item 16/30) and "«modeling approach» is easy to understand" (Item 17/31) measure both accessibility as component of the *Quality in Use*. Therefore, we reduce the data set in such a way that only one single value exists for each key component of the *Quality in Use*. Furthermore, we only use the repetitively assessed items and the modeling errors. We also removed statement items 15/29 and 20/34 because they are negative items included as control mechanism to identify response tendencies in the questionnaire. The variables are combined using the mean average of all concerning item values. The combination is done as shown in Table 1.

## 6.3 Hypotheses Testing

The first hypothesis regarding higher effectiveness for master students using the LEMMA modeling approach compared to UML is evaluated using Wilcoxon's signed-rank test with significance level of $\alpha = 0.05$. We use this test because we applied both treatments on the same subject group and visualized data does not look normally distributed which is mandatory for using the more common t-test [31]. For validation we performed a Shapiro-Wilk test [15] for normality on UML and LEMMA errors (both $p - values < \alpha; \alpha = 0.05$) which supports the impression of a non-normal distribution. The test result for the signed-rank test is shown in Table 2. The second,

**Table 2: Results from the Wilcoxon signed-rank test for LEMMA versus UML.**

| Variable | V-value | p-value |
|---|---|---|
| **Effectiveness** | 0 | 8.087e-06 |

third and fourth hypotheses regarding the efficiency, satisfaction, and accessibility for master students using LEMMA modeling MSAs compared to the usage of UML are tested with a paired one-tailed t-test with significance level of $\alpha = 0.05$. We assume normal distribution. Results of the t-tests are shown in Table 3.

From Table 2 it can be concluded that $H_0$ of Hypothesis 1 is rejected. The effectiveness of master students modeling MSA using the LEMMA is significantly greater compared to the UML modeling approach. As the p-value is very low the effect is highly significant.

**Table 3: Results from the paired t-tests for LEMMA versus UML.**

| Variable | Mean diff. | df | t-value | p-value |
|---|---|---|---|---|
| **Efficiency** | -0.274 | 27 | -1.1235 | 0.8644 |
| **Satisfaction** | 0.064 | 27 | 0.33691 | 0.3694 |
| **Accessibility** | -0.008 | 29 | -0.039455 | 0.5156 |

From Table 3 it can be concluded that each $H_0$ of Hypotheses 2 to 4 is not rejected. Efficiency, satisfaction, and accessibility of master students modeling MSA do not significantly improve using LEMMA compared to UML.

# 7 DISCUSSION

In our experiment and its analysis we have investigated whether the *Quality in Use* of the LEMMA modeling approach is higher compared to the UML approach which are both presented in Section 3.

With the means of descriptive as well as inductive statistics we are able to show that master students who were taught both approaches during a microservice course produce MSA models with a higher effectiveness, i.e., they make fewer modeling errors, using LEMMA. This is consistent with our previous assumptions because textual notations are generally characterized as being easier to analyze and check for consistency [25]. We assume that one major reason for the better effectiveness is the syntax highlighting and auto completion provided by the Eclipse IDE when creating LEMMA models. This would also explain the virtually non-existence of syntax errors in LEMMA. Although Visual Paradigm provides means to verify for syntactical correctness of UML models, we assume that students simply did not use these mechanisms or intentionally omitted them. However, it can be argued whether modeling errors are generally a reasonable means to measure effectiveness depending on the intended purpose of the created model. For example, in order to explain a relationship between two microservices, to a human it may not be necessary that an association is depicted using the wrong arrowhead, while this is highly relevant when the intended purpose is to generate source code based on the model.

Concerning *Quality in Use*'s other components efficiency, satisfaction, and accessibility we were not able to show that students significantly prefer LEMMA to UML. Although the hypotheses tests (cf. Section 6.3) do not allow a conclusion as to whether both approaches are equal or whether the UML approach is superior to LEMMA regarding the three components, the descriptive analysis indicates towards equality.

Although the LEMMA approach performed comparable to UML, this is not consistent with our assumption that students perform the modeling with a higher efficiency using LEMMA because of LEMMA's domain specificity and the students familiarity with programming in general. We think of three possible explanations: (1) there could actually be no difference in the efficiency of the modeling approaches; (2) we did not objectively measure the efficiency, e.g. by measuring time, but relied on self-assessment using the questionnaire, this could have led to distortions; and (3) the task in the experiment may has been too small, since certain disadvantages like the demand of huge on-screen space of graphical notations [16] only become apparent in larger real world applications. Other explanations may also be found, therefore, we argue that further research is needed.

We falsely assumed that the students are more satisfied with LEMMA compared to the UML modeling approach as well as LEMMA providing a better accessibility. Although there may be several factors which further influence the satisfaction and accessibility, like

the fact that the students were comparably inexperienced with developing MSAs, we take the results as an impulse to optimize LEMMA in the future especially with regard to the user experience.

## 8 THREATS TO VALIDITY

According to Campbell et al. [8] threats to four different kinds of validity need to be considered in our experiment, namely internal, external, conclusion, and construct validity. In the following we address each of these regarding our conducted experiment.

Since all students of the course took part in the experiment, we assess the *internal validity* as little threatened. A possible threat pose the learning effects during our experiment because we chose the paired comparison design. We tried to address this threat by striving for a balanced design in which we randomly assigned with which modeling approach the students had to start the assignment (cf. Section 4).

Concerning *external validity* it is highly likely that similar results can be obtained when running the course in a similar way. However, it is probably difficult to generalize the results to other students that do not have the necessary understanding of the MSA domain and therefore can not understand the domain-specific constructs in LEMMA. It should be possible to generalize the results to other students who have the knowledge about MSA.

The major threat to the *conclusion validity* is the fact that we were not able to gather the data completely anonymous. Because we awarded points for the course exams based on the correctness of the created models as incentive for participation, the students had to submit the models using their account for the universities e-learning platform. We addressed this threat by assuring that the modeling artifacts were evaluated by two persons not involved in the lectures and anonymized before further processing. We made this known to the students before they had started the assignment.

Concerning the *construct validity*, which deals with the operationalization of the measures in the study in relation to the constructs in the real world, we identify two major threats: an improper designed questionnaire and the measurement of effectiveness using errors in the submitted models. As explained in Section 7 what is called effective concerning a model is highly dependent on the intended purpose. However, the underlying paradigm in MDD is to use models as first class citizens in the development process, e.g., to generate source code, and thus models have a need for formal correctness. Therefore, we argue that errors are a fitting measurement for effectiveness. Regarding the design of the questionnaire we rigorously applied provided guidelines with the question design and stick to a simple design relying on closed items (cf. Section 4). However, the evaluation of modeling approaches clearly lacks standardized methods for quality measurement.

## 9 CONCLUSION & FUTURE WORK

We conducted an experiment study to evaluate the LEMMA modeling approach. LEMMA comprises a set of domain-specific modeling languages to describe an MSA. We compared LEMMA with using a UML-based approach. The experiment consisted of an assignment in a master course in which 32 students in informatics created an MSA model using LEMMA and UML. We designed the experiment as a balanced paired comparison design. For assessing the *Quality in Use* of the modeling approaches we measured the effectiveness, efficiency, satisfaction, and accessibility with a questionnaire and by evaluating the modeling artifacts created by the students during the assignment.

The results show that students make fewer mistakes in modeling MSAs when using LEMMA which indicates that LEMMA has a better effectiveness than using UML as modeling means for MSA. However, the results do not show that LEMMA is significantly superior to UML regarding efficiency, satisfaction, and accessibility.

Although the results were obtained by only one observation and the experiment was conducted in a university environment, we clearly identify the user experience as an essential field for future improvements of LEMMA. Therefore, we plan to investigate the directions for these improvements with qualitative research, e.g., by conducting in-depth interviews with LEMMA users or using focus groups.

## APPENDIX A: SURVEY ITEMS

In the following we present a translated and shortened version of the questionnaire. The repetitive section was asked twice in a row within the questionnaire for the respective modeling approach (items 12 to 25 for LEMMA; 26 to 39 for UML). As response option in the repetitive section we use a ranking from 1 to 6 with the following scale: 1 (completely agree), 2 (agree), 3 (slightly agree), 4 (slightly disagree), 5 (disagree), 6 (completely disagree). In the comparison section we use single choice as response option with LEMMA (L), UML (U) or both.

| Item | Description | Response Option |
|---|---|---|
| **Demographic Information** | | |
| 1 | University handle | numeric |
| 2 | Variant of the questionnaire | A\|B |
| 3 | Master semester | numeric |
| 4 | Field of study of the Bachelor degree | list |
| 5 | Years of experience in software development | numeric |
| 6 | Self-assessment of modeling skills in general | ranking |
| 7 | Years of experience with UML | numeric |
| 8 | Self-assessment of UML skills | ranking |
| 9 | Years of experience with MDD | numeric |
| 10 | Self-assessment of MDD skills | ranking |
| 11 | Already familiar modeling languages | free text |
| **Repetitive Section** | | |
| 12/26 | I think «approach» is good for describing MSAs. | agreement |
| 13/27 | With «approach» the essential characteristics of an MSA can be described well. | agreement |
| 14/28 | I can remember the language elements of «approach» well. | agreement |
| 15/29 | «approach» is difficult to learn. | agreement |
| 16/30 | The use of «approach» is intuitive. | agreement |
| 17/31 | «approach» is easy to understand. | agreement |
| 18/32 | I can imagine using «approach» for future projects. | agreement |
| 19/33 | Planning errors in the architecture can be prevented by using «approach». | agreement |
| 20/34 | It is easy to make mistakes when modeling with «approach». | agreement |
| 21/35 | Modeling errors can be easily detected in «approach». | agreement |
| 22/36 | «approach» is a helpful modeling language in the context of MSAs. | agreement |
| 23/37 | The effort required to describe an MSA using «approach» is reasonable compared to the benefits. | agreement |
| 24/38 | The use of «approach» has a positive effect on a later implementation. | agreement |
| 25/39 | The use of «approach» is worthwhile for the development of MSAs. | agreement |
| **Comparison Section** | | |
| 40 | More suitable for modeling MSA? | L\|U |
| 41 | More precise? | Both\|L\|U |
| 42 | Easier to learn? | Both\|L\|U |
| 43 | Greater benefit in relation to required effort? | Both\|L\|U |

## REFERENCES


[1] Nuha Alshuqayran, Nour Ali, and Roger Evans. 2016. A Systematic Mapping Study in Microservice Architecture. In *Proc. of the 9th Int. Conf. on Service-Oriented Computing and Applications (SOCA)*. IEEE, 44–51.

[2] David Ameller, Xavier Burgués, Oriol Collell, Dolors Costal, Xavier Franch, and Mike P. Papazoglou. 2015. Development of service-oriented architectures using model-driven development: A mapping study. *Information and Software Technology* 62, 1 (2015), 42–66.

[3] Anonymous Authors. Omitted per double blind reviewing. [n.d.].

[4] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software* 33, 3 (2016), 42–52.

[5] Ankica Barišić, Vasco Amaral, Miguel Goulão, and Bruno Barroca. 2011. Quality in use of domain-specific languages. In *Proceedings of the 3rd ACM SIGPLAN workshop on Evaluation and usability of programming languages and tools - PLATEAU '11*. ACM Press.

[6] Victor R Basili and H Dieter Rombach. 1988. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering* 14, 6 (1988), 758–773.

[7] J. Bogner, J. Fritzsch, S. Wagner, and A. Zimmermann. 2019. Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 187–195.

[8] Donald T. Campbell and Julian C. Stanley. 1966. *Experimental and Quasi-Experimental Designs for Research*. R. McNally.

[9] John M. Chambers, William S. Cleveland, Beat Kleiner, and Paul A. Tukey. 1983. Graphical Methods for Data Analysis. (1983).

[10] Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim Steel, and Didier Vojtisek. 2017. *Engineering Modeling Languages: Turning Domain Knowledge into Tools*. CRC Press.

[11] Sara Dolnicar, Bettina Grün, and Friedrich Leisch. 2011. Quick, Simple and Reliable: Forced Binary Survey Questions. *International Journal of Market Research* 53, 2 (March 2011), 231–252.

[12] Thomas F. Düllmann and André van Hoorn. 2017. Model-driven Generation of Microservice Architectures for Benchmarking Performance and Resilience Engineering Approaches. In *Proc. of the 8th Int. Conf. on Performance Engineering Companion (ICPE)*. ACM, 171–172.

[13] Thomas Erl. 2005. *Service-Oriented Architecture (SOA) Concepts, Technology and Design*. Prentice Hall.

[14] Eric Evans. 2004. *Domain-Driven Design*. Addison-Wesley.

[15] Asghar Ghasemi and Saleh Zahediasl. 2012. Normality tests for statistical analysis: a guide for non-statisticians. *International journal of endocrinology and metabolism* 10, 2 (2012), 486.

[16] Thomas Goldschmidt, Steffen Becker, and Axel Uhl. 2008. Classification of Concrete Textual Syntax Mapping Approaches. In *Model Driven Architecture – Foundations and Applications*. Springer, 169–184.

[17] Stefan Haselböck, Rainer Weinreich, and Georg Buchgeher. 2019. Using Decision Models for Documenting Microservice Architectures: A Student Experiment and Focus Group Study. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 37–3709.

[18] Nane Kratzke and Peter-Christian Quint. 2017. Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study. *Journal of Systems and Software* 126 (2017), 1–16.

[19] Santiago Meliá, Cristina Cachero, Jesús M. Hermida, and Enrique Aparicio. 2016. Comparison of a textual versus a graphical notation for the maintainability of MDE domain models: an empirical pilot study. *Software Quality Journal* 24, 3 (01 Sep 2016), 709–735.

[20] Fabrizio Montesi, Claudio Guidi, and Gianluigi Zavattaro. 2014. Service-oriented programming with jolie. In *Web Services Foundations*. Springer, 81–107.

[21] Irakli Nadareishvili, Ronnie Mitra, Matt Mclarty, and Mike Amundsen. 2016. *Microservice Architecture*. O'Reilly Media.

[22] Sam Newman. 2015. *Building Microservices*. O'Reilly Media.

[23] Object Management Group. 2014, Version 2.0. *Model Driven Architecture (MDA) Guide*. Object Management Group. https://www.omg.org/cgi-bin/doc?ormsc/14-06-01

[24] Object Management Group. 2017. OMG Unified Modeling Language (OMG UML). Version 2.5.1 formal/2017-12-05 (2017).

[25] Marian Petre. 1995. Why Looking Isn'T Always Seeing: Readership Skills and Graphical Programming. *Commun. ACM* 38, 6 (June 1995), 33–44.
[26] F. Rademacher, S. Sachweh, and A. Zündorf. 2019. Aspect-Oriented Modeling of Technology Heterogeneity in Microservice Architecture. In *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 21–30.
[27] F. Rademacher, J. Sorgalla, and S. Sachweh. 2018. Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective. *IEEE Software* 35, 3 (2018), 36–43.
[28] F. Rademacher, J. Sorgalla, S. Sachweh, and A. Zündorf. 2019. Viewpoint-Specific Model-Driven Microservice Development with Interlinked Modeling Languages. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 57–5709.
[29] A. Singleton. 2016. The Economics of Microservices. *IEEE Cloud Computing* 3, 5 (Sep. 2016), 16–20.
[30] Branko Terzić, Vladimir Dimitrieski, Slavica Kordić, Gordana Milosavljević, and Ivan Luković. 2017. MicroBuilder: A Model-Driven Tool for the Specification of REST Microservice Architectures. In *Proc. of the 7th Int. Conf. on Information Society and Technology (ICIST)*. ICIST.
[31] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Berlin Heidelberg.
[32] Olaf Zimmermann. 2017. Microservices tenets. *Computer Science - Research and Development* 32, 3 (Jul 2017), 301–310.