Institute of Software Engineering
Software Quality and Architecture

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master’s Research Project

Analyzing the Comprehensibility of the Visualized Deployment Model Provided by DeMAF

Felix Heidrich, Rene Tischler, and Marcel Wolkober

Course of Study:	Software Engineering
Examiner:	Prof. Dr.-Ing. Steffen Becker
Supervisor:	Marcel Weller, M.Sc.

Abstract

The deployment of cloud-based applications is often a complex task for stakeholders in the deployment process. One aspect of this complexity is the wide range of deployment technologies available to choose from. To address this particular challenge, the Deployment Model Abstraction Framework (DeMAF) by Weller et al. was developed. The DeMAF allows these Technology-Specific Deployment Models to be transformed into an agnostic model that conforms to the Essential Deployment Metamodel (EDMM) and can be visualized using Eclipse Winery to improve comprehensibility.

This research project analyzed the comprehensibility and usability of the visualizations generated by the DeMAF in a controlled study in comparison to a Kubernetes visualization tool – Monokle. For this purpose, an intuitive web user interface was developed that allowed the users to analyze two Kubernetes-based deployment models.

The results show that the DeMAF significantly improves the accuracy of the user's responses, for simpler deployment models, while reducing the perceived cognitive load. While the average task completion times of both tools are similar, participants consistently rated the DeMAF more positively in terms of comprehensibility, ease of use, and visual clarity. The preference for the DeMAF was particularly clear among participants with little prior knowledge of deployment. Overall, the study shows the potential of the technology-agnostic, semantically enriched visualization approach of the DeMAF to present deployment models in a more comprehensible way without creating additional cognitive burden. The results underline the practical benefits of the DeMAF both for educational and practical deployment contexts.

Contents

List of Acronyms

API Application Programming Interface

DeMAF Deployment Model Abstraction Framework

EDMM Essential Deployment Metamodel

FOSS free and open-source software

GUI graphical user interface

IDE integrated development environment

K8s Kubernetes

NASA-TLX NASA Task Load Index

REST Representational State Transfer

SLO Service Level Objective

TADM Technology-Agnostic Deployment Model

TOSCA Topology and Orchestration Specification for Cloud Applications

TSDM Technology-Specific Deployment Model

UI user interface

1. Introduction

In recent years the options to deploy cloud-based applications have changed and become a complex task for stakeholders in the deployment process [VB18]. One main problem is the availability of many different deployment technologies, such as Kubernetes (K8s), Helm, Ansible, Terraform, and others, to choose from, each requiring in-depth knowledge [WBF+20].

A solution to this problem is using a Technology-Agnostic Deployment Model (TADM), i.e., a model that is independent of the specific deployment technology, like a TADM based on Essential Deployment Metamodel (EDMM) [WBF+20]. The Deployment Model Abstraction Framework (DeMAF) by Weller et al. [WBSB23] was developed to transform a Technology-Specific Deployment Model (TSDM) into a EDMM model. The DeMAF offers multiple plugins for different deployment technologies, such as Kubernetes, Helm, Ansible, and Terraform, to create the EDMM model. Additionally, the DeMAF provides a visualization of the created model – provided by Winery [KBBL13; KWH+24] – to enhance comprehensibility.

Since the DeMAF is unique in the aspect that it can handle multiple different deployment technologies and visualize the generated EDMM models through Winery, we want to explore the following question: *How easily can deployment models be comprehended using the DeMAF visualization compared to other visualization tools?* This holds important value because we want to justify the existence of the framework and further development, including adding more plugins and extending the accessibility of the tool. Currently, the DeMAF has no own graphical user interface (UI) and is only usable through its shell, the *demaf-shell*, or using a REST API.

Our research project analyzes the comprehensibility and usability of the visualized EDMM models generated by the DeMAF. To facilitate our research, we first created a deployment configuration targeting a Kubernetes cluster provided by the university. Previously, only a Docker Compose configuration was available, which was limited to local deployments. By developing the new configuration, we enabled the deployment of the DeMAF in a more realistic, cloud-based environment. Second, a user study to compare the DeMAF visualization with other available visualization tools in the industry. To achieve this, we explored various deployment options and UIs, researched and compared different visualization tools, and developed an appropriate study design.

The findings of our study demonstrate that the DeMAF visualization leads to a significant improvement in user comprehension of deployment models, particularly for simpler configurations. Participants utilizing the DeMAF exhibited higher levels of accuracy in task responses and reported lower cognitive load scores compared to those employing Monokle. It is noteworthy that DeMAF users demonstrated a higher degree of success in identifying service interactions and configuration details, particularly in scenarios where prior knowledge was limited. While the mean task completion times were similar across both tools, the DeMAF was consistently rated higher in terms of perceived usability and visual clarity. The DeMAF was the preferred option, as it was found to be more navigable and informative. These findings demonstrate that the semantically enriched, technology-agnostic visualizations produced by DeMAF can facilitate more rapid and precise interpretation of deployment models, thereby validating its potential for educational and practical applications. In sum, the results provide a robust response to our central question: deployment models can be comprehended more readily using the DeMAF visualization than with existing tools.

This report further details our methodology, i. e., what our research questions are, how our study was designed, who participated in our study, which deployment models we used, what our hypothesis are, how the study was performed, what our evaluation criteria and a small excerpt from our survey. Afterward, we present the complete results of our study, discuss them and try to interpret our findings. A *threats to validity* analysis, featuring statistical tests, is also performed and evaluated. Finally, we finish off with a conclusion and how future research on this topic or future development could look like.

Report Structure

Chapter 2 – *Foundations and Related Work*: Here, we provide the necessary information about the DeMAF and give an overview of the existing visualization technologies.

Chapter 3 – *Methodology*: With this chapter, we present the methodology of our research project, including the research questions, participants, hypothesis, procedure, evaluation criteria, and survey questionnaire.

Chapter 4 – *Results*: In this chapter we present the results of our user study, including the most important objective and subjective metrics, and the statistical analysis of the results.

Chapter 5 – *Discussion*: Here, we discuss the different shown results, and additionally explain the threats to validity and the implications towards our hypotheses.

Chapter 6 – *Conclusion*: Finally, we summarize the results of our research project and present future work.

2. Foundations and Related Work

First, in Section 2.1, we discuss the main subjects related to our research project. Afterward, in Section 2.2, we present related work and highlight the importance of the chosen topic and why it was not sufficiently solved in other research.

2.1. Foundations

In the following section, we discuss the foundations of our research project. First we explain what DeMAF is and how it works. Afterward, we take a look at the importance of visualization tools in the comprehension of deployment models and define what a user study in this aspect is.

2.1.1. Deployment Model Abstraction Framework (DeMAF)

The Deployment Model Abstraction Framework (DeMAF) is a transformation framework from Weller et al. [WBSB23] that enables the user to transform a wide variety of TSDMs into TADMs using its plugins. The technology-agnostic metamodel EDMM is used for this purpose [WBF+20]. The DeMAF already has plugins for the deployment technologies Helm, K8s, Ansible and Terraform. The analysis manager is the core of the DeMAF to which all plugins register and which provides the user with an interface. At this interface, the user can request a transformation of a deployment model using the DeMAF shell. The analysis manager first sends this request to the model service, which initializes an empty EDMM model. The analysis manager then creates an analysis task and sends it to the corresponding plugin via a message broker. The respective plugin then requests the saved models from the model service and updates them at the end of the analysis task. If a to-be-transformed deployment model contains an embedded model, the plugin sends an analysis request for this model to the analysis manager, which forwards the analysis tasks to the corresponding plugin. If nothing else was specified in the transformation request, the visualization service requests the TADM from the model service after successful transformation and outputs it in a format understandable for Eclipse Winery. Eclipse Winery by Kopp et al. [KBBL13; KWH+24] is an HTML5-based environment for the graph-based modeling of application topologies and the definition of reusable component and relationship types. Winery uses TOSCA, an OASIS standard, as internal storage, import and export format. To visualize the updated TADM, the visualization service layouts the model using Graphviz and outputs it to TOSCA files which are then imported into Winery for display.

2.1.2. Visualization

Technology-Specific Deployment Models (TSDMs) are typically in YAML or another text format, which makes it difficult for beginners to understand. Cerny et al. [CAYT24] describe how visual models can represent the structure and behavior of microservice architectures by extracting and organizing information from static code analysis; this approach can similarly be used to visualize Technology-Specific Deployment Models (TSDMs), making them more accessible and understandable. A visualization of these TSDMs is a graphical representation or diagram that illustrates the structure, components, relationships, and key aspects. Visualizations allow experts and beginners to develop a common understanding. Further, they are used to communicate complex concepts and designs to various stakeholders, including developers, architects, project managers, and other non-technical stakeholders. Visualizations provide a clear and concise way to communicate the high-level and low-level aspects of TSDMs.

2.1.3. User Studies

In research, user studies are a widely used method to identify the needs and requirements for a particularly positive user experience with products, services, or systems. By examining target users in real-life contexts or controlled environments, researchers can discover valuable information about user preferences, problems, and expectations, which can be used to inform the design, development, and improvement of products and services [LFH17]. This helps developers to better understand the user's perspective, validate design decisions, and iteratively refine prototypes or implementations to improve usability, user satisfaction, and comprehensibility.

2.2. Related Work

In the following section, we present *visualization tools* from widely used deployment technologies and *user study techniques* that can be used to examine the effectiveness and comprehensibility of our visualization tool. Additionally, related work on how to design and execute a user study to evaluate the comprehensibility of visualization tools is presented.

2.2.1. Visualization Tools

There are multiple visualization tools available for different deployment technologies. We give an example of various tools, focusing on K8s, Ansible and Terraform. In our research on this topic we have not been able to tell whether there is a state-of-the-art visualization tool – a standard tool for a specific TSDM that is commonly used – for any of these deployment technologies, and instead we will focus on free and open-source software (FOSS). Further, we were not able to find a tool that is able to visualize TSDMs from multiple different deployment technologies. In the following we will present some of these tools as there are dozens if not more tools available.

Ansible

First, there is *Ansible Playbook Grapher* [hai24] which uses a Python package to graph an Ansible playbook. The installation is straightforward using the Python package manager `pip`. Two renderers are supported: *Graphviz* [Aut24] and *Mermaid* [SVC+24].
Both renderers generate node-link diagrams, where tasks, roles, and dependencies are represented as nodes and edges in a directed graph. This approach provides a global overview of the playbook structure and execution flow.

Secondly, there is *Ansigenome* [nic17] which was last updated seven years ago. It also provides a Python package that may be installed through `pip` and is able to generate a visualization using *Graphviz* [Aut24].
Ansigenome focuses on visualizing the relationships between roles and tasks as a dependency graph, again using a node-link diagram. Although this project seems abandoned, it may still provide a potential visualization for the proposed research project.

Kubernetes (K8s)

To begin, there is *SPEKT8*, a visualization tool which uses a graphical user interface (GUI) to visualize a K8s cluster either as a graph or as a table [eejs20]. The tool allows the user to inspect each pod individually and see it's configuration in an intuitive way. *SPEKT8* also provides information on ingresses, nodeports, load balancers, cluster-ip and persistent volume (claims). Further, *SPEKT8* is able to show service details.
The main visualization technique is an interactive node-link diagram, where cluster resources (pods, services, etc.) are represented as nodes and their relationships as edges. The tool also offers tabular views for detailed inspection.

Next, *KubeView* [ben22] allows for a visualization inside a web browser. It offers the same capabilities as *SPEKT8*: viewing the configuration of the pods and details on the services. *KubeView* further includes K8s objects like Deployments, ReplicaSets/StatefulSets/DeamonSets or Ingresses, as well as LoadBalander IPs, PersistenVolumeClaims, Secrets and ConfigMaps.
KubeView primarily uses a force-directed graph layout to visualize the cluster, where each resource is a node and their relationships (such as ownership or connectivity) are edges. This provides a holistic, interactive overview of the cluster state.

Both tools are not up-to-date, i.e., their development is not being continued. *SPEKT8* was last updated four years ago while *KubeView* is a little more recently updated, two years ago.

Monokle Monokle [Kub25] is a visualization tool for K8s deployment models. It offers a desktop integrated development environment (IDE) in which a user can open a K8s resource directory to analyze how the deployment is set up. Specifically, this means how the K8s resources like *Deployments*, *StatefulSets*, *Services*, *Ingresses*, *PersistentVolumes* or *ReplicaSets* connect to each other – i. e., which *Deployment* uses which *Service* or *PersistentVolume*. Monokle uses the user's `kube/config` configuration file to connect to registered clusters which in return lets Monokle deploy the visualized deployment model. Furthermore, Helm charts are supported and can be both visualized and deployed on a cluster. Since Monokle is open-source, its source code can be found in Kubeshop's – the developers behind the application – GitHub repository [Kub24a]. A documentation is also available online, hosted by Kubeshop [Kub24b].

Monokle's visualization technique is hybrid: it combines a source code editor with contextual, object-specific graph visualizations. Users can view and edit YAML manifests directly, while also accessing interactive dependency graphs for selected resources (e.g., showing how a Deployment is connected to Services and PersistentVolumes). This enables both detailed inspection and high-level structural understanding.

Terraform

First, Terraform provides a built-in method to generate a graph from the provided configuration. This can be generated using the following command: `terraform graph -type=plan`. The result may then be rendered as a PNG image using the `dot` command from Graphviz [Aut24].
The visualization is a static directed acyclic graph, where resources and modules are nodes and dependencies are edges. This gives a global overview of resource relationships and execution order.

Next, there is a tool called *InfraMap* [cyc24] which provides another approach on visualizing a Terraform deployment configuration. *InfraMap* provides a Docker container for easy PNG generation. This may be accomplished using
```
docker run --rm -v ${PWD}:/opt --entrypoint "/bin/ash" inframap -c './inframap generate /opt/PATH_TO_HCL_STATE | dot -Tpng > /opt/graph.png'
```
.
InfraMap also generates a static dependency graph, but with additional filtering and grouping features to highlight relevant resources and their interconnections, improving clarity for large infrastructures.

Summary of Findings and Choice of Tool for Study

There are several visualization tools available for different deployment technologies such as K8s, Ansible, and Terraform. However, all presented tools are technology-specific and lack cross-technology capabilities. Furthermore, many tools, including *Ansible Playbook Grapher*, *KubeView*, and *SPEKT8*, show limited maintenance or have been discontinued.

From a visualization technique perspective, most tools rely on node-link diagrams (graph-based visualizations) to represent resources and their dependencies. These can be static (as with Terraform's built-in graph and *InfraMap*) or interactive (as with *SPEKT8* and *KubeView*). Some tools, such as *Monokle*, combine these graph visualizations with source code or manifest editors, allowing users to switch between textual and graphical representations and to focus on specific objects or relationships within the deployment.

Monokle emerges as a suitable option in this context due to its maintained open-source nature and comprehensive support for K8s deployment models. It offers an IDE-like experience that allows users to analyze resource configurations, inspect interconnections between deployments, services, and persistent volumes, and supports Helm charts natively. Moreover, *Monokle* connects directly to clusters using the users' kubeconfig file, enabling seamless validation and deployment of configurations.

Given the absence of a state-of-the-art tool and the fragmented tool landscape, *Monokle* provides a robust and actively maintained alternative for visualizing and managing K8s models. It ensures both usability for practitioners and extensibility for research projects requiring inspection and manipulation of Kubernetes configurations.

2.2.2. Visualization User Study Techniques

This section presents some of the techniques used in studies to analyze the comprehensibility of models and their visualization. Razali and Garratt [RG06] measure the comprehensibility of a UML-B and a B model and used the approach of two cross-over trial groups. The B model is a formal method which includes tool support and is based on an abstract machine notation. UML-B describes the B model extended with UML elements. This means that each participant is exposed to both models and the order of exposure is randomized, to reduce the learning effect. To measure the comprehensibility, they used a scoring system based on questions about the models and the time taken to answer the questions. Their approach may be used as a basis for the user study in this research project.

3. Methodology

This chapter presents the methodology of our study. We begin by outlining the research objectives and the comparative analysis approach (Section 3.1 – *Goal*). Section 3.2 – *Study Design* details the participant selection, the deployment models and assessment tools used, the pilot study, and the tested hypotheses. We then define the evaluation criteria used to assess comprehensibility and performance as well as an examplatory excert from our employed survey questionnaire (Section 3.3 – *Evaluation*). The chapter concludes with the experimental procedure outlined in Section 3.4 – *Procedure*.

3.1. Goal

This research project is driven by the objective to how the DeMAF visualization influences the comprehensibility of deployment models, compared to alternative visualization tools such as *Ansible Playbook Grapher*, *KubeView*, and *InfraMap* (see Section 2.2). Given the increasing complexity of deployment configurations, effective visualization plays a critical role in aiding developers and operators in understanding system architectures.

To achieve this, we investigate two core aspects:

1. *Visualization Effectiveness:* We analyze how the DeMAF's visualization approach influences the comprehensibility of deployment models compared to other tools. We are particularly interested in comparing the DeMAF with the tool *Monokle*, which is a suitable visualization tool for K8s deployment configurations. The effectiveness of such a tool is pivotal as they offer fast and easy to understand (visual) information to developers, potentially leading to significant improvements in understandability and bug finding.
2. *Cognitive Load and Accuracy:* We assess whether participants using the DeMAF make fewer errors and require less time to extract key insights from deployment models. This is especially important to allow non-domain users to grasp the whole deployment configuration without requiring additional research on each deployment technology.

These aspects are examined through a controlled user study, where participants perform predefined tasks while their performance and subjective assessments are recorded. This leads to the following research questions:

- **RQ_1**: *How easily can an arbitrary deployment model be comprehended using the DeMAF visualization compared to other visualization tools?*
- **RQ_2**: *Are there differences in quality (i.e., affordance, etc.) and performance (i.e., task completion time, etc.) measurements when comparing the DeMAF visualization to other visualization tools?*

3.1.1. Comparative Analysis

In our study, we carried out a detailed comparison to really get to the bottom of the two important (research) questions:

1. *Visualization Effectiveness:* **RQ_1** is about whether the visualization tool we chose to compare to the DeMAF, Monokle, is not more effective in visualizing deployment models. The DeMAF enables the user to get a TADM, i. e., to get a more abstract model, and view connections like depends-on relations between services as well as service type recognition. We wanted to see if this really helps in comprehending a deployment model.
2. *Cognitive Load and Accuracy:* For **RQ_2** we look into how big the cognitive load and answer accuracy is when using both visualization tools. The main question here is: Does our tool, the DeMAF, really lower the cognitive load and increase the accuracy of users?

To do this comparison properly, we collected two main types of information from our participants: numbers (quantitative data) and free texts, thoughts or opinions (qualitative data). This mix of quantitative and qualitative data, helped us to not only see if there were any noticeable differences but also to understand why those differences might exist. By doing this, we hoped to learn more about how the DeMAF compares to other widely used visualization tools like Monokle and if our tool actually increases the accuracy in deployment model comprehension while decreasing the cognitive load.

3.2. Study Design

With our study, we employed a counterbalanced, within-subjects design. Participants were assigned to two groups, Group A and Group B. Both groups analyzed the same two deployment models in a fixed order. Group A used our tool, the DeMAF, for the first deployment model and *Monokle* for the second, while Group B began with Monokle and then used the DeMAF. This approach controlled for potential order effects and tool-specific learning biases. After each task, i. e., each deployment model, the participants were tasked to answer nine (for Task 1) or seven (for Task 2) questions in order for us to grasp their true comprehension of the deployment model. Finally, after each task questionnaire, we asked the participants about the task load and the tool's practical use.

We chose a *within-subjects* study design because our participant pool was small and had limited prior knowledge. By exposing all participants to every condition, we minimized the impact of individual differences, such as cognitive abilities or prior experience, on the results. This approach allowed us to reduce error variance and increase statistical power despite the low sample size, as within-subjects designs are known to be more sensitive in detecting effects under such constraints [CGK12]. Our participants were able to compare conditions based on their own experiences, which was particularly important given their limited background knowledge. A *between-subjects* design would have required a larger sample to balance individual characteristics and ensure comparable groups, which was not feasible in our context. Thus, the *within-subjects* design provided a more efficient and internally valid approach for our study conditions.

Thankfully, our supervisor provided us with access to the institute's computer science laboratory where multiple computers and required utensils are situated. This made enabled us to perform the study *in situ* and within the same environment for each participants, reducing any possible interference from an *ex situ* study.

In the following sections we want go into more detail on how our study is designed by first discussing our participant selection, followed by presenting the deployment models used and the tool used to assess the participant's task load. We then move on to summarizing our pilot study before concluding with our hypotheses.

3.2.1. Participants

For our study, we specifically *did not* select participants with a required expertise in the field of Computer Science. To recruit our participants we chose to send out mails through the university's distribution list to ask as many current students and graduatees. This mail sent two times to maximazie participation count. Further, we used other communication chanels, like Telegram groups, to advertise for our study because these groups are generally used more frequently by students.

The content of both the mail and the message were identical – inviting potential participants to take part in our study and get financially compensated. To organize the study, we added three links to register for a time slot where at least one of use was available. We also asked the participants to provide us with their names and contact information to get in touch with them if anything came up. The latter – getting in touch with the participants – was especially helpful if one did not show up at their time slot.

Participants were categorized into two groups:

- *Group A:* This group first used our tool, the DeMAF, for the T2-Microservices deployment model before switching to *Monokle* to work on the OpenTelemetry Astronomy Shop deployment model.

- *Group B:* The second group began their work on the T2-Microservices deployment model using *Monokle* before they were tasked to use the DeMAF for the OpenTelemetry Astronomy Shop deployment model.

This division was important to our study in order to see how the participant's comprehensibility would be affected when sarting with the DeMAF, compared to when starting with Monokle as visualization tool for the tasks.

3.2.2. Deployment Models and Assessment Tool Used

For our study, we decided to use *Monokle*, which was introduced in Section 2.2.1, as a comparative visualization tool to our tool, DeMAF, in order to evaluate both the *visualization's effectiveness* and the *cognitive load and accuracy* of comprehension tasks.

Additionally, we performed the study on *two* different deployment models of different complexity – both being K8s deployment models.

Furthermore, we used a widely acclaimed and used assessment tool – in our case the NASA Task Load Index (NASA-TLX) by Hart and Staveland [HS88]. The NASA-TLX is a state-of-the-art, broadly cited questionnaire that rates perceived workload.

T2-Microservices

The T2-Project [SK24b; SKd24] is a reference application provided in two variants: T2-Microservices (using the saga pattern) [Kop24a; SK24c] and T2-Modulith (a monolith with the same modular structure) [Kop24b; SK24a]. Initially, only the microservices variant existed; the modulith was later added for comparison. T2-Project now refers to both variants collectively. The project was originally designed to trigger Service Level Objective (SLO) violations related to response time and availability.

For our purposes only the *T2-Microservices* variant is of interest as it fits being deployed using a deployment technology like K8s. The source code can be found on GitHub [Kd24; Kop24a] while the implementation we used is from Weller [Wel22]. This implementation features a K8s deployment model as well as a Docker Compose and Terraform variant and can be deployed using multiple different Bash scripts.

OpenTelemetry Astronomy Shop

The OpenTelemetry Astronomy Shop [Aut25b] is a reference application that is a microservices-based distributed system designed to illustrate the implementation of OpenTelemetry [Aut25a] in a realistic environment. The goal of the reference application is to provide a realistic example of the deployment of a distributed system. The project offers the possibility of deployment with the help of the deployment technology Docker and K8s. The latter variant is more interesting for our purposes, as these deployment models can also be visualized by the DeMAF. The source code of the reference application can be found on GitHub [Par25].

For evaluation purposes, the OpenTelemetry Astronomy Shop was expanded to include a *MongoDB* integration and the deployment technologies Ansible and Terraform as part of a development project in which we participated. Further, all deployment models for all deployment technologies were translated to EDMM and added to the GitHub repository (see the `/edmm` directory). This extended source code can also be found on GitHub [MHS+24].

NASA Task Load Index (NASA-TLX)

The NASA-TLX by Hart and Staveland [HS88], is a subjective tool for assessing perceived workload. It measures six dimensions: mental demand, physical demand, temporal demand, performance, effort, and frustration. Each is rated on a scale from 0 to 100, and a weighted average is computed based on pairwise comparisons of their importance. The NASA-TLX is widely used in fields such as aviation, healthcare, and human-computer interaction to evaluate task difficulty and system usability. It is valued for its sensitivity to workload differences and its reliability when combined with objective performance metrics.

For our study we dropped the physical demand dimension as our research focuses on a non-physical, mental task. Thus, we only used the other dimensions. We chose to use Likert scales from 1 to 7 instead of the NASA-TLX's proposed 0 to 100 scale as our study mainly consists of Likert scales from 1 to 5. With this we hoped to make answering the questionnaire easier for our participants.

To evaluate the results, we followed Hart and Staveland's approach in averaging the answer. Though, we did not weight the dimensions and mapped the aggregated sum back to the original 1 to 7 interval. This made interpreting and comparing results to our other Likert scale question more straightforward as the diagrams and box plots then share almost the same dimension on the axis that correlates to the Likert scale.

3.2.3. Pilot Study

To ensure that the study we designed would not overwhelm potential participants, we conducted a pilot study. We planned for the study to take between 45 and 60 minutes, depending on how experienced each participant was with deployment models and the visualization tools. Our pilot did not have a higher education level, was studying computer science, had previously worked with deployment technologies but had not used any visualization tools yet and therefore had no prior experience with the tools we were comparing. Thus, they ideally represented one of the possible inexperienced participants in our study.

The pilot needed a total of 52 minutes to complete the study. Although this was within our expectations, we decided to reduce the amount of comprehension questions as the participant in the pilot study did not look at the documentation of the visualization tools as much as we hoped. Furthermore, we noticed that the constant switching between the online survey and the tool was unnecessarily time-consuming. This is why we decided to print an instruction sheet (see Appendix A.3 – *Instruction Sheet*) with both tasks as well.

3.2.4. Hypotheses

In addressing the research inquiries, we established corresponding null hypotheses. The outcomes of these analyses are detailed in *Chapter 4 – Results*. Our null and alternative hypotheses are as follows:

- $\boldsymbol{H_0^{(1)}}$**:** *There* ***is no*** *statistically significant difference, with respect to comprehensibility of deployment models when using our tool, the DeMAF, compared to other visualization tools.*
- $\boldsymbol{H_1^{(1)}}$**:** *There* **is** *statistically significant difference, with respect to comprehensibility of deployment models when using our tool, the DeMAF, compared to other visualization tools.*

- $\boldsymbol{H_0^{(2)}}$**:** *There* ***is no*** *statistically significant difference in quality (i. e., affordance, etc.) and performance (i. e., task completion time, etc.) measurements between using our DeMAF visualization and Monokle.*
- $\boldsymbol{H_1^{(2)}}$**:** *There* **is** *statistically significant difference in quality (i. e., affordance, etc.) and performance (i. e., task completion time, etc.) measurements between using our DeMAF visualization and Monokle.*

To determine whether we can reject these null hypothesis we use the following metric. For the first hypothesis $H^{(1)}$, we use: task completion time, comprehension question performance (task performance), and the aggregated and normalized NASA-TLX score.
For the second hypothesis $H^{(2)}$, we use the metrics: task completion time and the individual answers of the task load and the tool's practical use questionnaires.

3.3. Evaluation

We want to see how well participants could comprehend the deployment models using the two visualization tools. Further, we were interested in how easily the participants could use the tools in order to get the required information for the two tasks. To do this, we looked at two main things: how easy the deployment models was for them to understand and how well they could use the visualization tools.

To make sure we were fair in our evaluation, we asked everyone specific questions about the deployment model they worked on. This included simple questions at first, but also more complicated ones later on. These questions were *not* always answerable as the used visualization tool might not give the required information. We also kept track of how long it took the participants to answer these questions, but we did not tell them we were doing this. We thought, knowing they were being timed might make them feel pressured or nervous and effect their answers.

Next, we want to present some example questions that were part of the survey. Participants were asked to use the provided visualization tools and deployment models and provide an understanding of the deployed services and how they interact witch each other. The following questions exemplify the types of inquiries presented to participants during the evaluation process:

1. *Question on the general deployment model comprehensibility:*

   - How many services get deployed?
     - Free text field: ________________
   - How many databases get deployed?
     - Free text field: ________________

   For this, participants can use the overview of both visualization tools as seen in Figure A.2 and Figure A.3. There, in the DeMAF the services and databases could simply be counted and in Monokle there is already a count available in the service list.

2. *Question about the deployment model's environmental variables, i. e., settings from a specific service or database:*

   - Which version does the "UI" use? (Task 1, T2-Microservices)
     - ○ `main`
     - ○ None
     - ○ `5.2.4`
     - ○ `0.18.0.RELEASE`
     - ○ Don't know
   - What port gets mapped to the internal port `27017` of the MongoDB? (Task 2, Open-Telemetry Astronomy Shop)
     - ○ `27017`
     - ○ `27020`
     - ○ `8080`
     - ○ `999`
     - ○ Don't know
   - What are the username and password for the `<DataBase>`? (both Task 1 and 2)
     Please answer in this format: `username:password`
     If you don't know the answer please leave this blank!
     - Free text field: ________________

   For these questions, participants could again rely on the overviews of both tool (again, as seen in Figure A.2 and Figure A.3), find the required service or database and click on it to get more details. With a thorough search, the answers could then be found.

Correctness of answers and the time taken to respond were used to evaluate the impact of the visualization tools on deployment model comprehensibility.

3.4. Procedure

In our approach, we divided the participants into two main groups. Each group got the same two tasks but began work with a different visualization tool:

1) *Start with the DeMAF:* Group A began their work on the T2-Microservices deployment model with the DeMAF before switching to Monokle for the OpenTelemetry Astronomy Shop deployment model.
2) *Start with Monokle:* Group B was tasked to use Monokle for the T2-Microservices deployment model first before switching to the DeMAF to work on the OpenTelemetry Astronomy Shop deployment model.

Since the OpenTelemetry Astronomy Shop deployment model has more K8s resources and is larger than the T2-Microservices deployment model we chose to use the former for our **Task 1** and therefore OpenTelemetry Astronomy Shop for **Task 2**. We hoped to reduce any learning effect by first introducing the participants to deployment models using the less complex model with the first tool before moving on to the more complex one with the second tool.

In both cases, we first introduced each task and handed out an instruction sheet to each participant in which the tasks were explained (see Appendix A.3 – *Instruction Sheet*). The participants were given the advice to use the provided documentation for both tools and were told that they can ask for help if required.

As a small remark, since our study was conducted on Apple iMacs and most participants were not familiar in using MacOS, we offered guidance on how to use the Apple Magic Mouse as well as on how to use the operating system (i. e., shortcuts, window management, multi tasking, etc.). If participants were still unable to make any progress after about 5 minutes or asked us directly for help, we gave appropriate advice on which part of the documentation could help with their current problem.

After each task, we asked each participants 5 questions regarding the task load (see Section 3.2.2 – *NASA Task Load Index (NASA-TLX)*), which they could answer with a Likert scale ranging from 1 to 7. Where 1 stood for *very low* or *perfect* and 7 for *very high* or *failure*. It was therefore desirable to achieve the lowest possible score. See Table 3.1 for our full naming convention of these Likert scales.

	Meaning	
Likert scale	Task Demand	Success
1	very low	perfect
2	moderately low	almost perfect
3	slightly low	good
4	neutral	neutral
5	slightly high	bad
6	moderately high	almost failure
7	very high	failure

Table 3.1.: Naming convention for our 1-7 NASA-TLX Likert scales.

To measure the tool's practical utility, we gave each participant 10 statements to which they were asked to indicate their agreement using a Likert scale. In the rating, the participants could choose between a 1, which stood for strongly disagree, and a 5, which stood for strongly agree. It was therefore desirable to achieve the highest possible score. For our full naming convention of this Likert scale, see Table 3.2.

Likert scale	Meaning
1	strongly disagree
2	disagree
3	neutral
4	agree
5	strongly agree

Table 3.2.: Naming convention for our 1-5 agreement Likert scale.

4. Results

Section 4.1 – *Participants* provides an overview of the participants in the study and their backgrounds. In Section 4.1.1 – *Knowledge*, the results of the participant's knowledge and their self-evaluation are then presented. The following section, Section 4.2 – *Task Completion Time*, deals with the participants' task completion times, and task performance is processed in Section 4.3 – *Task Performance*. Then, Section 4.4 – *Task Load* deals with the task load rated by the participants. Section 4.5 – *Tool's Practical Utility* deals with the question of how helpful the tools and their visualizations of the deployment models were for understanding the deployment models themselves. Section 4.6 – *Preference and Qualitative Rating* addresses how the DeMAF compares to Monokle regarding preference and qualitative rating among the participants and which tool they would use to visualize deployment models in the future. The final section, Section 4.7 – *Statistical Significance of Results*, covers the assumptions, significance tests, and results regarding statistical significance.

Our study results as well as this document can be found at https://github.com/UST-DeMAF/FoPro-WebUI-Study-Results.

For illustration purposes, box plots are repeatedly used in the following sections. The boxes of these plots correspond to the interval between the first and third quartile $[Q_1, Q_3]$, the whiskers have a maximum length of 1.5 times the interquartile range (IQR $= Q_3 - Q_1$) and represent the minimum and maximum values of the data if no outliers are shown in the chart in the form of circles. The solid line within the box represents the median, while the dashed line stands for the mean.

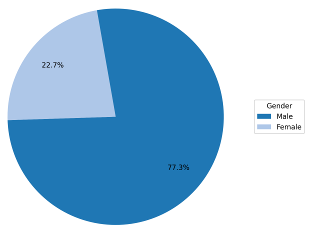
As previously mentioned in Section 3.4 – *Procedure*, when talking about Task 1, we are always referring to the T2-Microservices deployment model (Section 3.2.2) and the OpenTelemetry Astronomy Shop deployment model (Section 3.2.2) for Task 2.

4.1. Participants

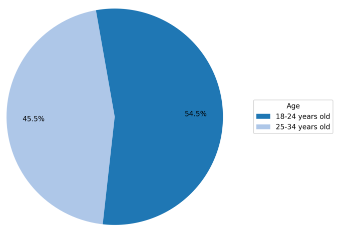
In total we had 22 participants with 77.3% being male, 22.7% female and 0% being diverse (see Figure 4.1a). We grouped them up so that both Group A and Group B had 11 participants each. All participants are between 18 and 34 years old (see Figure 4.1b) and either already have a Bachelor's, Master's degree or do not have any higher educational level yet (see Figure 4.1c). The participants that are still studying for their degree are from a broad range of semesters, going all the way up to their sixth or more semester. We mostly had participants from the field of

Computer Science but also some from related fields such as Mechanics, Electrical Engineering, Simulation Technology and Mathematics (see Figure 4.1d).

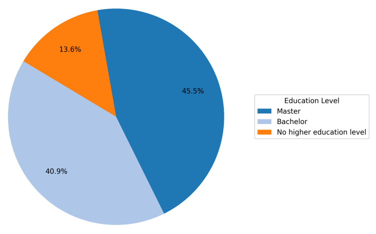
Roughly 60% of our participants have worked with deployment technologies before, with most of them having used them for university or work. The most used deployment technologies were *Docker compose* and K8s (see Figure 4.1e and Figure 4.1f) – although the familiarity with the deployment technologies was rather low. The terms TADM, TSDM, EDMM, OpenTOSCA and the DeMAF were mostly unknown, though TSDM (27.3%), EDMM (22.7%) and OpenTOSCA (22.7%) was known to some participants (see Figure A.1). 82.6% of participants have never used a visualization tool for deployment models before, with only a few having used *KubeView*, *Ansible Playbook Grapher*, *Ansible Run Analysis (ARA)*, *InfraMap*, *Terraform Graph (built-in)*, *OpenTOSCA* and *Winery*.



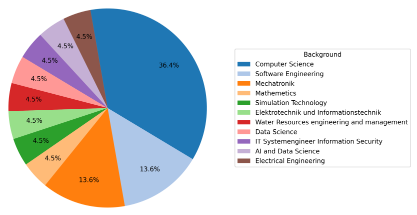
(a) This pie chart shows the gender distribution of our participants. None were "Diverse" or preferred to not answer.



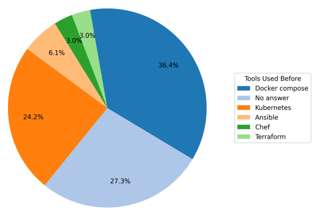
(b) This pie chart shows the age distribution of our participants. None were younger than 18 years old or older than 34 years.



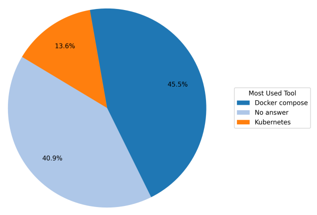
(c) This pie chart shows the education level distribution of our participants. None had a PhD.



(d) This pie chart shows the study field distribution of our participants.



(e) This pie chart shows what deployment technologies our participants used before. None used Helm before.



(f) This pie chart shows which deployment technologies our participants used the most, if any.

Figure 4.1.: In these pie charts, the demography of our participants can be seen. We depict the gender, age and education level distribution and the deployment technologies used before as well as the participants' most used deployment technology, if any.

4.1.1. Knowledge

As previously mentioned in Section 3.2.1, 59.1% of our 22 participants have used deployment technologies before. Nevertheless, 36.4% of the participants stated that they are *not* familiar with deployment technologies. 27.3% are *slightly* familiar. 18.2% of the participants stated that they are *moderately* familiar. Again, 18.2% stated that they are *quite* familiar with deployment technologies. None of the participants indicated that they are *very* familiar. Our naming convention for our 1-5 familiarity Likert scale can be seen in Table 4.1.

Knowledge level		
Likert scale	Meaning	Participant count
1	*not* familiar	8
2	*slightly* familiar	6
3	*moderately* familiar	4
4	*quite* familiar	4
5	*very* familiar	0

Table 4.1.: Naming convention for our 1-5 familiarity Likert scale.

Of our 22 participants, 13 indicated that they were most familiar with a particular deployment technology. 76.9% said that they were most familiar with Docker Compose, while the remaining 23.1% stated Kubernetes.

Just 18.2% of our participants have used a visualization tool for deployment models.

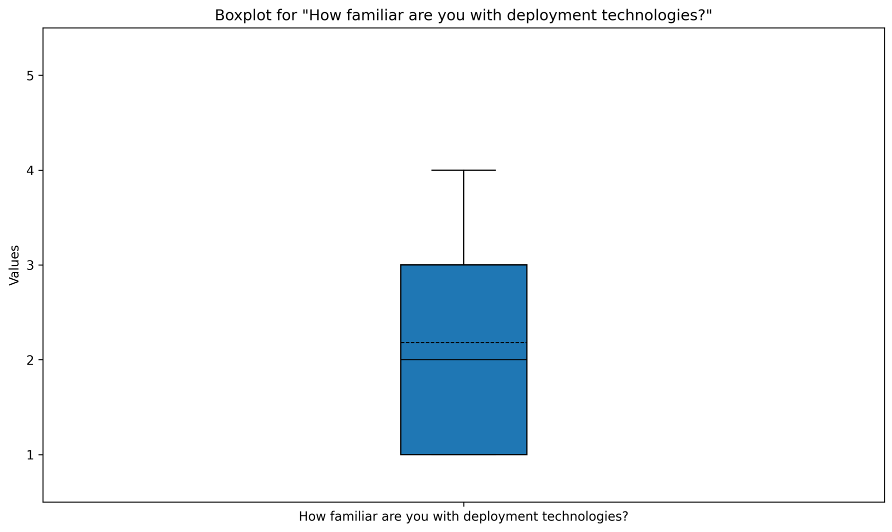


Figure 4.2.: The box plot shows the distribution of how the participants assessed their own level of knowledge regarding deployment technologies. The value 1 stands for not familiar and 5 for very familiar.

4.2. Task Completion Time

As described in Chapter 3 – *Methodology*, the participants had to visualize two deployment models in our study and answer questions about them. For the task completion time, independent of the task, the average time for the DeMAF was 23.12 minutes, while for Monokle it was 21.91 minutes. When considering the average completion times independently of the tool, we see that the participants required approximately the same amount of time for both deployment models: around 22.37 minutes for the T2-Microservices and 22.66 minutes for the OpenTelemetry Astronomy Shop. These differences can be seen in both Table 4.2 and Figure 4.3.

There are apparent differences in the tool-specific times for each task. With our tool, the DeMAF, participants on average took 3.12 minutes *longer* to answer the survey questions about the OpenTelemetry Astronomy Shop deployment model than the T2-Microservices deployment model. In contrast, when using Monokle, they were on average 2.53 minutes *faster* with answering the questions about the OpenTelemetry Astronomy Shop than with the T2-Microservices – the reverse.

Our participants required less time to process the T2-Microservices deployment model with the DeMAF than with Monokle, whereas the reverse was true for the larger deployment model the OpenTelemetry Astronomy Shop.

During the evaluation, we observed greater variance in times when the DeMAF was used, compared to Monokle. This variability is illustrated in Figure 4.3's taller box and whiskers of the box plots.

Looking at Task 1 in more detail, participants' prior knowledge appears to influence their performance. While the average task completion time tends to decrease with increasing familiarity, this trend is not strictly linear and this observation is also evident in Figure 4.4a. For instance, participants who reported being only *slightly* or *moderately* familiar on average completed the task faster than those who indicated being *quite* familiar. Notably, participants with *slight* familiarity completed Task 1 fastest when using the DeMAF, while participants with the same knowledge level took significantly longer with Monokle — resulting in a notable time average difference of over 6 minutes. Further details on these group-wise differences can be found in Table 4.2. Again, see Table 4.1 for more details on our naming convention.

For Task 2, the overall differences across knowledge levels were less pronounced, see Figure 4.4b. Participants who were *slightly* familiar took the longest, while those who were *quite* familiar completed it fastest. Also here, tool-specific trends were observed: with the DeMAF, participants with the knowledge level *slightly* familiar needed the most time, while this group of participants was among the fastest with Monokle. For the full breakdown by tool and level of knowledge, see Table 4.2.

Tool	Task(s)	Aggregated, Normalized
Both	1	22.37 min
	2	22.66 min
DeMAF	Both	23.12 min
	1	21.56 min
	2	24.68 min
Monokle	Both	21.91 min
	1	23.18 min
	2	20.65 min

(a)

		Knowledge level				
Tool	Task	1	2	3	4	5
Both	1	25.06 min ($n = 8$)	18.59 min ($n = 6$)	20.59 min ($n = 4$)	24.44 min ($n = 4$)	– ($n = 0$)
	2	22.43 min ($n = 8$)	24.57 min ($n = 6$)	21.54 min ($n = 4$)	21.39 min ($n = 4$)	– ($n = 0$)
DeMAF	1	25.80 min ($n = 4$)	14.37 min ($n = 2$)	21.17 min ($n = 3$)	20.86 min ($n = 2$)	– ($n = 0$)
	2	23.61 min ($n = 4$)	26.76 min ($n = 4$)	21.12 min ($n = 1$)	24.41 min ($n = 2$)	– ($n = 0$)
Monokle	1	24.32 min ($n = 4$)	20.70 min ($n = 4$)	18.85 min ($n = 1$)	28.02 min ($n = 2$)	– ($n = 0$)
	2	21.24 min ($n = 4$)	20.19 min ($n = 2$)	21.68 min ($n = 3$)	18.38 min ($n = 2$)	– ($n = 0$)

(b)

Table 4.2.: Aggregated, normalized average (mean) task completion time in minutes and split between the five knowledge levels. The n indicates the number of participants for the respective tool and knowledge level. There were no participants for knowledge level 5.

4.3. Task Performance

As shown in Table 4.3 and in Figure **??**, the aggregated, normalized average performance over both tasks is 58.02% for the DeMAF and 49.15% for Monokle. Regardless of the tool used, participants answered an average of 51.89% of the questions correctly for Task 1 and 46.27% for Task 2. The task questions with the highest performances for Task 1, independent of the tool, are *"What port does the Orchestrator use?"* and *"What are the username and password for the spring datasource?"*, both 77.27% correctly solved. For Task 2 the highest performance with 81.82% is for the question *"What port does the Kafka queue use?"*.

Study participants performed better on average when completing the tasks using our tool, the DeMAF. On average, they answered 64.39% of the questions on the T2-Microservices and 51.64% of the questions on the OpenTelemetry Astronomy Shop correctly. Whereas with Monokle they only answered 39.39% of the questions on the T2-Microservices and 40.91% of the questions on the OpenTelemetry Astronomy Shop correctly. Using the DeMAF, the study participants' average performance was the highest for the questions *"How big is the Inventory"*, with a 100% average performance, for Task 1 and *"Which service(s) interact with the Checkout Service?"*, with a 90.91% average performance for Task 2. The highest average performance using Monkle is for the questions *"What port gets mapped to the internal port `8080` of the Payment Service?"*, with a 72.73% average performance, for Task 1 and *"What port does the Kafka queue use?"*, with a 90.91% average performance, for Task 2. When comparing the difference of average performances, for Task 1, the participants using the DeMAF have a 72.73 percent points higher performance for the question

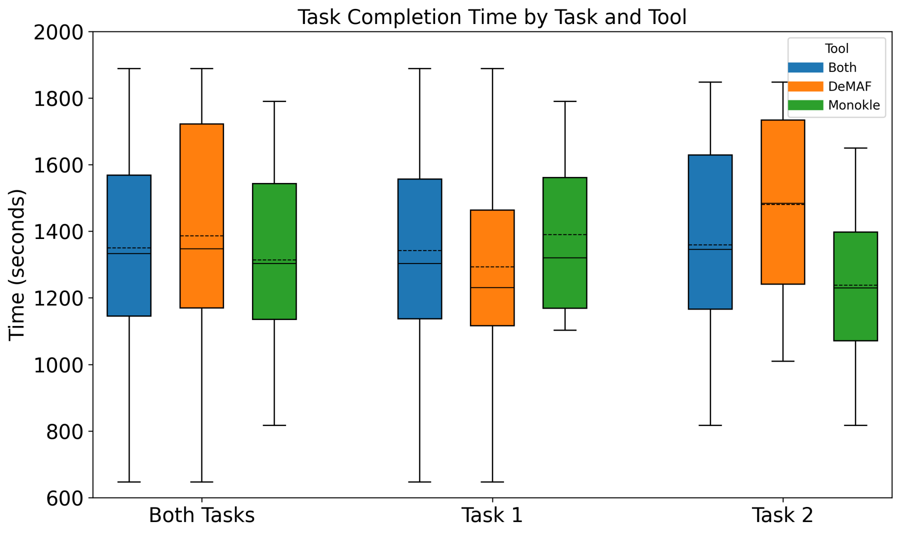
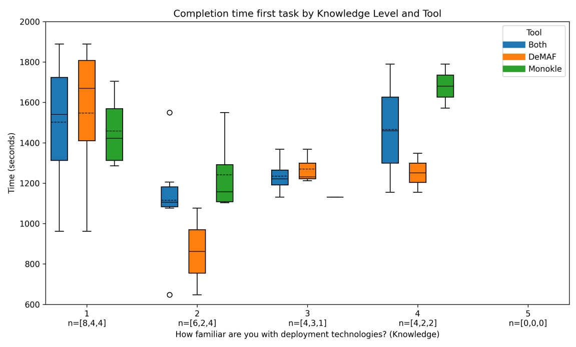
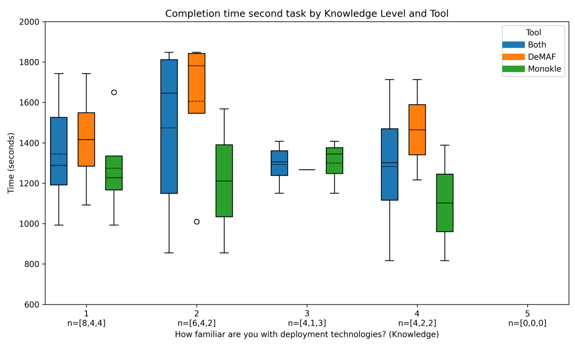


Figure 4.3.: These box plots show the aggregated and normalized task completion times for both tools, the DeMAF and Monokle. “Both Tasks” shows the aggregated time for both tasks, while “Task 1” and “Task 2” show the times grouped by task.



(a) This box plot shows the distribution of the time required for Task 1.



(b) This box plot shows the distribution of the time required by the participants for Task 2.

Figure 4.4.: In these box plots, the times are shown both tool-dependent and independently, subdivided according to the participants’ self-assessed level of knowledge. The value n on the x-axis represents the number of participants who assigned themselves to this value for the respective tool.

“Which service(s) interact with the Cart Service?”. For the participants using Monokle *“How many services get deployed?”* has a 45.45 percent points higher average performance compared to the participants using the DeMAF. For Task 2 the participants have a 63.64 percent points higher performance using the DeMAF for the task question *“Which service(s) interact with the Checkout Service?”*. While for *“How many services get deployed?”* the average performance is 63.64 percent points higher when using Monokle.

The Table 4.3b shows the aggregated and normalized task performances for the different knowledge levels. For both tools combined for Task 1, the highest average performances, with 60.42%, is for the participants with *moderately* familiarity of deployment models (knowledge level 3) and for Task 2, with 51.67%, the participants with *slightly* familiarity of deployment models (knowledge level 2) performed the best. When using the DeMAF for both tasks, the participants with knowledge level 1 have the highest average performances with 68.75% and 57.00%. For the Task 1 with Monokle, the highest average performance, with 48.83%, is achieved by the participants with the knowledge level 2 and for Task 2 the highest average performances, with 48.00%, are achieved by the participants with knowledge levels 2 and 4.

Tool	Task(s)	Aggregated, Normalized
Both	1	51.89%
	2	46.27%
DeMAF	Both	58.02%
	1	64.39%
	2	51.64%
Monokle	Both	49.15%
	1	39.39%
	2	40.91%

(a)

		Knowledge level				
Tool	Task	1	2	3	4	5
Both	1	53.12% ($n = 8$)	51.39% ($n = 6$)	60.42% ($n = 4$)	41.67% ($n = 4$)	– ($n = 0$)
	2	46.00% ($n = 8$)	51.67% ($n = 6$)	45.00% ($n = 4$)	40.00% ($n = 4$)	– ($n = 0$)
DeMAF	1	68.75% ($n = 4$)	62.50% ($n = 2$)	66.67% ($n = 3$)	54.17% ($n = 2$)	– ($n = 0$)
	2	57.00% ($n = 4$)	55.00% ($n = 4$)	50.00% ($n = 1$)	35.00% ($n = 2$)	– ($n = 0$)
Monokle	1	37.50% ($n = 4$)	45.83% ($n = 4$)	41.67% ($n = 1$)	29.17% ($n = 2$)	– ($n = 0$)
	2	35.00% ($n = 4$)	45.00% ($n = 2$)	43.33% ($n = 3$)	45.00% ($n = 2$)	– ($n = 0$)

(b)

Table 4.3.: Aggregated, normalized average (mean) performance split into the five knowledge levels. The n indicates the number of participants for that specific tool and knowledge level. Knowledge level 5 did not have any participants.

4.4. Task Load

In order to determine whether there is a difference in task load between the tools, the NASA-TLX described in Section 3.2.2 was used in a modified form in our study. As a small reminder, a higher NASA-TLX score indicates a higher task load, which is considered worse in this case.

For the DeMAF, the average normalized NASA-TLX score aggregated over both tasks is 3.65 and for Monokle it is 4.49. Looking at the normalized NASA-TLX score independently of the tool, the average score for Task 1 was 3.83. Monokle achieved an average score of 4.42 and a median of 4.4, which is significantly higher than the DeMAF's. Our tool, the DeMAF, achieved an average score of 3.24 and a median of 3.4. There is a clear difference in the question "*How successful were you in accomplishing what you were asked to do?*". Participants gave an average score of 4.82 for

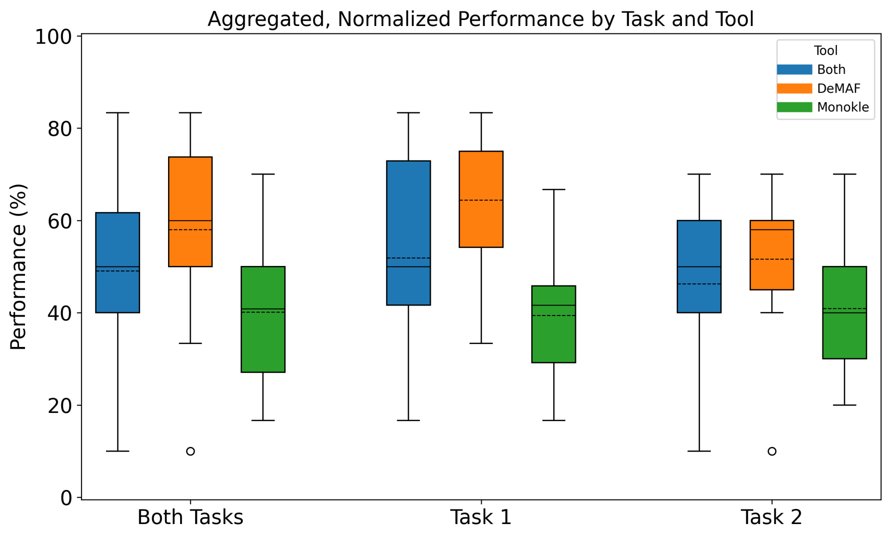
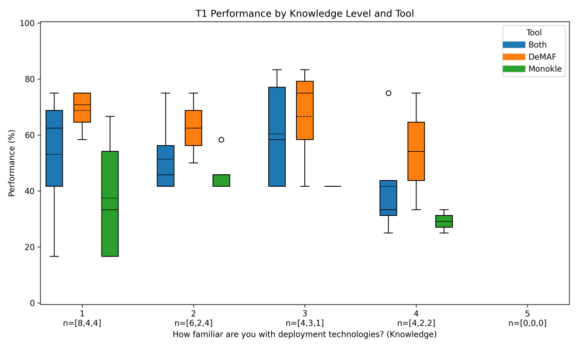
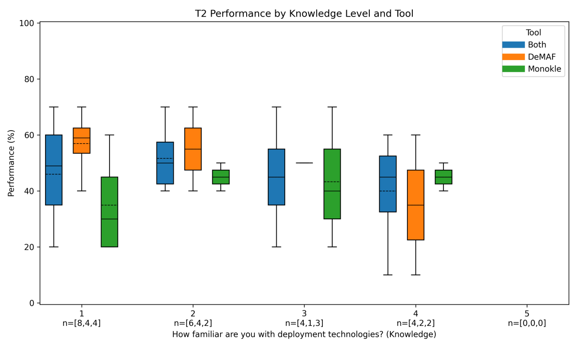


Figure 4.5.: These box plots show the aggregated and normalized task performance for both tools, the DeMAF and Monokle. “Both Tasks” shows the aggregated performance for both tasks, while “Task 1” and “Task 2” show the performances grouped by task.



(a) This box plot shows the performance distribution for Task 1.



(b) This box plot shows the performance distribution for Task 2.

Figure 4.6.: The two figures show the performance achieved, broken down according to the self-assessed knowledge levels. The n value on the x-axis indicates the number of participants who ordered this level of knowledge for the respective tool.

Monokle, while participants using the DeMAF only gave a score of 3.18. There is also a large difference in scores for the question “*How insecure, discouraged, irritated, stressed, and annoyed were you when working on the tasks using this tool?*”. Here, participants using the DeMAF gave an average and median score of 3.0. Participants who used Monokle gave an average score of 4.64 and a median of 5.0.

The tool independent, normalized NASA-TLX score of Task 2 on average is 0.49 points worse than that of Task 1 and is therefore at 4.32. Here, too, the DeMAF achieves a better average and median score than Monokle. The average score for the DeMAF is 4.07 compared to Monokle's score of 4.56. The median of Monokle is 4.8 while that of the DeMAF is 4.4.

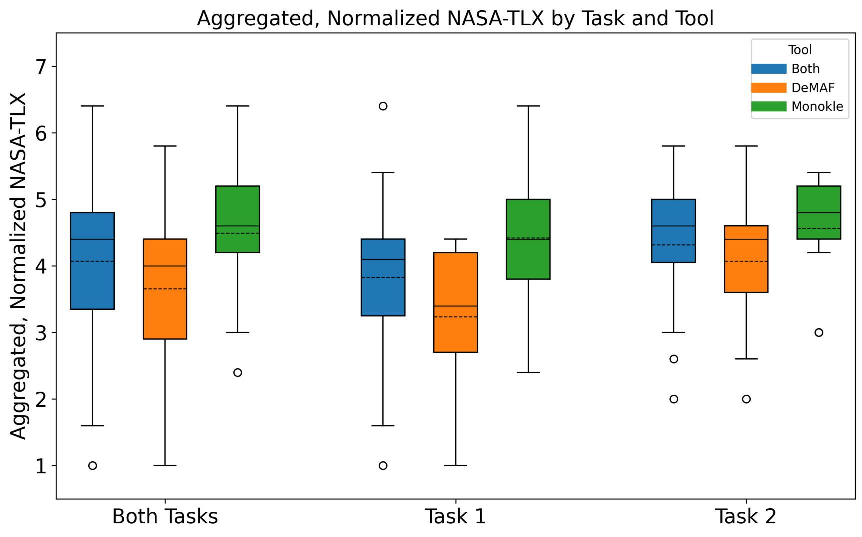


Figure 4.7.: These box plots show the aggregated and normalized NASA-TLX scores for both tools, the DeMAF and Monokle. "Both Tasks" shows the aggregated NASA-TLX score for both tasks, while "Task 1" and "Task 2" show the scores grouped by task.

Tool	Task(s)	Aggregated, Normalized
Both	1	3.83
	2	4.32
DeMAF	Both	3.65
	1	3.24
	2	4.07
Monokle	Both	4.49
	1	4.42
	2	4.56

Table 4.4.: Aggregated, normalized average (mean) task load. *Lower* values are *better* in this case, as this indicates a *lower* task load, i. e., a *better* user experience.

4.5. Tool's Practical Utility

As described in Section 3.4 – *Procedure*, the participants were asked to indicate their agreement with the statements regarding the visualization support of the respective tool when answering the questions.

Looking at the agreement obtained regarding the statements on the two tools, regardless of the tasks or deployment models, the DeMAF received consistently higher scores for all statements. On average, across all statements, the DeMAF achieved a score of 3.35 while Monokle only achieved an average score of 2.67. Table 4.5 provides an overview of the five statements with the greatest absolute difference in agreement between the two tools.

Statement	DeMAF	Monokle	Absolute Difference
The visualization helped me understand the deployment model clearly.	3.23	2.23	1.00
The layout and design made it easy to follow the model structure.	3.59	2.68	0.91
The tool was easy to navigate.	3.73	2.95	0.78
I could analyze the deployment model quickly using this tool.	3.18	2.45	0.73
The information presented was easy to interpret.	3.27	2.59	0.68

Table 4.5.: The five statements with the largest absolute difference in aggregated and normalized agreement.

The trend that our tool, the DeMAF, clearly performs better than Monokle in several key aspects is also evident in Task 1. This was especially clear in the statement "*The visualization helped me understand the deployment model clearly*", where the DeMAF performed much better with an average score of 3.45 . In terms of the comprehensibility of the information presented ("*The information presented was easy to interpret*"), the DeMAF also performed better with an average score of 3.64 compared to 2.45. Another example is the statement "*The tool was easy to navigate*", where the DeMAF scored 4.09 and Monokle 3.36. A similarly differentiated picture also emerged in the other statements, with the DeMAF generally receiving higher approval ratings. The distribution of approval scores can be seen, for example, for the statement "*The tool made the analysis process more efficient*" in Figure 4.8.

In Task 2, the scores for both tools dropped slightly, with Monokle performing worse overall. This was particularly noticeable for the statement "*The visualization helped me understand the deployment model clearly*", where the DeMAF achieved a value of 3.0 and Monokle only 1.82. There was also a clear difference in favor of the DeMAF about layout and design (3.36 vs. 2.27). For several other statements, such as "*The information presented was easy to interpret*", " *I could efficiently find the information I needed*" and "*I was able to extract the required details without difficulty*", both tools received only moderate approval (average value < 3). For the first two statements mentioned, the agreement distribution can be viewed in Figure 4.9a and Figure 4.9b. The exact values are also listed in Table A.1.

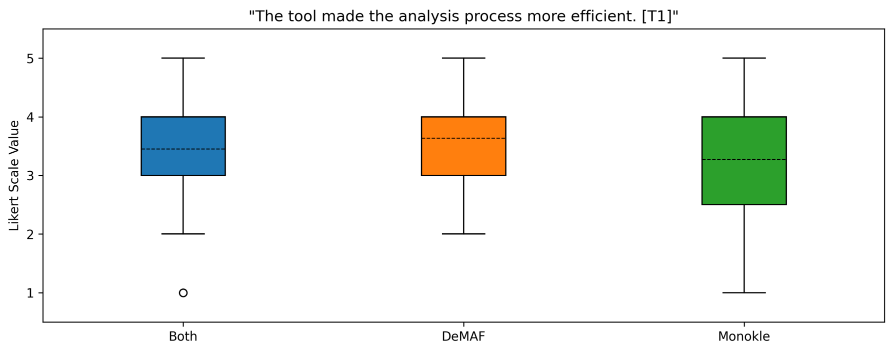
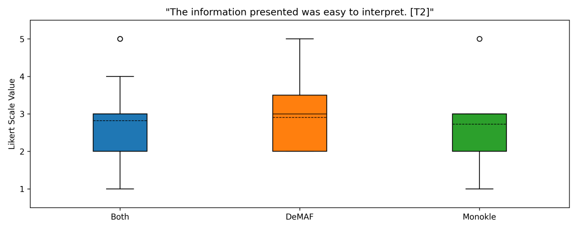
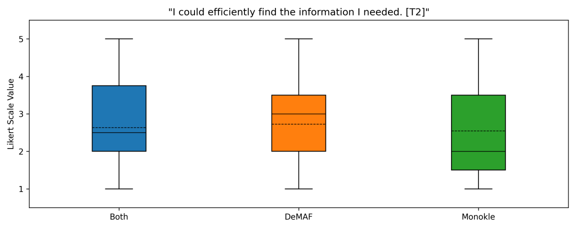


Figure 4.8.: The chart shows the distribution of agreement on how the tool made the analysis process more efficient. The value 1 stands for strongly disagree and the value strongly agree.



(a) The figure shows the distribution of agreement regarding how easy it was to interpret the presented information.



(b) The chart shows the distribution of agreement regarding how efficiently the required information could be found.

Figure 4.9.: The box plots show the distribution of agreement with the statements for the respective tool in Task 2. The value 1 stands for strongly disagree and the value 5 for strongly agree.

4.6. Preference and Qualitative Rating

In our study, the participants preferred the DeMAF in various statements regarding their preference and qualitative rating (see Table 4.6). It was considered by the majority ($> 50\%$) to be easier to learn, more confidence-enhancing in understanding, and the preferred choice for visualizing deployment models among participants. It is worth mentioning that the DeMAF was consistently preferred over Monokle, with only a small proportion of participants remaining undecided or expressing a preference for the comparison tool. These results are illustrated in Figure 4.10 and Figure 4.11, which show the distribution of tool preference and overall favorability.

The tendency towards the DeMAF can also be seen in the breakdown of the participants' responses according to their self-assessed level of knowledge (see Table 4.7). Especially for participants who were *not* or *moderate* familiar with deployment models, the DeMAF was preferred. *Slightly* familiar participants had no clear preference regarding the tools compared, which is why the rating was uniformly distributed between the options, while those with *quite* familiarity still tended to favor the DeMAF, albeit to a lesser extent.

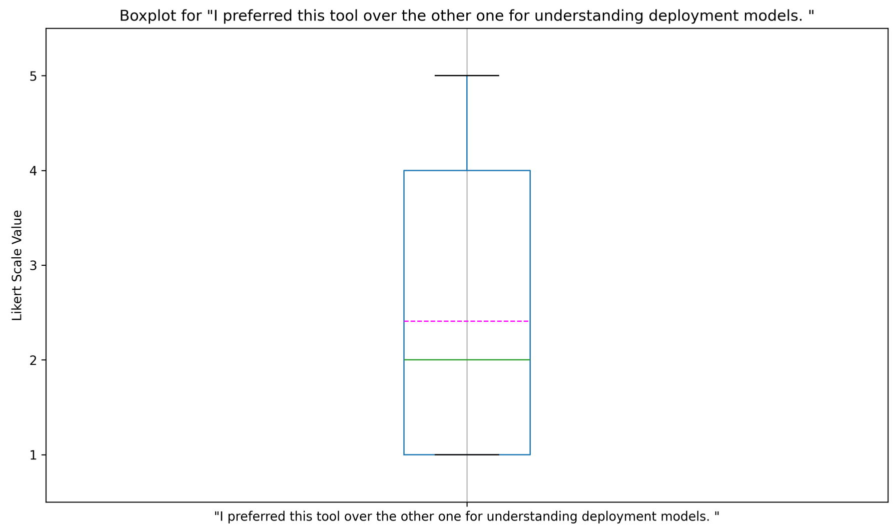


Figure 4.10.: The box plot shows the distribution of which tool was preferred for understanding the deployment models. Where the value 1 stands for strongly preferred the DeMAF and the value 5 for strongly preferred Monokle.

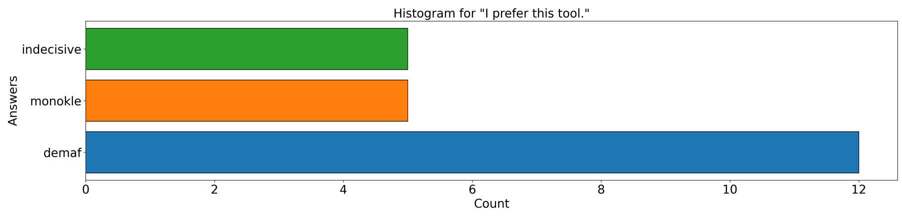


Figure 4.11.: This histogram shows how many participants preferred which tool.

4.7. Statistical Significance of Results

This section presents the prerequisites, tests, and results of our statistical tests for both tools and both tasks for the following metrics:

- Task performance (aggregated and normalized)
- Task completion time
- NASA-TLX score (aggregated and normalized)
- Tool's practical utility Likert scale questions

The task performance, task time, and NASA-TLX score are the main metrics of the study.

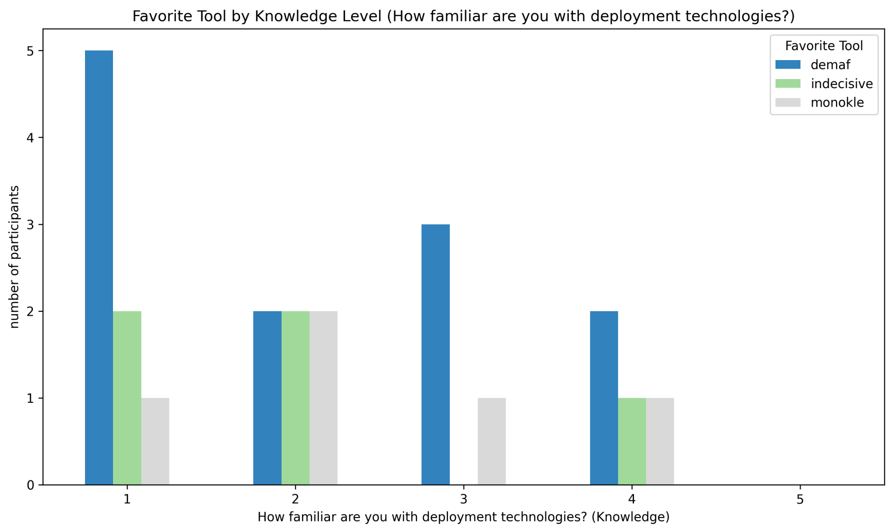


Figure 4.12.: The figure shows the distribution of the preferred tools according to the participants' self-assessed level of knowledge.

Statement	DeMAF	Indecisive	Monokle
I preferred this tool over the other one for understanding deployment models.	59.1%	4.5%	36.3%
This tool was easier to learn compared to the other tool.	63.6%	4.5%	31, 8%
I felt confident in my understanding when using this tool.	54.6%	9.1%	36.4%
I would use this tool again for deployment model visualization.	63.7%	9.1%	27.3%
This tool provided better insights than the alternative tool I used.	45.4%	36.4%	18.2%
I prefer this tool.	54.5%	22.7%	22.7%

Table 4.6.: Preference of the participants in percent regarding the respective statement.

4.7.1. Prerequisites and Tests

The data was tested for normal distribution using the Shapiro-Wilk test [SW65] and the variance was tested for homogeneity using Levene's test [Lev60]. These prerequisites tests resulted in the following:

Knowledge	DeMAF	Indecisive	Monokle
not familiar	62.5%	25.0%	12.5%
slightly familiar	33.3%	33.3%	33.3%
moderately familiar	75.0%	–	25.0%
quite familiar	50.0%	25.0%	25.0%
very familiar	–	–	–

Table 4.7.: Participants' preference regarding the tool in percent split according to their self-estimated level of knowledge.

- *Task performance* and *task completion time*: The data is normally distributed both across tasks and for the individual tasks, and the variance is homogeneous so we were able to use the Student's t-test [Stu08].
- *NASA-TLX score*: Only for the combined tasks and Task 1 is the data normally distributed, and the variance is homogeneous. For Task 2, we used the Mann-Whitney U test [MW47].
- *Tool's practical utility*: The data is neither normally distributed for the individual tasks nor across tasks, and the variance is not homogeneous, thus we use the Mann-Whitney U test here as well.

4.7.2. Results

For the significance tests, we used two-tailed statistical significance tests and a significance level of $p < 0.05$. The results for the overall task significance tests for our main metrics are available in Table 4.8. It shows that our results for the *task performance* and the *NASA-TLX score* are significant towards the DeMAF. The results of the significance tests for the remaining metrics across both tasks are shown in Table A.2. Table 4.9 for Task 1 and Table 4.10 for Task 2 show the results of the significance tests for the respective tasks. Also for Task 1, of the main metrics only the *task performance* and the *NASA-TLX score* are significant. Additionally, some of the *tool's practical utility* Likert scale questions are significant for Task 1 and Task 2.

Since our tests are two-tailed we then have to argue for which side they are significant, i. e., whether the DeMAF is better or worse than Monokle. Because for some of our metrics a higher score corresponds to a better result while for others a lower is, we have to specify this properly:

- Aggregated and normalized performance: *higher* is better
- Task completion time: *lower* is better
- Aggregated and normalized NASA-TLX: *lower* is better
- Tool's practical utility (Likert scales): *higher* is better

We chose to compare the mean values for this and the resulting choice can be seen in Table 4.8 for both tasks and Table 4.9 and Table 4.10 for each individual task. For the remaining metrics, the results for Task 1 are in Table A.3 and for Task 2 in Table A.4.

			Significant	
Metric	Test	p-value	$p < 0.05$	towards
Performance (aggregated & normalized)	t-test	0.00096	yes	DeMAF
Task-competion time	t-test	0.43558	no	–
NASA-TLX (aggregated & normalized)	t-test	0.01312	yes	DeMAF
How mentally demanding were the tasks?	Mann-Whitney U	0.01287	yes	DeMAF
How hard did you have to work to accomplish your level of performance?	Mann-Whitney U	0.03425	yes	DeMAF
How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool?	Mann-Whitney U	0.01448	yes	DeMAF
The visualization helped me understand the deployment model clearly.	Mann-Whitney U	0.01281	yes	DeMAF
The tool was easy to navigate.	Mann-Whitney U	0.02394	yes	DeMAF
The layout and design made it easy to follow the model structure.	Mann-Whitney U	0.01677	yes	DeMAF

Table 4.8.: The table shows the results of the significance tests for the main metrics (performance, time and NASA-TLX) of the study, the metrics that showed significance in *both tasks* and which significance tests were used.

			Significant	
Metric	Test	p-value	$p < 0.05$	towards
Performance (aggregated & normalized)	t-test	0.00160	yes	DeMAF
Task-competion time	t-test	0.46695	no	–
NASA-TLX (aggregated & normalized)	t-test	0.02254	yes	DeMAF
How successful were you in accomplishing what you were asked to do?	Mann-Whitney U	0.01467	yes	DeMAF
How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool?	Mann-Whitney U	0.01965	yes	DeMAF
The information presented was easy to interpret.	Mann-Whitney U	0.01416	yes	DeMAF
I could efficiently find the information I needed.	Mann-Whitney U	0.00949	yes	DeMAF
The tool provided all necessary information about the deployment model.	Mann-Whitney U	0.03540	yes	DeMAF
I was able to extract the required details without difficulty.	Mann-Whitney U	0.04386	yes	DeMAF

Table 4.9.: The table shows the results of the significance tests for the main metrics (performance, time and NASA-TLX) of the study, the metrics that showed significance in *Task 1* and which significance tests were used. The first two rows that feature the Mann-Whitney U test are part of the NASA-TLX while the rest are part of our own tool practicality questions.

			Significant	
Metric	Test	p-value	$p < 0.05$	towards
Performance (aggregated & normalized)	t-test	0.14922	no	–
Task completion time	t-test	0.06502	no	–
NASA-TLX (aggregated & normalized)	Mann-Whitney U	0.14649	no	–
The visualization helped me understand the deployment model clearly.	Mann-Whitney U	0.04574	yes	DeMAF
The layout and design made it easy to follow the model structure.	Mann-Whitney U	0.03868	yes	DeMAF

Table 4.10.: The table shows the results of the significance test for the most important metrics (performance, time and NASA-TLX) of the study, the metrics that showed significance in *Task 2* and which significance tests were used. The last two rows feature questions from our own tool practicality questions as these were the only significant ones.

5. Discussion

Section 5.1 – *Discussion of Results* deals with our results from the study introduced in Chapter 4 – *Results* and its subsections therefore follow the same structure. Afterward, in Section 5.2 – *Threats to Validity*, we discuss the limitations of our study and what threats to validity it poses. In the final section, Section 5.3 – *Implications*, the implications of the results with regard to our research hypotheses are discussed.

5.1. Discussion of Results

This section discusses the results of our study, starting with the participants and their knowledge. Next, the task completion time and task performance are discussed. Afterwards, the task load is covered before the tool's practical utility as well as preference and qualitative rating are discussed. Finally, we discuss the statistical significance of your results. As a note, we primarily looked at the results with dependency on the specific Tasks. We decided to do so as evaluating the results for each task separately allows us to better understand the impact of the different deployment models and tools on the participants' performance. Additionally, some significant differences only occur for one of the two tasks, which would be lost if we only looked at the overall results.

5.1.1. Participants

Our uniformity in rather low expertise and educational background ensured that each participant possesses a comparable understanding of deployment technologies and visualization tools. Our methodology (see Chapter 3 – *Methodology*) included a preliminary skill assessment to guarantee that all participants have the same level of skills. This assessment is structured as a small block of questions about tools they might have used before and how comfortable they feel in using them. By conducting this pre-test, we aimed to ensure the homogeneity of knowledge among participants before starting the research, thus minimizing any discrepancies in understanding that may arise from different skill levels. As a safety measure we gave a short explanation about deployment models and technologies. This setup allowed us to more precisely measure the impact of our visualization tools on how they assist in the comprehensibility of deployment models.

Knowledge

Taking a closer look at how familiar the participants were with deployment technologies in general, participants are *not* familiar (8 participants) or *slightly* familiar (6 participants) with deployment technologies. This under representation of participants with higher familiarity of deployment technologies can also be seen in the mean value of the Likert scale responses, which is 2.18. Ideally, we would like to have a more balanced distribution (mean around 3) of participants with high and low familiarity, as this would have allowed us to better evaluate the impact of the DeMAF on different user groups. However, this level of knowledge with deployment technologies is expected as we had no requirements for previous knowledge, except that the participants should be affine with computer science. Additionally, even for advanced or graduated computer science students, deployment models are not necessarily well-known, making it even harder to have participants with a high familiarity.

For the specific deployment technology familiarity, it was expected that the majority chose Docker Compose as deployment technology, because this tool is the most prominent and practical one for deployment, especially on a local computer. The second highest deployment technology that our participants were the most familiar with was K8s with 61.5%, which, again, was expected as this is well-known to most users who have used or heard about deployment technologies before. Looking at the different visualization tools the participants have used before, it is also no surprise that only less than 25% used any before, as they are even more niche than deployment technologies themselves.

5.1.2. Task Completion Time

Looking at Figure 4.3, we can see that, on average, the participants have a slightly higher (5.52%) task completion time using the DeMAF than using Monokle over both tasks. However, with a high standard deviation of 5.66 minutes for the DeMAF and 4.42 minutes for Monokle, this difference neglectable. Additionally, they needed approximately the same amount of time to answer the questions on both models, regardless of the tool. Even though the OpenTelemetry Astronomy Shop is a much more comprehensive deployment model.

If we look at the task completion times of the participants who used the DeMAF, we can see that the variance in the processing time required is significantly greater than for the participants who used Monokle.

The participants who used the DeMAF for visualization took longer to complete Task 2, both on average and in the median. This is in line with our expectations, as OpenTelemetry Astronomy Shop requires significantly more services to be visualized than T2-Microservices and the graph is therefore noticeably larger and more complex. And, in order to answer the general questions, more nodes had to be counted and viewed by the participants. This is further complicated by the increased K8s resources OpenTelemetry Astronomy Shop offers compared to T2-Microservices. This is an essential disadvantage of the graph-based visualization of deployment models.

Looking at the task completion times in Figure 4.3 by the participants who used Monokle to visualize the tasks, it can be seen that the users needed even less time on average for Task 2, where OpenTelemetry Astronomy Shop was visualized – as described in Section 4.2 – *Task Completion Time* – than for Task 1, where the smaller deployment model T2-Microservices was visualized. It could be argued that this time saving is due to a learning effect on the participants. However, this can be ruled out due to our study design, as none of the participants visualized both deployment models with the same tool. A potential part of the time saving can be related to the fact that in the sidebar of Monokle, for example, the number of deployed services was directly visible and thus one of the deployment questions could be answered by the participants without counting each individual service and deciding whether or not it was indeed a service or some other deployment specific resource (like a storage device).

Looking at the times concerning the participants' self-assessed knowledge, no clear trend can be seen, neither in Figure 4.4a nor Figure 4.4b. As mentioned in Section 4.2, participants in Task 1 who indicated a higher level of knowledge (4 or 5) were, on average, faster than participants who indicated that they were not familiar. However, Figure 4.4a also shows that the variance is greatest for participants who stated that they were *not* familiar and were sometimes faster than other participants in other knowledge levels. It is also notable that *slightly* familiar participants were on average the fastest, and the remaining higher knowledge levels (3, 4, and 5) took longer. For Task 2, Figure 4.4b shows that the variance in the time required has increased for all knowledge levels except the *not* familiar. This is in line with our expectations, as the OpenTelemetry Astronomy Shop deployment model is larger and therefore more complex. However, the expected effect of increasing prior knowledge of deployment technologies and models, significantly reducing the required working time, did not occur. This may be because our participants consisted only of students and graduates, and they simply overestimated their knowledge in these subject areas.

5.1.3. Task Performance

The task independent task performance is, as shown in Table 4.3, 8.87 percent points higher than Monokle. This is a positive result towards accepting our first hypothesis $H_1^{(1)}$ in favor of our tool, the DeMAF. For the task dependent task performance, as introduced in Section 3.4, we considered the OpenTelemetry Astronomy Shop deployment model more complex due to its greater extensiveness compared to the T2-Microservices. This is affirmed by the drop in average task performance of 5.62 percentage points for Task 2 compared to Task 1, over both tools. The generally low (less than 50%) task performance for both tasks is expected because of the lack of higher familiarity with deployment models. This is most likely due to the background of the participants, who are mostly students and graduates with no expertise. Additionally, this overall low task performance is also reflected by the individual question performances, where only the simple questions with easily found solutions in both tools have a high performance.

For the tool-dependent task performance, the participants using the DeMAF achieved a 25.0 percent points higher performance for Task 1 and a 10.7 percent points higher performance for Task 2 compared to the participants using Monokle. This is, especially for the first task, a positive result towards accepting $H_1^{(1)}$. An explanation for why the study participants performed better while using the DeMAF for Task 1 may primarily be due to the high difference of 72.73 percent points for the question *"Which service(s) interact with the Cart Service?"*. For this question, the participants using Monokle never provided the correct answer, which shows that for this specific case Monokle has a large disadvantage. In contrast, the participants using Monokle performed better for both task for the question *"How many services are deployed?"*. This can be explained by the fact that the number of deployed services is directly visible in the sidebar of Monokle, while in the DeMAF, users must count the nodes in the graph and identify which nodes represent services.

When evaluating the general average task performance, independent of tool, by the participants' self-assessed knowledge, we can see that contrary to our expectations, the participants with the highest recorded knowledge level (4, i. e., *quite* familiar) performed the worst in both tasks (see Table 4.3b). However, the number of participants for this knowledge level is low, with only four individuals. Additionally, the high standard deviation of 22.57 for Task 1 and 21.60 for Task 2 prevents us from concluding that users with higher knowledge levels generally perform worse. It is possible that these participants either overestimated their familiarity with deployment technologies or lacked specific knowledge relevant to our models and tools. For the two lower knowledge levels (1 and 2), the general task performance is similar for both tasks, which is in line with our expectations, as they are both inexperienced users and thus have similar task performances. The participants with knowledge level 3 performed the best in Task 1, which is in line with our expectation assuming participants with knowledge level 4 overestimated their knowledge. Interestingly, this group (knowledge level 3) of participants has the largest drop (15.42 percent points) in average performance for Task 2 compared to Task 1. This result is generally in line with our initial expectations, although we expected the largest drop to occur for participants with the lower knowledge levels (1 or 2). This deviation from our expectations is likely due to the low number of participants with knowledge level 4, which was only four. Additionally, their Task 1 performance was high compared to the other knowledge levels, which increases the relative drop in performance. For the participants with the knowledge levels 1 and 2, the average task performance is similar for Task 1, which is expected, as they are both inexperienced users and thus have similar task performances. However, this is not the case for Task 2, where the participants with knowledge level 1 have a drop of 7.12 percent points, while the participants with knowledge level 2 have an increase of 0.28 percent points. The larger drop for the participants with knowledge level 1 is expected, as earlier mentioned, but the increase for the participants with knowledge level 2 is unexpected. Additionally, the participants with knowledge level 2 have the overall best average performance for Task 2, which is also unexpected. We have no clear explanation for this, but we assume it is due to the overall small sample size, which limits the statistical power of our analysis.

For the tool-dependent task performance, split into the participants' self-assessed knowledge, we can see that the participants using the DeMAF performed better than the participants using Monokle for all knowledge levels in both tasks, except for the participants with knowledge level 4 in Task 2. This difference is greatest for Task 1, where the participants with knowledge level 1 achieved a 31.25 percent points higher performance using the DeMAF than using Monokle.

Additionally, also for Task 2, the participants with knowledge level 1 achieved the highest difference of 22.00 percent points. These results are in line with our hypothesis $H_1^{(1)}$, stating that the DeMAF is better suited for understanding deployment models than Monokle. Nonetheless, the number of participants for each of the knowledge levels is low and splitting them into the different tools used reduces the number of participants even further. This makes it difficult to draw meaningful conclusions about how well the DeMAF performs for each knowledge level compared to Monokle. However, the results show a clear trend that the DeMAF is better suited for inexperienced participants, of which the data has the highest statistical power due to the larger number of participants.

5.1.4. Task Load

The task load and thus the corresponding aggregated, normalized NASA-TLX score for the DeMAF is lower than for Monokle by 23% over both tasks. This is a positive result towards accepting our second hypothesis $H_1^{(2)}$ in favor of our tool, the DeMAF. For the task dependent task load, it increases by 0.49 points for Task 2 compared to Task 1, which is within the range of what we expected. We chose the T2-Microservices as the first deployment model due to its lower complexity compared to OpenTelemetry Astronomy Shop in order not to overwhelm our participants completely during the introduction. The fact that Monokle, on average, received 1.18 more points for Task 1 than the DeMAF shows that our tool ensures that comprehension tasks are not quite as mentally demanding for users, at least for deployment models that are not too complex. The fact that the DeMAF received a similarly high aggregated, normalized NASA-TLX score for Task 2 (4.07) as Monokle (4.56), indicates that the graph representation reaches its limits with this size and complexity of the deployment model, such as that of OpenTelemetry Astronomy Shop. However, the difference of 0.49 points between the tools shows that our tool still provides better support for this task than Monokle.

5.1.5. Tool's Practical Utility

For the tool's practical utility, the DeMAF received a higher approval rating than Monokle for all ten statements for both Task 1 and Task 2. This result supports the observations regarding the task load from Section 5.1.4 and shows that our tool relieved the user in processing the questions and supported them with its functions.

Both tools receiving less agreement with these statements is most likely due to the high number of K8s resources of the OpenTelemetry Astronomy Shop deployment model. Because of the increased size of the deployment model compared to T2-Microservices, the sidebar list Monokle offers and the graph of the DeMAF were significantly larger. This made the visual presentation of the tools less user-friendly.

Monokle scored significantly worse in the statements regarding the visualization due to the fact that inexperienced users (knowledge level 1-3) were confused by the apparent self-referencing of services in the Monokle representation. This might also be caused by these participants not reading the whole task, as it was specifically stated that services do not interact with themselves. Monokle also displays the connections between K8s resources – in our case, most often image, service, and deployment – in its graph representation, giving the appearance of multiple self-references. This further confused inexperienced users. This could have led to a different rating for these statements.

The DeMAF exhibited a distinct behavior in this regard, as it grouped all K8s resources into a single, self-contained microservice node. To elaborate, a database service that has a deployment, a service, a persistent volume, and a persistent volume claim would manifest as a single node in the graph. This phenomenon distinguishes it from the problem that occurred in Monokle. The information regarding the Docker images is represented as artifacts, which must first be actively displayed by the user. This is also mentioned in the documentation, which 86.36% used. Although not showing these artifacts in the first place improves the clarity of the graph, it makes it more difficult to find this type of information – especially with the different wording (*artifacts* vs. *image*). Since the EDMM's *artifacts*, inherently are different from Docker images – as they refer to *any* artifact – it would be inconvenient to name all kinds of artifacts inside the DeMAF's documentation. Despite this additional burden, the DeMAF received a better average rating than Monokle from the participants in terms of how efficiently the information could be found.

5.1.6. Preference and Qualitative Rating

The results regarding preference and qualitative rating show a clear tendency towards the DeMAF. In five out of six questions, the majority ($\geq 50\%$) of the participants stated that they preferred our tool. Only in terms of which tool provided the better insights did the DeMAF not achieve major agreement (45.4%), as 36.45% of participants thought that neither tool was better. For participants with *not* familiar and *moderately* familiar knowledge levels, the DeMAF was the most popular. Participants who rated their knowledge level as *slightly* familiar or *quite* familiar had no clear favorite. Users with little to moderate prior knowledge of deployment models and technologies seem to benefit most from the graph-based visualization. Experienced users only benefited from this visualization to a limited extent, as they were able to answer the comprehension questions more quickly using the source view of Monokle, as they did not have to search through every node as with the graph visualization. The same could be said with the raw EDMM model view of the DeMAF if the user is familiar with the meta-model.

5.1.7. Statistical Significance of Results

As mentioned in Section 3.2.4, we defined our hypothesis $H^{(1)}$ to be evaluated through the main metrics, namely task performance, task completion time, and the NASA-TLX. Additionally, for our second hypothesis $H^{(2)}$, we defined the task completion time and the individual practicality Likert scale questions to be evaluated. We chose these metrics because they are the most relevant for our study and can be easily measured and evaluated.

As explained in Section 4.7.1, we do not have a normal distribution for all metrics. We assume, this is due to the relatively small sample size of participants ($n = 22$) and their rather low expertise which biases the data. However, using the mentioned Mann-Whitney U test, we can still calculate a p-value comparable with the p-value of the t-test. For our significance tests we chose a $p < 0.05$ to determine whether we can reject our null hypotheses, as this is a widely accepted threshold in statistical analysis. This threshold is appropriate for our study because it balances the risk of Type I errors (false positives) while maintaining sufficient sensitivity to detect meaningful effects in our relatively small sample size.

For our first hypothesis $H^{(1)}$, we argue that we can reject the null hypothesis $H_0^{(1)}$ as two out of our three main metrics (task performance and NASA-TLX score) are significant. The reasons for this are that the task performance is significant with a p-value of 0.00096, which is well below our threshold of 0.05. Additionally, we argue that the task completion time should be weighted less in our context because it can be influenced by factors unrelated to tool effectiveness, such as individual differences in reading speed, familiarity with similar interfaces, or varying strategies for task completion. In contrast, task performance and task load (NASA-TLX) more directly reflect the tool's ability to support accurate deployment model comprehension. When evaluating the hypothesis $H^{(1)}$ with regards to the individual tasks, we observe that the same metrics are significant for Task 1. However, they are not significant for Task 2. This shows that this significant difference in comprehension is especially pronounced for the simpler T2-Microservices deployment model.

For our second hypothesis $H^{(2)}$, we *cannot* reject the null hypothesis $H_0^{(2)}$ as only seven out of the fifteen task Likert scale questions are significant, in addition to the insignificant task completion time (see Table 4.8). However, the three personal preference questions show a significant difference in favor of the DeMAF in terms of the visualization and the ease of use. Additionally, when looking at the tasks individually, we can see some further significant differences, as ease of use and information content understanding. This shows that the task dependent evaluation also provides some meaningful insights, that would not be found with only a general evaluation.

5.2. Threats to Validity

Nevertheless, our study also has some limitation that need to be addressed when interpreting the results. One limitation is the narrow sample of mostly bachelor and master students from the computer science field, which is especially emphasized by the rather small sample size ($n = 22$). This may limit the generalizability and applicability of the results to other programmers, software architects and DevOps with different levels of experience or backgrounds. To increase external validity, our study could include a larger and more diverse sample of users.

Another limitation is our measurement of quality (i. e., affordance and UI/UX) which was only assessed using our own, custom Likert scale questions. Even though we tried to attune them to our needs, these measurements might neglect other important factors of tool quality, such as ease of use, the documentation or participant satisfaction. To increase construct validity, our study could introduce additional measures, which could provide a more comprehensive picture of tool quality. Furthermore, researching and adopting standardized, state-of-the-art questionnaires related to software architecture comprehension and tool quality, would help to remove the aforementioned limitations.

Thirdly, our study focused only on a singular visualization tool (for the deployment technology K8s) and two deployment models, namely T2-Microservices and OpenTelemetry Astronomy Shop. There might be other visualization tools for other deployment technologies that were not discovered by our initial research, as we only looked for FOSS on GitHub. There might even be a tool that can visualize multiple, different deployment technologies or an EDMM model. Other deployment models that are either easier or more difficult to comprehend could also change the significance of our study and its evaluation.

Finally, we mainly evaluated and compared the DeMAF and Monokle within each task. We chose this approach because, as mentioned in Section 3.2, we only used two deployment models with an within-subjects study design. However, a study with a broader range of deployment models and visualization tools and a between-subjects study design could provide a more comprehensive evaluation. This, especially if the study is conducted with more visualization tools and several different deployment models, could lead to an assessment of the actual added value of the DeMAF.

5.3. Implications

The findings of this study offer several implications for research and practice in deployment model visualization. Firstly, the findings underscore the significance of technology-agnostic visualizations in enhancing the comprehensibility of complex deployment models. Participants utilizing the DeMAF demonstrated higher performance on comprehension tasks (see Section 4.3), despite the models' complexity, and reported lower cognitive load (Section 4.4), suggesting that abstracted visual representations facilitate more effective understanding of deployment configurations.

Secondly, the comparative analysis demonstrates that the DeMAF offers a substantial advantage in early learning and usability contexts. As demonstrated in Table 4.7, participants who self-assessed as having a lower level of familiarity consistently preferred the DeMAF and rated it as more accessible. This suggests that the tool has the capacity to reduce entry barriers for novice users, such as students or practitioners who are new to deployment technologies. Furthermore, it supports the incorporation of abstracted model visualizations into educational and onboarding environments.

Thirdly, the DeMAF demonstrated superior performance in several pivotal usability metrics when compared with Monokle. The efficacy of the DeMAF in enhancing task performance was statistically significant ($p = 0.00096$; Table 4.8). Additionally, the DeMAF received a significantly lower NASA-TLX score ($p = 0.01312$; Table 4.8), indicating a notable reduction in frustration and an enhancement in task success (NASA-TLX, Section 4.4). This is why, we argue that we can reject the null hypothesis $H_0^{(1)}$ and accept our alternate hypothesis $H_1^{(1)}$. These results suggest that abstracted, graph-based representations, such as those generated by the DeMAF, may be more suitable in contexts requiring rapid comprehension or decision-making based on deployment structure.

Fourthly, the findings of this study indicate that visualizations that facilitate semantic abstraction, such as the recognition of service types and dependency relations (see Section 3.1.1), may prove to be more advantageous than low-level resource representations. This finding is consistent with the observation that users were more successful in deriving higher-level architectural insights using the DeMAF (e. g., Figure 4.6 and Figure 4.9), despite the fact that Monokle provided more detailed low-level configuration access.

In conclusion, the present study underscores the necessity for tools that are autonomous of particular deployment technologies. The DeMAF's capacity to transform and unify disparate TSDMs into a unified EDMM-based representation has been demonstrated to enhance tool maintainability while facilitating cross-technology comparisons and training (see Section 2.1.1). This positions the DeMAF as a promising foundation for future multi-platform deployment management interfaces.

6. Conclusion

In this final chapter, we summarize the objective and key findings of our research project in Section 6.1 – *Summary*. Finally, in Section 6.2 – *Future Work*, we outline potential options and objectives for future research and development of the DeMAF.

6.1. Summary

The objective of this research is to investigate the comprehensibility and usability of deployment model visualizations generated by the Deployment Model Abstraction Framework (DeMAF) compared to an established visualization tool, Monokle. In order to assess the impact of the DeMAF's abstracted, technology-agnostic visualization approach, a controlled user study was conducted with 22 participants using a within-subjects design (Section 3.2 – *Study Design*). Each participant analyzed two Kubernetes-based deployment models – T2-Microservices and OpenTelemetry Astronomy Shop – once using the DeMAF and once using Monokle.

The findings indicate that the DeMAF enhances task performance. Participants utilizing the DeMAF demonstrated a substantially higher mean percentage of correct responses on average across both tasks (58.02%) in comparison to those using Monokle (49.15%) (Section 4.3, Table 4.3a). This enhancement in precision is statistically significant ($p = 0.00096$; Table 4.8). While the mean task completion times were comparable between tools (see Section 4.2, Table 4.2), participants generally performed faster on simpler models with the DeMAF. However, the difference in performance times between the two tools did not reach statistical significance.

The participants further reported a lower cognitive load of 23% when utilizing the DeMAF, as indicated by the modified NASA Task Load Index (NASA-TLX) questionnaire. This lower cognitive load is statistically significant ($p = 0.01312$; Table 4.8). For Task 1, the NASA-TLX score was significantly lower for the DeMAF (average: 3.24 vs. 4.42; $p = 0.02254$), with notably lower frustration and higher perceived success (Section 4.4, Table 4.4). For Task 2, although the difference was not statistically significant, the DeMAF still yielded superior average scores (see Section 4.4, Table 4.10).

Subjective evaluations further support the superiority of the DeMAF. As demonstrated in Section 4.5, Figure 4.8, and Figure 4.9, participants consistently provided higher ratings to the DeMAF in terms of comprehensibility, ease of use, and visual clarity. The data indicates a clear preference for the DeMAF, particularly among participants with limited familiarity with deployment technologies (see Section 4.6, Table 4.7, Figure 4.12).

In summary, the present study corroborates the hypothesis that the DeMAF's abstracted visualization approach enhances the comprehensibility of deployment models without introducing additional cognitive burden. Its favorable performance across objective and subjective metrics supports its potential for broader adoption in both educational and practical deployment analysis contexts. These findings underscore the significance of technology-agnostic, semantically enriched visualizations in facilitating deployment-related decision-making (see Section 5.3 – *Implications*).

6.2. Future Work

To further evaluate and substantiate the effectiveness of the DeMAF in enhancing the comprehensibility of deployment models, several directions for future work are necessary.

First, a broader comparison with existing visualization tools should be conducted. While our current study focused on tools within the K8s ecosystem, many deployment models are not based on K8s. Including tools that visualize Ansible, Terraform, or Helm-based models would provide a more comprehensive benchmark and allow us to assess the DeMAF's performance in a wider range of deployment contexts.

Second, the initial study was limited in scale and scope, with a relatively small participant pool and a limited set of deployment models. A follow-up study should include a larger and more diverse sample of participants and deployment scenarios to improve the statistical significance and generalizability of our findings. This would help in more confidently assessing whether the DeMAF truly enhances model comprehensibility across varying levels of complexity and user expertise.

Third, the DeMAF is still under development and currently supports only a subset of technologies through its plugin architecture. As additional plugins are added, such as the in-development Docker plugin that enables semantic categorization of services (e.g., databases vs. generic services), the visualizations generated by the DeMAF may offer new insights into system architecture. Future studies should investigate whether these new features influence user comprehension and whether they lead to different conclusions regarding the DeMAF's effectiveness.

Finally, our second research question (RQ_2) was assessed using custom Likert scale questions. While these were tailored to our study's context, they lack external validation. Future work should involve identifying and adopting standardized, validated questionnaires related to software architecture comprehension. Conducting a new study using such instruments would allow for more reliable measurement and enable better comparison with related work. This could either reinforce or challenge our current findings regarding hypothesis $H^{(2)}$.

Bibliography

[Aut24] T. G. Authors. *Graphviz*. 2024. URL: https://graphviz.org/ (cit. on pp. 5, 6).

[Aut25a] O. Authors. *OpenTelemetry*. 2025. URL: https://opentelemetry.io/ (cit. on p. 12).

[Aut25b] O. Authors. *OpenTelemetry Demo Docs*. 2025. URL: https://opentelemetry.io/docs/demo/ (cit. on p. 12).

[ben22] benc-uk. *KubeView*. 2022. URL: https://kubeview.benco.io/ (cit. on p. 5).

[CAYT24] T. Cerny, A. S. Abdelfattah, J. Yero, D. Taibi. "From static code analysis to visual models of microservice architecture." In: *Cluster Computing* (2024), pp. 1–26 (cit. on p. 4).

[CGK12] G. Charness, U. Gneezy, M. A. Kuhn. "Experimental methods: Between-subject and within-subject design." In: *Journal of Economic Behavior & Organization* 81.1 (2012), pp. 1–8. ISSN: 0167-2681. DOI: https://doi.org/10.1016/j.jebo.2011.08.009. URL: https://www.sciencedirect.com/science/article/pii/S0167268111002289 (cit. on p. 11).

[cyc24] cycloidio. *InfraMap*. 2024. URL: https://github.com/cycloidio/inframap (cit. on p. 6).

[eejs20] elliotxkim, ed-roh, jamesvillarrubia, stevefan1999-personal. *SPEKT8*. 2020. URL: https://github.com/spekt8/spekt8 (cit. on p. 5).

[hai24] haidaraM. *Ansible Playbook Grapher*. 2024. URL: https://github.com/haidaraM/ansible-playbook-grapher (cit. on p. 5).

[HS88] S. G. Hart, L. E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research." In: *Human Mental Workload*. Ed. by P. A. Hancock, N. Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9. URL: https://www.sciencedirect.com/science/article/pii/S0166411508623869 (cit. on pp. 12, 13).

[KBBL13] O. Kopp, T. Binz, U. Breitenbücher, F. Leymann. "Winery–a modeling tool for TOSCA-based cloud applications." In: *Service-Oriented Computing: 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings 11*. Springer. 2013, pp. 700–704 (cit. on pp. 1, 3).

[Kd24] D. Kopp, delvh. *T2-Project*. 2024. URL: https://github.com/t2-project/t2-project (cit. on p. 12).

[Kop24a] D. Kopp. *T2-Microservices*. 2024. URL: https://github.com/t2-project/microservices (cit. on p. 12).

[Kop24b] D. Kopp. *T2-Modulith*. 2024. URL: https://github.com/t2-project/modulith (cit. on p. 12).

[Kub24a] Kubeshop. *monokle*. 2024. URL: https://github.com/kubeshop/monokle (cit. on p. 6).

[Kub24b] Kubeshop. *Monokle Desktop*. 2024. URL: https://docs.monokle.io (cit. on p. 6).

[Kub25] Kubeshop. *Monokle Desktop IDE*. 2025. URL: https://monokle.io/ (cit. on p. 5).

[KWH+24] O. Kopp, M. Wurster, L. Harzenetter, C. Kleine, K. Saatkamp, T. Binz, U. Breitenbücher, C. T. Sungur, P. Hirmer. *Eclipse Winery*. 2024. URL: https://projects.eclipse.org/projects/automotive.winery (cit. on pp. 1, 3).

[Lev60] H. Levene. "Robust Tests for Equality of Variances." In: *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*. Ed. by I. Olkin, H. Hotelling. Stanford University Press, 1960, pp. 278–292 (cit. on p. 31).

[LFH17] J. Lazar, J. H. Feng, H. Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017 (cit. on p. 4).

[MHS+24] E. Mueller, F. Heidrich, P. Schur, R. Tischler, L. Rönsch. *OpenTelemetry Demo*. 2024. URL: https://github.com/UST-DeMAF/opentelemetry-demo (cit. on p. 13).

[MW47] H. B. Mann, D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other." In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. ISSN: 00034851. URL: http://www.jstor.org/stable/2236101 (visited on 05/22/2025) (cit. on p. 32).

[nic17] nickjj. *Ansigenome*. 2017. URL: https://github.com/nickjj/ansigenome (cit. on p. 5).

[Par25] A. Parker. *OpenTelemetry Demo*. 2025. URL: https://github.com/open-telemetry/opentelemetry-demo (cit. on p. 13).

[RG06] R. Razali, P. Garratt. "Measuring the Comprehensibility of a UML-B Model and a B Model." In: 16 (Nov. 2006) (cit. on p. 7).

[SK24a] S. S. Stieß, D. Kopp. *T2-Modulith*. 2024. URL: https://t2-documentation.readthedocs.io/en/latest/modulith/index.html (cit. on p. 12).

[SK24b] S. S. Stieß, D. Kopp. *T2-Project Documentation*. 2024. URL: https://t2-documentation.readthedocs.io/en/latest/index.html (cit. on p. 12).

[SK24c] S. S. Stieß, D. Kopp. *T2-Project Microservices*. 2024. URL: https://t2-documentation.readthedocs.io/en/latest/microservices/index.html (cit. on p. 12).

[SKd24] S. S. Stieß, D. Kopp, delvh. *t2-project*. 2024. URL: https://github.com/t2-project (cit. on p. 12).

[Stu08] Student. "The Probable Error of a Mean." In: *Biometrika* 6.1 (1908), pp. 1–25. ISSN: 00063444, 14643510. URL: http://www.jstor.org/stable/2331554 (visited on 05/22/2025) (cit. on p. 32).

[SVC+24] K. Sveidqvist, S. Vinod, N. Cuzon, A. Jain, T. Liu, R. Al Sulais, A. Klink, N. Rozhkov, J. Greywolf, M. Morel, S. Huynh, M. Faber, Y. Singh, N. Orlandoni, P. Brolin, M. Laganeckas. *Mermaid*. 2024. URL: https://mermaid.js.org/ (cit. on p. 5).

[SW65] S. S. SHAPIRO, M. B. WILK. "An analysis of variance test for normality (complete samples)†." In: *Biometrika* 52.3-4 (Dec. 1965), pp. 591–611. ISSN: 0006-3444. DOI: 10.1093/biomet/52.3-4.591. eprint: https://academic.oup.com/biomet/article-pdf/52/3-4/591/962907/52-3-4-591.pdf. URL: https://doi.org/10.1093/biomet/52.3-4.591 (cit. on p. 31).

[VB18] B. Varghese, R. Buyya. "Next generation cloud computing: New trends and research directions." In: *Future Generation Computer Systems* 79 (2018), pp. 849–861 (cit. on p. 1).

[WBF+20] M. Wurster, U. Breitenbücher, M. Falkenthal, C. Krieger, F. Leymann, K. Saatkamp, J. Soldani. "The essential deployment metamodel: a systematic review of deployment automation technologies." In: *SICS Software-Intensive Cyber-Physical Systems* 35 (2020), pp. 63–75 (cit. on pp. 1, 3).

[WBSB23] M. Weller, U. Breitenbücher, S. Speth, S. Becker. "The Deployment Model Abstraction Framework." In: *Enterprise Design, Operations, and Computing. EDOC 2022 Workshops*. Ed. by T. P. Sales, H. A. Proper, G. Guizzardi, M. Montali, F. M. Maggi, C. M. Fonseca. Cham: Springer International Publishing, 2023, pp. 319–325. ISBN: 978-3-031-26886-1 (cit. on pp. 1, 3).

[Wel22] M. Weller. *kube*. 2022. URL: https://github.com/Well5a/kube (cit. on p. 12).

All links were last checked on May 26, 2025.

A. Appendix

A.1. Tables

Table A.1.: This table shows the mean agreement of the participants to the respective statements regarding the tool's practical utility, split into task and tool.

	Task 1			Task 2		
Statement	Both	DeMAF	Monokle	Both	DeMAF	Monokle
The visualization helped me understand the deployment model clearly.	3.05	3.45	2.64	2.41	3.0	1.82
The information presented was easy to interpret.	3.05	3.64	2.45	2.82	2.91	2.73
The tool was easy to navigate.	3.73	4.09	3.36	2.95	3.36	2.55
I could efficiently find the information I needed.	2.86	3.36	2.36	2.64	2.73	2.55
The tool provided all necessary information about the deployment model.	3.23	3.73	2.73	3.36	3.45	3.27
I was able to extract the required details without difficulty.	2.68	3.09	2.27	2.5	2.55	2.45
The graphical representation was visually appealing.	3.68	3.73	3.64	2.91	3.27	2.55
The layout and design made it easy to follow the model structure.	3.45	3.82	3.09	2.82	3.36	2.27
I could analyze the deployment model quickly using this tool.	3.0	3.27	2.73	2.64	3.09	2.18
The tool made the analysis process more efficient.	3.45	3.64	3.27	3.0	3.45	2.55

Table A.2.: This table shows the results of the significance tests for **both tasks** and which significance test was used.

Metric	Test	p-value	Significant ($p < 0.05$)
Performance (Aggregated)	t-test	0.00096	yes
"Don't know" count	Mann-Whitney U	0.09253	no
Task completion time	t-test	0.43558	no
NASA-TLX	t-test	0.01312	yes
How mentally demanding were the tasks?	t-test	0.01287	yes
How hurried or rushed were the pace of the tasks?	t-test	0.53673	no
How successful were you in accomplishing what you were asked to do?	t-test	0.20325	no
How hard did you have to work to accomplish your level of performance?	Mann-Whitney U	0.03425	yes
How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool?	Mann-Whitney U	0.01448	yes
The visualization helped me understand the deployment model clearly.	Mann-Whitney U	0.01281	yes
The information presented was easy to interpret.	t-test	0.05106	no
The tool was easy to navigate.	Mann-Whitney U	0.02394	yes
I could efficiently find the information I needed.	Mann-Whitney U	0.06499	no
The tool provided all necessary information about the deployment model.	Mann-Whitney U	0.10163	no
I was able to extract the required details without difficulty.	Mann-Whitney U	0.16554	no
The graphical representation was visually appealing.	t-test	0.31616	no
The layout and design made it easy to follow the model structure.	Mann-Whitney U	0.01677	yes

I could analyze the deployment model quickly using this tool.	t-test	0.06067	no
The tool made the analysis process more efficient.	t-test	0.06458	no

Table A.3.: This table shows the results of the significance tests for **Task 1** and which significance test was used.

Metric	Test	p-value	Significant ($p < 0.05$)
Performance (Aggregated)	t-test	0.00160	yes
“Don’t know” count	Welch’s t-test	0.05541	no
Task completion time	t-test	0.46695	no
NASA-TLX	t-test	0.02254	yes
How mentally demanding were the tasks?	t-test	0.06355	no
How hurried or rushed were the pace of the tasks?	t-test	0.45259	no
How successful were you in accomplishing what you were asked to do?	t-test	0.01467	yes
How hard did you have to work to accomplish your level of performance?	Mann-Whitney U	0.12757	no
How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool?	Mann-Whitney U	0.01965	yes
The visualization helped me understand the deployment model clearly.	Mann-Whitney U	0.14538	no
The information presented was easy to interpret.	t-test	0.01416	yes
The tool was easy to navigate.	Mann-Whitney U	0.05181	no
I could efficiently find the information I needed.	Mann-Whitney U	0.00949	yes
The tool provided all necessary information about the deployment model.	Mann-Whitney U	0.03540	yes
I was able to extract the required details without difficulty.	Mann-Whitney U	0.04386	yes

The graphical representation was visually appealing.	t-test	0.87309	no
The layout and design made it easy to follow the model structure.	Mann-Whitney U	0.16467	no
I could analyze the deployment model quickly using this tool.	t-test	0.34077	no
The tool made the analysis process more efficient.	t-test	0.43311	no

Table A.4.: This table shows the results of the significance tests for **Task 2** and which significance test was used.

Metric	Test	p-value	Significant ($p < 0.05$)
Performance (Aggregated)	t-test	0.14922	no
"Don't know" count	Mann-Whitney U	0.45809	no
Task completion time	t-test	0.06502	no
NASA-TLX	Mann-Whitney U	0.14649	no
How mentally demanding were the tasks?	Mann-Whitney U	0.06449	no
How hurried or rushed were the pace of the tasks?	t-test	0.78189	no
How successful were you in accomplishing what you were asked to do?	Mann-Whitney U	0.50018	no
How hard did you have to work to accomplish your level of performance?	t-test	0.12617	no
How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool?	t-test	0.22080	no
The visualization helped me understand the deployment model clearly.	Mann-Whitney U	0.04574	yes
The information presented was easy to interpret.	Mann-Whitney U	0.83401	no
The tool was easy to navigate.	Mann-Whitney U	0.14538	no
I could efficiently find the information I needed.	t-test	0.75024	no

The tool provided all necessary information about the deployment model.	*t*-test	0.73518	no
I was able to extract the required details without difficulty.	Mann-Whitney U	0.97125	no
The graphical representation was visually appealing.	Mann-Whitney U	0.17207	no
The layout and design made it easy to follow the model structure.	*t*-test	0.03868	yes
I could analyze the deployment model quickly using this tool.	*t*-test	0.06864	no
The tool made the analysis process more efficient.	*t*-test	0.06300	no

A.2. Figures

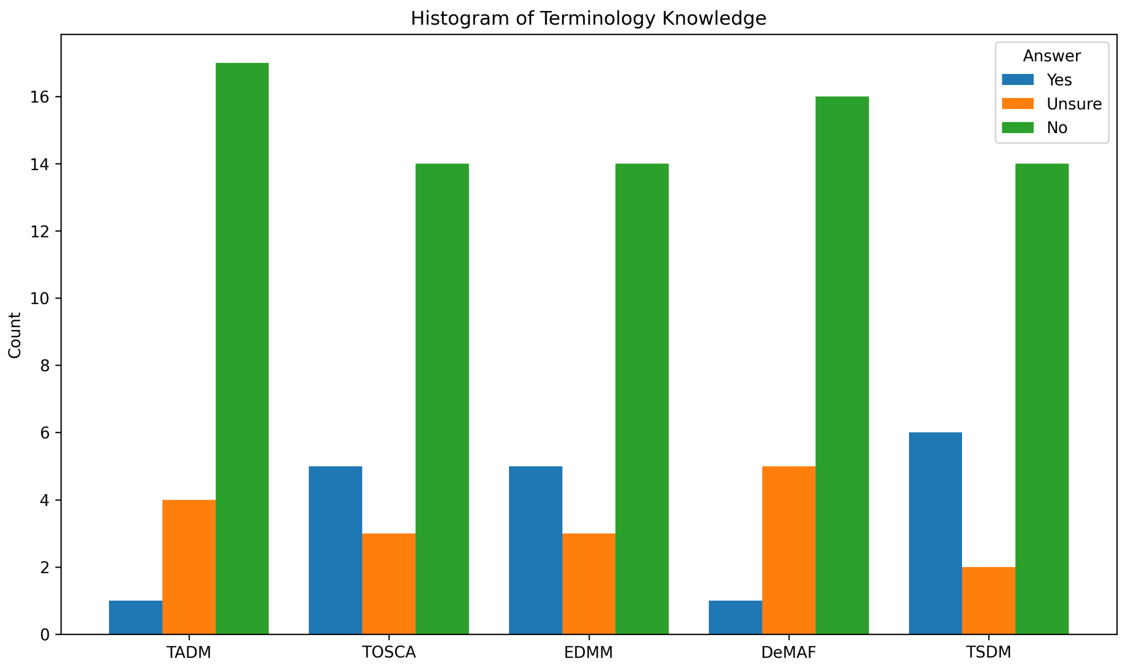


Figure A.1.: This histogram shows the participants' familiarity with the terms used in the study. The terms are: TADM, TOSCA, EDMM, DeMAF and TSDM.

A.2.1. Visualization Tools

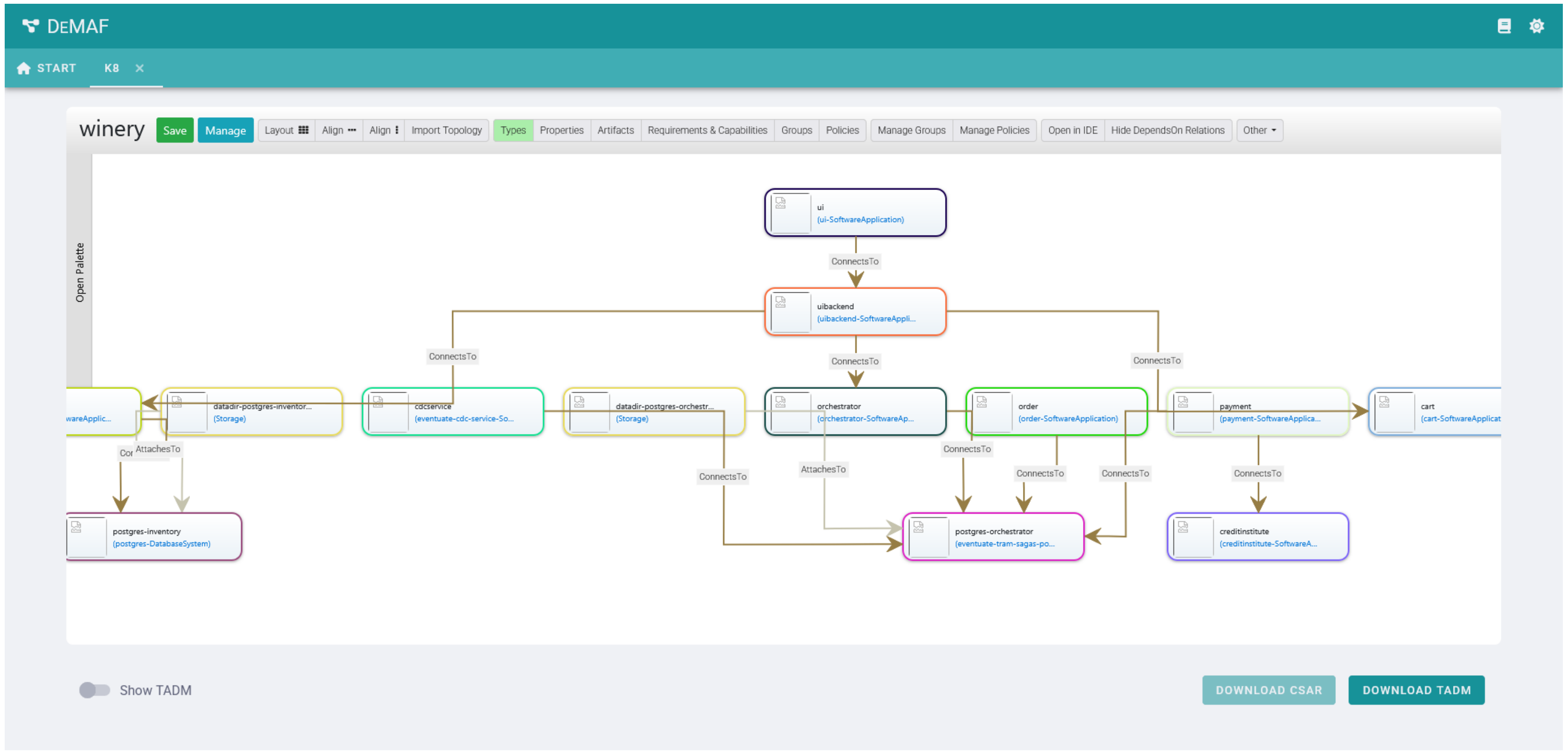
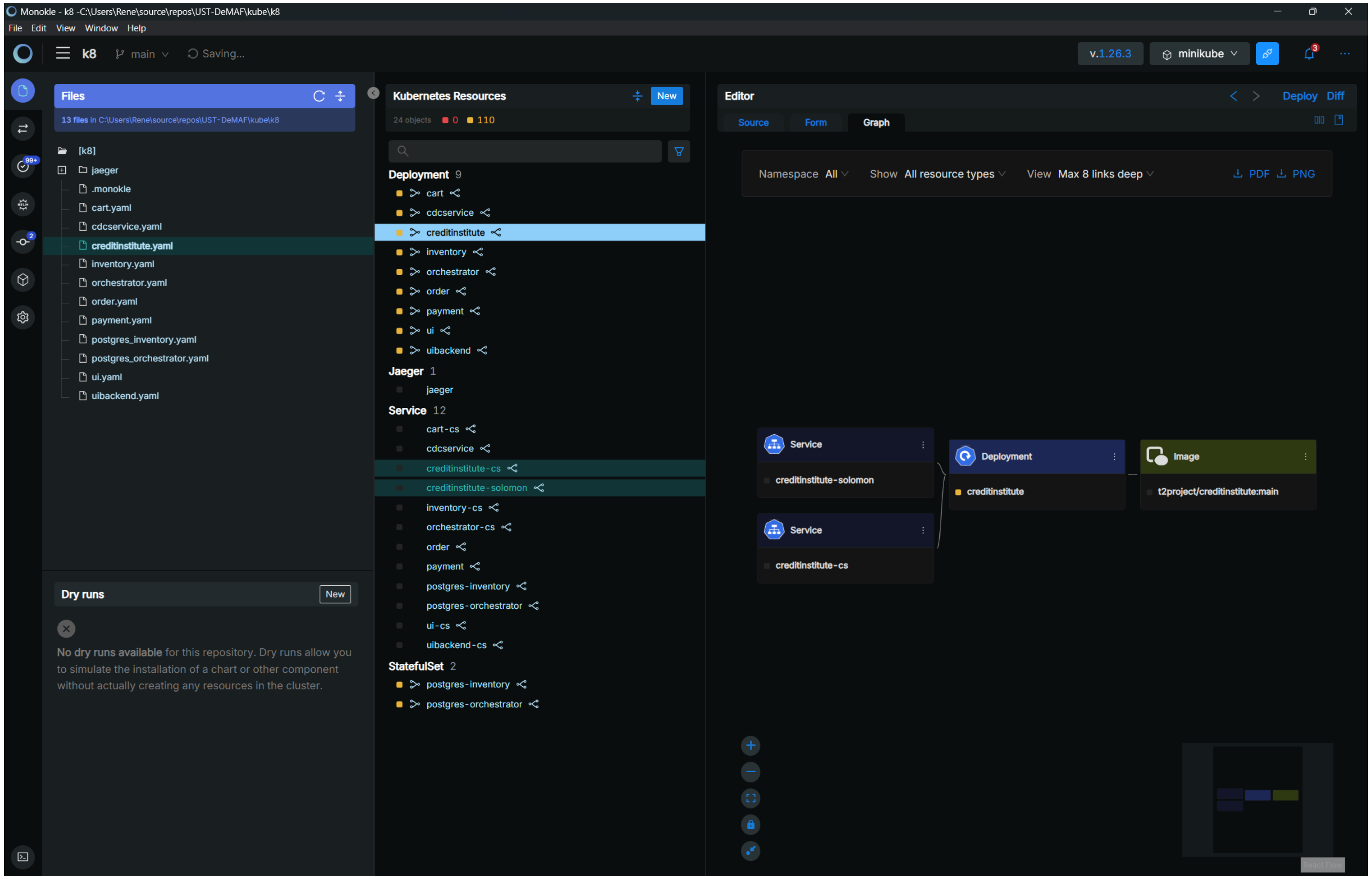


Figure A.2.: DeMAF visualizing the T2-Microservices deployment model.



13 files in C:\Users\Rene\source\repos\UST-DeMAF\kube\k8

[k8]
jaeger
.monokle
cart.yaml
cdcservice.yaml
creditinstitute.yaml
inventory.yaml
orchestrator.yaml
order.yaml
payment.yaml
postgres_inventory.yaml
postgres_orchestrator.yaml
ui.yaml
uibackend.yaml

No dry runs available for this repository. Dry runs allow you to simulate the installation of a chart or other component without actually creating any resources in the cluster.

24 objects 0 110

Deployment 9
cart
cdcservice
creditinstitute
inventory
orchestrator
order
payment
ui
uibackend
Jaeger 1
jaeger
Service 12
cart-cs
cdcservice
creditinstitute-cs
creditinstitute-solomon
inventory-cs
orchestrator-cs
order
payment
postgres-inventory
postgres-orchestrator
ui-cs
uibackend-cs
StatefulSet 2
postgres-inventory
postgres-orchestrator

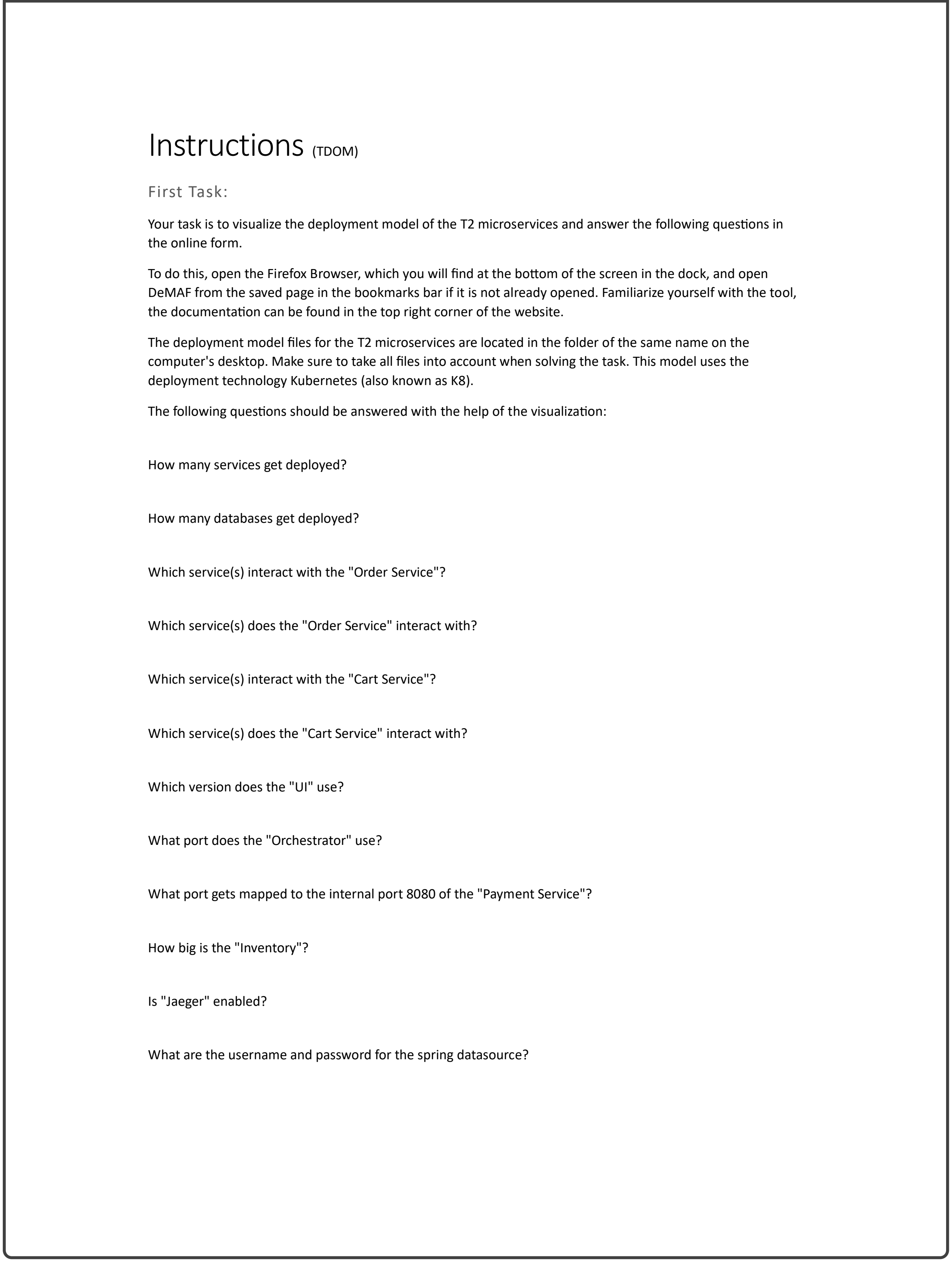
Namespace All
Show All resource types
View Max 8 links deep

Service: creditinstitute-solomon
Service: creditinstitute-cs
Deployment: creditinstitute
Image: t2project/creditinstitute:main

Figure A.3.: Monokle visualizing the T2-Microservices deployment model.

A.3. Instruction Sheet

Instruction Sheet A [1]



Instructions (TDOM)

First Task:

Your task is to visualize the deployment model of the T2 microservices and answer the following questions in the online form.

To do this, open the Firefox Browser, which you will find at the bottom of the screen in the dock, and open DeMAF from the saved page in the bookmarks bar if it is not already opened. Familiarize yourself with the tool, the documentation can be found in the top right corner of the website.

The deployment model files for the T2 microservices are located in the folder of the same name on the computer's desktop. Make sure to take all files into account when solving the task. This model uses the deployment technology Kubernetes (also known as K8).

The following questions should be answered with the help of the visualization:

How many services get deployed?

How many databases get deployed?

Which service(s) interact with the "Order Service"?

Which service(s) does the "Order Service" interact with?

Which service(s) interact with the "Cart Service"?

Which service(s) does the "Cart Service" interact with?

Which version does the "UI" use?

What port does the "Orchestrator" use?

What port gets mapped to the internal port 8080 of the "Payment Service"?

How big is the "Inventory"?

Is "Jaeger" enabled?

What are the username and password for the spring datasource?

Instruction Sheet A [2]

Second Task:

Now visualize the deployment model of the OpenTelemetry Astronomy Shop Demo and answer the following questions from the online form.

Open the Monokle application, which you will find at the bottom of the screen in the dock, it looks like this:

The tool's documentation is saved in Firefox's bookmarks bar, which you will also find in the dock, with the label Monokle Docs. Alternatively, you can call up the documentation in the menu at the top right-hand corner of the application.

The OpenTelemetry Astronomy Shop Demo deployment model files can also be found in the folder with the same name on the machine's desktop. Make sure to take all files into consideration while solving the tasks. This model also uses the deployment technology Kubernetes (also called K8).

The following questions should be answered with the help of the visualization:

How many services get deployed?

How many databases get deployed?

Which service(s) interact with the "Checkout Service"?

Which service(s) does the "Checkout Service" interact with?

Which service(s) interact with the "Product Catalog Service"?

Which service(s) does the "Product Catalog Service" interact with?

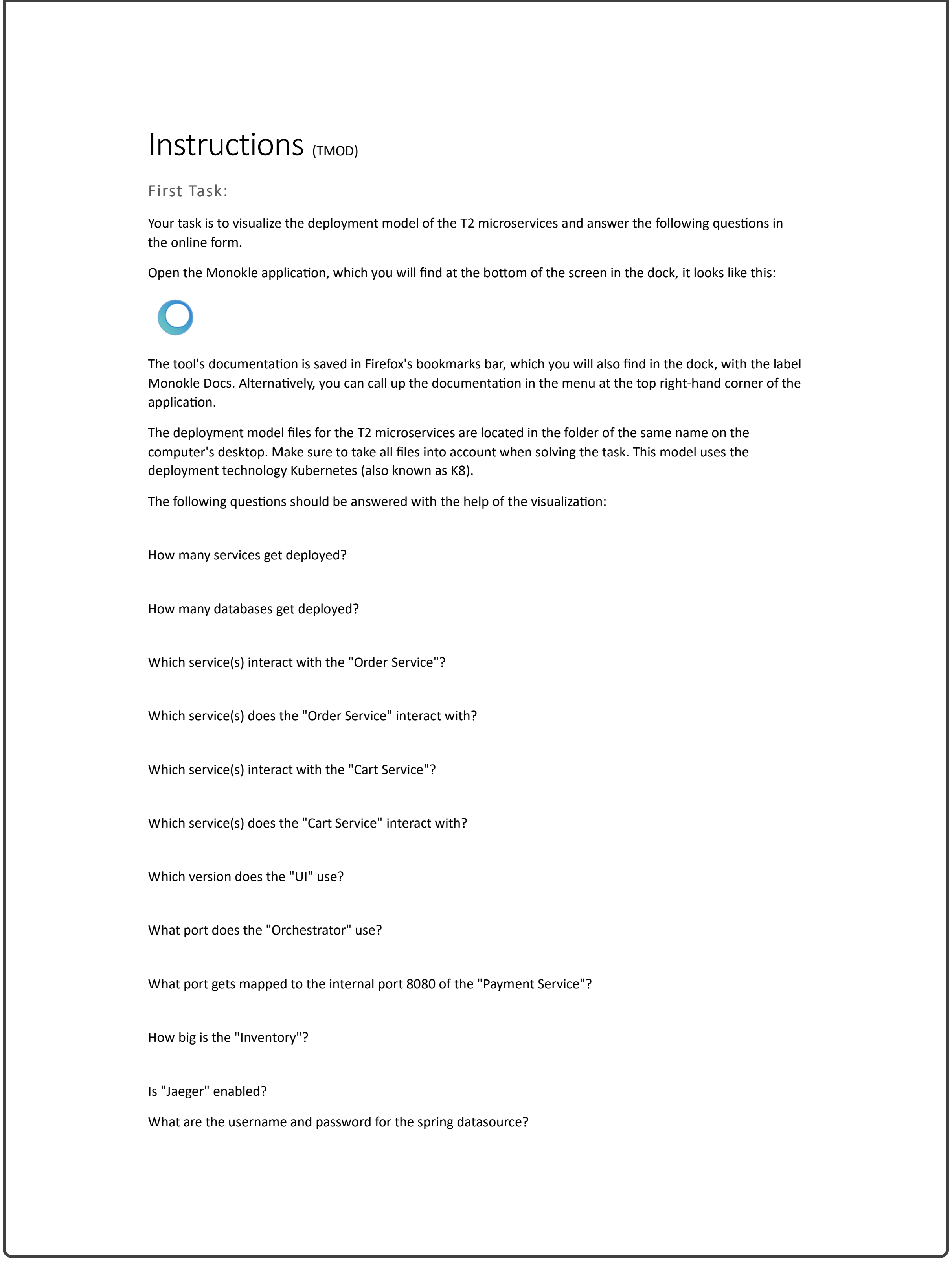
Which version does the Mongo database use?

What port does the Kafka queue use?

What port gets mapped to the internal port 27017 of the MongoDB?

What are the username and password for the MongoDB?

Instruction Sheet B [1]



Instructions (TMOD)

First Task:

Your task is to visualize the deployment model of the T2 microservices and answer the following questions in the online form.

Open the Monokle application, which you will find at the bottom of the screen in the dock, it looks like this:

The tool's documentation is saved in Firefox's bookmarks bar, which you will also find in the dock, with the label Monokle Docs. Alternatively, you can call up the documentation in the menu at the top right-hand corner of the application.

The deployment model files for the T2 microservices are located in the folder of the same name on the computer's desktop. Make sure to take all files into account when solving the task. This model uses the deployment technology Kubernetes (also known as K8).

The following questions should be answered with the help of the visualization:

How many services get deployed?

How many databases get deployed?

Which service(s) interact with the "Order Service"?

Which service(s) does the "Order Service" interact with?

Which service(s) interact with the "Cart Service"?

Which service(s) does the "Cart Service" interact with?

Which version does the "UI" use?

What port does the "Orchestrator" use?

What port gets mapped to the internal port 8080 of the "Payment Service"?

How big is the "Inventory"?

Is "Jaeger" enabled?

What are the username and password for the spring datasource?

Instruction Sheet B [2]

Second Task:

Now visualize the deployment model of the OpenTelemetry Astronomy Shop Demo and answer the following questions from the online form.

To do this, open the Firefox Browser, which you will find at the bottom of the screen in the dock, and open DeMAF from the saved page in the bookmarks bar if it is not already opened. Familiarize yourself with the tool, the documentation can be found in the top right corner of the website.

The OpenTelemetry Astronomy Shop Demo deployment model files can also be found in the folder with the same name on the machine's desktop. Make sure to take all files into consideration while solving the tasks. This model also uses the deployment technology Kubernetes (also called K8).

The following questions should be answered with the help of the visualization:

How many services get deployed?

How many databases get deployed?

Which service(s) interact with the "Checkout Service"?

Which service(s) does the "Checkout Service" interact with?

Which service(s) interact with the "Product Catalog Service"?

Which service(s) does the "Product Catalog Service" interact with?

Which version does the Mongo database use?

What port does the Kafka queue use?

What port gets mapped to the internal port 27017 of the MongoDB?

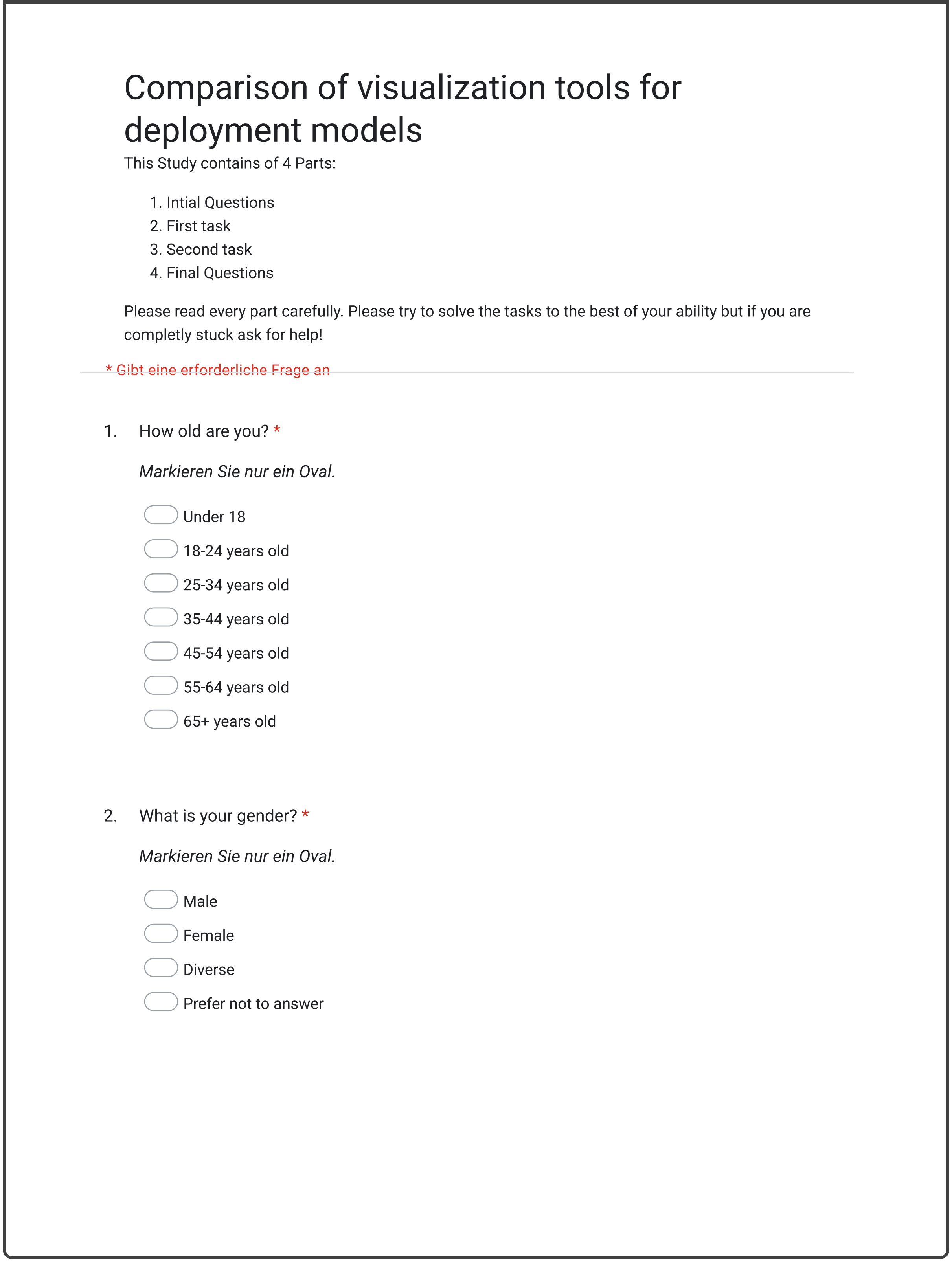
What are the username and password for the MongoDB?

A.4. Questionnaire

This section presents all the questions from our study questionnaire. The following notes address issues encountered during printing, such as text truncation or missing columns:

- **Questionnaire [4]:** For question *9.*, the left most option is labeled *"Not familiar"*.
- **Questionnaire [7]:** The question *17.* has an additional column named *"Don't know"*.
- **Questionnaire [9]:** For question *26.*, the left most option is labeled *"Not useful"*.
- **Questionnaire [9] – [10]:** For questions *27. – 31.*, the left most option is labeled "Very low", except for question *29.* where it is *"Perfect"*.
- **Questionnaire [10] – [13]:** For questions *32. – 41.*, the left most option is labeled *"Strongly disagree"*.
- **Questionnaire [15]:** The question *48.* has additional columns, including the cut-off one, named:
  - *"Recommendation Service"*
  - *"Accounting Service"*
  - *"Fraud Detection Service"*
  - *"Payment Service"*
  - *"Product Catalog Service"*
  - *"None"*
  - *"Don't know"*
- **Questionnaire [17]:** For question *55.*, the left most option is labeled *"Not useful"*.
- **Questionnaire [17] – [18]:** For questions *56. – 60.*, the left most option is labeled *"Very low"*, except for question *58.* where it is *"Perfect"*.
- **Questionnaire [19] – [20]:** For questions *61. – 70.*, the left most option is labeled *"Strongly disagree"*.
- **Questionnaire [21]:** For questions *74. – 78.*, the left most option is labeled *"Strongly preferred DeMAF"*.
- **Questionnaire [24]:** For questions *84. – 86.*, the left most option is labeled *"Strongly agree"*.

Questionnaire [1]



Comparison of visualization tools for deployment models

This Study contains of 4 Parts:

1. Intial Questions
2. First task
3. Second task
4. Final Questions

Please read every part carefully. Please try to solve the tasks to the best of your ability but if you are completly stuck ask for help!

* Gibt eine erforderliche Frage an

1. How old are you? *

Markieren Sie nur ein Oval.

- ◯ Under 18
- ◯ 18-24 years old
- ◯ 25-34 years old
- ◯ 35-44 years old
- ◯ 45-54 years old
- ◯ 55-64 years old
- ◯ 65+ years old

2. What is your gender? *

Markieren Sie nur ein Oval.

- ◯ Male
- ◯ Female
- ◯ Diverse
- ◯ Prefer not to answer

Questionnaire [2]

3. What is your Level of Education? *

   Markieren Sie nur ein Oval.

   - ◯ Bachelor
   - ◯ Master
   - ◯ PhD
   - ◯ No higher education level
   - ◯ Sonstiges: ____________________

4. What is your Field of Study?

   Markieren Sie nur ein Oval.

   - ◯ Computer Science
   - ◯ Software Engineering
   - ◯ Data Science
   - ◯ Medieninformatik
   - ◯ Wirtschaftsinformatik
   - ◯ AI and Data Science
   - ◯ Sonstiges: ____________________

5. If you're still studying, what semester are you in? Dropdown

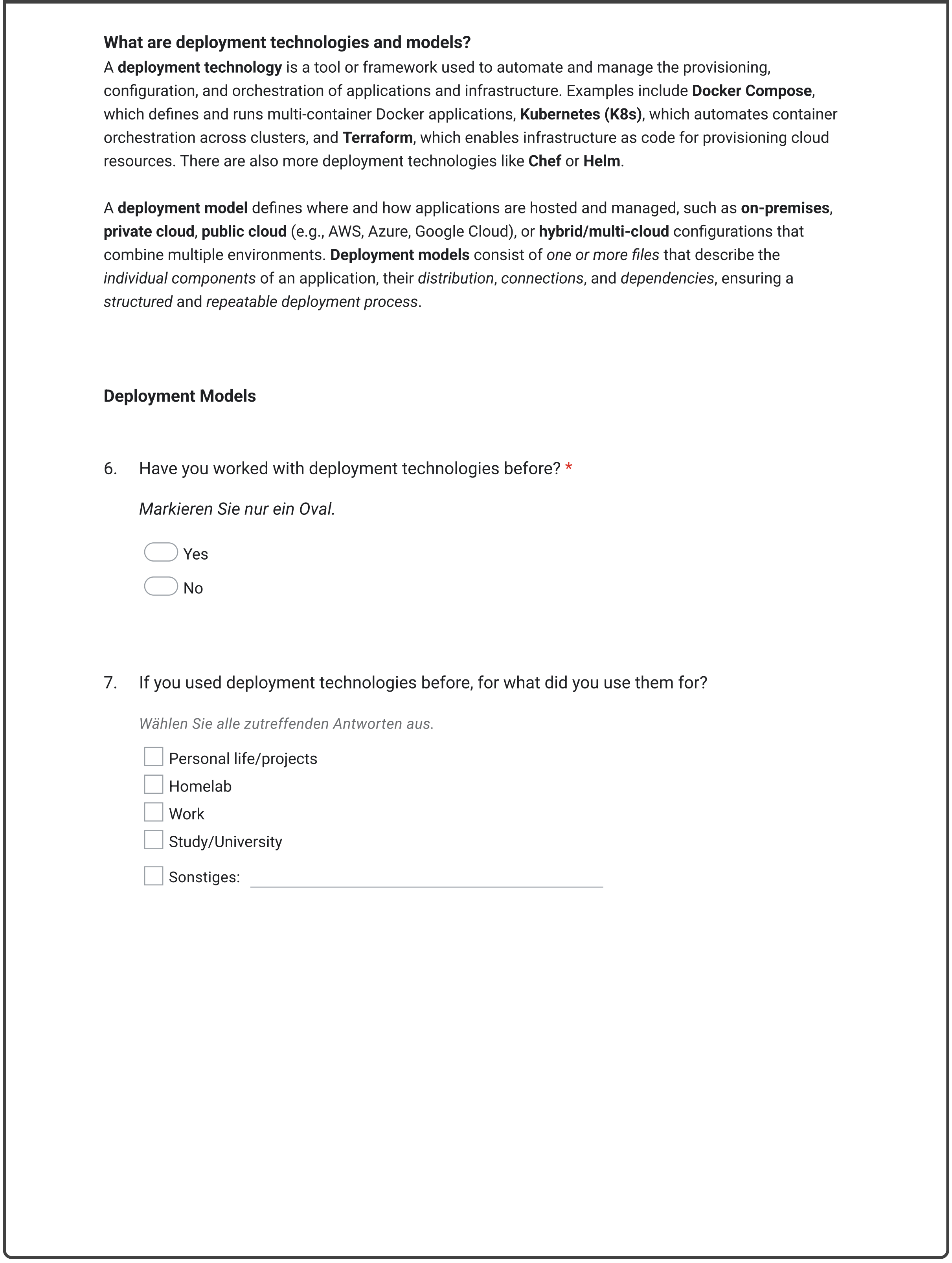
   Markieren Sie nur ein Oval.

   - ◯ 1
   - ◯ 2
   - ◯ 3
   - ◯ 4
   - ◯ 5
   - ◯ 6
   - ◯ 6+

Experience with Deployment Technologies & Tools

This section is about your prior knowledge on deployment technologies and tools used to visualize their structure.

Questionnaire [3]



What are deployment technologies and models?
A **deployment technology** is a tool or framework used to automate and manage the provisioning, configuration, and orchestration of applications and infrastructure. Examples include **Docker Compose**, which defines and runs multi-container Docker applications, **Kubernetes (K8s)**, which automates container orchestration across clusters, and **Terraform**, which enables infrastructure as code for provisioning cloud resources. There are also more deployment technologies like **Chef** or **Helm**.

A **deployment model** defines where and how applications are hosted and managed, such as **on-premises**, **private cloud**, **public cloud** (e.g., AWS, Azure, Google Cloud), or **hybrid/multi-cloud** configurations that combine multiple environments. **Deployment models** consist of *one or more files* that describe the *individual components* of an application, their *distribution*, *connections*, and *dependencies*, ensuring a *structured* and *repeatable deployment process*.

Deployment Models

6. Have you worked with deployment technologies before? *

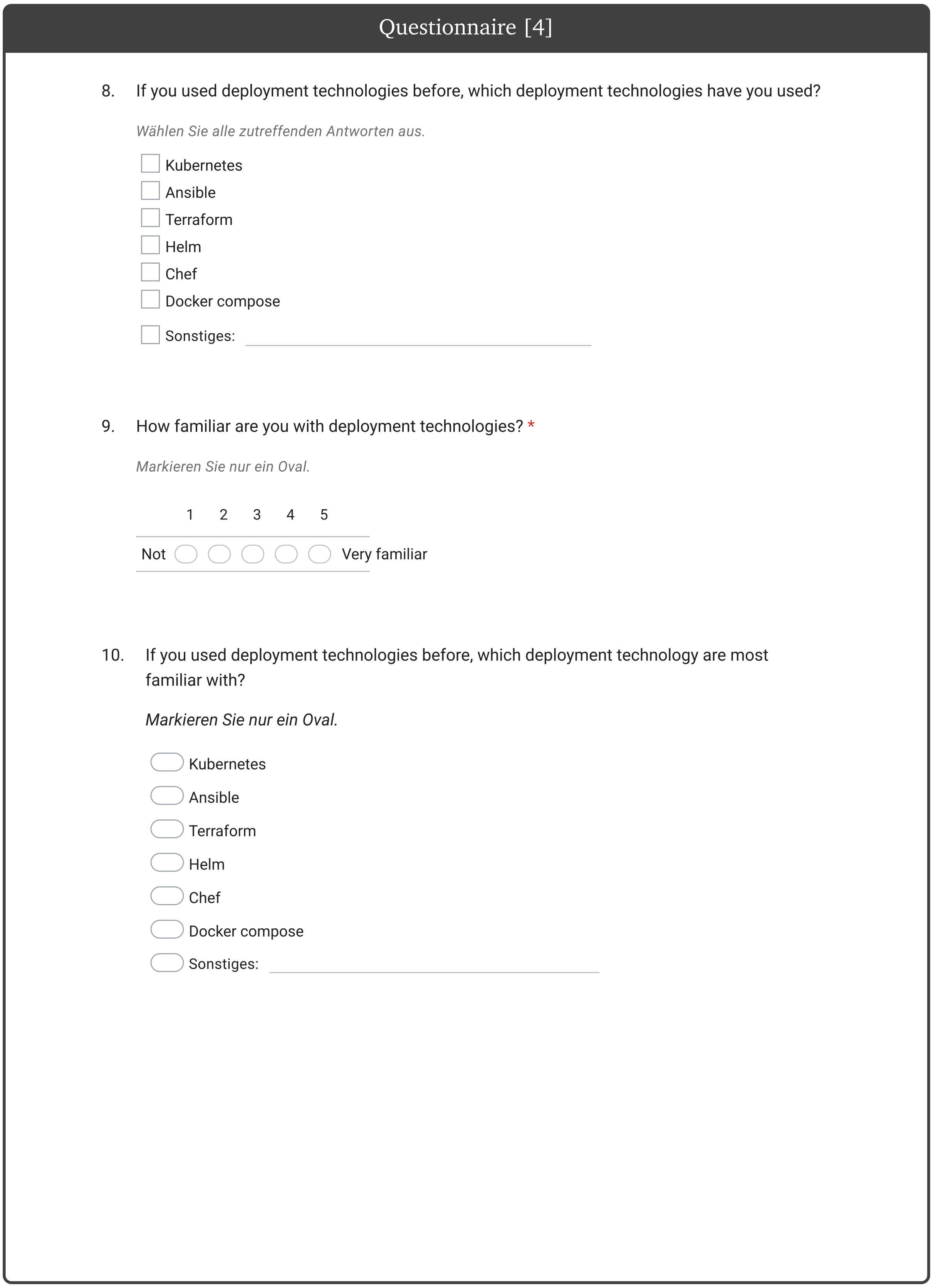
   Markieren Sie nur ein Oval.

   ◯ Yes
   ◯ No

7. If you used deployment technologies before, for what did you use them for?

   Wählen Sie alle zutreffenden Antworten aus.

   ☐ Personal life/projects
   ☐ Homelab
   ☐ Work
   ☐ Study/University
   ☐ Sonstiges: ____________________



Questionnaire [4]

8. If you used deployment technologies before, which deployment technologies have you used?

Wählen Sie alle zutreffenden Antworten aus.

- ☐ Kubernetes
- ☐ Ansible
- ☐ Terraform
- ☐ Helm
- ☐ Chef
- ☐ Docker compose
- ☐ Sonstiges: ____________________

9. How familiar are you with deployment technologies? *

Markieren Sie nur ein Oval.

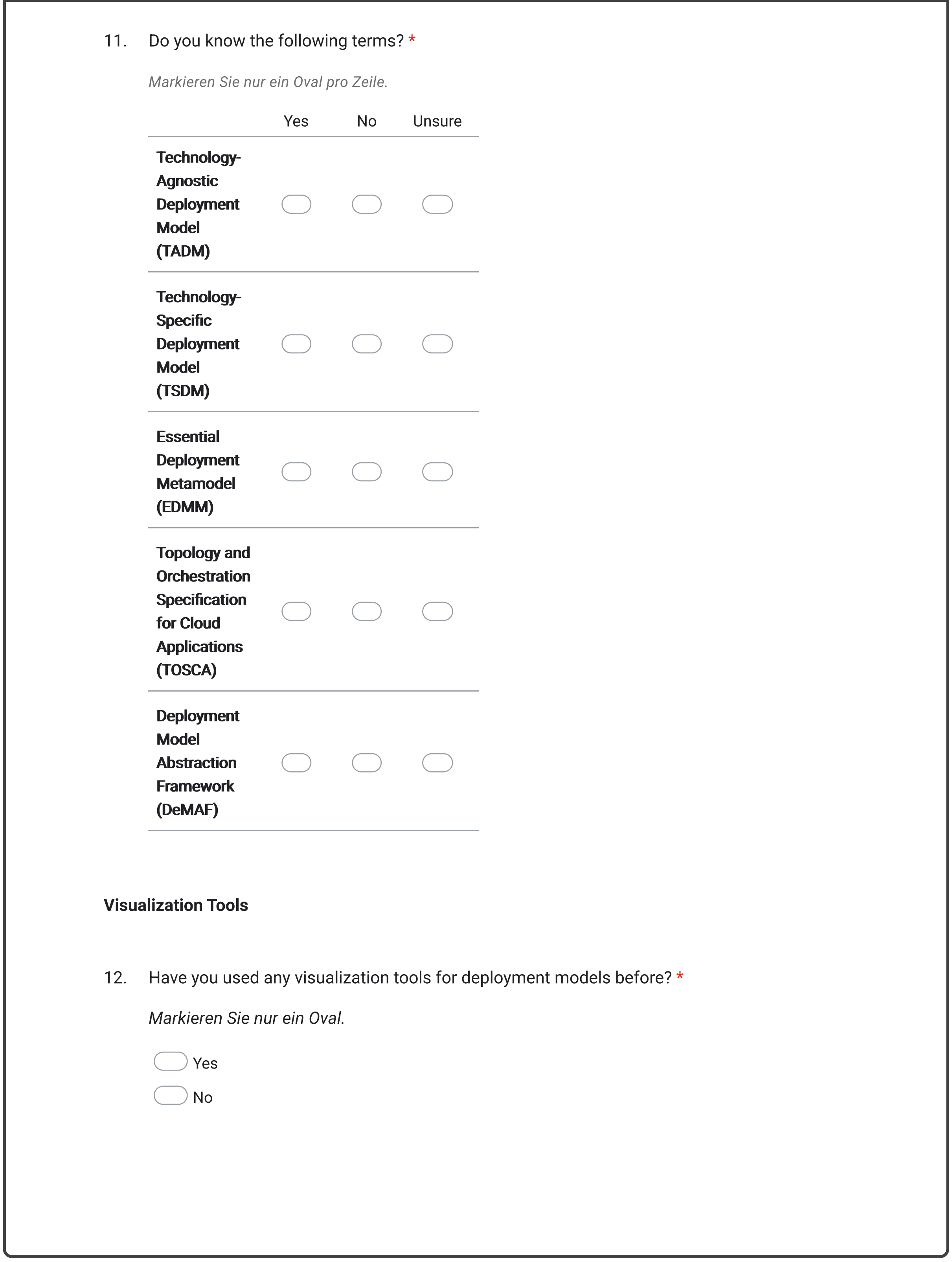
	1	2	3	4	5	
Not	○	○	○	○	○	Very familiar

10. If you used deployment technologies before, which deployment technology are most familiar with?

Markieren Sie nur ein Oval.

- ○ Kubernetes
- ○ Ansible
- ○ Terraform
- ○ Helm
- ○ Chef
- ○ Docker compose
- ○ Sonstiges: ____________________

Questionnaire [5]



11. Do you know the following terms? *

Markieren Sie nur ein Oval pro Zeile.

	Yes	No	Unsure
Technology-Agnostic Deployment Model (TADM)	◯	◯	◯
Technology-Specific Deployment Model (TSDM)	◯	◯	◯
Essential Deployment Metamodel (EDMM)	◯	◯	◯
Topology and Orchestration Specification for Cloud Applications (TOSCA)	◯	◯	◯
Deployment Model Abstraction Framework (DeMAF)	◯	◯	◯

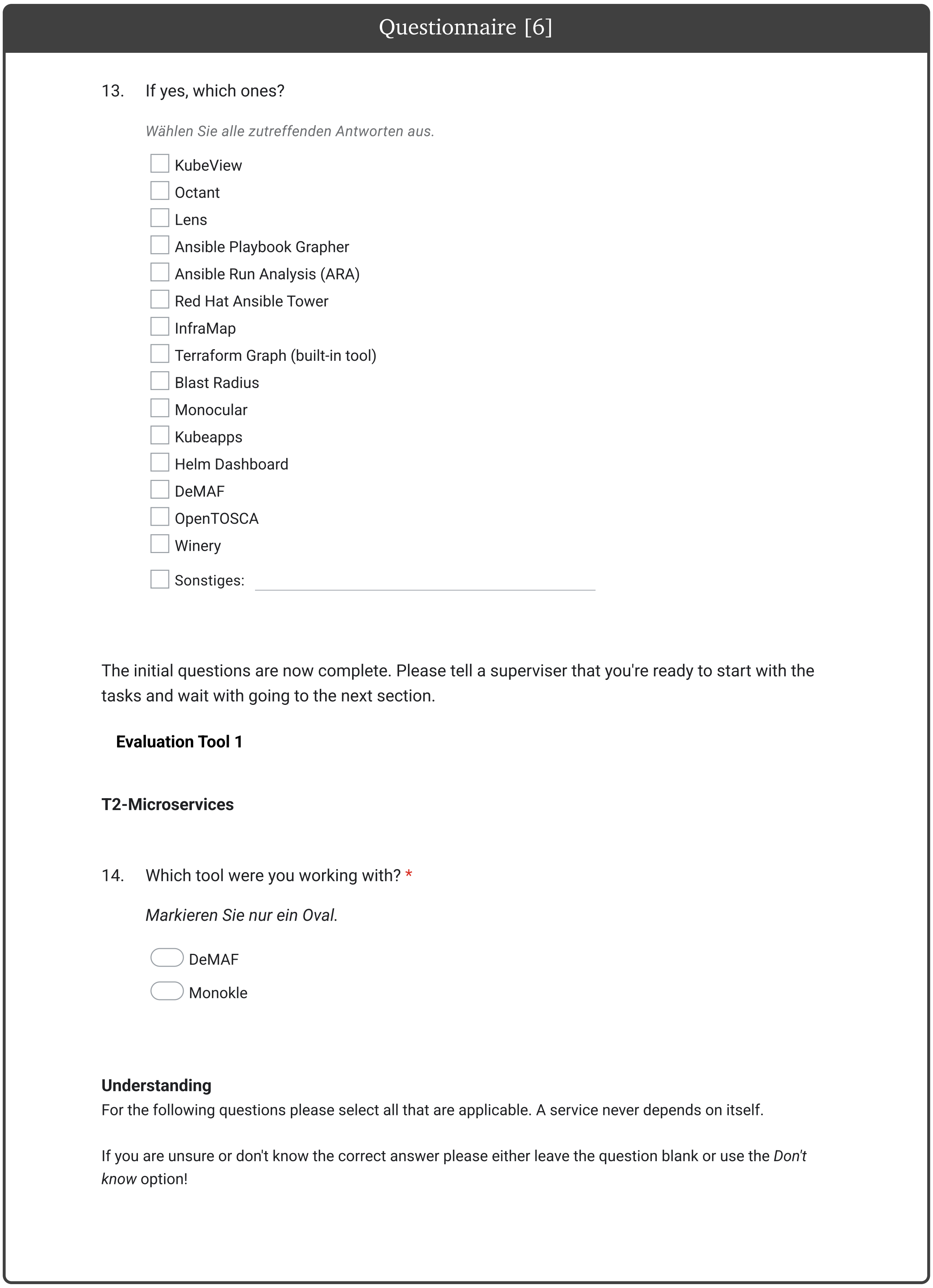
Visualization Tools

12. Have you used any visualization tools for deployment models before? *

Markieren Sie nur ein Oval.

◯ Yes

◯ No



Questionnaire [6]

13. If yes, which ones?

Wählen Sie alle zutreffenden Antworten aus.

- ☐ KubeView
- ☐ Octant
- ☐ Lens
- ☐ Ansible Playbook Grapher
- ☐ Ansible Run Analysis (ARA)
- ☐ Red Hat Ansible Tower
- ☐ InfraMap
- ☐ Terraform Graph (built-in tool)
- ☐ Blast Radius
- ☐ Monocular
- ☐ Kubeapps
- ☐ Helm Dashboard
- ☐ DeMAF
- ☐ OpenTOSCA
- ☐ Winery
- ☐ Sonstiges: ____________________

The initial questions are now complete. Please tell a superviser that you're ready to start with the tasks and wait with going to the next section.

Evaluation Tool 1

T2-Microservices

14. Which tool were you working with? *

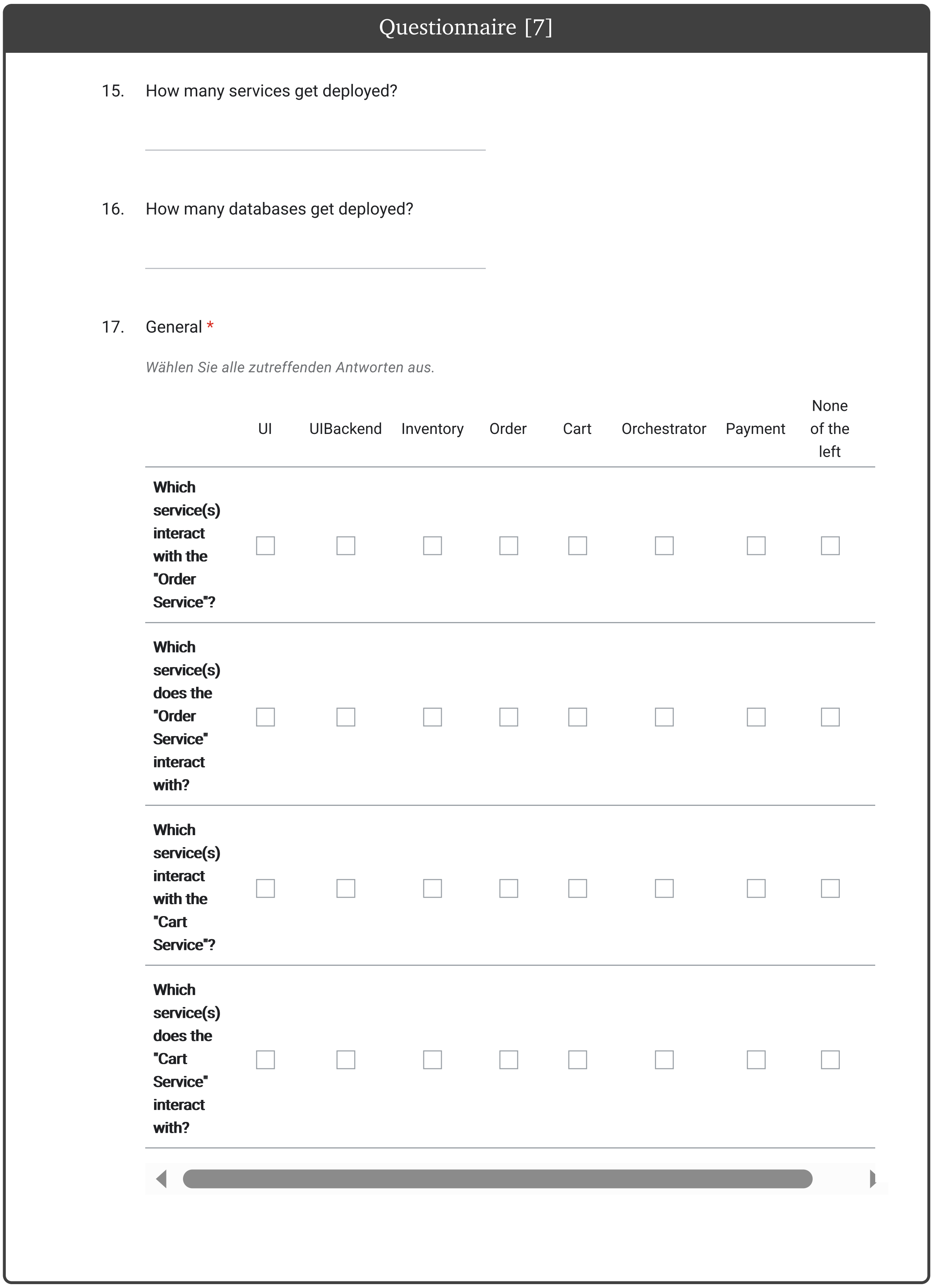
Markieren Sie nur ein Oval.

- ◯ DeMAF
- ◯ Monokle

Understanding

For the following questions please select all that are applicable. A service never depends on itself.

If you are unsure or don't know the correct answer please either leave the question blank or use the *Don't know* option!



Questionnaire [7]

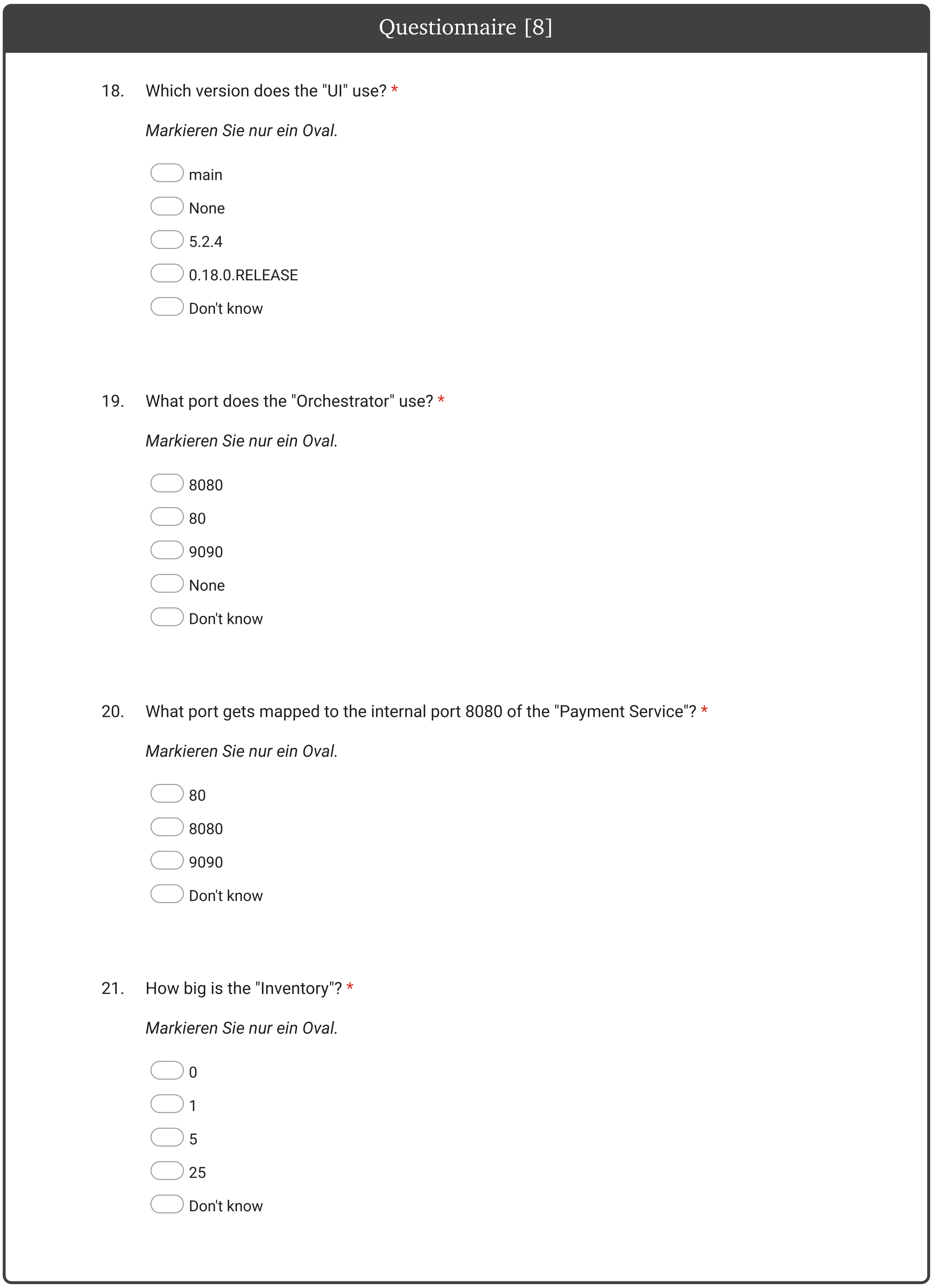
15. How many services get deployed?

16. How many databases get deployed?

17. General *

Wählen Sie alle zutreffenden Antworten aus.

	UI	UIBackend	Inventory	Order	Cart	Orchestrator	Payment	None of the left
Which service(s) interact with the "Order Service"?	☐	☐	☐	☐	☐	☐	☐	☐
Which service(s) does the "Order Service" interact with?	☐	☐	☐	☐	☐	☐	☐	☐
Which service(s) interact with the "Cart Service"?	☐	☐	☐	☐	☐	☐	☐	☐
Which service(s) does the "Cart Service" interact with?	☐	☐	☐	☐	☐	☐	☐	☐



Questionnaire [8]

18. Which version does the "UI" use? *

Markieren Sie nur ein Oval.

- main
- None
- 5.2.4
- 0.18.0.RELEASE
- Don't know

19. What port does the "Orchestrator" use? *

Markieren Sie nur ein Oval.

- 8080
- 80
- 9090
- None
- Don't know

20. What port gets mapped to the internal port 8080 of the "Payment Service"? *

Markieren Sie nur ein Oval.

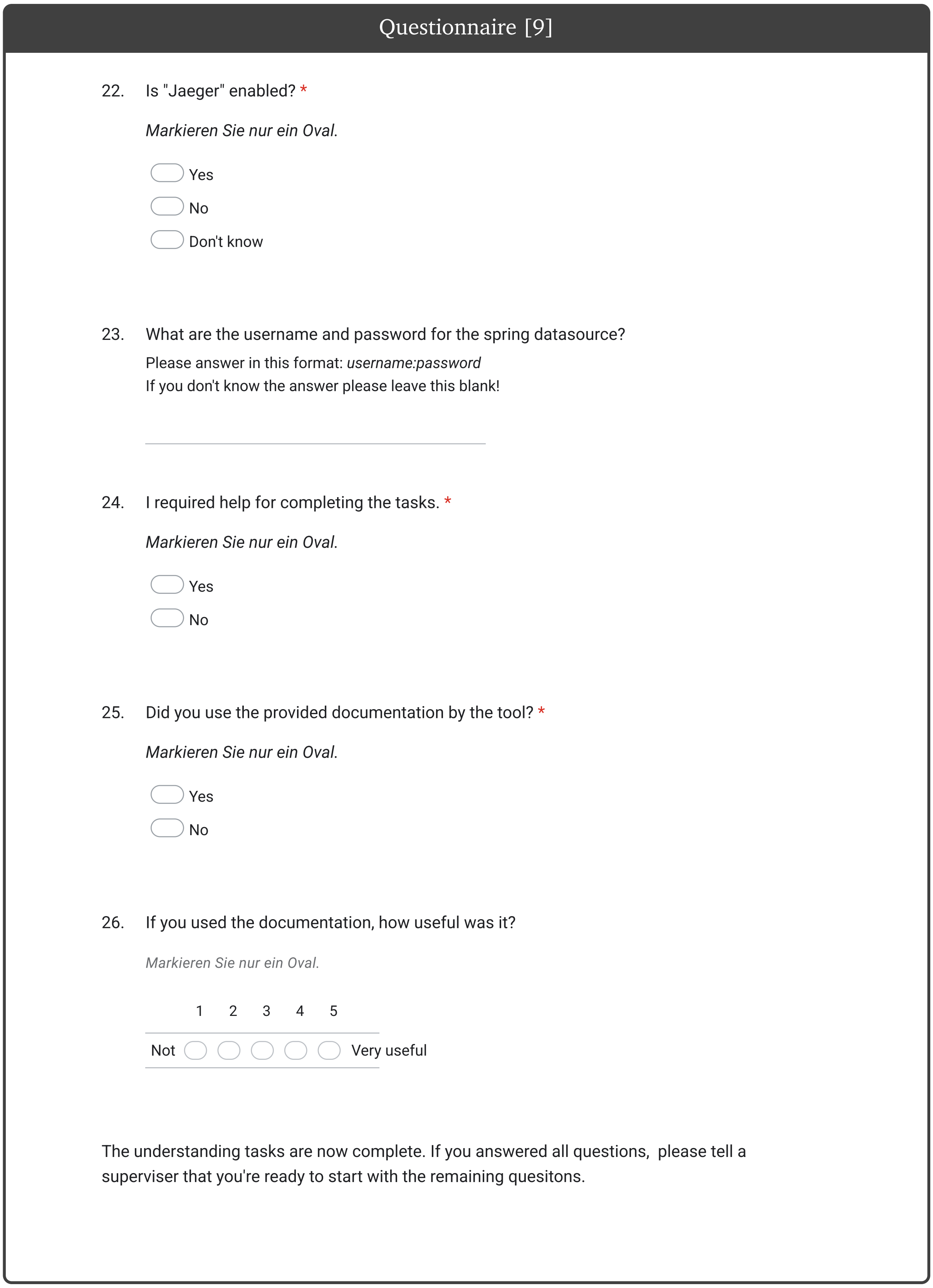
- 80
- 8080
- 9090
- Don't know

21. How big is the "Inventory"? *

Markieren Sie nur ein Oval.

- 0
- 1
- 5
- 25
- Don't know

Questionnaire [9]



22. Is "Jaeger" enabled? *

Markieren Sie nur ein Oval.

- Yes
- No
- Don't know

23. What are the username and password for the spring datasource?

Please answer in this format: *username:password*
If you don't know the answer please leave this blank!

24. I required help for completing the tasks. *

Markieren Sie nur ein Oval.

- Yes
- No

25. Did you use the provided documentation by the tool? *

Markieren Sie nur ein Oval.

- Yes
- No

26. If you used the documentation, how useful was it?

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Not						Very useful

The understanding tasks are now complete. If you answered all questions, please tell a superviser that you're ready to start with the remaining quesitons.

Questionnaire [10]

Task Load
In this we want to evaluate your demand on the task to find out how performant you were when using this tool.

27. How mentally demanding were the tasks? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

28. How hurried or rushed were the pace of the tasks? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

29. How successful were you in accomplishing what you were asked to do? *

Markieren Sie nur ein Oval.

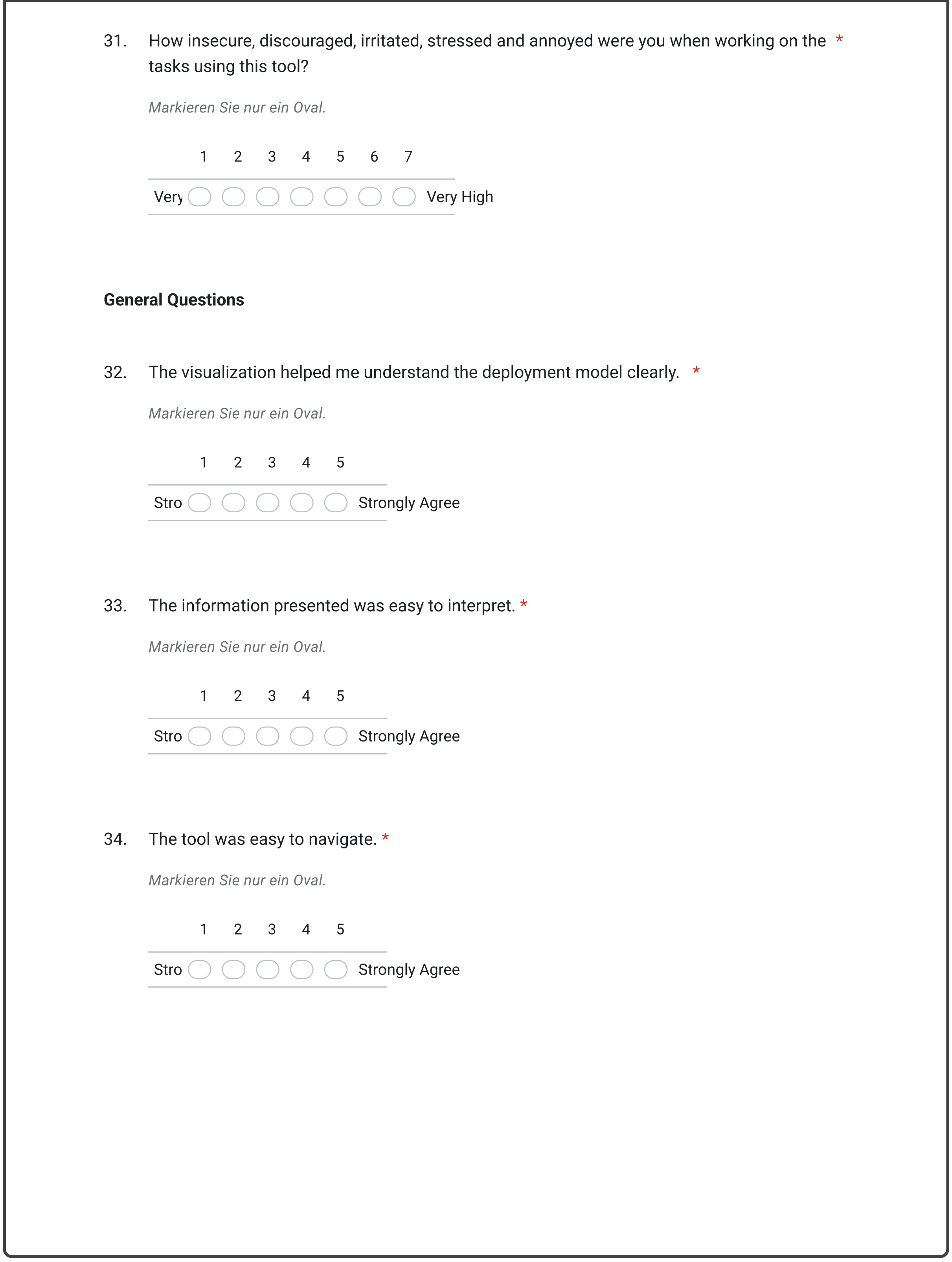
	1	2	3	4	5	6	7	
Perf	◯	◯	◯	◯	◯	◯	◯	Failure

30. How hard did you have to work to accomplish your level of performance? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

Questionnaire [11]



31. How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

General Questions

32. The visualization helped me understand the deployment model clearly. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

33. The information presented was easy to interpret. *

Markieren Sie nur ein Oval.

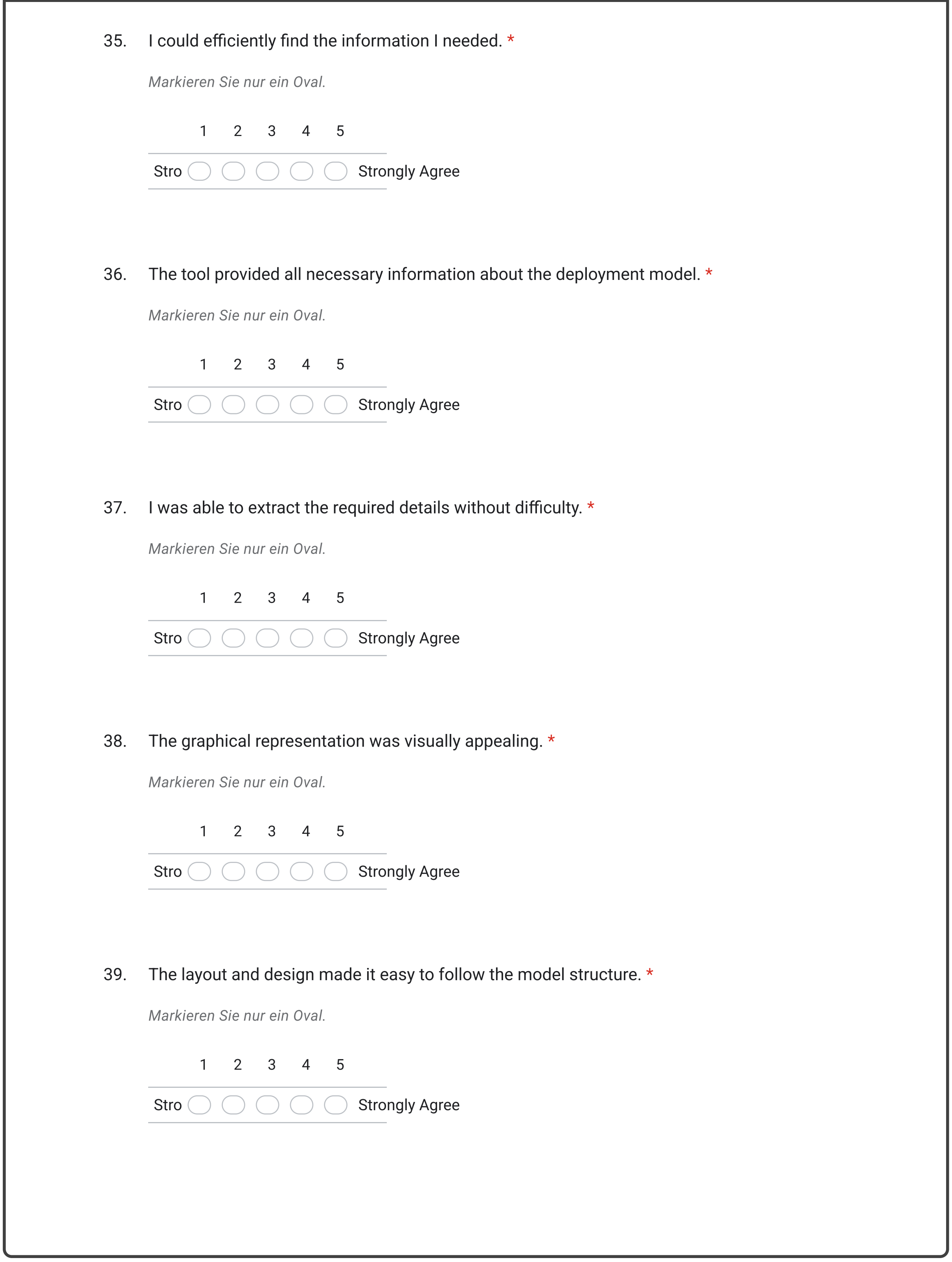
	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

34. The tool was easy to navigate. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

Questionnaire [12]



35. I could efficiently find the information I needed. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

36. The tool provided all necessary information about the deployment model. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

37. I was able to extract the required details without difficulty. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

38. The graphical representation was visually appealing. *

Markieren Sie nur ein Oval.

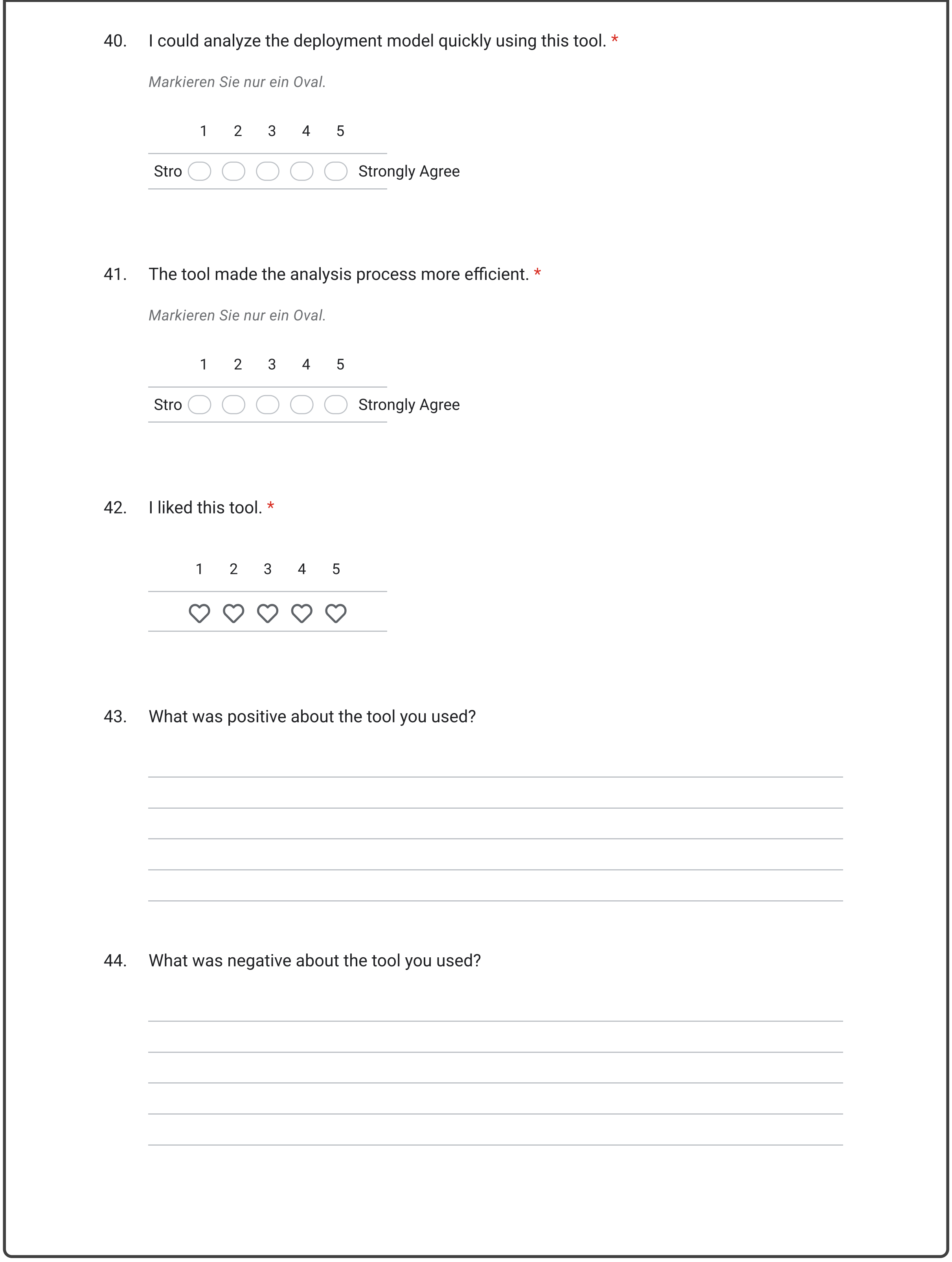
	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

39. The layout and design made it easy to follow the model structure. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

Questionnaire [13]



40. I could analyze the deployment model quickly using this tool. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

41. The tool made the analysis process more efficient. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

42. I liked this tool. *

1	2	3	4	5
♡	♡	♡	♡	♡

43. What was positive about the tool you used?

44. What was negative about the tool you used?

Questionnaire [14]

The first task is now complete. Please tell a superviser that you're ready to start with the second task and wait with going to the next section.

Evaluation Tool 2

OpenTelemetry Astronomy Shop Demo

45. Which tool were you working with? *

Markieren Sie nur ein Oval.

◯ DeMAF

◯ Monokle

Understanding
For the following questions please select all that are applicable. A service never depends on itself.

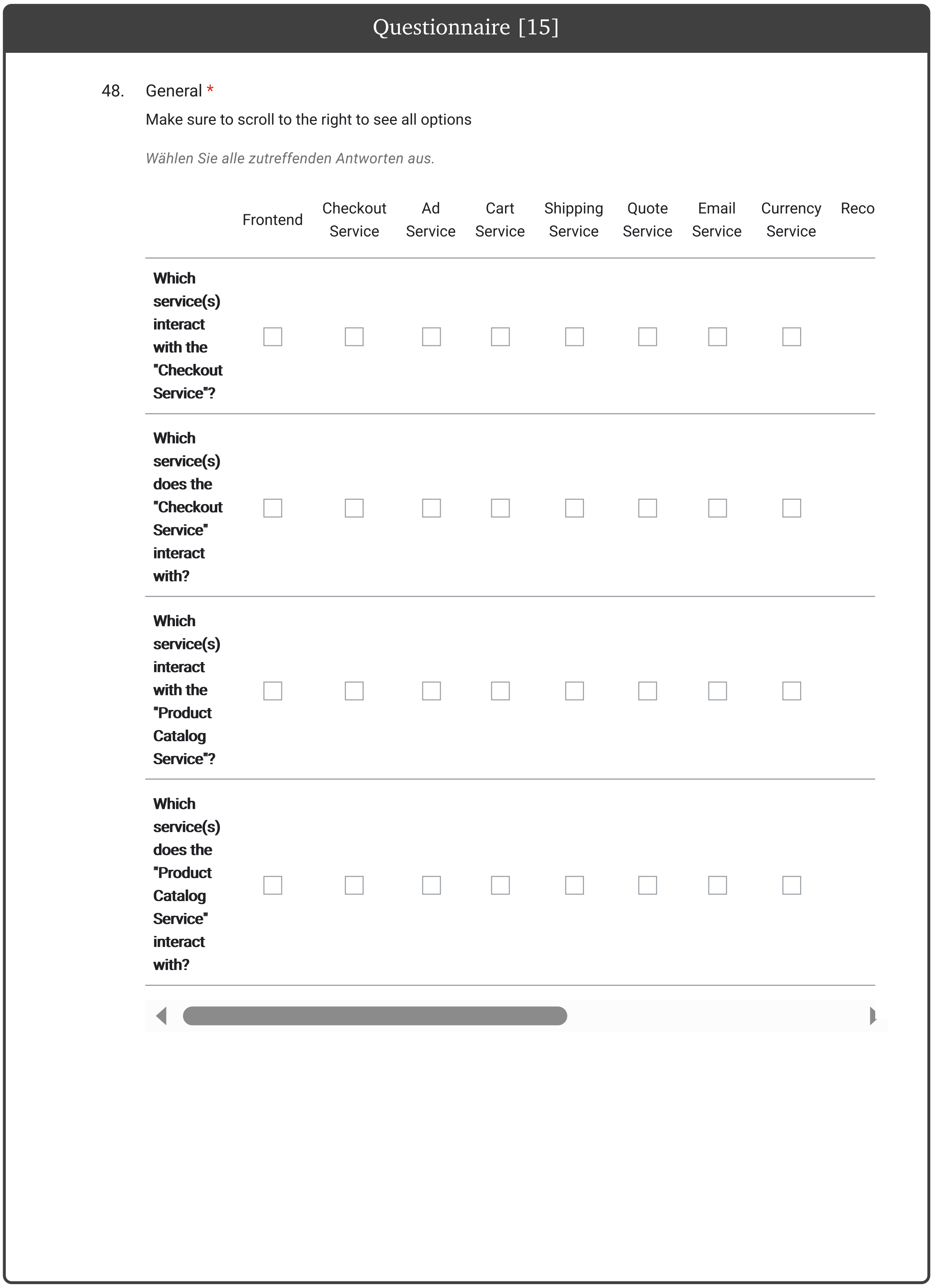
If you are unsure or don't know the correct answer please either leave the question blank or use the *Don't know* option!

46. How many services get deployed?

____

47. How many databases get deployed?

____



Questionnaire [15]

48. General *

Make sure to scroll to the right to see all options

Wählen Sie alle zutreffenden Antworten aus.

	Frontend	Checkout Service	Ad Service	Cart Service	Shipping Service	Quote Service	Email Service	Currency Service	Reco
Which service(s) interact with the "Checkout Service"?	☐	☐	☐	☐	☐	☐	☐	☐	
Which service(s) does the "Checkout Service" interact with?	☐	☐	☐	☐	☐	☐	☐	☐	
Which service(s) interact with the "Product Catalog Service"?	☐	☐	☐	☐	☐	☐	☐	☐	
Which service(s) does the "Product Catalog Service" interact with?	☐	☐	☐	☐	☐	☐	☐	☐	

Questionnaire [16]

49. Which version does the Mongo database use? *

Markieren Sie nur ein Oval.

- ◯ 8.0.0-rc9
- ◯ 8.0.3
- ◯ 7.0.16
- ◯ 7.2-alpine
- ◯ 1.11.1
- ◯ Don't know

50. What port does the Kafka queue use? *

Markieren Sie nur ein Oval.

- ◯ 9092
- ◯ 9093
- ◯ 9094
- ◯ None
- ◯ Don't know

51. What port gets mapped to the internal port 27017 of the MongoDB? *

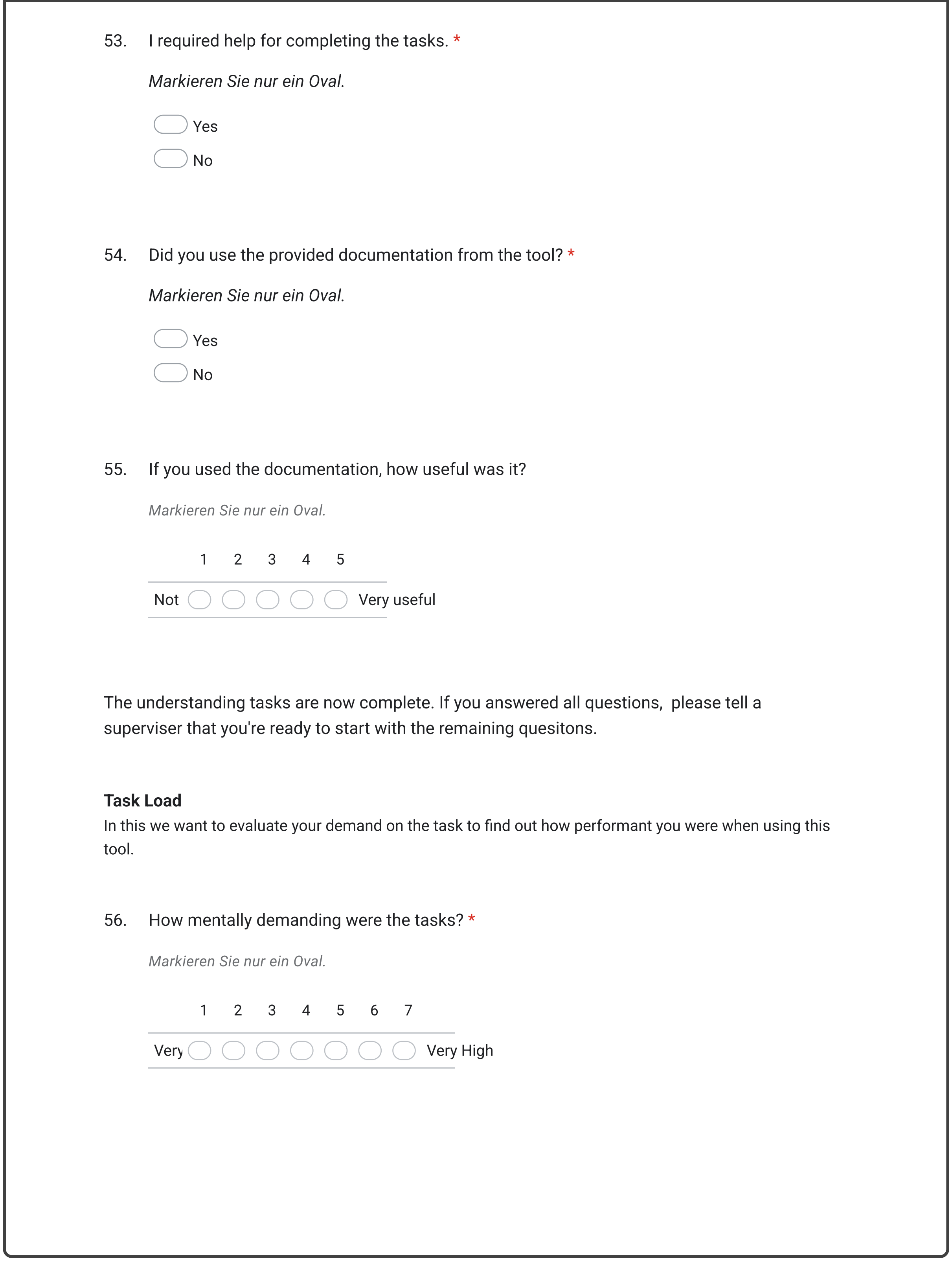
Markieren Sie nur ein Oval.

- ◯ 27017
- ◯ 27020
- ◯ 8080
- ◯ 999
- ◯ Don't know

52. What are the username and password for the MongoDB?

Please answer in this format: *username:password*.
If you don't know the answer please leave this blank!

Questionnaire [17]



53. I required help for completing the tasks. *

Markieren Sie nur ein Oval.

- Yes
- No

54. Did you use the provided documentation from the tool? *

Markieren Sie nur ein Oval.

- Yes
- No

55. If you used the documentation, how useful was it?

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very useful

The understanding tasks are now complete. If you answered all questions, please tell a superviser that you're ready to start with the remaining quesitons.

Task Load
In this we want to evaluate your demand on the task to find out how performant you were when using this tool.

56. How mentally demanding were the tasks? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

Questionnaire [18]

57. How hurried or rushed were the pace of the tasks? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

58. How successful were you in accomplishing what you were asked to do? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Perf	◯	◯	◯	◯	◯	◯	◯	Failure

59. How hard did you have to work to accomplish your level of performance? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

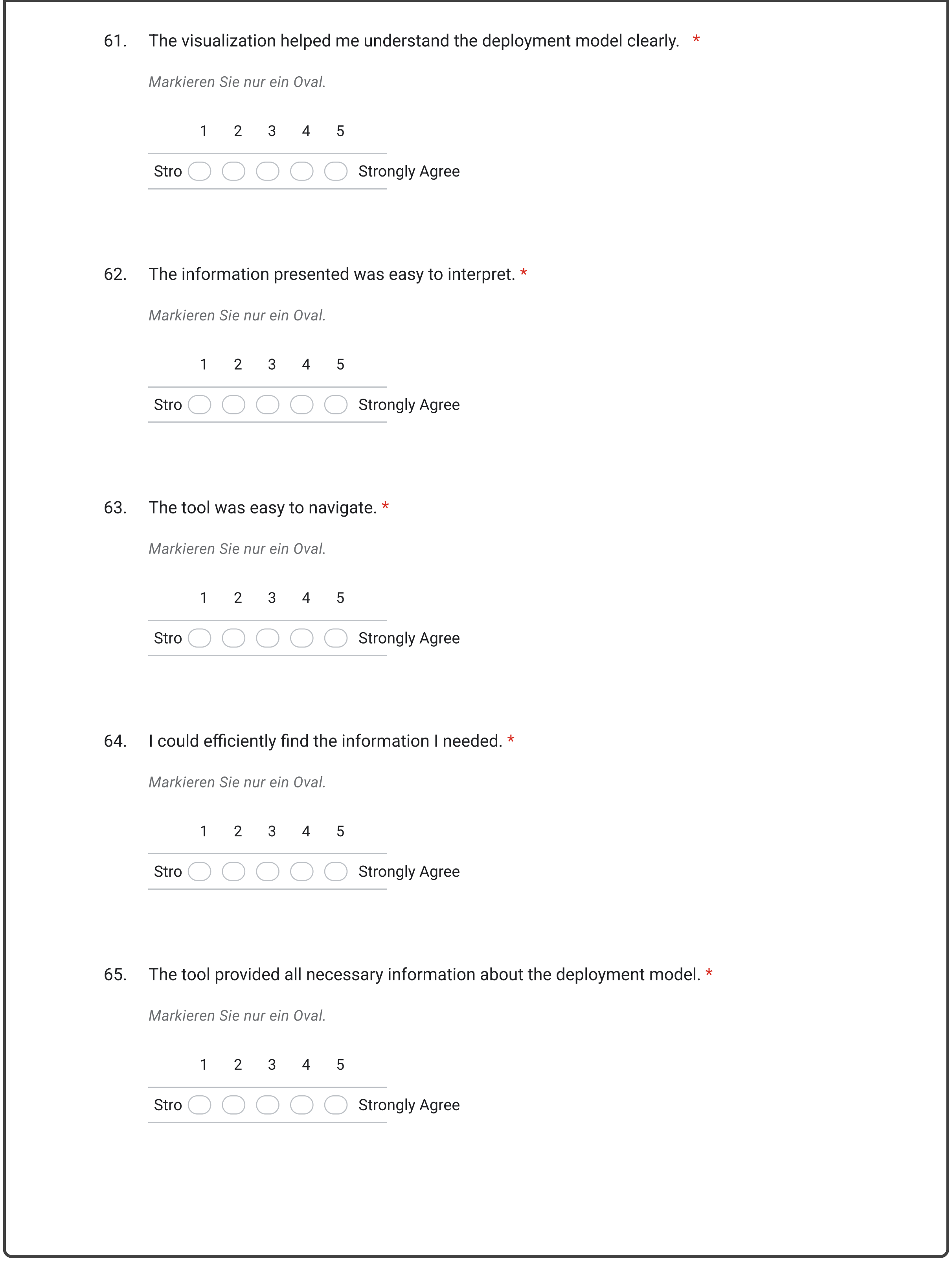
60. How insecure, discouraged, irritated, stressed and annoyed were you when working on the tasks using this tool? *

Markieren Sie nur ein Oval.

	1	2	3	4	5	6	7	
Very	◯	◯	◯	◯	◯	◯	◯	Very High

General Questions

Questionnaire [19]



61. The visualization helped me understand the deployment model clearly. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

62. The information presented was easy to interpret. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

63. The tool was easy to navigate. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

64. I could efficiently find the information I needed. *

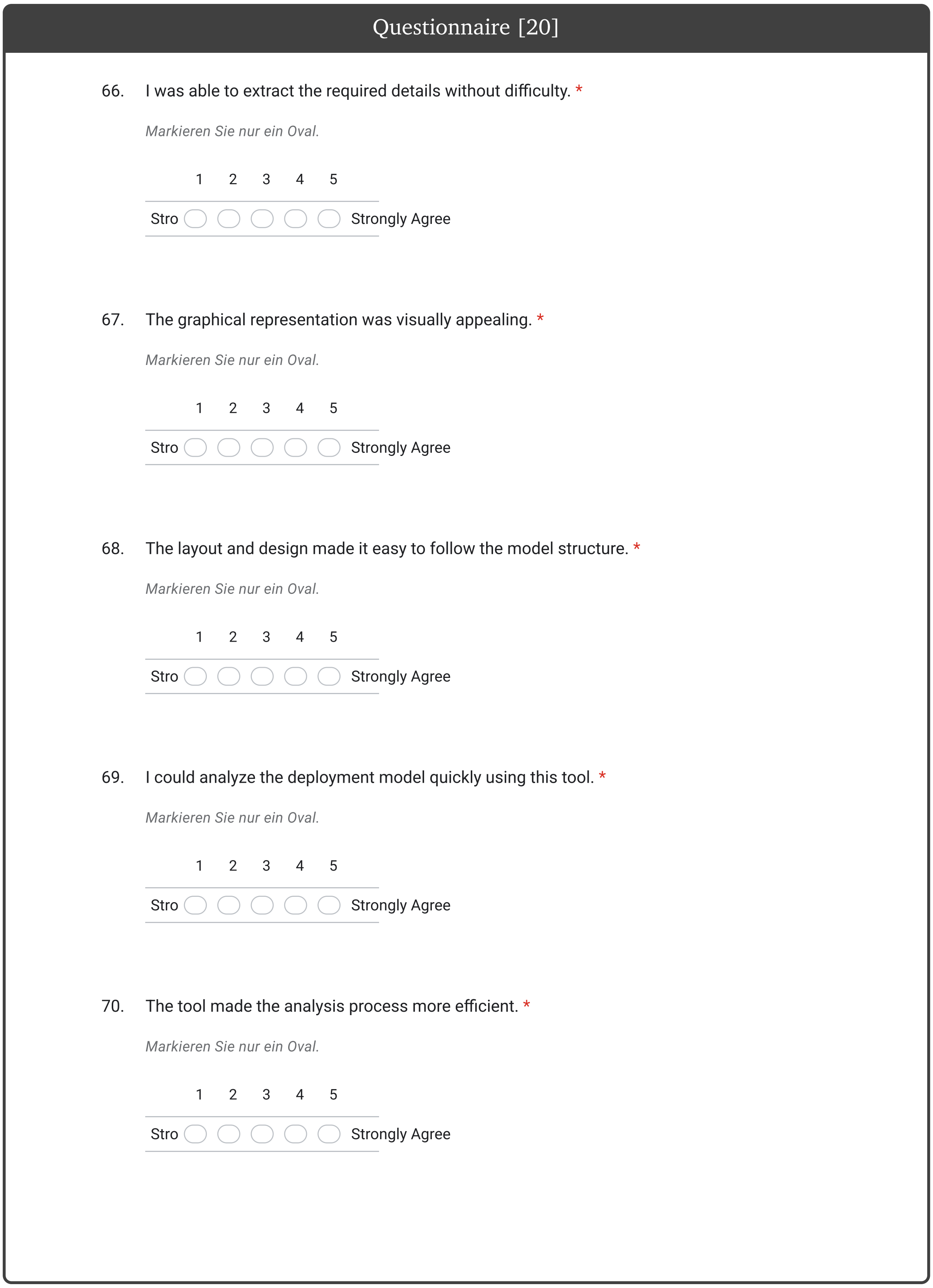
Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

65. The tool provided all necessary information about the deployment model. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree



Questionnaire [20]

66. I was able to extract the required details without difficulty. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

67. The graphical representation was visually appealing. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

68. The layout and design made it easy to follow the model structure. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

69. I could analyze the deployment model quickly using this tool. *

Markieren Sie nur ein Oval.

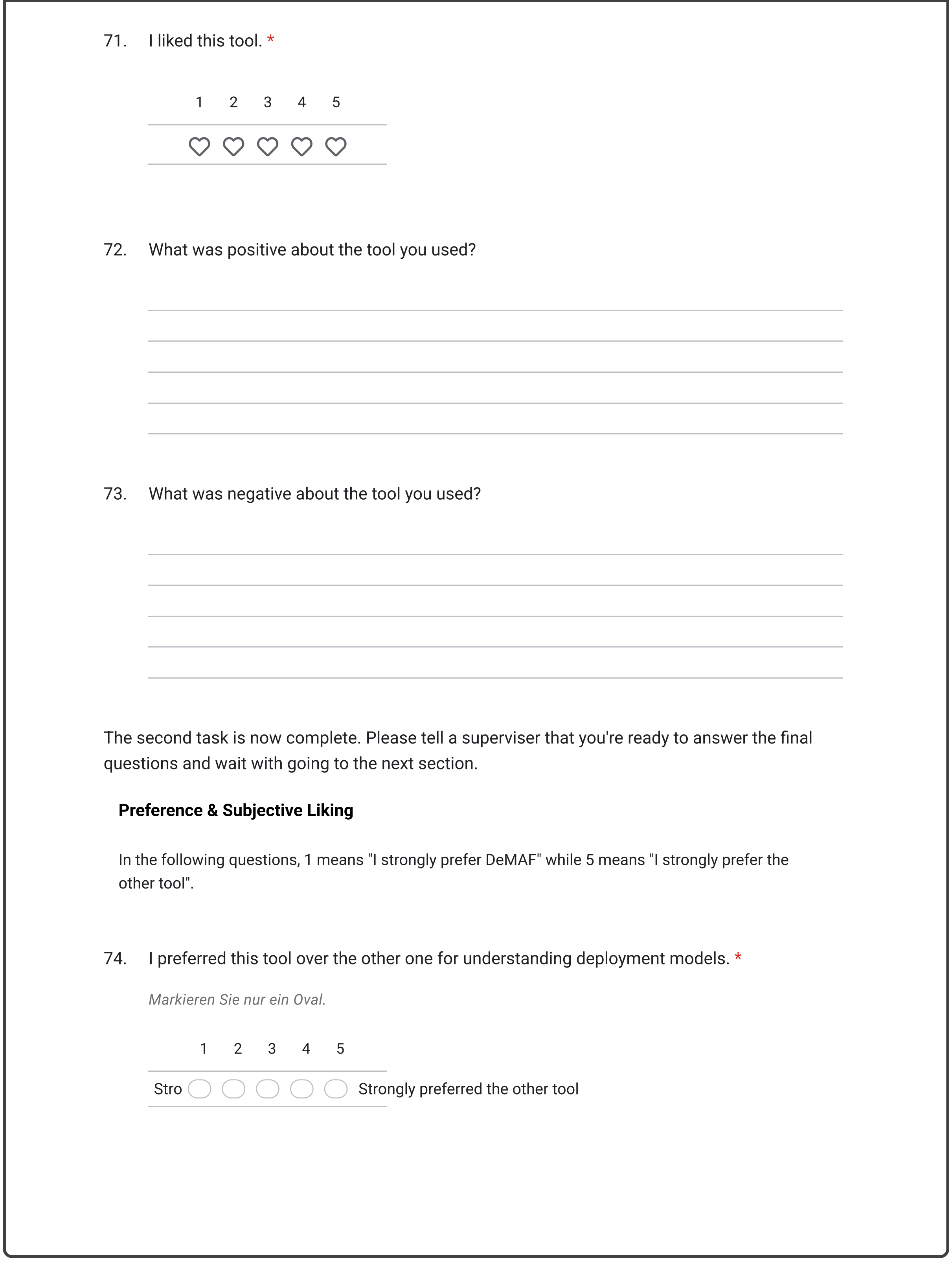
	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

70. The tool made the analysis process more efficient. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly Agree

Questionnaire [21]



71. I liked this tool. *

1	2	3	4	5
♡	♡	♡	♡	♡

72. What was positive about the tool you used?

73. What was negative about the tool you used?

The second task is now complete. Please tell a superviser that you're ready to answer the final questions and wait with going to the next section.

Preference & Subjective Liking

In the following questions, 1 means "I strongly prefer DeMAF" while 5 means "I strongly prefer the other tool".

74. I preferred this tool over the other one for understanding deployment models. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	○	○	○	○	○	Strongly preferred the other tool

Questionnaire [22]

75. This tool was easier to learn compared to the other tool. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly preferred the other tool

76. I felt confident in my understanding when using this tool. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly preferred the other tool

77. I would use this tool again for deployment model visualization. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly preferred the other tool

78. This tool provided better insights than the alternative tool I used. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly preferred the other tool

79. I prefer this tool. *

Markieren Sie nur ein Oval.

◯ DeMAF

◯ Other tool

◯ Indecisive

Questionnaire [23]

80. Do you have additional feedback *on the tools* you used in this study?

General Feedback on the Study/Survey

81. What did you like most about this study?

82. What did you find challenging or unclear?

83. Do you have any suggestions for improving the study or our tool (DeMAF) used?

Questionnaire [24]

84. The survey was well-structured and easy to follow. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly disagree

85. The study helped me understand all relevant aspects of deployment model visualization. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly disagree

86. The study helped me understand the given deployment models and how they are structured. *

Markieren Sie nur ein Oval.

	1	2	3	4	5	
Stro	◯	◯	◯	◯	◯	Strongly disagree

87. Would you participate in a similar study again? *

Markieren Sie nur ein Oval.

◯ Yes

◯ No

◯ Maybe

88. Any additional comments?