Institute of Software Engineering
Software Quality and Architecture

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master's Thesis

Evaluation of Explainability in Autoscaling Frameworks

Markus Zilch

Course of Study:	M.Sc. Informatik
Examiner:	Prof. Dr.-Ing. Steffen Becker
Supervisor:	Floriment Klinaku, M.Sc. Sandro Speth, M.Sc.
Commenced:	March 1, 2022
Completed:	September 12, 2022

Abstract

With the introduction of container- and microservice-based software architecture, operators face increasing workloads for monitoring and administrating these systems. Operators now need more and more software support to keep up with the complexity of these software architectures. This increased reliance on support software results in explainability becoming increasingly important. This effect is amplified as machine-learning-based autoscaling for container-based systems is steadily growing. Explainability of these new machine-learning approaches is vital for expert users to verify, correct and reason about these approaches.

Many scaling approaches based on machine-learning do not offer suitable methods for explainability. Therefore, expert users have difficulties identifying the reasons behind problems, such as suboptimal resource utilization, in these scaling approaches. Unfortunately, this also prevents them from effectively improving resource utilization. This thesis aims to improve the tools developers and operators have to build and evaluate autoscaling frameworks with explainability. Our first objective is to build a base autoscaling framework for Kubernetes that can be easily enhanced with machine-learning and explainability approaches. Our second objective is to elicit requirements that capture the explainability needs of operators in an autoscaling environment. Our third objective is to build an evaluation scheme that helps operators evaluate autoscaling frameworks regarding their explainability capabilities.

We re-implement the autoscaler "Custom Autoscaler" (CAUS) developed by Klinaku et al. in 2018. We also conduct an expert user survey with industry experts and researchers to gather the data needed for eliciting the requirements. Additionally, we use these requirements to build the evaluation scheme for autoscaling frameworks with explainability.

Ultimately, we show our research's benefits and limitations and how it can be expanded.

Kurzfassung

Mit der Einführung von Container- und Microservice-basierten Software-Architekturen steigt die Arbeitsbelastung von Betreibern bei der Überwachung und Verwaltung dieser Systeme. Um mit der Komplexität dieser Software-Architekturen Schritt halten zu können, benötigen die Betreiber nun immer mehr Software-Unterstützung. Diese zunehmende Abhängigkeit von Unterstützungssoftware führt dazu, dass Erklärbarkeit zunehmend an Bedeutung gewinnt. Dieser Effekt wird noch dadurch verstärkt, dass die auf maschinellem Lernen basierende automatische Skalierung für containerbasierte Systeme ein stetig wachsender Bereich ist. Die Erklärbarkeit dieser neuen Ansätze des maschinellen Lernens ist von entscheidender Bedeutung für erfahrene Benutzer, um diese Ansätze zu überprüfen, zu korrigieren und zu begründen.

Viele Skalierungsansätze, die auf maschinellem Lernen basieren, bieten von sich aus keine geeigneten Methoden zur Erklärbarkeit. Daher fällt es Experten schwer,die Gründe hinter Problemen, wie etwa einer suboptimale Ressourcennutzung, in diesen Skalierungsansätzen zu erkennen. Dies hindert sie leider auch daran, die Ressourcennutzung effektiv zu verbessern. Diese Arbeit zielt darauf ab, die Werkzeuge zu verbessern, die Entwicklern und Betreibern zur Verfügung stehen, um automatisierte Skalierungsframeworks zu erstellen und zu bewerten. Unser erstes Ziel ist es, ein Basis-Framework für die automatische Skalierung für Kubernetes zu entwickeln, das leicht mit Ansätzen des maschinellen Lernens und der Erklärbarkeit erweitert werden kann. Unser zweites Ziel ist es, Anforderungen zu ermitteln, die den Erklärungsbedarf von Betreibern in einer Umgebung mit automatischer Skalierung erfassen. Unser drittes Ziel ist es, ein Bewertungsschema zu entwickeln, das Betreibern hilft, automatisierte Skalierungsframeworks hinsichtlich ihrer Erklärungsfähigkeit zu bewerten.

Wir reimplementieren den Autoscaler “Custom Autoscaler” (CAUS), der von Klinaku et al. 2018 entwickelt wurde. Außerdem führen wir eine Nutzerumfrage mit Branchenexperten und Forschern durch, um die für die Erhebung der Anforderungen erforderlichen Daten zu sammeln. Darüber hinaus verwenden wir diese Anforderungen, um das Bewertungsschema für automatisch skalierende Frameworks mit Erklärbarkeit zu erstellen.

Schließlich zeigen wir die Vorteile und Grenzen unserer Forschung und wie sie erweitert werden kann.

Contents

List of Figures

List of Tables

1 Introduction

Software development over the last decades has seen changes from monolithic to more component-based structures [Aoy+98; Crn01; HC01]. This resulted in increasing utilization of microservices. It is no surprise that technology to better support these microservices emerged. One of the most used ones is the containerization of software. These containers only use minimal software stacks to provide a defined microservice or functionality. Additionally, containers allow to easily deploy multiple replicas of the same microservice to meet request demands. This allows companies to spend resources on specific components and functionalities, such as components with high load demands. Companies can save money on resources that would otherwise be unused that way. These developments brought forth a new field of research, the field of autoscaling. With an increasing amount of components, the workload required to monitor and administrate them has also increased. As a result, there have been developments that automate deploying components should the need arise, reducing the workload for system administrators. These approaches include autoscalers such as Azure Autoscale [Mic], AWS EC2 Auto Scaling [Ama] or OpenTOSCA [Stu].

In recent years, there also has been an increase in research on machine-learning methods [Mah20]. The developed approaches have started to find applications in other fields of computer science, such as the field of autoscaling applications. Especially for dominant infrastructure providers for container-based deployments, such as Kubernetes, the research in this field has experienced growth over the last half-decade. The previously used methods in this field utilized rule-based approaches and methods from the field of statistics to deploy an appropriate amount of containers. These methods are designed to deploy enough resources to guarantee service level objectives and other possible system metrics. Simultaneously they try to minimize the number of containers deployed to save resources and minimize the cost for stakeholders. This trade-off is the core principle of autoscaling methods [FMD+22]. Due to the complex nature of autoscaling and the increasing amount of container-based deployments, operators have an increased need for transparency in this context. As a result, they need an increased amount of support from software in order to be able to manage resources efficiently.

Recently developed approaches for autoscaling such as Toka et al. [TDFS20], Imdoukh et al. [IAA20] and Balla et al. [BSM20] try to improve scaling accuracy by using machine-learning. These new approaches are promising and show improved resource utilization depending on the machine learning approach used [RGLT19]. However, this increased performance for machine-learning-based autoscalers comes with a trade-off. More complex methods are less transparent and provide little insight into the decision-making process of how many containers to deploy. This results in the need to employ additional explainability methods. Research on explainability has experienced a lot of growth in general as can be seen in the multitude of approaches developed by Belle and Papantonis [BP21], Koh and Liang [KL17], Speth et al. [SSB22b] and Bhatt et al. [BXS+20] among others.

However, the field of autoscaling has not yet developed explainability approaches which consider the unique factors the autoscaling environment provides and cover the needs of system operators in the autoscaling field. Autoscaling approaches that utilize machine-learning generally do not provide any additional explainability features for system operators.

This forces these operators to make a difficult decision. They have to decide if the trade-off of increased performance is worth the risk and potential additional work caused by the decreased transparency and little insight machine-learning provides. While they know the information necessary to administrate the system they are entrusted with, they do not necessarily have expertise in explainability approaches. Operators are forced to deploy other explainability methods alongside these autoscalers, which are neither tailored to the scenario of autoscaling nor the machine learning methods utilized. In addition, it can be hard to deploy these explainability methods alongside existing machine-learning autoscalers since they do not provide sufficient interfaces to accommodate additional software.

Our first goal is to re-implement the autoscaler CAUS developed by Klinaku et al. [KFB18] in a programming language that supports machine-learning in order to build a base autoscaler that can function as a baseline for explainability approaches. We chose Python as our language since it provides a wide array of machine-learning libraries.

However, we noticed a few problems with this approach during the re-implementation. The number of possible combinations of machine-learning and explainability methods made it hard to decide on an optimal combination for the new framework. Additionally, no research is available that gives insight into what operators expect from explainability in autoscaling frameworks. We then came up with additional goals.

Our second goal is to design a study that gives these insights and to build requirements from them. This resulted in an expert user study designed to allow operators of different fields to provide their perspectives on what an autoscaling framework should provide.

While the insights from this expert user study are valuable, we wanted to provide further assistance for system operators. In order to achieve this, we want to design an evaluation scheme on the most important aspects an autoscaling framework has to provide in terms of explainability and configuration ability. This results in our third goal of building an evaluation scheme out of our previously elicited requirements. We build on the foundations provided by Rosenfeld and Richardson [RR19], Samek et al. [SWM17] and Doshi-Velez and Kim [DK17] in order to create this evaluation scheme. By combining our insights with the previous research, we try to create a helpful checklist for developers, researchers and operators for researching, implementing and utilizing explainable autoscaling frameworks.

In the shown autoscaling context and with respect to the problems that result from this context, this paper has three goals it tries to fulfil:

RG1 Re-implementing CAUS in Python and updating dependencies on the Kubernetes API

RG2 Building requirements for autoscaling frameworks with explainability from the insights of the expert user study

RG3 Building an evaluation scheme for autoscaling frameworks with explainability components

The following chapters deal with fulfilling these research goals. We show how we try to gather the necessary information and construct our results. Ultimately, we discuss our results and give our thoughts on what points future research based on our results can improve upon.

Thesis Structure

The thesis is structured into the following chapters:

Chapter 2 – Foundations and Related Work We provide terms necessary for understanding the topic of this paper and give an overview of related work from other researchers.

Chapter 3 – Methods We show our approach for re-implementing CAUS. Additionally, we explain the steps taken in preparation for building the expert survey and the evaluation scheme and which requirements engineering methods were applied during the thesis.

Chapter 4 – Results We show what our re-implementation of CAUS resulted in and the results our workshop produced. Additionally, we provide the requirements we came up with regarding these results. We also show how the evaluation scheme was constructed and the end result of this process.

Chapter 5 – Evaluation: We provide an evaluation of our findings and reflect on possible shortcomings and strong points of both.

Chapter 6 – Conclusion We conclude our paper with an outlook on possible further research. We provide a perspective for improvements to the evaluation scheme and possible implications of how our results could impact future developments.

2 Foundations and Related Work

In the following sections, we explain the concepts and technologies needed to understand this thesis. Additionally, we look at related work in the areas of autoscaling, explainability of machine-learning and evaluation of explainability. We also show what differentiates this thesis from existing work. We show our contributions to each area and how they expand on the work of other researchers. We start by explaining the process used for conducting the literature research and the tools we used to find and gather information on the abovementioned topics.

2.1 Literature Research Methodology

The primary method in the literature research used was searching Google Scholar [Goo] for fitting papers. The keywords used included “Autoscaling”, “Kubernetes”, “Machine Learning”, “Container Scaling”, “Explainability”, “Explainability Evaluation” and other combinations of these phrases. We then evaluated the found papers for relevance in context to this thesis.

The criteria for considering a paper differed depending on the research direction. For autoscalers, we included papers that are set into context with Kubernetes. Of particular interest were papers that directly compare their performance to the standard autoscaler provided by Kubernetes. We included papers that refined their approach beyond simple rule-based autoscaling.

For explainability methods, we included papers that either give a good overview over the combination possibilities of machine-learning algorithms and explainability approaches. Furthermore, we included explainability approaches we deemed promising for the autoscaling field. We chose explainability approaches which pair well with the machine-learning methods we expected to see in autoscaling approaches. Papers that did not fulfil at least some of the set criteria were removed and not further considered.

We chose papers that offer guidance in evaluating explainability approaches for the explainability evaluation. Additionally, we chose papers which provide vital points on what criteria an evaluation scheme has to fulfil. In a second step, the connections to other papers were looked into and evaluated for relevance. In addition to papers found manually, the search included the tool Litmaps.co [Lit]. We used it to find previously missed papers. The tool offers forward and backwards snowballing functionality based on a set of papers the user enters. As for the requirements engineering process, we used library books to gain insights into different possible approaches. Additionally, we searched for further information on the “Living Lab” approach and requirement elicitation on Google Scholar.

2.2 Foundations

The following concepts and term definitions will be needed to understand the thesis. They include design patterns of the autoscaling field and formal definitions of concepts of the fields of autoscaling, machine-learning and explainability.

2.2.1 MAPE Control Loop

MAPE is a pattern for designing control loops used in autoscaling frameworks [IAA20]. It is built from four components: monitoring, analyse, planning and execution. In the monitoring step, input data is gathered for the later steps. The analyse step filters through the gathered data, removes unnecessary entries, transforms formats and computes new metrics that depend on the previously gathered data if necessary. The planning step takes the gained information and decides how many instances of a given application should be deployed. Typical options are scaling down, maintaining the current status and scaling up. Systems using MAPE can achieve these decisions through different means. One of the most commonly used approaches is the rule-based approach. Rule-based approaches follow a structure similar to decision trees, where conditions are preset to trigger defined outputs when met. In the last step, the execution, the previously made decision is put into practice and resources are allocated accordingly.

2.2.2 Rule-based Autoscaling

Autoscalers use different approaches to decide how deployments should be modified. Their common goal is maintaining previously defined standards in a consistently changing environment. One of the most commonly utilized methods is the rule-based approach. These approaches operate on a predefined set of inputs and decision processes modelled on those. A commonly used rule-based autoscaler is the horizontal pod autoscaler (HPA) by Kubernetes [Kuba]. This autoscaler uses the Kubernetes API to obtain the state of a deployment and required performance data, such as load metrics, to decide how to modify the deployment state. The HPA then again uses the Kubernetes API to implement this new state and scale the number of replicas accordingly.

Another example of rule-based autoscaling is the elasticity controller for a containerized microservice: Custom Autoscaler (CAUS). CAUS is a static rule-based scaling framework to scale Kubernetes containers vertically. The framework was proposed by Klinaku et al. [KFB18] to make container scaling less prone to under-provisioning due to rapidly growing load spikes. The principle behind CAUS is to provide so-called buffer containers that are used when high workload spikes occur. This allows the framework to react appropriately to rapidly growing workloads and long-term changes while not violating service level objectives.

2.2.3 Machine-learning-based Autoscaling

While rule-based autoscaling approaches are already well established in the industry, other methods, such as machine-learning-based approaches, are gaining more and more traction. Machine-learning-based approaches make decisions based on a model or belief system. These models are obtained through observations of monitoring data and performance metrics of the deployed application. These

approaches have shown to save up to 10% or more of used resources [RGLT19]. They do so by either timing scale-up and scale-down decisions more precisely to predicted workloads or adapting to changes in the environments in ways rule-based approaches cannot. The machine-learning methods utilized vary from approaches from the field of control theory to approaches from deep learning, such as neural networks. The autoscaler utilizes different principles to fulfil its purpose depending on the type of machine-learning used. Therefore, the quality of the results and transparency of these approaches can vary by large degrees.

2.2.4 Explainability

Explainability is a loosely defined term in the field of machine-learning. In its most loose definition, explainability can be any information. This includes visual, textual or any other form of data representation. As long as the data helps a human understand the reasoning behind the behaviour of a machine-learning algorithm, the data is included in the definition of explainability. However, a common consensus is that this definition does not apply to all fields [IBM]. It needs to be much more narrow in fields where understanding is crucial to resolve an issue regarding the system. Explainability in machine-learning aims to help characterize model accuracy, fairness, transparency and outcomes in machine-learning-based decision-making.

2.3 Related Work

This section shows the related work this thesis expands upon. This thesis contributes to the fields of autoscaling methods, explainability methods and explainability evaluation.

2.3.1 Autoscaling Methods

Several autoscaling methods and frameworks have been published in the past few years. A particular interest has been in autoscalers for Kubernetes containers due to its growth to something of an industry standard for container-based service deployments in clusters. Our paper focuses on improving the development of autoscaling frameworks for Kubernetes. Therefore, the following papers all focus on autoscalers for Kubernetes or similar platforms. In the following paragraphs, we show what autoscalers have been developed and their shortcomings regarding explainability.

Zhao et al. [ZHH+19] show how empirical mode decomposition (EMD) combined with a differential autoregressive moving average model (ARIMA) can build a robust prediction model for pod replica autoscaling. The new predictive model optimised the replica provision compared to the standard Kubernetes model, resulting in a substantially reduced mean square error. Compared, however, to the proposed method of this paper, EMD-ARIMA does not provide methods to present explainability to users. Additionally, the prediction model may struggle with vastly different load profiles of real-life applications compared to the load profile used for benchmarking. Due to the static prediction model, users may have to adjust the prediction model to fit different applications manually. A machine-learning algorithm, however, should be able to adjust to different environments without user intervention.

Balla et al. [BSM20] provide a similar approach to the planned methods of the proposed paper. Their autoscaler Libra uses linear regression to predict the load required in the next scaling window. Their benchmark measurements show better performance than the default HPA autoscaler provided by Kubernetes. Additionally, Libra provides both horizontal as well as vertical scaling possibilities. While vertical scaling is interesting in its own right, the proposed paper will only focus on horizontal scaling.

Toka et al. [TDFS20] brought forth an approach which combines multiple machine learning approaches into one framework. The different approaches work in parallel to create possible solutions. The best solution among them is selected and executed for scaling the Kubernetes set-up. This approach significantly reduces the over-provisioning of resources and loss of HTTP requests according to their benchmark. Additionally, with their introduction of an excess parameter, their frameworks provide an easy way to influence the trade-off between lost requests and the over-provisioning of resources. Their approach, however, leaves some blind spots. It remains not explainable with the previously defined criteria and offers no additional guidance for a user trying to understand the decisions made. Their approach also does not provide insights into the method for scaling at a given time step. It leaves the user guessing which prediction method predicted which load and what alternatives have been considered.

The approach Imdoukh et al. [IAA20] introduced uses a long short-term memory (LSTM) neural network to reliably predict incoming loads. Their approach is as accurate in its prediction as other established methods, such as auto-regression (ARIMA). However, it provides significant speed-ups and reduced costs compared to approaches based on ARIMA prediction models. In contrast, their approach suffers significantly more under the aspect of explainability due to the inherent black box properties of neural networks.

Rattihalli et al. [RGLT19] proposed a vertical autoscaling system instead of the usual horizontal approaches. Their resource utilization-based autoscaling system (RUBAS) adjusts the provided resources for a given container instead of starting new containers with new resources. Their work expands upon providing non-disruptive service scaling based on container migration. Their approach improves CPU and memory utilization by 5-20% compared to the standard Kubernetes vertical pod autoscaler. Their approach, however, does not include any horizontal scaling and no additional explainability ability for users.

Zheng and Yen [ZY18] proposed an autoscaling method implemented deeper into the Kubernetes logic. Their approach optimises edge cases in which latency between container and client is one of the most critical metrics. Their method improves latency and resource utilization in Kubernetes pods for these so-called fog computation cases. Their work neither utilizes machine-learning nor offers any explainability advantages, never the less their approach might be worth analysing for benchmarking more complex edge cases, such as these fog computation cases or similar scenarios. Additionally, these might provide better insight into the performance of other autoscalers, deviation of load profiles and response times. Their work also has to be considered in application cases, where performance and latency are of much higher importance than cost-effectiveness and reducing over-provisioning.

Lorido-Botrán et al. [LML12] provide an overview of different autoscaling techniques used in elastic cloud environments. The approaches shown might fit into this paper more or less. They, however, suffer from different problems, as Lorido-Botrán et al. describe. The shown approaches were tested under different test environments and pretences. They also mainly deal with elastic

cloud providers such as Amazon's AWS and provide much more rigid VMs instead of containers in an elastic cluster, as we aim for in this thesis. This makes the shown approaches, while interesting, hard to put into a perspective for the applications of this thesis.

The shown approaches all have no additional functionality aiding explainability. Our thesis aims to provide insights into possible improvements of the shown approaches. We aim to provide requirements operators have regarding autoscalers and explainability. Additionally, we provide an evaluation scheme to help further the development of these autoscalers.

2.3.2 Explainability Methods

Belle and Papantonis [BP21] show in their paper, that different machine-learning approaches need different approaches of explainability. Their paper shows a practical approach to how developers can find suitable explainability methods for their machine-learning environment to tackle their explainability needs. They list different machine-learning approach classes and show which approaches need which explainability methods to gain further knowledge. They also show what each approach class offers naturally and what explainability approaches may add to an explanation objective. In contrast to their work, we are not trying to guide developers in terms of explainability in general. We are focusing strictly on autoscaling and try to gain further insights into what aids developers of this field, specifically in evaluating their machine-learning approaches and explainability methods together. Further, we try to gather requirements in the context of an explainable autoscaling framework. Therefore, the insights gained from Belle and Papantonis may not be necessarily applicable in the context of building an evaluation scheme.

Bhatt et al. [BXS+20] explore how machine-learning and explainability are used in industry settings. They focus on local explainability techniques. They are gathering best practices and other information through interviews with industry experts. The findings of those interviews are then built into an explainability framework for establishing clear goals when deploying local explainability techniques to guide organisations and companies in their decision regarding machine-learning and explainability. Their knowledge is valuable for building a framework combining machine-learning and explainability approaches.

Koh and Liang [KL17] developed a method to use influence functions robust statistics to trace a model's prediction through a learning algorithm and back to its training data. The approach finds high relevance data points from the training data for a given prediction. Their approach can be used in model debugging and increasing model explainability. They showed that their approach also makes training-set attacks on prediction models possible and changes the results of a prediction with visually-indistinguishable changed training data sets.

In the research of Kambhampati [Kam19] machine-learning approaches are modified to work better in environments with humans around. The paper focuses on agent-based machine-learning approaches, which incorporate a model of human beliefs to adapt better to the needs of humans. This approach includes explainability in the human-believe model and tries to improve through explicable planning on the part of the agent based system. The paper tries to balance explicable behaviour and explanation-based decisions when the first is too costly. The centre of the paper is built around the human-believe model of the agent. It deals with problems of multiple users, different environments and the abstraction of the belief system to stay efficient and explainable.

Samek and Müller [SM19] gathered approaches for explainability in deep learning applications. They describe the differences between different methods and explain the positive and negative aspects of each mentioned approach. They also show fallacies which are commonly done regarding explainability and machine-learning. Their book describes the differences between transparency and interpretability.

Montavon et al. [MLB+17] developed an extension of the mathematical principles of so-called Taylor compositions for deep neural networks. Their so-called deep Taylor composition approach offers transparency enhancements for deep neural networks, especially image processing. They build a relevance distribution propagation to identify relevant input data and build explanations, such as heat maps from the received returns of their approach. Their approach improves previous methods through higher degrees of clarity by deconstructing each neuron of a deep neural network with Taylor compositions which leads to a higher degree of insight into the model of a neural network.

Ribeiro et al. [RSG16] produced the explanation technique LIME, which offers a new way to explain and interpret different models of machine-learning methods. Their approach uses local interpretations of a model to give a user insight if the underlying model is appropriate for the task at hand. They find their approach has high accuracy in determining if a given model is "good" or "bad". Additionally, they proposed SP-LIME, a further developed method to make global predictors in models more transparent to a user. In their experiments, they managed to get persistently better prediction results than comparable approaches for model explanation.

Baehrens et al. [BSH+10] have proposed a framework for explaining individual classification decisions of black-box machine-learning techniques regardless of the technique used. Their approach uses explanation vectors, support vector machines and Gaussian processes to make the reasoning behind classification decisions of a machine-learning approach more understandable for humans. Their approach offers visual aids through vector representations of gradients to visualize classification decisions. One of the most significant advantages of their approach is being independent of the model representation of the machine-learning approach and therefore being independent of the machine-learning approach as a whole for providing explanations for users.

Shrikumar et al. [SGK17] proposed a new framework for dependency interpretation in neural networks. Their approach, called DeepLIFT, interprets the activation of neurons in a neural network against its "reference activation". The resulting differences are assigned contribution scores. These scores can show dependencies between neurons which were not visible before. Therefore, connections between input and output of the neural network can be made visible, and connections between input can be made more apparent. The proposed framework improves transparency for users and can improve the possible usage of neural networks in scenarios where transparency is needed.

Sundararajan et al. [STY17] propose an explanation technique for providing explainability to neural networks. Their approach introduces two axioms, "sensitivity" and "implementation invariance", which describe attribution characteristics explanation techniques for neural networks should cover. Their approach is an extension of explanation techniques via gradients called integrated extension. Where standard gradient explanation techniques violate at least one of their proposed axioms, their new approach fulfils both axioms fully. Their approach helps to find errors and differences in different attribution methods. It helps identify the source of such errors. Therefore it improves the explainability of the neural network.

The above explainability techniques have all been developed to fit one or multiple machine-learning techniques. However, their deployment usually involves additional work by the user to properly combine the used machine-learning method and the chosen explainability approach. Additionally, these methods do not utilise unique characteristics the environment they are deployed in provides, such as monitoring data in an autoscaling setting. Our work aims to incentivise researchers to incorporate these characteristics and develop new approaches to autoscaling that take advantage of their deployment environment.

2.3.3 Explainability Evaluation

This thesis aims to bridge the gap between machine-learning and explainability in the autoscaling environment. We try this by building an evaluation scheme tailored to the autoscaling frameworks. The following research papers are aimed at building standards for explainability techniques and how to evaluate them. Therefore, they lay important foundations for our thesis. In the following paragraphs, we summarize these research papers and show their relevance to our research.

In order to evaluate the explainability of each approach and compare them to existing approaches, standards have to be established. As Rosenfeld and Richardson [RR19] establish, there are currently no commonly agreed upon definitions of explainability used by the majority of researchers. They propose a formal definition of explainability and other important structures. Additionally, they try to establish a scoring system to help identify which aspects of a machine-learning system are sufficient to explain its decisions. They propose to rank those aspects in the following questions that need to be answered:

- Why should the system be explainable?
- Who is the target of the explanation?
- What interpretation can be generated?
- When should the information be presented?
- How can explanations be evaluated?

The proposed paper aims to answer those questions in the context of autoscaling technology for Kubernetes clusters and expand on the proposed evaluation scheme. The paper evaluates existing and newly developed methods for further qualities introduced by Rosenfeld and Richardson.

In his paper Samek et al. [SWM17] focuses explainability and evaluation of explanations. The main focus is the explanation of deep learning models, especially image processing. The paper gives insight into methods for visualizing, interpreting and explaining these deep learning models. It also tries to evaluate the quality of explanations and experiments with different environments where explanations in image processing can be helpful.

Doshi-Velez and Kim [DK17] proposed a set of qualities an explanation of a machine-learning approach should satisfy. In their paper, they try to define interpretability which is commonly used as a fallback for difficult-to-handle metrics or where numerical metrics are not handling the underlying quality fully or well enough. They define interpretability in different dimensions, including task- and method-related latent dimensions and additional factors that play a factor in interpretability.

Their goal is to come to a unifying definition of interpretability while not neglecting scenario-specific high-value metrics. They show fellow researchers what qualities to include in individual interpretability definitions to get more comparable results between different researchers and fields of study. We try to use the knowledge and insights gained by their work and apply them to the field of autoscaling frameworks for Kubernetes in particular. We want to expand on the factors given by their research and improve on area-specific qualities as much as possible to gain a useful evaluation scheme for our area and keep comparability to other fields as high as possible.

3 Methods

This chapter presents the methods we use to achieve the thesis objectives. In the following, we give an overview of our work processes during the thesis and how our research goals shifted based on the problems encountered. We show how we design the different parts of the workshop. Additionally, we explain our requirement elicitation process and show how we design the evaluation scheme.

3.1 Overview

Our initial goal was to update and enhance the outdated code base of the autoscaler CAUS developed by Klinaku et al. The original code base was rendered non-functional due to dependencies on no longer available Kubernetes API features. Our development process for updating and refactoring CAUS can be seen in Section 3.2.

Additionally, we wanted to enhance CAUS and transform it into a full-scale autoscaling framework that provides more than just a decision-making process by utilizing machine-learning for the decision-making process and automatically delivering explanations to the user. Due to the vague terminology of an autoscaling framework and understanding of explainability in this context, we first have to define this for our research. We argue an autoscaling framework is more than just the decision making process responsible for scaling a deployment. In our opinion it is comprised of multiple relevant parts. As shown in Figure 3.1, these parts include the infrastructure responsible for monitoring and making metrics available. They also include the controller which gathers the data, analyses it, makes a scaling decision and then executes the decision in the Kubernetes cluster. In case of utilizing machine-learning approaches, another part is added to the framework. This component is responsible for computing and providing explanations to the user in regards of the decision making process as shown in Figure 3.2. Explainability, in our definition, therefore includes all information the framework presents regarding these components.

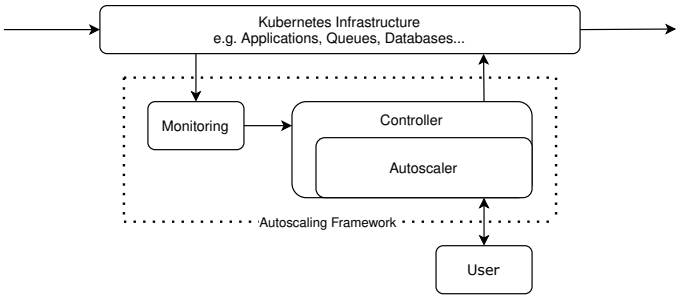

Figure 3.1: Standard Autoscaling Framework.

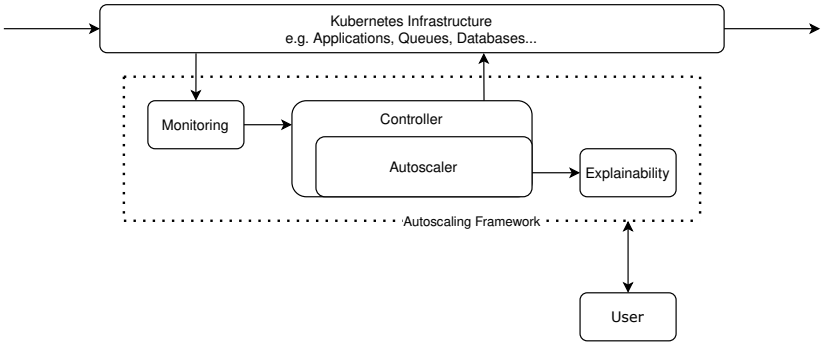

Figure 3.2: Machine-Learning Autoscaling Framework.

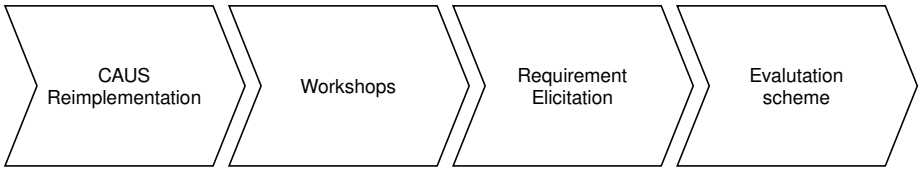

Figure 3.3: Development and Requirements Engineering Process.

After defining the base terms of our research, we abandoned our original goal of developing CAUS into an autoscaling framework due to several reasons. First, the amount of machine-learning, explainability approaches and especially their combinations is very high. This high amount of choices makes it difficult to select appropriate combinations for our use case. Second, the amount of information available on requirements users of autoscaling framework have is minimal. This is especially true for modern machine-learning autoscalers and frameworks with explainability functions. To our knowledge, no available research covers autoscaling frameworks in that regard. While some insights on explainability and evaluation of explainability are available, they only cover a broad spectrum of applications. The available research does not cover explainability in an autoscaling environment.

We then decided to focus our research on the second problem and contribute research on requirements operators have for autoscaling frameworks. We chose to conduct an expert user survey in order to gather data on the expectations of operators. The design of the expert user survey can be seen in Section 3.3. Additionally, we chose to contribute to the evaluation of autoscaling frameworks and, more specifically, of the explainability autoscaling frameworks should provide. We decided to design an evaluation scheme in order to give operators and researchers a checklist for machine-learning-based autoscaling frameworks with explainability. The design is shown in Section 3.5.

The points mentioned above result in a development and requirements engineering process for this thesis, as shown in Figure 3.3. The process is compromised of four underlying phases. In the first phase we update and refactor CAUS. The second phase is conducting the expert user survey in the form of a workshop in order to gather data. The third phase is eliciting requirements from the data gathered in the second phase. In the fourth phase we use the requirements we elicited to build the evaluation scheme.

3.2 Python-CAUS Design

The original goal of this thesis was to improve the autoscaling approach by Klinaku et al. [KFB18] with machine-learning to increase performance and make the decisions the autoscaler makes explainable. We wanted to modify the MAPE loop of CAUS to incorporate machine-learning as the decision-making process and add a loop step where CAUS generates explanations of the decision-making process for the user.

For this purpose, we converted the original code base from Go to Python. We did this to use the high number of machine-learning libraries available for Python. During the implementation, we updated CAUS's interface references for interacting with Kubernetes. The original version used API references which are no longer supported. In the new version, the Python implementation utilizes the official Kubernetes Python client [Kubb] for interacting with the Kubernetes API. This way, we ensure an easy way to upgrade the implementation for future development should the API change again.

Next, we tried to find machine-learning and explainability approaches which fit the CAUS approach. During the research, we discovered the many possibilities of the combination of machine-learning and explainability from which we can choose. Additionally, we realized our need for explainability is more complex when viewed in detail. We did not find papers that sufficiently guide what explainability approach is the most fitting for an autoscaling environment. Additionally, most machine-learning-based autoscalers we discovered during our literature review offer no explainability features or even provide guidance on what explainability methods could be helpful for the respective autoscaler. This makes the choice of what combination of explainability and machine-learning nearly insurmountable. In order to gain these insights and narrow down the number of possible combinations we deem useful for our use case, we set the goal of this thesis to gather fitting requirements.

In addition, we want to build an evaluation scheme from these requirements to improve the methods other developers and researchers have to tackle explainability for autoscaling methods. We want to build this evaluation scheme on the opinions of researchers and industry experts. This way, we want to ensure that we can provide valuable evaluations of autoscalers concerning explainability. Additionally, we want to provide some guidance on explainability for the autoscaling research field and contribute to the ongoing research in that area.

3.3 Expert User Survey

In order to build robust and helpful explainability tools for autoscaling, we have to know what information is useful for users. Additionally, we have to gain insights into what explanations give additional meaningful inputs for the users' understanding. To gain these insights, we must know what information users are looking for and how to present this information properly.

For this purpose, we conducted an expert user survey on system operators. Since operators have the highest degree of contact and work experience in these systems, their insights are the most reliable source of information in building an explainability framework. We conducted this expert user study in the form of a small workshop.

We started by building a scenario which mimics real scenarios operators encounter during their daily tasks. Simultaneously the scenario is stripped of any information that would make it too narrow or specific. The goal was to build a scenario with which every participant felt familiar and could participate in a discussion centred around the scenario. Additionally, we built a questionnaire the participants were tasked with answering at the end of the workshop. The questionnaire aims to capture the participants' thoughts, demographic data, and previous knowledge levels. The workshop and questionnaire can both be found archived here [Zil22]. In the following sections, we go into further detail about the expert user study, workshop and questionnaire.

3.3.1 Workshop Design

In the first stage of the expert user study, the participants work together in a workshop. Each workshop consists of one to two participants. In workshops with two participants, we tasked them to discuss with one another what expectations they have for the system. In workshops with one participant, this discussion was held between the participant and the researcher. This allows the participants to voice their opinions freely while having someone to discuss and improve their ideas. More participants per workshop may result in not every participant being able to voice their ideas or some individuals dominating the discussion.

In the beginning, we presented the participants with an autoscaling scenario involving an autoscaler. Next, we gave the participants 15-20 minutes to discuss the scenario and come up with the information they would want the autoscaler to present them with to gain a deeper understanding of autoscaling and the underlying decision process. Afterwards, we asked the participants to fill out a questionnaire to formulate their opinions and make them available for the requirements engineering step later. Additionally, we record the discussions for later review and extraction of requirements with prior permission from participants.

We designed the scenario and workshop discussion to stimulate the participants' creativity and get them to think about what they would want from an ideal autoscaling framework instead of what they currently have. We aim to get the participants to generate needs and drive their imagination. Through this design, we hope to increase creative thought and gain insights into the participants' needs regarding autoscaling frameworks.

3.3.2 Questionnaire Design

The questionnaire aims to provide information on what requirements the participants have regarding an explainable autoscaling framework. Additionally, the questionnaire aims to collect data regarding the participants' circumstances to make the results of the workshop quantifiable. The questions in the questionnaire offer different modes of answering. The three answering modes are (1) text-based, (2) selection and (3) selection and text-based. The text-based mode usually involves a text field where participants can write their opinions down freely. This mode is used on questions aiming at high creative input by the participants and questions with no clear answers that could be previously defined. It offers no guidance on what we expect as answers for these questions and therefore influences the participants the least. The selection mode is a set of pre-written answers the participants can choose from. We use this mode for questions where the participants have to sort themselves into categories we defined, such as company size. The participants cannot give

1	Questionnaire Sections
1.1	Demographic questions
1.2	Questions about previous experience
1.3	Questions about requirements of autoscaling framework
1.4	Questions regarding further participation

Table 3.1: Questionnaire Sections.

1.1	Demographic Questions	Type
1.1.1	Which of the job descriptions fits most to your actual job?	selection and text-based
1.1.2	How big is the company you are working for?	selection
1.1.3	In which country do you work?	text-based
1.1.4	How experienced with autoscaling frameworks would you count yourself from 1 (little to none) to 5 (I regularly deploy and maintain it in projects)?	selection
1.1.5	(Optional) In which company are you working? Please provide an answer only if you allowed to and like to provide the information.	text-based

Table 3.2: Demographic Questions.

additional information or their own options. Selection and text-based mode is a combination of the previous two modes. It includes a selection of pre-written answers and an additional option of answering with free text. This gives the users ideas on what kind of answer we expect and guides them to more detailed instead of generic open-ended answers. We use this mode on questions where we want to give the users some common examples before answering.

The questionnaire is structured as described in Table 3.1. In the first section, the participants are given questions about their demographics to set the scope in which the expert survey is valid. Additionally, the participants are tasked with rating their expertise regarding autoscaling. This section contains the questions shown in Table 3.2.

In the second section, we ask the participants to give insights into their previous knowledge in the autoscaling field. We ask the participants about previously used autoscalers and to give some insight into their work processes regarding autoscalers. The section contains the questions presented in Table 3.3.

1.2	Current Usage Questions	Type
1.2.1	Can you state which autoscalers you already have used?	selection
1.2.2	What type of autoscaler would you prefer?	selection
1.2.3	Which metrics do you use for autoscaling decisions?	selection
1.2.4	What is your current process to debug and understand autoscaling behaviour?	selection

Table 3.3: Questions about Current Autoscaler Usage.

1.3	Questions about Required Explanations	Type
1.3.1	Which explainability questions would you like the autoscaler to answer?	text-based
1.3.2	What kind of interface are you looking for (e.g. ChatBot, REST Interface, etc)?	text-based
1.3.3	What additional metrics to the ones stated before are you looking for to understand autoscaling behaviour?	text-based
1.3.4	On what occasions do you need information from the system (e.g. system failure)? Does this depend on the type of information (and if, in what combinations)?	text-based

Table 3.4: Questions about Required Explanations.

In the third section, they are given the opportunity to describe what information they require to understand the decision process of an autoscaler, what information and information representation they would want from an autoscaling framework and when the operator requires the information of the autoscaling framework. This section contains the questions shown in Table 3.4.

In the last section of the questionnaire, we ask the participants if they would like to participate in the follow-up reevaluation and to provide contact information if so. The questionnaire results are collected after the workshop and used in further requirements engineering methods as shown in Section 3.4.

3.4 Requirements Extraction

Gathering the user feedback of the workshop allows us to use requirements engineering to gain detailed requirements for information representation for each component of the framework. This should allow us to gain more insights into the participants' requirements for an autoscaling system.

We use the COSMOD-RE approach Klaus Pohl describes in his book [Poh10] to build the requirements. The approach described considers four layers of abstraction, as seen in Figure 3.4. We use a reduced version of this approach as we are not interested in the hardware/software layer nor the deployment layer of a specific implementation. Instead, we focus on the first two layers.

The first layer is the system layer. This is where we focus on users' requirements regarding the system's interactions with its environments and how it is embedded into it. In this layer, the autoscaling framework is regarded as a black box. All requirements focus on how the framework interacts with its environment and the user [Poh10].

The second layer is the functional decomposition layer. Here the system is decomposed into an abstract functional block. Here we collect users' requirements regarding features the system has to accommodate. Additionally, we lay out what interactions the functional blocks have between each other [Poh10].

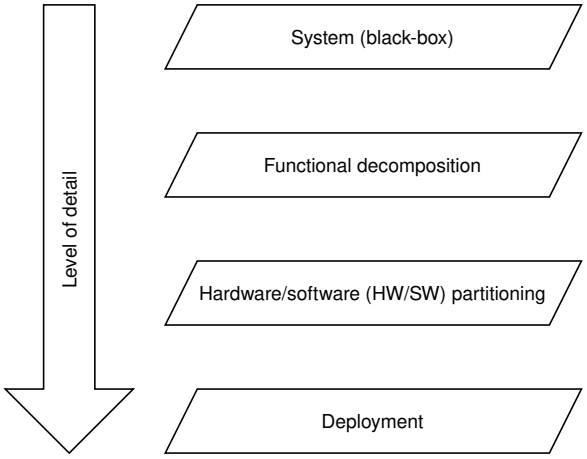

Figure 3.4: COSMOD-RE Abstraction Layers [Poh10].

3.5 Evaluation Scheme

The qualities we want to measure with the evaluation scheme are a direct result of the requirements engineering of the expert user workshop described in Section 3.4. We use the goal-question-metric approach laid out by Basili et al. [BCR94] to build a valid and fitting evaluation scheme. Additionally, we use it to ensure we focus each element of the evaluation scheme on a specific goal and correctly set the context, environment and goals of the evaluation scheme. This approach lets us identify aspects and appropriate metrics for each aspect we want to evaluate in an autoscaling framework. We use this approach to build a quality model around the extracted requirements for the framework. After this, we suggest fitting rating systems for each quality. This approach process promises a high-quality scheme to evaluate operators' requirements for an autoscaling framework.

After requirements elicitation the evaluation scheme is built with additional knowledge gained from Rosenfeld and Richardson [RR19], Samek et al. [SWM17] and Doshi-Velez and Kim [DK17]. Their works contribute qualities the evaluation scheme has to accommodate in addition to the qualities identified in the requirement elicitation step. They play a role in how the evaluation scheme is constructed, and their insights are used to further refine the qualities and metrics regarding explainability.

4 Results

This chapter deals with the requirements elicitation from the results of the workshop. The answers and questions are grouped in the same sections as they were presented to the participants in the questionnaire. The following section focus on showing what the participants answered in the questionnaire. Additional information from the discussions is added where the participants did not submit them in written form.

4.1 Implementation of Python-CAUS

We archived the resulting code-base of our thesis here [ZD22]. For our re-implementation of CAUS, we started by structuring the autoscaler into classes. Our goal was to separate functional components from each other and keep the autoscaler modular to some degree. As seen in Figure 4.1, we separated monitoring, replication computation and process controller from each other.

The monitoring component is responsible for gathering monitoring data, such as requests per second or response time. In our case, we used the Prometheus client library to connect to a Prometheus instance that monitors the traffic of a message queue and other metrics the message queue provides. We use the client to gather the rate of messages per second and make these metrics available for the other classes.

The replication computation class offers functions for computing the replica count appropriate for the current workload. Additionally, it offers the functionality to compute the number of buffer replicas CAUS deploys based on the publishing rate. We modelled these functionalities according to the work of Klinaku et al. [KFB18]. Like in the work of Klinaku et al., these functionalities use the elasticity of the scaled application for the computation. The scaled application's elasticity is a parameter provided by the controller class. We assume the elasticity of the application is already measured when deploying the CAUS. We provide a configuration class named elasticity for storing these measurements and other configuration values such as initial buffer size and the maximum amount of replicas the autoscaler should be able to deploy.

The process controller class takes the previous classes and handles communication between them. It reads the configuration class and hands the needed data to the functional components. Additionally, it provides the main functionality. At an interval, it computes the new replica amount based on the current workload and updates the deployment of the scaled application by using the Kubernetes API.

In our code-base, we provide a Dockerfile which can be used to package the autoscaler into a Docker image. This Docker image can then be used in the Kubernetes ".yaml" file to deploy the autoscaler in a Kubernetes cluster.

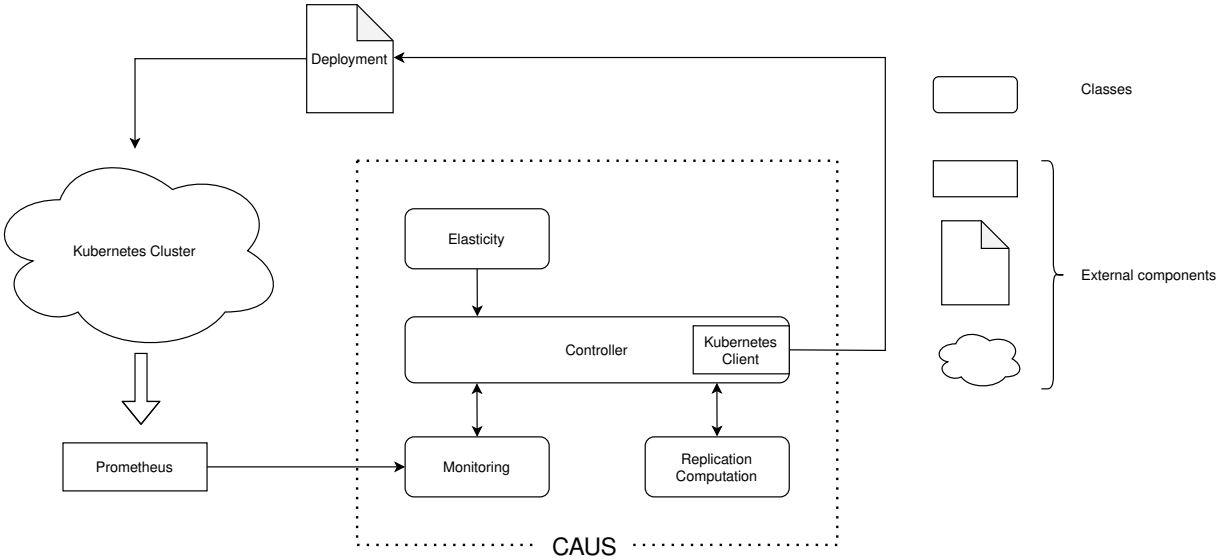

Figure 4.1: CAUS Structure.

Our code base provides a foundation on which future researchers and developers can expand upon. These adaptations can push further development in other frameworks, such as the T2-project developed by Speth et al. [SSB22a]. As their framework already includes the original CAUS, it can be used as a reference architecture to test and benchmark different machine-learning and explainability combinations. Our reimplementation provides the necessary foundation for especially this and similar cases.

4.2 Workshop and Questionnaire Results

4.2.1 Demographics

Ten people participated in the workshop. The participants' demographic data show that we managed to attract people from diverse backgrounds. We managed to include participants from different economic directions with ties to the field of computer science. Figure 4.2 shows that the participants come from different IT backgrounds, including university students, software developers, system administrators, and researchers.

The participants predominantly work at big companies with 250 or more employees, as Figure 4.3 shows. Some of the participants are working in medium and small-sized companies. One of the participants is a university student.

As can be seen in Figure 4.4, most participants are working in Germany. One participant is working in Switzerland.

The participants show a high spread in previous knowledge about autoscaling frameworks, as seen in Figure 4.5. The participants had to score themselves on a scale of 1 (low) to 5 (high). The average score the participants scored themselves was 2.9.

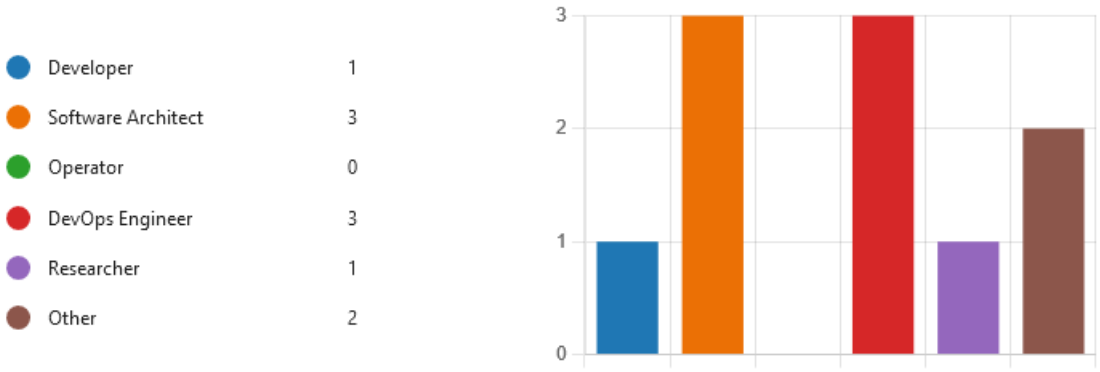

Figure 4.2: Job Title of Participants.

Figure 4.3: Company Size of Participants.

The participants work in companies with very different economic focuses, ranging from software development companies to companies predominantly offering administrative services to customers or other companies. Figure 4.6 shows these companies.

4.2.2 Current Usage of Autoscalers

This section focuses on the first section of the questionnaire. The goal of the questions in this section was to identify the level of current use the participants have of autoscalers.

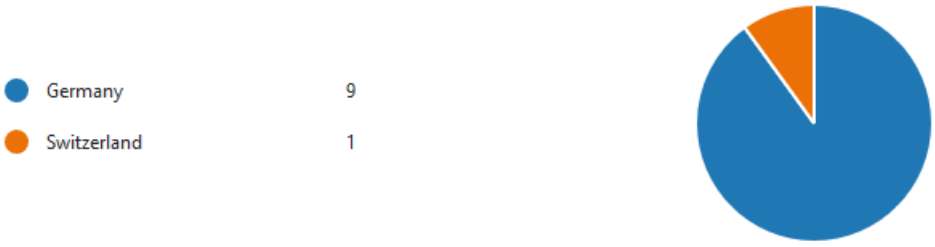

Figure 4.4: Country of Employment of Participants.

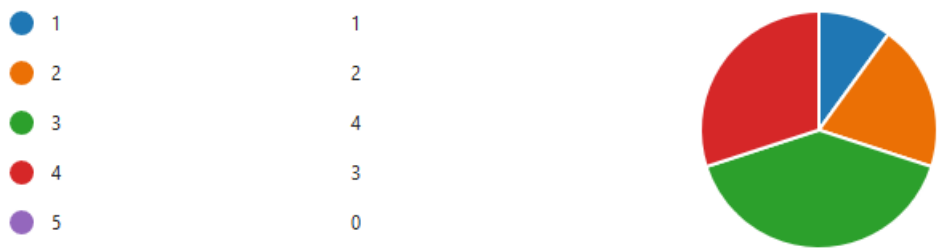

Figure 4.5: Previous Knowledge of the Participants.

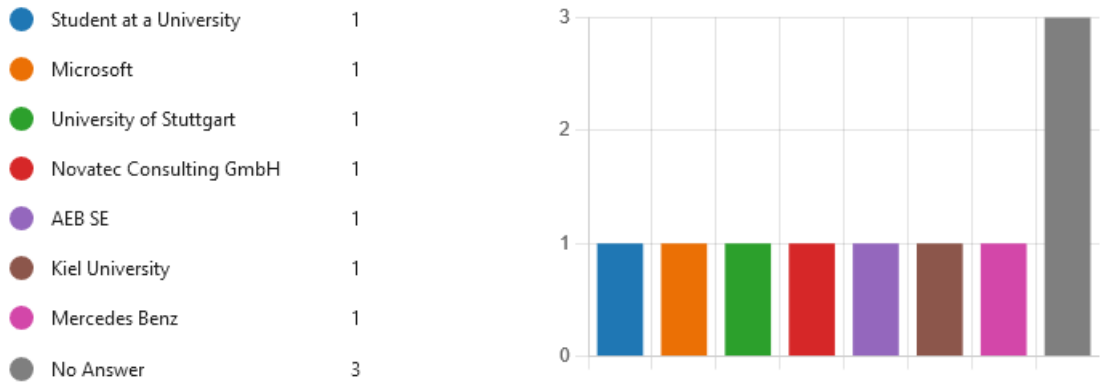

Figure 4.6: Companies of Participants.

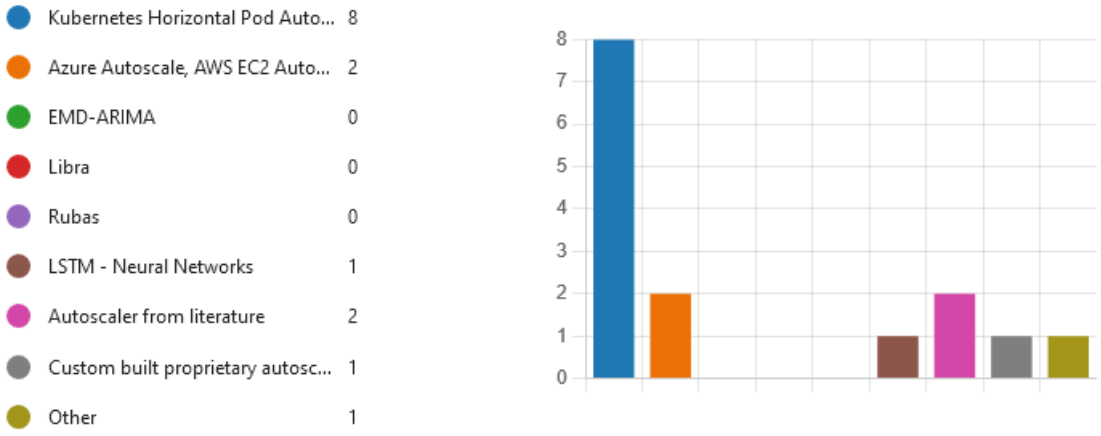

Figure 4.7: Previous Autoscaler Usage.

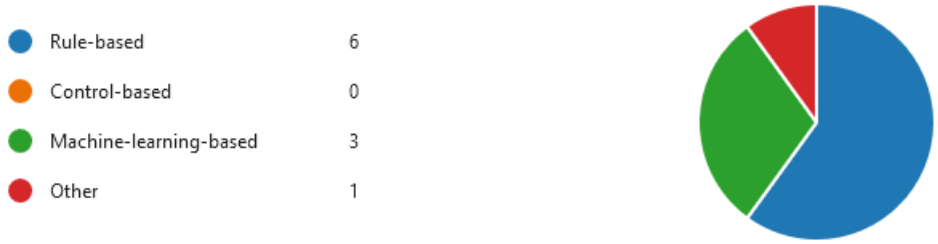

Figure 4.8: Preferred Type of Autoscaler.

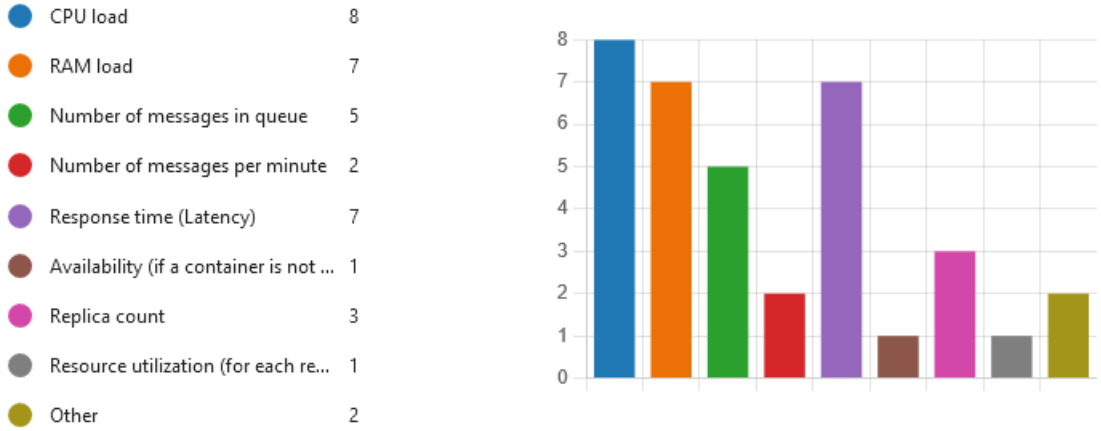

Figure 4.9: Used Metrics for Autoscaling.

The results of the first question regarding the current use of autoscalers paint a picture of diverse usage of existing autoscalers with a clear dominance of previously built and ready-to-use solutions such as the Kubernetes HPA [Kuba] or cloud autoscalers such as Azure Autoscale [Mic] or others. As can be seen in Figure 4.7, the participants predominantly use the Kubernetes HPA as their autoscaler. However, as can be seen, there is a wide spread of autoscalers in use, aside from the Kubernetes HPA. This spread does not only include more rule-based autoscalers. Some of the participants are using machine learning autoscalers as well.

The previous observation is reflected in the second question as well. As seen in Figure 4.8, a majority of the participants prefer to use a rule-based autoscaler. Around a third of the participants prefer machine-learning autoscalers or a mix of rule-based and machine-learning autoscalers. None of the participants prefers to use control-theory-based autoscalers.

In the third question, we gained insight into what metrics the participants use in their autoscaling setups. In Figure 4.9, we see the most predominantly used metrics are CPU load, RAM load, number of messages per queue and response time. More rarely used metrics include the number of messages per minute, container availability, replica count and resource utilization per replica. Some participants used additional metrics such as historical data points and metrics and container availability. According to the participant, container availability is their measurement for a certain amount of time a container is not reachable).

#	Answer
1	Have a look in the HPA and metrics server logs, try to find out why the HPA scaled, and then adapt the rules if necessary
2	reading logs, checking metrics (timeseries)
3	Looking at metrics/graphs in cloud dashboard.
4	None
5	I'll check the events of the HPA.
6	Kibana and KPIs
7	Look at resource usages at time of decision
8	no process so far
9	Kubernetes Logs
10	None really, looking at the autoscaler, comparing it with the metrics and adjusting

Table 4.1: Current Process to Debug and Understand Autoscaling Behaviour.

In question four, the participants gave examples of their process when finding problems in the behaviour of their autoscaler. Table 4.1 shows the participants' answers. Many participants answered that they did not have a set process to identify the reason behind the misbehaviour. Most of the participant gave some of the most common steps they take. These steps usually involve looking at metrics and log files as entry steps to the debugging process. As most participants use rule-based autoscalers, they try to identify the central rules which lead to the erroneous behaviour and modify these rules to fit the expected behaviour better.

4.2.3 Erroneous Behaviour Needs

This section aims at gaining insight into the participants' need for additional information. In the discussions, the participants were tasked with creating scenarios for which they wished to have more information. Additionally, we tasked them with imagining what information the autoscaling framework could provide to make their work easier and faster. The discussions were aimed at scenarios where the autoscaling framework showed erroneous behaviour. Additionally, we questioned the participants if the autoscaling framework should provide different information when working as intended.

In the erroneous behaviour needs section, the first question aimed at the needs of the participants. The question tried to get the participants to answer what metrics, graphics and other general information they wanted the autoscaling framework to provide. As can be seen in the given answers in Table 4.2, most participants want the autoscaler to explain why it scaled (or did not scale) at a specific time. Some participants want insights into possible alternative scenarios where the autoscaler would behave differently at a given time. Some participants want the weight maps of a neural network or the general data of the autoscaler.

The second question the participants faced was what kind of interface they were looking for when interacting with the autoscaling framework. The participants answered this question, as seen in Table 4.3, mainly with a combination of graphical user interfaces and REST API. The participants

#	Answer
1	Why did it scale? How (much) did it scale? When did it scale? How did this affect other services (if applicable)? What would have happened if the the metric / config for the autoscaler would have been x?
2	decission tree, matrix
3	Weights of hyperparameters for a certain decissions. Weights of goal metrics that played into decisions (did the model optimize for cost, latency, messages in queue, ...) Feedback loop so that I can take the data into my training dataset + automation so that the model notices anomalies and adds that data to the data set. Forecast of which decisions the model would make next depending on different future request patterns that I control (i.e. I adjust the requests and it shows me how it would react next).
4	Unknown due to lack of experience
5	Why did the autoscaler (not) scale? What was the forcast of the autoscaler? Which dimensions caused the autoscaler to scale/ not scale?
6	Why did the performance summer?
7	Why did it (not) scale
8	* Reason for when autoscalling did happen * Reason for why did the autoscaler decided to scale up/down to X instances
9	Why a scaling has happend or not happened.
10	why did you scale? why did you scale the way (amounts) you did?

Table 4.2: Explainability Questions the Participants would you like the Autoscaler to Answer.

wanted a graphical user interface for presenting metrics and the REST API for programmatic interaction. For the graphical interface, most participants wanted an interface like Grafana with interactive charts for log files and Prometheus-like interfaces for metrics.

The third question aims at identifying metrics which are important for explainability in autoscaling frameworks and have not been named in the previous section. As seen in Table 4.4, the participants came up with a broad width of metrics they wanted to be presented by the autoscaler. The answers involve data on the correlation between inputs and outputs, precision ratings regarding the error rate of the autoscaler, application-specific metrics and importance ratings of inputs. Participants also suggest different data types for measuring inputs, such as bytes per minute instead of the number of messages per minute.

In the last question, we asked the participants about their information needs. We tasked them with pointing out the information they wanted on which occasions. As seen in Table 4.5, the participants have similar views on when and what information should be presented. The answers regarding the occasions when to show information include “during system failure”, “during an unexpected scaling result”, “when examining user satisfaction”, “during any unusual events in the system”, “during SLO violations” and “continuously”. In none of the cases the participants expect additional information not previously included in the explainability information they already wanted present.

#	Answer
1	graphical web interface with diagrams, logs, optional: REST API for programmatic interaction
2	rest interface
3	Visual (for playing around, understanding) + API based (to integrate with other systems)
4	Mix of GUI / REST / Command-Line
5	Metrics and Event
6	Kibana / Grafana
7	REST Interface, visual interface (graphs etc.)
8	Interactive charts (Grafana etc.) and metrics (e.g. Prometheus)
9	GUI ala Kubernetes Dashboard. Notifications optional
10	dashboard with the most recent decision and a graph showing past decisions also an on demand text/graph export

Table 4.3: Desired Interfaces.

#	Answer
1	how does the scaling correlate with the input values
2	Unknown
3	- Precision of the approach
4	Application metrics
5	* load on the system in messages/bytes * specific SLOs (maybe not just pure technical ones, but SLOs incorporating business information)
6	which input was weighted how much and is it really that important? how does the system come to a decision? why were decisions made if they were bad decisions?

Table 4.4: Desired Additional Metrics to the ones stated before in order to Understand Autoscaling Behaviour.

4.2.4 Opinions from the Discussions

In addition to the answers to the questionnaire above, we extracted additional statements of impact from the interviews. These statements were paraphrased from the discussions and counted by the number of participants who voiced the same or similar statements. We grouped these statements into two groups regarding the need for visualisation and information. Additionally, we grouped them into statements with high and low confidence. Low confidence statements have only been voice by one or two participants. If three or more participants voice the same statement, it is regarded as a high confidence statement.

#	Answer
1	definitely on failure, but also when unexpected scaling result, but also on completely normal behavior (even in that case, it would be interesting to see why it scaled when) all the questions are interesting for all cases imho, but the ”what if”questions are especially interesting for the failure and unexpected behavior
2	massive scaling issues, system failure
3	System failure are the extreme cases in which I need information. Other cases like user satisfaction are also relevant cases in which I need additional information.
4	Continuuous stream of metrics, visualized + alerting due to wrong decisions
5	I only need the information in case something goes wrong with my system.
6	Anything unusal: way to many requests, ressources. Autoscaling was triggered but did not create the expected results.
7	At system failures. And before huge expected loads (e.g. a new product launch) to ensure scaling will work correctly
8	system failures, SLO violations
9	For Verification and Debugging. For Verification more historical data, how the expectation deviates from reality. For Failure more which options where available, was the right one even considered?
10	when the scaler decided to scale up/down way too much

Table 4.5: Occasions on which the Participants need Information from the System.

Information and visualisation Requirements

In this section, we generate statements from the discussions that provide insights into the information and visualisation needs of the participants. Table 4.6 shows statements we extracted from the discussions that are of high confidence. Table 4.7 shows statements we extracted that are of low confidence. We tried to show the participants' thought processes, which resulted in our generated statements.

The high confidence statements consist of 4 different topics. The first one is regarding graphical visualisation. All participants except one said they prefer graphical visualisation to other types. One participant preferred data being represented in the form of matrices. He especially mentioned data regarding used machine-learning models as being of interest. The resulting statement of this is statement 1.

In the discussion, several participants voiced their desire for interactions between charts, such as including the event timestamps in the metrics graph to quickly identify areas of high importance, as seen in statement 2. Some participants proposed interactability in the form of timelines. In these timelines, one can interact with a date to gain explainability and other visualisations computed from the data at that point in time.

Some participants explicitly stated that heatmaps could help them in case neural networks are the chosen machine learning approach. Some participants stated they could use heatmaps for pruning input metrics with a very low impact on scaling decisions. In their opinion, this could reduce computation time. Others stated that heatmaps could be help understand interactions between inputs and identify dependencies and other correlations. We summarised these opinions in statement 3.

A few participants have stated it could be interesting to be able to recompute decisions of some machine learning approaches with different learning parameters. Some participants mentioned that observing future changes in scaling behaviour could be interesting when the system simulates a different scaling decision for a given time. They argue they could gain insights from the consequences a different scaling decision encompasses for a given load profile. We summarised their opinions in statement 4.

There are seven further statements with low confidence we extracted from the discussion. A few participants noted that displaying the workloads the autoscaler predicts for future time steps ahead of time can help understand scaling decisions. Additionally, one noted that displaying this information can aid in configuring the autoscaler, should these two functionalities be compatible. We summarised this in statement 5.

Two participants want data representation in the form of matrices, as stated in statement 6. They argued that matrices contain all the information they need in order to understand how the machine-learning model works. They mentioned this in the context of neural networks. They believe a straightforward and easy-to-understand representation of the actual weights is the best for them.

Two participants stressed the importance of historical data. They argued that insights could be derived from looking at the historical data and understanding the reasons behind trends of changes in workloads and other metrics. We summarised this in statement 7.

Two participants wanted to be able to see how different deployed components that are managed by the framework influence each other. They argued that this would increase their understanding of how their application components interact and help them understand interactions between these components in terms of autoscaling behaviour. Statement 8 shows this and is a consequence of a feature the participant argued over, as stated in statement 16.

Two participants stressed the need for especially detailed resource utilization visualisation. They argued that this is especially important in their work area and greatly impacts their work duties. They mentioned that detailed reports help them react to inadequate resource utilization and take active measures for critical infrastructure. They also mentioned they would want the same in terms of autoscaler performance, in particular, objective function values and similar metrics should they apply to the used autoscaling approach. We summarised this in statement 9.

Statement 10 shows an addition to statement 8. One participant mentioned the desire for explanations divided by managed components. The participant explained that he desires a structure for multiple explainability modules where each is tailored to a specific autoscaler. We summarised this in statement 10.

One participant voiced the desire to log scaling decisions and other events. The participant mentioned that this is interesting for reviewing scaling decisions on a larger scale and gaining insights. Additionally, retaining this information is important for later discussions among peers, according to him. We summarised this information in statement 11.

# Statement	Statement	# Participants
1	We want graphical visualisation for metric data	9
2	We want a timeline visualisation with detailed information by timestamp	4
3	Heatmaps are useful explanations for input pruning and understanding input interactions	3
4	There could be benefit from being presented alternative scaling scenarios for a given point in time	3

Table 4.6: High Confidence Statements of Interest regarding visualisation and Information Needs.

# Statement	Statement	# Participants
5	A forecast of predicted workloads could help understand scaling decisions	2
6	We want data representation in the form of matrices	2
7	Historical data gives us additional insights	2
8	We want to know how different components we autoscale influence each other	2
9	We want to be able to see resource utilization and review the performance of the autoscaler	2
10	We want explanations presented per autoscaler	1
11	We want to log scaling decisions and events to review them at a later time	1

Table 4.7: Low Confidence Statements of Interest regarding visualisation and Information Needs.

4.2.5 Configuration and Structure Requirements

Similar to Section 4.2.4, we divided requirements regarding the structure and configuration of an autoscaling framework into two tables. Table 4.8 shows statements with high confidence. Table 4.9 shows statements with low confidence. In the following, we show the context in which we collected the statements.

Around half of the participants said they wanted some form of anomaly detection. Some participant also wanted the system to give them graphical data on what load it predicted to understand if the prediction was wrong or the process computing the replica count from the forecast was the problem.

In the discussion, the participants clarified that the used metrics heavily depend on the application the autoscaler is responsible for. For some applications, overall CPU and RAM load are critical. For others, these metrics have near to no impact. Instead, they emphasised the importance of response latency. We summarised this in statement 13.

# Statement	Statement	# Participants
12	We would appreciate anomaly detection for autoscaling behaviour	4
13	We want to be able to manage custom metrics as inputs for the autoscaler	3
14	We can see training a rule-based autoscaler with machine-learning approaches as a worthy approach to pursue	3

Table 4.8: High Confidence Statements of Interest regarding the Structure of and the Ability to Configure the Autoscaler.

In the discussion, some participants voiced the idea of compromising between rule-based and machine-learning approaches, as seen in statement 14. They voiced the idea of using machine-learning to explore the realm of rules the autoscaler can use, and for the actual decision process, only use these rules. In their opinion, this could optimise the autoscaler sufficiently and without compromising rule-based autoscaling's easy understandability and modifiability.

From the discussions, some participants believe that setting up complex autoscalers takes more time. This results in a cost-benefit trade-off between saved resources in the long run and the amount of work required to set up and maintain the more complex autoscaling system. The participants voiced that a system with "good enough" results and a low amount of maintenance needed is preferable to a system with perfect results but high amounts of setup cost and maintenance, resulting in statement 15.

Some of the participants argued over a more extensive framework. Their argument introduced the possibility of multiple autoscalers being deployed simultaneously in one framework. They argued that multiple components being scaled by individual decision processes could result in good autoscaling results while maintaining low complexity per decision process. In their opinion, this could be useful when multiple levels of scaleable components are built on top of each other. We summarised this in statement 16.

The participants brought up the possibility of measuring the elasticity of an application during run time, as stated in statement 17. They argued that this is a valuable feature as it reduces their workload by removing the need to benchmark an application to have appropriate elasticity values. Another point they argued was detecting changes in elasticity due to autoscaling as an additional benefit. They mentioned that elasticity changes could occur when components are scaled due to dependencies between automatically scaled components.

In the discussion, one participant stressed the need for the ability to scale multiple containers in one step. They mentioned the need to be able to shorten the time needed to scale up to a certain amount of containers through this method. We summarised this in statement 18.

One participant mentioned the desire to be able to configure the time interval of the autoscaler. The participant stressed that this is vital, especially in fast-changing environments. Statement 19 shows this opinion.

One participant also mentioned the desire to be able to configure rigid upper and lower limits for an autoscaler. This way, the autoscaler has built-in fail saves and can not dramatically over or under-scale. We represent this opinion in statement 20.

# Statement	Statement	# Participants
15	"Good enough" results and low amount of maintenance needed in an autoscaler is preferable to perfect results but high amounts of needed maintenance.	2
16	We want multiple autoscalers for different components in one framework	2
17	We want the autoscaler to be able to automatically measure elasticity and be able to use it as input for learning	2
18	Ability for fast spin-up or multiple container spin-up is necessary in some environments to meet possible needs fast enough	1
19	We want to be able to configure variable time steps for the autoscaler and explainability	1
20	We want to be able to define hard upper and lower limits the autoscaler can not scale higher or lower from	1

Table 4.9: Low Confidence Statements of Interest regarding the Structure of and the Ability to Configure the Autoscaler.

4.3 Resulting Requirements

This section shows the requirements we elicited from the participants' feedback on the questions in the sections above. We grouped the elicited requirements from Section 4.2.2 by the two low detail layers of the COSMOD-RE approach, the system layer and the functional decomposition layer. The requirements we extracted and are part of the system layer are show in Table 4.10. They are requirements with high confidence, as they are backed by either the questionnaire or multiple participants in the discussion. We show additional requirements with low confidence in Table 4.11. We have elicited the requirements from the statements the participants made. This includes opinions from the discussion as well as the answers to the questionnaire. Table 4.10 shows the requirements ID, the requirement and the statement IDs the requirement has been elicited from. Requirements with no attached statement ID are derived from the questionnaire answers directly. Requirements with only one statement ID are directly derived from a statement and may be backed by answers in the questionnaire. Requirements with multiple statement IDs are combined with other requirements and incorporate the answers from the questionnaire. The requirements of the functional decomposition layer are shown in Table 4.12. They are elicited the same way as the system layer requirements. However, they are heavily dependent on the requirements of the system layer and are, therefore, mainly derived from them. We marked requirements derived from other requirements as R##. We only used strong requirements from the system layer to build the requirements for the functional decomposition layer. Additionally, we limited the functional decomposition requirements to general requirements to not limit the possible scope structures of the autoscaling framework. We limited them to the minimum functionalities required by an autoscaling framework.

#	Requirement	Original Statement IDs
1	The system should be able to present visual data on metrics in the form of graphs.	1,2
2	The system should be able to present explainability information at any given time.	2,5,9
3	The system should be able to deliver input metrics as a time series.	2
4	The system should orientate itself on Prometheus or Graphana dashboards in terms of appearance.	1,2,7,11
5	The system should be able to display heatmaps.	3
6	The system should be able to present alternative decisions an autoscaler could have made at a given time.	4
7	The system should be able to present the error of the objective function of the used machine learning approach at a given time.	4,9
8	The system should be able to grant easy access to information from a timeline for the user.	2,7,8,9
9	The system should be able to display historical data.	2,7
10	The system should be able to show influence between different autoscalers and their respective applications.	3,8
11	The system should be able to present logging information corresponding to scaling decisions on demand.	2,9,10,11
12	The system should be able to support different explainability approaches.	3,4,5,6
13	The system should provide anomaly detection for workloads and autoscaling decisions	12
14	The system should allow the user to make changes in the parameters of the machine-learning or rule-based scaling approach.	13
15	The system should be able to allow data access by REST API	
16	The system should have a graphical user interface for all functionalities	
17	The system should be able to be accessed by a CLI	

Table 4.10: Strong System Layer Requirements.

4.4 Evaluation Scheme

We start our evaluation scheme by answering key questions that Rosenfeld and Richardson [RR19] propose to be able to evaluate the explainability of a system accurately.

The first question is used to clarify why the system should be explainable. In our case, this is relatively obvious. The system should be explainable to make scaling decisions understandable for system operators. This allows operators to maintain the system and make configuration changes to improve system performance.

#	Requirement	Original Statement IDs
18	The system should be able to produce a forecast of estimated workloads for future time steps	5
19	The system should be able to display data in the form of matrices	6
20	The system should be able to retain information over longer time periods for later review.	7,11
21	The system should be able to accommodate multiple autoscalers.	10,11
22	The system should allow to manage multiple autoscalers for different components	16
23	The system should allow to benchmark the automatically scaled application	17
24	The system should allow to configure the autoscaling intervals	19
25	The system should allow to configure upper and lower container limits for the autoscaler	20

Table 4.11: Weak System Layer Requirements.

#	Requirement	Original Statement IDs
26	The system needs a module for autoscaling	
27	The system needs a module for delivering explainability data	R1,R2
28	The system needs a graphical user interface for displaying metrics	R1,R17
29	The system needs graphical user interface for displaying explainability information	R2,R17
30	The system needs to display information of R27 when demanded in R26	R1,R2,R3,R9
31	The system needs to be able to exchange data between autoscaling module and explainability module	R2,R5,R6,R7,R10,R12,R15
32	The system needs an interface for configuring the autoscaling module	R12
33	The system needs to provide a REST API to allow data access	R15
34	The system needs to provide a CLI to access functionalities	R17

Table 4.12: Functional Decomposition Layer Requirements.

For the second question, we set the target of the explanations explicitly to operators. We do this because operators are the primary users of autoscaling systems. Other stakeholders have little to no direct use of explainability in autoscaling systems.

The third and fourth questions are part of the goals of our research. The answers to these questions can partly be found in our related work. We provide additional new answers with our requirements from the previous chapter. The last question deals with how explanations can be evaluated. We aim to answer this question in this chapter by building an evaluation scheme.

By utilizing the Goal-Question-Metric approach as shown by Basili et al. [BCR94] we build an evaluation scheme for evaluating autoscaling frameworks and explainability of those frameworks. We start by identifying central questions our evaluation scheme should evaluate. The central goal we identify is: We want to assess an autoscaling framework that utilizes machine-learning and provides explainability.

We then brake this central point down into subcategories. We brake the assessments down into assessments of functionalities regarding (1) explainability , (2) configuration and (3) accessibility. We come up with questions regarding each of these categories by considering the insights we gathered from our requirement elicitation process. We further added questions and metrics derived from information we gained from the papers of Samek et al. [SWM17] and Doshi-Velez and Kim [DK17].

We group each strong requirement we elicited for the system layer we show in Table 4.10 into the appropriate category. We formulate them as questions on the ability of the framework to fulfil the requirements. Additionally, we come up with metrics that can be used to measure the completion on each question.

Figure 4.10 and Figure 4.11 show our results in term of explainability. Here we explore how we can quantify the frameworks abilities to represent data and deliver meaningful information to the operator. As can be seen we focus on what types of data the framework can present and how explainability is provided. Most of the questions are in part measured by the subjective evaluation of the framework operator due to the importance of the subjective opinions of the operator. For representation standard qualities such as “Scene organization” “Occlusion Culling” as described by Post et al. [PNB02] we use the shorthand “Quality of presentation” or “Quality of visualisation”.

Our questions regarding the configuration ability of an autoscaling framework are shown in Figure 4.12. We explore the frameworks’ ability to fulfil the needs of operators regarding the ability to configure the autoscaling approach used. The configuration needs focus on the ability to tweak machine-learning parameters and define a set of metrics used as input for the decision-making process of the autoscaler. The metrics we came up with in terms of these configuration qualities focus on checking if the possibility exists and if they are helpful. Figure 4.13 shows the same in terms of accessibility.

Q1 Is the framework able to satisfyingly present metrics visually?
- Q1.1 Is the framework able to satisfyingly display metrics in the form of graphs?
  - M1.1.1 Quality of graph visualisation
  - M1.1.2 Subjective evaluation by framework operator
- Q1.2 Is the framework able to satisfyingly display data in the form of time series?
  - M1.2.1 Quality of time series visualisation
  - M1.2.2 Subjective evaluation by framework operator
- Q1.3 Is the framework able to satisfyingly display data in the form of historical data?
  - M1.3.1 Quality of historical data visualisation
  - M1.3.2 Subjective evaluation by framework operator

Figure 4.10: Explainability Questions and Metrics. Part 1.

Q2 Does the framework provide explainability?

- Q2.1 Does the framework support multiple explainability approaches for a machine-learning autoscaling approach?
  - M2.1.1 Support of meaningful explainability methods
  - M2.1.2 Subjective evaluation by framework operator
- Q2.2 Can the framework show underlying machine-learning models as heatmaps?
  - M2.2.1 Fitness of heatmaps depending on the used autoscaling method
  - M2.2.2 Ability to display heatmaps meaningfully
  - M2.2.3 Quality of presentation
  - M2.2.4 Subjective evaluation by framework operator
- Q2.3 Can the framework provide alternative scaling scenarios based on slight variations of the input?
  - M2.3.1 Availability of presentation of possible alternative scaling decisions
  - M2.3.2 Quality of presentation of results
  - M2.3.3 Subjective evaluation by framework operator
- Q2.4 Can the framework display the error of objective function value?
  - M2.4.1 Possibility to display objective function value of autoscaling method
  - M2.4.2 Quality of presentation
  - M2.4.3 Subjective evaluation by framework operator
- Q2.5 Can the framework show relationships between managed applications?
  - M2.5.1 Possibility to display relationships between managed application
  - M2.5.2 Possibility to display influence of managed application on each other
  - M2.5.3 Quality of presentation
  - M2.5.4 Subjective evaluation by framework operator
- Q2.6 Can the framework provide logging information in a meaning full way?
  - M2.6.1 Amount of logging information retained
  - M2.6.2 Possibility to display logging information
  - M2.6.3 Quality of presentation
  - M2.6.4 Subjective evaluation by framework operator

Figure 4.11: Explainability Questions and Metrics. Part 2.

Q3 Does the framework provide the ability to configure the autoscaler?
- Q3.1 Does the framework allow configuration of autoscaler machine-learning parameters?
  - M3.1.1 Possibility of configuration
  - M3.1.2 Selection of possible configurations
  - M3.1.3 Subjective evaluation by framework operator
- Q3.2 Does the framework allow configuration of the autoscaler interval?
  - M3.2.1 Possibility of configuration
  - M3.2.2 Allowed granularity of configuration
  - M3.2.3 Subjective evaluation by framework operator
- Q3.3 Does the framework allow configuration of input metrics and objective functions?
  - M3.3.1 Possibility of configuration
  - M3.3.2 Subjective evaluation by framework operator
- Q3.4 Does the framework allow configuration of the rules of a rule-based autoscaler?
  - M3.4.1 Possibility of configuration
  - M3.4.2 Fitness of configuration possibility to used autoscaling method
  - M3.4.3 Subjective evaluation by framework operator

Figure 4.12: Configuration Questions and Metrics.

Q4 Does the framework provide accessibility for the user?
- Q4.1 Does the framework provide a sufficient graphical user interface?
  - M4.1.1 Presence of a graphical user interface
  - M4.1.2 Quality of the graphical user interface
- Q4.2 Does the framework provide a sufficient CLI?
  - M4.2.1 Presence of a CLI
  - M4.2.2 Quality of the CLI
- Q4.3 Does the framework provide a sufficient REST API?
  - M4.3.1 Presence of a REST API
  - M4.3.2 Quality of the REST API

Figure 4.13: Accessibility Questions and Metrics.

5 Evaluation

In this section, we evaluate our research methodology and results. We show the shortcomings of our survey design. We go into our evaluation scheme design and show its shortcomings. After this, we discuss our research goals and if we have achieved them. Ultimately, we discuss the threats to validity and show our steps in minimizing their impact.

5.1 Survey Design

We designed our questionnaire to assess the participants' previous knowledge of autoscalers. We question the participants on their previous use of autoscalers. We inquire about what types of autoscaler they use and what metrics they use as inputs to these autoscalers. Additionally, we inquire about their debugging process and information needs during this process. With these questions, we try to establish a foundation for what our participants see as state-of-the-art autoscaler usage. This allows us to confirm whether our assumptions while creating the workshop and questionnaire reflect the situation in a productive environment.

After establishing this foundation, we ask the participants what information they expect from an autoscaling framework that provides explainability. This inquiry contains questions regarding information they imagine can be helpful. We inquire about the features an autoscaling framework with explainability features should provide. Additionally, we ask about the type of explanations they deem useful. Some questions deal with the type of data representation and accessibility the participants prefer. These questions are designed to bring forth the needs the participants are experiencing while working with autoscaling frameworks. They allow us to align the requirements we develop better with the participants' vision.

We collected the questionnaire's answers after the workshop with an online form. Should some of the questions in the questionnaire not be clear for the participants, they were given the opportunity to ask the researcher for clarification. This allows the participants to answer questions without direct interaction with the researcher, should they desire this, while not neglecting potential questions they might have.

Our approach, of course, has shortcomings. Due to the exploratory nature of our expert survey, the questions we put into the questionnaire may severely limit the imagination of the participants. Generating generalized questions that fit the exploratory approach while simultaneously not losing important details is complicated and often impossible. In order to minimize limitations on the participants' imagination, we are bound to lose out on details. This trade-off is an apparent shortcoming of our approach.

The same holds true for defining the scope of the questions. Experts' time is highly valued, and there are natural limitations on how much time a workshop can take up. Therefore, the amount and detail of questions must be carefully selected to not lose experts due to their time constraints. More

RG1	Re-implement CAUS in Python and update dependencies on the Kubernetes API
RG2	Build requirements for autoscaling frameworks with explainability from the insights of the expert user study
RG3	Build an evaluation scheme for autoscaling frameworks with explainability components

Table 5.1: Research Goals.

expedited questions are valuable for clarifying details. More general questions help provide room to explore different ideas and concepts simultaneously on a broad scale. Due to the time constraints of our workshop and questionnaire, we must choose which questions to include carefully. Therefore, we have to leave out questions which we might consider helpful but do not provide enough return. This results in another shortcoming of our design.

5.2 Evaluation Scheme Design

We designed our evaluation scheme to capture participants' requirements regarding autoscaling frameworks. We defined a central goal we wanted to evaluate. We split the goal into subgoals we have to evaluate in order to be able to evaluate the central goal. Each subgoal is formulated into a question. They cover three main parts of an autoscaling system: System accessibility, system configuration, and explainability. Each part contains evaluation questions derived from the participants' requirements. This structure allows us to categorize the participants' requirements into the category each subgoal provides. Additionally, it allows future users of the evaluation scheme to quickly identify relevant parts for the functionalities they want to evaluate.

We then formulated each of the requirements into a question. We came up with metrics that must be considered when evaluating these questions. These metrics provide the evaluation scheme with measurable qualities and tie it directly to the framework it is designed to evaluate. These metrics allow for comparing multiple frameworks and provide guidance on how the qualities operators look for can be evaluated.

However, our evaluation scheme is not exhaustive. We only cover the features operators named in the workshop and questionnaire. In a fully implemented autoscaling framework, more features than the ones we show can be of interest. Additionally, there can be multiple additional metrics for each quality that we have not covered. The metrics we provide are our best effort for making the named qualities quantifiable. These points are a shortcoming of our evaluation scheme. Possible modifications of the evaluation scheme include adding more qualities to evaluate and expanding them with more metrics.

5.3 Discussion

At the beginning of this thesis, we formulated three research goals (see Table 5.1). In the following, we show if and how we managed to fulfil our objectives.

As for the first goal, we successfully managed to re-implement CAUS. We improved its maintainability and interfacing with Kubernetes by utilizing officially supported libraries instead of manually targeting API endpoints with requests. Our implementation allows other developers and researchers to further modify and improve CAUS easily.

For RG2, we managed to build requirements for an autoscaling framework. Our requirements include information about what operators want in terms of explainability. Our requirements give insights regarding autoscaling, explainability and configuration needs expected to be satisfied by an autoscaling framework. We managed to complete RG2 through these requirements.

Our evaluation scheme may not capture everything system operators are looking for. However, we are confident that our research at least provides first insights into the topic and will help future researchers and developers build autoscaling frameworks that cater to the needs of system operators. In that regard, we argue that our RG3 is also fulfilled.

5.4 Threats to Validity

In this section, we discuss what might threaten the validity of our results. We used the classification of these threats laid out by Runeson and Höst [RH09] to identify threats. We show the techniques we used to minimize these identified threats.

5.4.1 Construct Validity

Construct validity is defined as the accumulation of evidence to support the interpretation of what a measure reflects [RH09]. Our approaches from the workshop to the requirement elicitation to the evaluation scheme all suffer in some form from this validity concern. Our personal bias in constructing the questions for the questionnaire could influence the results. To remedy this, we tried to keep the questions as open as possible. For the workshop, the discussion and its course over the workshop may influence the participants differently. The base scenario of the workshop may influence the participants' answers. In particular, if the scenario is too restricted, the participants may focus too much on one particular direction. We tried to compensate for this effect by making the provided scenario as general as possible and asking the participants to apply their personal use cases instead. We provided some basic questions in each discussion to get the participants to imagine what qualities a potential autoscaling framework should have and keep the discussion between the participants going.

The threat of misinterpreting participants' statements and eliciting wrong requirements exists in the requirements elicitation process. We mitigate this by relying on multiple statements for eliciting requirements. Additionally, we reviewed the discussion to ensure we aligned the requirements we elicited with the direction the participants stated them in. We weight the statements backed by multiple participants higher than those backed by few participants. We structure the requirements after the COSMOD-RE principles [Poh10] to cross-reference the requirements of different layers and make the whole process more robust.

5.4.2 External Validity

"External validity is of concern when we try to generalize the findings we developed in our investigated case" [RH09]. We maximize generalization possibility by collecting data from multiple participants and building our requirements from there. Additionally, we sought participants from different companies and occupational backgrounds. This way, we can improve the generalization of our requirements for an autoscaling framework. However, our findings are only viable in the context of autoscaling. They are not suited for general explainability due to the construction of our workshop and the focus on finding requirements for a creative new software solution.

5.4.3 Internal Validity

The properties we introduced in our research, as shown above, are also of interest for internal validity. "Internal validity is of concern when causal relations are examined. When the researcher is investigating whether one factor affects an investigated factor, there is a risk that the investigated factor is also affected by a third factor" [RH09]. All requirements we derived are either supported by multiple participants or multiple other requirements. Combined with the participant selection, this reduces the risk of some outside factors influencing our findings.

5.4.4 Reliability

"Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers" [RH09]. As the researcher was a participant in the discussion, the reliability of our findings may be in question. The participating researcher tried to stay out of the discussion as much as possible. Additionally, the researcher tried to keep the discussion going with the same questions in all workshops. This way, we improve the reliability of our results.

6 Conclusion

In the following, we summarise our findings. We show the benefits our research provides and the limitations of the context in which our findings are valid. We show our research experiences and the areas in which we improved our knowledge. In the end, we show what possible future work can benefit from our research and the directions this future research can take to improve our findings in various aspects.

6.1 Summary

At first, we re-implemented the existing autoscaler CAUS [KFB18] in Python and improved its interface for Kubernetes. Our goal was to build a base autoscaler to which one can add machine-learning to the decision-making process and provide users with explainability. We then changed our approach to gain insights into what such an autoscaling framework should provide in terms of functionality. We conducted an expert user survey as a workshop for operators in the autoscaling field. Our goal was to identify operators' requirements for an autoscaling framework that combines machine-learning and explainability. After the workshop, we identified requirements from the data we collected and built an evaluation scheme from them.

The workshop involved a discussion phase and a questionnaire. The discussion was held to help participants explore their needs for a new autoscaling framework. They were confronted with a hypothetical scenario about an autoscaling application. They were then asked what information they wanted in this scenario and how a framework with explainability functionalities could help them. At the end of the discussion, they were tasked with answering a questionnaire. The questionnaire collected demographic data, data about the current usage of autoscalers in the industry, and information about the needs and desires participants have regarding the current state of the art.

After collecting the data, we formalized the requirements the participants had. We did this by running the data through a requirements elicitation process. We ranked statements the participants brought forth by relevance. Additionally, we identified hidden requirements that are indicated by directly stated requirements. We divided the requirements by layers of the COSMOD-RE approach [Poh10]. Additionally, we grouped requirements into strong and weak requirements depending on the confidence we attributed to each requirement.

We took the elicited requirements for the evaluation scheme and built formal attributes. We did this in order to be able to determine the degree to which an autoscaling framework fulfils the requirements operators have. The resulting evaluation scheme covers explainability properties, configuration properties and accessibility properties of autoscaling frameworks.

After this, we evaluated our survey design and evaluation scheme design. We also discussed if we reached our research goals. We ensured the validity of our results based on four metrics: Construct Validity, External Validity, Internal Validity and Reliability.

6.2 Benefits

First and foremost, our results can benefit researchers on autoscaling and explainability. Our insights and requirements can be used in further research and build a foundation upon which future researchers can expand on. With further work, the re-implemented CAUS can be used to build a full-scale autoscaling framework which combines machine-learning, rule-based autoscaling and explainability approaches.

Further stakeholders that can benefit from our research are developers. Our research can be used to set the general direction of further scenario generation and requirements engineering processes which could result in the development of better autoscaling frameworks. Additionally, our evaluation scheme can be used as a checklist for basic functionalities such a framework has to provide.

Last but not least, operators of autoscaling environments benefit from our research as pushing the abilities of autoscaling frameworks can improve their quality of work. Additionally, further research can reduce the risk of misbehaviour of autoscaling systems and therefore reduce the operators' workload. Better explainability makes it easier for operators to understand their system and can reduce the time it takes to identify errors.

6.3 Limitations

Due to how we approached this thesis and created the workshop, our results are limited to the autoscaling field. The requirements we elicited are only applicable when developing a framework for autoscaling and adding explainability to existing autoscaling approaches. Additionally, our findings are only applicable to autoscalers utilizing vertical scaling. The answers of the workshop participants were taken in the context of vertical scaling autoscalers and could change substantially when in the context of horizontal scaling autoscalers.

Due to the time constraints of this thesis, we could not reevaluate our findings with our participants. This makes our findings, at best, a good indicator for future research and further requirement elicitation processes. However, our findings can not solely be relied upon for developing an autoscaling framework. Future researchers and developers must build their own requirements and adapt elicitation processes to fit their goals.

Our work is also limited by the number of participants we managed to acquire. While we managed to spread our participants over a large area of expertise and backgrounds, only asking ten people is not enough to gather statistically relevant data. Since, in our findings, all participants had similar ideas, we are confident we managed to gather valuable insights. However, these insights might reflect a shared understanding of this group, not the wider community.

Since we built our evaluation scheme from the requirements we previously elicited, the above points are also relevant for the evaluation scheme. Additionally, we built the evaluation scheme vague in order for it to cover every aspect of interest for autoscaling frameworks. Points of our evaluation scheme that are not fulfilled can be interpreted as missing functionalities. On the other side, our evaluation scheme can not cover every possible functionality that could be seen as helpful or even necessary.

6.4 Lessons Learned

We learned a lot about the problems in setting up a workshop that does not frame the participants' opinions. These problems and the goal of exploring potential features of an imaginary creative new software solution make it very difficult to build a good experiment with reliable results. For example, building a setting for exploring these potential features already requires our own creativity. Not building the setting correctly could lead to not accurately reflecting the needs of the participants or stirring the discussions and opinions in a particular direction.

We also experienced how complex setting up Kubernetes can be and the difficulties in getting into autoscaling. We can see how this initial workload could stop operators from either using autoscaling or using more complex approaches in autoscaling instead of static solutions or already familiar simple autoscaling approaches.

6.5 Future Work

This thesis can enable several research possibilities. Our first suggestion for future work is further research into the requirements of autoscaling frameworks. Our requirements can give some insights into the needs operators have. However, more research is needed to come to definitive conclusions on the necessary features of an autoscaling framework with explainability. These can be further refined by using the requirements of this thesis in order to build a prototype framework. Using a prototype in future research and requirements elicitation can help participants imagine possible solutions. This should help researchers to define a more specific scope for their research and lead to more specific results. Especially using re-evaluations could help in getting more robust and specific results. The same improvements can be applied to our evaluation scheme. The scheme needs further re-evaluation and more specific reiterations once more insights in the autoscaling field are available. Using similar approaches as this thesis but with more participant engagement should improve the resulting evaluation schemes. Additionally, we suggest using more participants for a general scheme or more in-depth work with fewer participants for an evaluation scheme specific to a selected autoscaler and explainability approach.

Our insights can also lead the way for developing future autoscalers. We hope our research lets future researchers focus on the importance of explainability in autoscaling. Current autoscalers do not support explainability approaches to the extent they need. Contrary to the tendencies of the current research of autoscaler focusing on the performance of the autoscalers, our research shows that operators value easy-to-understand and manageable autoscalers higher than pure performance. Operators are willing to lose some performance if the quality of maintenance rises significantly. Future work in that regard can improve by explaining their scaling decisions without further work needed by the operators of these autoscalers.

Bibliography

[Ama] Amazon. URL: https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html (cit. on p. 1).

[Aoy+98] M. Aoyama et al. "New age of software development: How component-based software engineering changes the way of software development". In: *1998 International Workshop on CBSE*. 1998, pp. 1–5 (cit. on p. 1).

[BCR94] V. R. Basili, G. Caldiera, H. D. Rombach. "The goal question metric approach". In: *Encyclopedia of software engineering* (1994), pp. 528–532 (cit. on pp. 19, 36).

[BP21] V. Belle, I. Papantonis. "Principles and practice of explainable machine learning". In: *Frontiers in big Data* (2021), p. 39 (cit. on pp. 1, 9).

[BSH+10] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller. "How to explain individual classification decisions". In: *The Journal of Machine Learning Research* 11 (2010), pp. 1803–1831 (cit. on p. 10).

[BSM20] D. Balla, C. Simon, M. Maliosz. "Adaptive scaling of Kubernetes pods". In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. 2020, pp. 1–5. DOI: 10.1109/NOMS47738.2020.9110428 (cit. on pp. 1, 8).

[BXS+20] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. F. Moura, P. Eckersley. "Explainable machine learning in deployment". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 648–657 (cit. on pp. 1, 9).

[Crn01] I. Crnkovic. "Component-based software engineering—new challenges in software development". In: *Software Focus* 2.4 (2001), pp. 127–133 (cit. on p. 1).

[DK17] F. Doshi-Velez, B. Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017) (cit. on pp. 2, 11, 19, 36).

[FMD+22] I. Fé, R. Matos, J. Dantas, C. Melo, T. A. Nguyen, D. Min, E. Choi, F. A. Silva, P. R. M. Maciel. "Performance-Cost Trade-Off in Auto-Scaling Mechanisms for Cloud Computing". In: *Sensors* 22.3 (2022), p. 1221 (cit. on p. 1).

[Goo] Google. URL: https://scholar.google.com/ (cit. on p. 5).

[HC01] G. T. Heineman, W. T. Councill. "Component-based software engineering". In: *Putting the pieces together, addison-westley* 5 (2001) (cit. on p. 1).

[IAA20] M. Imdoukh, I. Ahmad, M. G. Alfailakawi. "Machine learning-based auto-scaling for containerized applications". In: *Neural Computing and Applications* 32.13 (July 1, 2020), pp. 9745–9760. ISSN: 1433-3058. DOI: 10.1007/s00521-019-04507-z. URL: https://doi.org/10.1007/s00521-019-04507-z (cit. on pp. 1, 6, 8).

[IBM] IBM. URL: https://www.ibm.com/watson/explainable-ai (cit. on p. 7).

[Kam19] S. Kambhampati. “Challenges of Human-Aware AI Systems”. In: *CoRR* abs/1910.07089 (2019). arXiv: 1910.07089. URL: http://arxiv.org/abs/1910.07089 (cit. on p. 9).

[KFB18] F. Klinaku, M. Frank, S. Becker. “CAUS: an elasticity controller for a containerized microservice”. In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*. 2018, pp. 93–98 (cit. on pp. iii, v, 2, 6, 13, 15, 21, 45).

[KL17] P. W. Koh, P. Liang. “Understanding Black-box Predictions via Influence Functions”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup, Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1885–1894. URL: https://proceedings.mlr.press/v70/koh17a.html (cit. on pp. 1, 9).

[Kuba] Kubernetes. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/ (cit. on pp. 6, 25).

[Kubb] Kubernetes-client. URL: https://github.com/kubernetes-client/python (cit. on p. 15).

[Lit] Litmaps. URL: https://www.litmaps.co/ (cit. on p. 5).

[LML12] T. Lorido-Botrán, J. Miguel-Alonso, J. A. Lozano. “Auto-scaling techniques for elastic applications in cloud environments”. In: *Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09* 12 (2012), p. 2012 (cit. on p. 8).

[Mah20] B. Mahesh. “Machine learning algorithms-a review”. In: *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), pp. 381–386 (cit. on p. 1).

[Mic] Microsoft. URL: https://azure.microsoft.com/en-us/products/virtual-machines/autoscale/ (cit. on pp. 1, 25).

[MLB+17] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller. “Explaining nonlinear classification decisions with deep Taylor decomposition”. In: *Pattern Recognition* 65 (2017), pp. 211–222. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2016.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0031320316303582 (cit. on p. 10).

[PNB02] F. H. Post, G. Nielson, G.-P. Bonneau. “Data visualization: The state of the art”. In: (2002) (cit. on p. 36).

[Poh10] K. Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 3642125778 (cit. on pp. 18, 19, 43, 45).

[RGLT19] G. Rattihalli, M. Govindaraju, H. Lu, D. Tiwari. “Exploring Potential for Non-Disruptive Vertical Auto Scaling and Resource Estimation in Kubernetes”. In: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. 2019, pp. 33–40. DOI: 10.1109/CLOUD.2019.00018 (cit. on pp. 1, 7, 8).

[RH09] P. Runeson, M. Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empir. Softw. Eng.* 14.2 (2009), pp. 131–164 (cit. on pp. 43, 44).

[RR19] A. Rosenfeld, A. Richardson. “Explainability in human–agent systems”. In: *Autonomous Agents and Multi-Agent Systems* 33.6 (2019), pp. 673–705 (cit. on pp. 2, 11, 19, 34).

[RSG16] M. T. Ribeiro, S. Singh, C. Guestrin. ""Why Should I Trust You?"" In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939778 (cit. on p. 10).

[SGK17] A. Shrikumar, P. Greenside, A. Kundaje. "Learning Important Features Through Propagating Activation Differences". In: *CoRR* abs/1704.02685 (2017). arXiv: 1704.02685. URL: http://arxiv.org/abs/1704.02685 (cit. on p. 10).

[SM19] W. Samek, K.-R. Müller. "Towards Explainable Artificial Intelligence". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer International Publishing, 2019, pp. 5–22. DOI: 10.1007/978-3-030-28954-6_1 (cit. on p. 10).

[SSB22a] S. Speth, S. Stieß, S. Becker. "A Saga Pattern Microservice Reference Architecture for an Elastic SLO Violation Analysis". In: *Companion Proceedings of 19th IEEE International Conference on Software Architecture (ICSA-C 2022)*. IEEE, Mar. 2022. DOI: 10.1109/ICSA-C54293.2022.00029 (cit. on p. 22).

[SSB22b] S. Speth, S. Stieß, S. Becker. "A Vision for Explainability of Coordinated and Conflicting Adaptions in Self-Adaptive Systems". In: *Proceedings of 14th Central European Workshop on Services and their Composition (ZEUS 2022)*. CEUR, Feb. 2022, pp. 16–19 (cit. on p. 1).

[Stu] U. of Stuttgart. URL: https://www.opentosca.org/ (cit. on p. 1).

[STY17] M. Sundararajan, A. Taly, Q. Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup, Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3319–3328. URL: https://proceedings.mlr.press/v70/sundararajan17a.html (cit. on p. 10).

[SWM17] W. Samek, T. Wiegand, K.-R. Müller. "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models". In: *arXiv preprint arXiv:1708.08296* (2017) (cit. on pp. 2, 11, 19, 36).

[TDFS20] L. Toka, G. Dobreff, B. Fodor, B. Sonkoly. "Adaptive AI-based auto-scaling for Kubernetes". In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. 2020, pp. 599–608. DOI: 10.1109/CCGrid49817.2020.00-33 (cit. on pp. 1, 8).

[ZD22] M. Zilch, Delvh. *zilchms/caus-python: Initial release of CAUS rework in Python*. 2022. DOI: 10.5281/ZENODO.7065707 (cit. on p. 21).

[ZHH+19] A. Zhao, Q. Huang, Y. Huang, L. Zou, Z. Chen, J. Song. "Research on Resource Prediction Model Based on Kubernetes Container Auto-scaling Technology". In: *IOP Conference Series: Materials Science and Engineering* 569.5 (July 2019), p. 052092. DOI: 10.1088/1757-899x/569/5/052092. URL: https://doi.org/10.1088/1757-899x/569/5/052092 (cit. on p. 7).

[Zil22] Zilch Markus. *Explainability in Autoscaling Frameworks Questionnaire and Answers*. en. 2022. DOI: 10.5281/ZENODO.7043866 (cit. on p. 16).

[ZY18] W.-S. Zheng, L.-H. Yen. "Auto-scaling in Kubernetes-based fog computing platform". In: *International Computer Symposium*. Springer. 2018, pp. 338–345 (cit. on p. 8).

All links were last followed on September 12, 2022.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature