A Quality-driven Machine Learning Governance Architecture for Self-adaptive Edge Clouds

Claus Pahl, Shelernaz Azimi, Hamid R. Barzegar and Nabil El Ioini
Free University of Bozen-Bolzano, Bolzano, Italy

Keywords: Machine Learning, Quality, Edge Cloud, Cloud Controller, Resource Management, AI Engineering, AI Governance, Explainable AI.

Abstract: Self-adaptive systems such as clouds and edge clouds are more and more using Machine Learning (ML) techniques if sufficient data is available to create respective ML models. Self-adaptive systems are built around a controller that, based on monitored system data as input, generate actions to maintain the system in question within expected quality ranges. Machine learning (ML) can help to create controllers for self-adaptive systems such as edge clouds. However, because ML-created controllers are created without a direct full control by expert software developers, quality needs to be specifically looked at, requiring a better understanding of the ML models. Here, we explore a quality-oriented management and governance architecture for self-adaptive edge controllers. The concrete objective here is the validation of a reference governance architecture for edge cloud systems that facilitates ML controller quality management in a feedback loop.

1 INTRODUCTION

Edge cloud systems are more and more using Machine Learning (ML) techniques if sufficient data is available to create ML models. A typical example is a self-adaptive system built around a controller that, based on monitored system data as the input, generates actions to maintain the system in question within expected quality ranges. Self-adaptive systems are prevalent in many IoT and edge cloud environments where manual adjustment is not feasible or not reliable. Virtualized systems are ideally suitable to apply some sort of control-theoretic mechanism to continuously and automatically adjust a dynamic system.

Our focus is edge clouds, i.e., connected device infrastructures that built on virtualization to allow dynamic resource management in line with application demands that create elasticity in the responses to demand changes. The management of the resources in terms of topology configuration, resource utilization and application is nowadays often based on models that are machine learned.

Machine learning can thus help to create controllers for self-adaptive edge cloud systems where automation is of critical importance (Mendonca et al., 2021). However, due to the ML approach to controller creation without a direct full control by expert software developers, quality needs to be specifically looked at. In broader terms, what is needed is a software engineering approach for ML-constructed software that addresses the specific quality concerns, often referred to as AI Engineering (Lwakatare et al., 2019). AI Governance based on better understanding (often referred to as explainability) and transparency of how the ML model works is here central in this wider AI engineering objective (Pahl and Azimi, 2021).

The observations above on ML-generated controllers and quality concerns related to this type of software (Azimi and Pahl, 2020b; Azimi and Pahl, 2020a; Azimi and Pahl, 2021) lead to challenges that we aim to address in our chosen technical application domain of edge cloud architectures (Pahl et al., 2018; Taibi et al., 2020; El Ioini et al., 2021). We develop a quality-driven governance architecture for self-adaptive edge-cloud applications that dynamically adjusts these their controllers through continuous model creation and improvements. The novelty lies in a comprehensive quality improvement framework beyond effectiveness of the controller only.

While both edge cloud management and in particular ML quality and governance are broad and challenging concerns, we will limit our investigation here to a clearly defined scope: resource-constrained edge clusters as the architectural setting and reinforcement learning as the ML technique for the controller itself

Pahl, C., Azimi, S., Barzegar, H. and El Ioini, N.
A Quality-driven Machine Learning Governance Architecture for Self-adaptive Edge Clouds.
DOI: 10.5220/0011107000003200
In *Proceedings of the 12th International Conference on Cloud Computing and Services Science (CLOSER 2022)*, pages 305-312
ISBN: 978-989-758-570-8; ISSN: 2184-5042

to determine self-adaptation mechanisms. The technical challenges to be addressed here include, firstly, scoring functions for considered qualities, including computability (for self-adaptable) and effective usage as the primary necessary aim and explainability as a second critical aim and, secondly, remedial strategies to address any detected quality deficiencies.

The paper is structured as follows. We first introduce a conceptual framework in Section 2, before analysing the state-of-the-art in Section 3. Section 4 presents our proposed architecture, which is the discussed in Section 5. We conclude with a summary and future work in Section 6.

2 CONCEPTUAL FRAMEWORK

2.1 Use Case and Motivation

In order to achieve meaningful results within the constraints given, we focus on a specific architectural setting: IoT and edge cloud architectures. Furthermore, the evaluation will involve concrete use cases from the domain of smart traffic and smart mobility.

The concrete setting here is edge cloud resource management (Scolati et al., 2019; von Leon et al., 2019). System adaption is required for edge cloud resource configuration, involving: monitor resource utilization and application performance and apply rules (an ML-generated rule model) for resource adaptation to meet performance and cost requirements. The rules adapt the resource configuration (e.g., size) to improve the performance/cost ratio. The chosen ML techniques here is reinforcement learning that works on a reward principle applicable in the self-adaptation loop. Finally, enact configuration change recommendation. The problem shall be described using a concrete use case:

- Problem: A resource controller for edge adaptation might suggests: if $Workload > 80\%$ then $Double(resource - size)$. The question is whether this recommendation is correct and overall the recommendation space is complete.
- Solution: Here a generated ML model could provide a recommendation for a 60% workload as a verified test case. Then, this recommendation can be scaled up to 80%.

ML-driven controller (model) creation and automated dynamic adaptation (the adaption by the controller is based on the rule model) cannot be done without proper quality monitoring. For this, the model quality needs to be assessed through score function regarding effectiveness, e.g., based on high accuracy. Detected quality deficiencies need to be subjected to a root cause analysis and suitable recommendation and enactment of remedies need to be determined.

2.2 Conceptual Challenges

Quality management of ML models in generated software and its link to underlying raw/training data is at the core of the problem here. Ground truth, which is the accuracy against the real world, in our cases means whether the system performance actually improves, if up-scaling was recommended.

Remedies include to improve data labelling, e.g., automate critical situation assessment (high risk of failure, based on past experience), which have a probability of discrimination and could be biased, be that through pre-processing (before ML training) and in-processing (while training). Research tasks are to find bias and remove bias (through a control loop), e.g., using favourable data labels. Examples are could smaller or bigger device clusters be favoured wrongly or specific types of recommended topologies or recommended configuration sizes (messages, storage etc)? Furthermore, explainability of the controller recommendations is a critical ingredient to map observed ML model deficiencies back to system-level properties (via the monitored input data to the ML model creation).

Our focus will be on the controller environment, i.e., the sensing devices that monitor the performance of the system and the actuators that enforce remedies proposed by the controller. Thus, we will consider faultiness in this environment as the root causes of quality problems. These faults could include sensor faults or communication faults in the network. In dynamic systems, a further complication is the need for continuous evaluation due to changing circumstances.

3 STATE-OF-THE-ART

We review related work, specifically focusing on ML for dynamic edge and IoT settings, before analysing the limitations and challenges.

3.1 Related Work

The application of ML techniques ranges from non-critical business applications to autonomous safety-critical software (Karkouch et al., 2016). Specifically, in distributed settings, the need for software dependability rises. A survey regarding the impact of ML on software development (Tokunaga et al., 2016) concludes that software practitioners still struggle to operationalize and standardize the software development practices of systems using ML. Instead of functional requirements that are often the focus in non-ML software, quantitative non-functional factors such as accuracy form central requirements for business-critical or safety-critical ML systems, which are in some business-critical or safety-critical domains of great importance.

It is noted that these systems are highly coupled. For instance, the performance of models is dependent on the quality of data processing. Poor data processing could prevent an effective ML model construction. In ML systems, “too low” and “too high” scores for performance measures as testing results both indicate defects. Not always is pre-construction validation [29] possible; thus, we aim here at an a-posteriori analysis to remedy problems.

The quality of raw data, the machine learning process perspective and the machine learning model quality are key building blocks in the construction process. Raw or source data quality has been investigated (Mohammedameen et al., 2019), resulting in quality frameworks that our earlier selection of quality attributes is based on. In (Mohammedameen et al., 2019)[23], data quality problems where classified into two groups of context-independent and context-dependent from the data and user perspective. In (De Hoog et al., 2019), a new architecture based on Blockchain technology was proposed to improve data quality and false data detection. In (Sridhar et al., 2018), a prototype of a distributed architecture for IoT was also presented, providing supporting algorithms for the assessment of data quality and security.

The ML process perspective is discussed in (Amershi et al., 2019). A machine learning workflow with nine stages is presented in which the early stages are data oriented. Usually the workflows connected to machine learning are non-linear and often contain feedback loops to previous stages in order to remedy quality concerns. If the system contains multiple, interconnected ML components, quality becomes even more critical. Investigating a broader loop from the final ML function construction stages to the initial data and ML training configuration stages has not been comprehensively attempted yet.

Another aspect is the machine learning model layer. Different supervised learning approaches were used. Specific quality metrics apply to ML techniques. The area under the receiver operating characteristic curve (AUC) is an example of quality for classification models. In (Sicari et al., 2016), a solution for model governance in production machine learning can build on provenance information to trace the origin of an ML prediction solution in order to identify the root cause of an observed problem symptom. Also the quality of data in ML has been investigated. An application use case was presented, but without a systematic coverage of quality aspects. Data quality as the root cause is important in many ML-supported applications. In (Deja, 2019), the authors investigate high-energy physics experiments. The work presented in (Deja, 2019) serves as an IoT setting. Some works highlight the need for a systematic, automated approach to achieve higher accuracy to remedy training problems arising from manual data labelling (Sheng et al., 2008). Here, ML techniques such as isolation forests as classification techniques or autoencoders as neural networks are looked at.

While most previous work looked for root causes of quality deficiencies in the ML construction, the IoT and edge cloud environment also needs to be considered as a consequence of the uncertainty of sensory data as a problem cause (Efron, 2020). Thus data quality needs a more prominent role in the construction process. (Efron, 2020) covers IoT root causes in the analysis, but not training/ML data problems.

3.2 Analysis and Challenges

Our task is to condense the different individual quality concerns into an integrated quality model that takes on board lessons learned from (De Hoog et al., 2019; Ehrlinger et al., 2019; Fang et al., 2016), but provides a closed feedback loop. We apply our proposed architecture to resource management and orchestration in edge clouds (Hong and Varghese, 2019), where controllers manage systems self-adaptively. Here, the compute, storage or network resources can be configured dynamically through a self-adaptation mechanism. Network configurations as one of the platform concerns have been investigated by a number of authors (Mantri et al., 2013; Tokunaga et al., 2016; Femminella and Reali, 2019). Another strategy is the dynamic allocation and management of tasks in distributed environments. Workload balancing has been a proposed solution (Zhao et al., 2019). Some authors have investigated specific applications such as streaming and other data-intensive contexts (Mohammedameen et al., 2019), where the interplay of communication, computation and storage needs to be coordinated. Equally, specific application domains such as vehicular networks have been looked at (Javed et al., 2020).

Our direction here is a specific type of architecture with layered and clustered edge clouds that are for instance suitable for automotive applications and networks. This requires a combination of horizontal and vertical orchestration techniques that have not been sufficiently explored for low-latency and high data volume applications.

Machine learning has been used in some of these advanced architectures (Wang et al., 2020). For instance, reinforcement learning (RL) is a promising solution. We also use reinforcement learning and build on our experience in QL and SARSA as two RL approaches that we used for resource management in central clouds and adapt this to edge architectures (Jamshidi et al., 2015; Arabnejad et al., 2017).

4 GOVERNANCE FRAMEWORK FOR ML EDGE MANAGEMENT

In the previous section, we already identified some concrete challenges and outlined the key architectural principles. This shall now be used to define a quality-driven architectural governance framework for edge cloud resource management. At the core is a ML-constructed controller.

4.1 ML-generated Controllers and Scoring Functions

Controllers that are ML-generated create the challenge of assessing and maintaining quality in a largely automated process. Intelligent quality management embedded into a systematic engineering framework is needed. The controller solution for self-adaptive system is build on reinforcement learning (RL) techniques that we have already successfully used for centralised cloud architectures (Jamshidi et al., 2015; Arabnejad et al., 2017), here adapted to the distributed and constrained edge context. It aims to maximize the notion of a cumulative reward. RL does not assume knowledge of an exact mathematical model and is thus suitable where exact methods become infeasible due to resource constraints and dynamic changes.

The controller manages workload and performance as qualities of the system in question, but the controller needs to be assessed in terms of ML qualities. Non-functional properties and respective scoring functions are relevant here for the controller (rather than functional testing), but this requires targeted score functions for quality measurement of the controller: accuracy of the controller actions is here the central scoring function, i.e., how accurate are the analyses of the controller and the remedial actions taken. We need to define score functions that are immediately computable (e.g., via group cardinalities as a suggestion), rather than observing future behaviour, which cause often unacceptable time delays.

4.2 A Reference Architecture for Edge Cloud Controller Quality

Adaptors and decision models at the core of controllers are suitable for ML-based creation. The outcome of the ML model is to implement an action, e.g., to adapt resources, divert traffic (IoT, roads), or to instruct machines. This should be a dynamic control loop, guided by defined quality goals. We use IoT-traffic use cases for the experimental validation.

4.2.1 Reference Governance Architecture

Our technical objectives are organised as follows into two layers to be reflected in our architectural framework: We develop a controller for self-adaptive edge cloud systems based on reinforcement learning as the ML technique. We develop an intelligent quality monitoring and analysis framework aligned with the needs of ML-generated controller software for self-adaptive software. This architecture aims to support quality management for ML-based adaptors. Figure 1 builds on a MAPE-K loop (Monitoring, Analysing, Planning, Executing – based on Knowledge) in two layers. The upper loop is the focus, but needs to take into account lower layer behaviour.

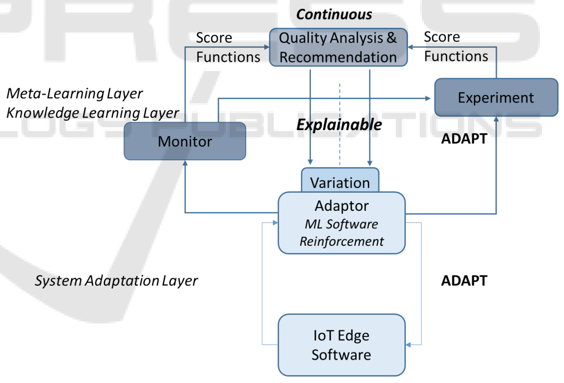


Figure 1: System Architecture – a lower edge controller loop and an upper ML quality management loop.

4.2.2 ML Controller and Quality Management

Important questions concern the full automation of the upper loop. The full automation is, however, not the objective here in this investigation. Nonetheless, this ultimate objective shall guide the work and shall aid its solution in the future. In concrete terms, the challenges are: Automate testing (M in MAPE), Automate the test evaluation (A in MAPE), Device a ML learning adjustment (P in MAPE). For this work, however, the upper loop is not meant be fully self-adaptive. Our objective here is to develop a core understanding that would allow full automation in the future. Thus, the objective here is to investigate the validity of the governance architecture, i.e., to demonstrate that an automation is beneficial and feasible.

For the MAPE-K controller design, we deploy a MAPE-K architecture pattern for the adaption as a specific type of an ML model self-adaption regarding accuracy. The evaluation of the controller is based on testing by checking variations and their effect the users experience regarding some metrics. The meta-layer upper MAPE-K loop is designed as follows:

- M: to consider are score functions for ML model quality, i.e., adaptor quality based on accuracy that is if possible linked back to application data quality, e.g., cloud resource utilization, application performance, with correctness and completeness as ML quality concerns.
- A: the analysis consisting of a root cause analysis for ML model quality problems and feeding into explainability concerns through partial dependency determination in order to identify what system factors can improve target quality most.
- P/E: To recommend and execute these recommendations, i.e., a rule update for the cloud adaptor with ML model creation configuration: training data size/ratio — linked to raw data correction (at least a recommendation).

This implements a meta-learning process that is not only learning to adapt the system, but learning to adapt the adaptor through a continuous testing/experimentation process that implements a knowledge learning layer. Continuous evaluation is, as in any self-adaptive system, a key component. The ultimate aim is to automate the model variation by MAPE-ing the ML model construction (meta-level optimization), e.g., labelling or test size/ratio.

5 EVALUATION

The target infrastructure is an IoT and edge cloud environment, where small devices (IoT) at the edge are linked to central cloud data centres.

Mobile edge use case: The key evaluation scenario is a video streaming application. The respective infrastructure will be fully developed as part of the EU H2020 project 5G-CARMEN that we are a partner of (Pomalo et al., 2020; Pomalo et al., 2020; Barzegar et al., 2020; Barzegar et al., 2021).

Lightweight edge devices and containers: We use a cluster of lightweight edge devices (specifically Raspberry Pi devices), deploying container-based applications on these.

Cloud and Edge Computing Lab (CECL): The CECL enables virtualization, cloud and edge computing experimentation. It consists of 3 HP DL380 gen10 servers (6 Virtual CPUs, 192GB RAM, 6 TB SSD) as central cloud/edge servers and 80 Raspberry Pi IV (BCM2711, Cortex-A72 quad-core (ARMv8) 64-bit SoC @ 1.5GHz, 4G RAM, 16G SD card) as lightweight edge devices.

Currently, a prototype is under development. A first controller version running on an RPi cluster is available (Gand et al., 2020).

5.1 Validation

We discuss the objectives, the setup of the validation experiments and report on our observations.

5.1.1 Evaluation Objectives

Results of the controller implementation – presenting the evaluation of the lower layer of our architecture from Fig. 1 – have been presented in (Gand et al., 2020). Here we focus on results for the upper layer of Fig. 1. In general, the evaluation of a quality controller (the upper layer) needs to cover the following solution components and respective strategies to evaluate: Regarding the monitoring of the controller quality, we need to carry out anomaly detection in the CECL lab. Evaluation activities are to induce anomalies into the testbed based on realistic patterns and to measure the impact by determining the accuracy as the selected scoring function of induced fault patterns.

In this paper, the primary objective is to evaluate the usefulness of the reference governance architecture. While the lower layer is already well explored, whether the proposed upper layer quality controller is beneficial and feasible remains an open question. For this, we need to demonstrate that by monitoring the controller, the root cause of detected quality anomalies can be detected and remedial actions be proposed, following the MAPE-K loop: M: monitor accuracy (and also related precision and recall) of the ML model. A: analyse the anomalies in the controller quality over time. P: plan a response using a rule system for remedial actions. E: provide a concrete, actionable remedy recommendation.

5.1.2 Experiments

In order to answer this research question, we conducted simulation experiments by inducing anomalies into two different input data sets: Data set 1 is a snapshot data of 72 rows counting volume of utilisation, 14 features that are largely independent. Data set 2 is a time series data of 197 rows collected at 49 data collection points on environmental data, 15 features that are partially interdependent. These were then analysed in terms of their impact of ML model accuracy (as well as precision and recall) and if a root cause and a respective remedy could be determined in order to complete the MAPE-loop described above. The use case will be the scenario already discussed: a smart traffic/mobility scenario.

The anomalies that we induced into the two data sets were:

- incompleteness: to simulate that monitors do not provide any data or that the connections between devices is down.
- incorrectness: to simulate that monitors provide incorrect data (because of faultiness of the monitors themselves or transmission faults).

The anomalies were systematically created, covering the following dimensions:

- extend or degree of incompleteness or incorrectness: We analysed different degrees of incompleteness and used incorrect data ranging from slightly out of normal ranges up to extreme and impossible values.
- variability of anomalies: the two types were both induced in a random ways as well as clustered on both data rows and features.

The example in Fig. 2 that shows blocks (clusters) of incorrect data for various features here using implausible negative values that could in an edge cloud setting indicate transmission or conversion errors.

These anomalies allow to map the anomaly types to systems faults, for instance. clustered incompleteness of rows can be associated with local network faults or time-clustered incorrectness can be associated with sensor faults. We created time series of anomalous model quality. These times series were then analysed with regard to possible change patterns regarding accuracy, precision and recall.

5.1.3 Results

We observed accuracy, precision and recall for the ML-generated model. Fig. 3 shows an example of a specific test case for increasing incorrect sensor values. Overall, more than 50 individual settings were explored. These graphs were then compared in terms of gradient, function type, and other criteria. Some key observations from the analysis of the ML model accuracy, precision and recall graphs are as follows:

- *Incorrectness is more significant than incompleteness.* The incorrectness has a bigger effect on the accuracy than the incompleteness. A possible reason here is that in incompleteness the machine learning an ML tool may ignore missing data rows or features and not include these in the predictions and calculations. However, for incorrectness a tool is forced to use all values, be them correct or not. Thus, it cannot control or minimize the negative impact on accuracy.
- *Rows in a data set are more significant than features.* Missing or invalid rows have a stronger impact on the accuracy than missing or invalid features in the data set tables. Here a number of are possible. A probable one is the fact that dealing with a complete missing or invalid row is more difficult than dealing with some missing or invalid features. Resolving the reduction of accuracy is consequently more difficult with missing/invalid rows than with missing/invalid features, resp.

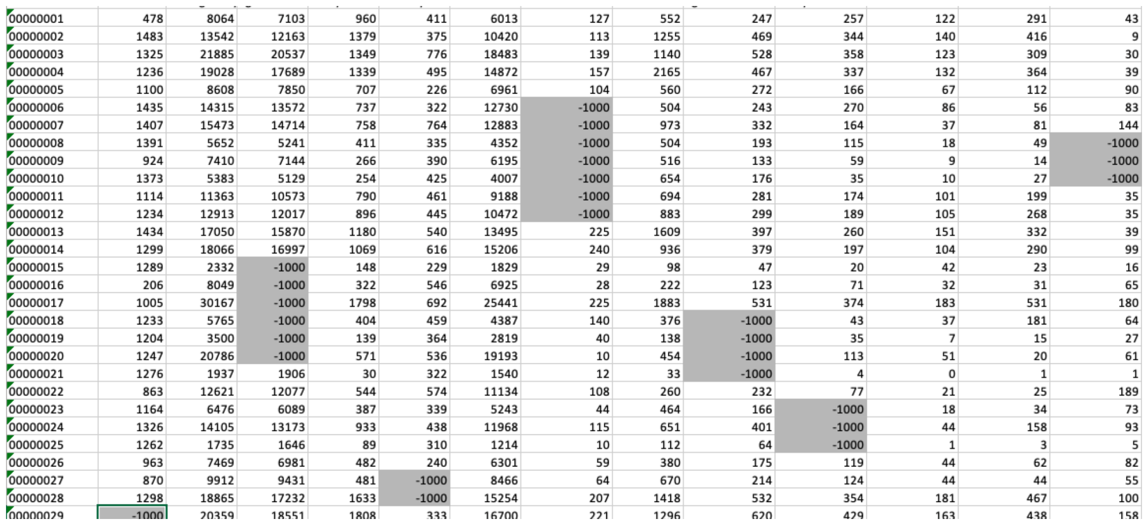
These observations represent identifiable change patterns, e.g., increasing incorrectness having a more significant (negative) impact on accuracy). As already mentioned, these changes can then be associated to root causes, e.g.,

- significant accuracy changes point to incorrectness cause most likely by sensor faults,
- faultiness of individual sensors has less impact then communication faults.

Using this kind of rule system, useful recommendations for remedial actions such as checking or replacing faulty sensors can be given. Thus, the experiments demonstrate that a quality controller is feasible and the reference architecture is valid.

6 CONCLUSIONS

Although our focus here is on edge cloud controllers as a typical example where automation is necessary and beneficial, the problem of quality control for ML-generated software is a broader one. Controllers can be effectively generated using machine learning from past monitoring and reaction data collected during dynamic resource management. While the benefits of automation and dynamic adaptation of application systems is clear, more work needs to be done on ongoing quality management for the ML-generated controllers themselves, using a meta-level feedback loop on top of the application management one. Managing AI (or more specifically ML) quality is a concern of AI governance and AI engineering that put more emphasis on understanding the AI systems in question and making quality management a more systematic engineering approach, respectively.



00000001	478	8064	7103	960	411	6013	127	552	247	257	122	291	43
00000002	1483	13542	12163	1379	375	10420	113	1255	469	344	140	416	9
00000003	1325	21885	20537	1349	776	18483	139	1140	528	358	123	309	30
00000004	1236	19028	17689	1339	495	14872	157	2165	467	337	132	364	39
00000005	1100	8608	7850	707	226	6961	104	560	272	166	67	112	90
00000006	1435	14315	13572	737	322	12730	-1000	504	243	270	86	56	83
00000007	1407	15473	14714	758	764	12883	-1000	973	332	164	37	81	144
00000008	1391	5652	5241	411	335	4352	-1000	504	193	115	18	49	-1000
00000009	924	7410	7144	266	390	6195	-1000	516	133	59	9	14	-1000
00000010	1373	5383	5129	254	425	4007	-1000	654	176	35	10	27	-1000
00000011	1114	11363	10573	790	461	9188	-1000	694	281	174	101	199	35
00000012	1234	12913	12017	896	445	10472	-1000	883	299	189	105	268	35
00000013	1434	17050	15870	1180	540	13495	225	1609	397	260	151	332	39
00000014	1299	18066	16997	1069	616	15206	240	936	379	197	104	290	99
00000015	1289	2332	-1000	148	229	1829	29	98	47	20	42	23	16
00000016	206	8049	-1000	322	546	6925	28	222	123	71	32	31	65
00000017	1005	30167	-1000	1798	692	25441	225	1883	531	374	183	531	180
00000018	1233	5765	-1000	404	459	4387	140	376	-1000	43	37	181	64
00000019	1204	3500	-1000	139	364	2819	40	138	-1000	35	7	15	27
00000020	1247	20786	-1000	571	536	19193	10	454	-1000	113	51	20	61
00000021	1276	1937	1906	30	322	1540	12	33	-1000	4	0	1	1
00000022	863	12621	12077	544	574	11134	108	260	232	77	21	25	189
00000023	1164	6476	6089	387	339	5243	44	464	166	-1000	18	34	73
00000024	1326	14105	13173	933	438	11968	115	651	401	-1000	44	158	93
00000025	1262	1735	1646	89	310	1214	10	112	64	-1000	1	3	5
00000026	963	7469	6981	482	240	6301	59	380	175	119	44	62	82
00000027	870	9912	9431	481	-1000	8466	64	670	214	124	44	44	55
00000028	1298	18865	17232	1633	-1000	15254	207	1418	532	354	181	467	100
00000029	-1000	20359	18551	1808	333	16700	221	1296	620	429	163	438	158

Figure 2: Sample anomaly-induced data set: clustered blocks of invalid features (implausible values).

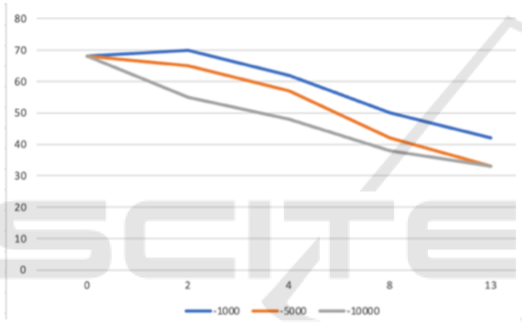


Figure 3: Observation of ML quality metrics (accuracy) for different incorrect (different degrees of incorrectness showing more impact for more unexpected/implausible values).

For our edge cloud setting, we have proposed a reference governance architecture here that embodies the two feedback loops in a layered setting. A common problem that applies here is that the quality of the controller is observable to users, but the root causes that might negatively impact on that quality are not under the control of the user or are not easily visible.

Based on experiments in a simulated setting, we have demonstrated that the proposed governance architecture is feasible, i.e., the controller quality can be monitored and an analysis of usually hidden root causes of root causes is possible, also allowing beneficial remedial actions to be proposed.

This paper reports on the first stage of the overall ambition, i.e., the validity of the reference model was demonstrated through experiments. In a second stage as part of the future work, we aim to fully implemented the layered feedback loop and validate the initial results with concrete implementations and real anomalous data from that system. Nonetheless, since the simulated experiments conducted here are based on induced anomalies that cover all common fault and anomaly situations in this architectural setting, we can be confident about the validity of our results.

REFERENCES

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software engineering for machine learning: A case study. In *Intl Conf on Software Engineering - Software Engineering in Practice track*. IEEE.

Arabnejad, H., Pahl, C., Jamshidi, P., and Estrada, G. (2017). A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In *International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017*. IEEE Computer Society / ACM.

Azimi, S. and Pahl, C. (2020a). A layered quality framework in machine learning driven data and information models. In *22nd International Conference on Enterprise Information Systems*.

Azimi, S. and Pahl, C. (2020b). Particle swarm optimization for performance management in multi-cluster iot edge architectures. In *International Conference on Cloud Computing and Services Science CLOSER*.

Azimi, S. and Pahl, C. (2021). The effect of iot data completeness and correctness on explainable machine learning models. In *Database and Expert Systems Applications*. Springer.

Barzegar, H. R., Ioini, N. E., Le, V. T., and Pahl, C. (2021). 5g-carmen: Service continuity in 5g-enabled edge clouds. In *Advances in Service-Oriented and Cloud Computing*. Springer.

Barzegar, H. R., Le, V. T., Ioini, N. E., and Pahl, C. (2020). Service continuity for ccam platform in 5g-carmen. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1764–1769.

De Hoog, J., Mercelis, S., and Hellinckx, P. (2019). Improving machine learning-based decision-making through inclusion of data quality. *CEUR Workshop Proceedings*, 2491.

Deja, K. (2019). Using machine learning techniques for data quality monitoring in cms and alice. *Proceedings of Science*, 350.

Efron, B. (2020). Prediction, estimation, and attribution. *Journal of the American Statistical Association*, 115(530):636–655.

Ehrlinger, L., Haunschmid, V., Palazzini, D., and Lettner, C. (2019). A daql to monitor data quality in machine learning applications. In *Database and Expert Systems Applications*, pages 227–237.

El Ioini, N., Hästbacka, D., Pahl, C., and Taibi, D. (2021). Platforms for serverless at the edge: A review. In *Advances in Service-Oriented and Cloud Computing*. Springer.

Fang, D., Liu, X., Romdhani, I., Jamshidi, P., and Pahl, C. (2016). An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation. *Future Gener. Comput. Syst.*, 56:11–26.

Femminella, M. and Reali, G. (2019). Gossip-based monitoring of virtualized resources in 5g networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019*, pages 378–384. IEEE.

Gand, F., Fronza, I., Ioini, N. E., Barzegar, H. R., Azimi, S., and Pahl, C. (2020). A fuzzy controller for self-adaptive lightweight container orchestration. In *International Conference on Cloud Computing and Services Science CLOSER*.

Hong, C. and Varghese, B. (2019). Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv.*, 52(5):97:1–97:37.

Jamshidi, P., Sharifloo, A. M., Pahl, C., Metzger, A., and Estrada, G. (2015). Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution. In *2015 International Conference on Cloud and Autonomic Computing*, pages 208–211. IEEE Computer Society.

Javed, A., Malhi, A., and Främling, K. (2020). Edge computing-based fault-tolerant framework: A case study on vehicular networks. In *Intl Wireless Communications and Mobile Computing Conference, IWCMC 2020*. IEEE.

Karkouch, A., Mousannif, H., Moatassime, H. A., and Noel, T. (2016). Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications*, 73:57 – 81.

Lwakatare, L. E., Raj, A., Bosch, J., Olsson, H. H., and Crnkovic, I. (2019). A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. In *Agile Processes in Software Engineering and Extreme Programming - 20th International Conference*. Springer.

Mantri, D., Prasad, N. R., and Prasad, R. (2013). BHCDA: bandwidth efficient heterogeneity aware cluster based data aggregation for wireless sensor network. In *Intl Confe on Advances in Computing, Communications and Informatics*. IEEE.

Mendonca, N. C., Jamshidi, P., Garlan, D., and Pahl, C. (2021). Developing self-adaptive microservice systems: Challenges and directions. *IEEE Software*, 38(2):70–79.

Mohammedameen, I., Mkwawa, I., and Sun, L. (2019). Follow-me prefetching for video streaming over mobile edge computing networks. In *2019 IEEE SmartWorld*, pages 1937–1942. IEEE.

Pahl, C. and Azimi, S. (2021). Constructing dependable data-driven software with machine learning. *IEEE Softw.*, 38(6):88–97.

Pahl, C., Jamshidi, P., and Zimmermann, O. (2018). Architectural principles for cloud software. *ACM Transactions on Internet Technology (TOIT)*, 18(2):17.

Pomalo, M., Le, V. T., Ioini, N. E., Pahl, C., and Barzegar, H. R. (2020). Service migration in multi-domain cellular networks based on machine learning approaches. In *Intl Conf on Internet of Things: Systems, Management and Security, IOTSMS 2020*. IEEE.

Scolati, R., Fronza, I., Ioini, N. E., Samir, A., and Pahl, C. (2019). A containerized big data streaming architecture for edge cloud computing on clustered single-board devices. In *Intl Conf on Cloud Computing and Services Science*.

Sheng, V. S., Provost, F. J., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *14th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622. ACM.

Sicari, S., Rizzardi, A., Miorandi, D., Cappiello, C., and Coen-Porisini, A. (2016). A secure and quality-aware prototypical architecture for the internet of things. *Information Systems*, 58:43 – 55.

Sridhar, V., Subramanian, S., Arteaga, D., Sundararaman, S., Roselli, D. S., and Talagala, N. (2018). Model governance: Reducing the anarchy of production ml. In *USENIX Annual Technical Conference*.

Taibi, D., Lenarduzzi, V., and Pahl, C. (2020). *Microservices Anti-patterns: A Taxonomy*. Springer.

Tokunaga, K., Kawamura, K., and Takaya, N. (2016). High-speed uploading architecture using distributed edge servers on multi-rat heterogeneous networks. In *IEEE International Symposium on Local and Metropolitan Area Networks, LANMAN 2016*, pages 1–2. IEEE.

von Leon, D., Miori, L., Sanin, J., Ioini, N. E., Helmer, S., and Pahl, C. (2019). A lightweight container middleware for edge cloud architectures. In *Fog and Edge Computing*, pages 145–170. Wiley.

Wang, F., Zhang, M., Wang, X., Ma, X., and Liu, J. (2020). Deep learning for edge computing applications: A state-of-the-art survey. *IEEE Access*, 8:58322–58336.

Zhao, H., Yi, D., Zhang, M., Wang, Q., Xinyue, S., and Zhu, H. (2019). Multipath transmission workload balancing optimization scheme based on mobile edge computing in vehicular heterogeneous network. *IEEE Access*, 7:116047–116055.