IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 June 2022, accepted 19 June 2022, date of publication 24 June 2022, date of current version 1 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3185990

A Data Protection Focused Adaptation Engine for Distributed Video Analytics Pipelines

CLEMENS LACHNER[1], (Member, IEEE), JAN LAUFER[2], SCHAHRAM DUSTDAR[1], (Fellow, IEEE), AND KLAUS POHL[2], (Member, IEEE)

[1]Distributed Systems Group, TU Wien, 1040 Vienna, Austria
[2]paluno—The Ruhr Institute for Software Technology, University of Duisburg-Essen, 45141 Essen, Germany

Corresponding author: Clemens Lachner (c.lachner@dsg.tuwien.ac.at)

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (FogProtect) under Grant 871525. The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

ABSTRACT The design, development, deployment, and operation of a distributed Video Analytics Pipeline (VAP) at the edge of the network is highly complex. In the domain of adaptive systems, several solutions are proposed in literature to optimize either one particular performance aspect of a VAP, e.g., execution time or latency, or focus on minimal energy consumption, or calculate a trade-off including some of those aspects. However, nowadays, most systems utilizing a VAP that records personally identifiable data have to adhere to some form of data protection regulation, such as the GDPR. Still, adaptations to increase data protection requirements are often second to previously mentioned performance or energy consumption characteristics of a VAP. While there is state of the art literature dealing with data protection related adaptations, most of them solely focus on increasing certain security or privacy aspects of a system, leaving previously mentioned performance or energy consumption characteristics out of scope. To the best of our knowledge, there is no solution that covers all of these aspects. In this paper, we present a data protection focused adaptation engine that leverages the application- and infrastructure based adaptation space of a distributed VAP. The engine employs an extended system model and adaptation rules that are based on previous research. It features an optimization algorithm to improve data protection, performance and energy consumption characteristics of a distributed VAP.

INDEX TERMS Adaptive systems, optimization, video analytics pipelines, data protection, edge computing.

I. INTRODUCTION

Edge Computing comprises a variety of different connected devices with minimal to average computing power. These devices continue to permeate deeper into our personal environment as well as in commercial and industrial areas, by sensing, processing, and storing all kind of data [1]. The design, development, and deployment of distributed edge applications, e.g., AI-assisted Video Analytics Pipelines (VAP), is highly complex [2]. Due to the heterogeneous hardware environment of such systems, concrete specifications of various infrastructural parts may not be fully known at design-time of a distributed application. Furthermore, various application parts (e.g., microservices) will be executed on different infrastructures and could be migrated during run-time. While the cloud is to be expected, mainly due to virtualization and containerization techniques, to have virtually unlimited computing power and storage, the capabilities of edge nodes may vary widely. This heterogeneity of capabilities increases even more in the area of IoT, where energy consumption becomes an additional important factor. Hence, not knowing the exact infrastructural details on where a distributed application will be deployed and executed on becomes a tremendous challenge [1], [3], [4]. Additionally, a VAP may use many different software components to fulfill its concrete task. Those software components may also be distributed across different nodes that may be maintained by different parties. Hence, software characteristics, such as efficiency, reliability or functionality, could also vary greatly. Another problem a VAP may likely face, is the handling of sensitive data, such as personally identifiable information (PII), in order to allow its workflow to comply to privacy policies [5], [6] or security provisions such as the General Data Protection Regulation (GDPR) of the EU. For such

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

applications, these circumstances call for data protection [7] to be the topmost priority, while performance and energy consumption come second. Integrity, confidentiality, availability, undetectability, and unobserveability are the key elements of such protection mechanisms. The overall risk that such a system is confronted with could be assessed by looking on its various attack surfaces that are exposed to an adversary. For example, in the context of AI-assisted video analytics, the potential leakage of PII (e.g., recorded faces) from video data, e.g., due to unencrypted transmission of images (frames), is a characteristic threat to data protection of such a system [8]. However, implementing adequate mechanisms to fulfill data protection requirements is often second to improvements for performance characteristics of a system, especially in the area of Edge Computing and IoT [9], [10]. The concept of adaptation aims to support tackling this challenge. Based on the classical definition of control theory an adaptive system monitors its own performance and adjusts its parameters in the direction of better performance [11]. Computing time, data storage or latency are concrete exemplary manifestations of this classical understanding of performance. For defining adaptation rules, it is crucial to understand (i) what changes in the environment may happen, (ii) what self-adaptations the system may perform, and (iii) how those changes and self-adaptations impact the relevant system properties. Hence, adaptations in the system have to be designed in a way so that this heterogeneous infrastructural and software environment is also taken into account. Regarding performance related adaptations, several solutions are proposed in literature to optimize either one particular performance aspect of a VAP, e.g., execution time or latency, or focus on minimal energy consumption, or calculate a trade-off including some of those aspects [12]–[15]. While there is state of the art literature dealing with data protection related adaptations, most of them solely focus on increasing certain security or privacy aspects of a system, leaving previously mentioned performance characteristics out of scope [16]–[18]. To the best of our knowledge, there is no solution that covers all of these aspects. In this paper, we present a data protection focused adaptation engine that leverages the application- and infrastructure based adaptation space of a distributed VAP. The engine employs an extended system model and adaptation rules that are based on previous research which was conducted within a collaborative H2020 project [19], namely FogProtect.[1] The motivating use case also stems from a real problem scenario by one of the projects partners. The model was specifically extended and enhanced to meet the requirements of AI-assisted VAPs at the Edge. The adaptation rules cover the application layer, as well as the infrastructure layer of a VAP system. Furthermore, the engine now features an optimization algorithm to improve performance, energy consumption and data protection of a distributed VAP and its functionalities. Based on a real-world use case, we can show that our approach efficiently and effectively mitigates data protection risks in a running system. Additionally, it opts to find the most suitable system configuration based on an operators preferences regarding performance, energy consumption, and available functionalities (QoS). The paper is structured as follows: Research related to this paper is discussed in Section II. In Section III, we state the problems and challenges a distributed VAP faces with respect to data protection, performance and energy consumption. Furthermore, we describe a real world use case, that is derived from a use case from FogProtect, motivating our approach and illustrating the associated challenges to data protection. Section IV gives the essential background information that is needed to understand our approach. Details on the implementation of the engine are described in Section V. In Section VI we evaluate the feasibility and applicability of our approach. Section VII discusses our findings and concludes the paper.

[1] https://fogprotect.eu/

II. RELATED WORK

Broadly speaking, the mitigation of threats to data protection of a system is a two-dimensional problem. First, a mitigation concept comprises either design-time activities or run-time activities or a combination of both. Second, it is typically specific to the system domain and underlying infrastructure. Research dealing with risk analysis, e.g., [20]–[23], as a design time activity is related to our work, i.e., it would be needed to identify and model Problematic Configuration Patterns (PCPs) and how to resolve them (see section IV for details on PCPs) as suggested by our approach. Our approach is agnostic to a specific risk analysis approach, therefore it could be seen as complimentary research. However, research that aims to mitigate data protection threats via run-time adaptations is closer related to our approach.

A generic approach was proposed by Bürger *et al.*, which handles *Essential Security Requirements*. This is achieved by triggering *Security Maintenance Rules* if changes in the *Security Context Knowledge* are detected [16], [17]. Another generic mechanism by Tsigkanos *et al.* describes a model checking approach to analyze threats and plan adaptations to mitigate those [18]. They use bigraphs to model the topology and security requirements of the system. Their approach takes a lot of execution time, while our approach builds upon previous research [24], which we could show is well performing even with larger system models. Additionally, both of the above mentioned proposed adaptation mechanisms focus solely on the mitigation of data protection threats, but do not take other parameters into account, such as QoS, performance or energy consumption. We identified several other papers related to our work that take such parameters into account. [25]–[27] take into account various types of costs, response time and/or performance. However, they mostly focus on specific mitigation strategies and/or approaches dedicated to specific attack vectors. For example, Nostro *et al.* solely deal with attacks performed by insiders and how to prevent those [28]. Nguyen *et al.* proposed a mitigation strategy that only involves migration of virtual machines [29]. Our proposed solution is able to process arbitrarily complex adaptations that

are optimized to also provide the desired QoS, performance and energy consumption.

In the context of privacy in video-based media spaces, Boyle *et al.* [30] proposed a framework – a descriptive theory – that defines how one can think of privacy while analyzing media spaces and their expected or actual use. The framework explains three normative controls: solitude, confidentiality and autonomy, yielding a vocabulary related to the subtle meaning of *privacy*. A more technical introduction to video surveillance is given by Senior in [31]. The paper briefly summarizes the elements in an automatic video surveillance system, including architectures, followed by the steps in video analysis, from preprocessing to object detection, tracking, classification and behaviour analysis. In [32], the authors introduce a video analytics framework to process real-time video streams and also do batch video analytics. However, they focus on the performance aspect of a distributed VAP, in terms of scalability, effectiveness and fault-tolerance, leaving data protection out of scope. Sada *et al.* proposed an edge computing based video analytics architecture [33]. Their solution leverages federated learning strategies in order to reduce the computational load of cloud infrastructure. Additionally, their training data remains on the edge servers, thereby increasing privacy. In contrast to our proposed solution, their approach incorporates federated learning, while we focus on local execution of AI-based tasks and protecting data agnostic to specific security and privacy mechanisms. Furthermore, we focus on the heterogeneity aspect in edge computing based VAPs, and the adaptation aspect of the system, hence increasing the flexibility and applicability of our approach.

Our proposed solution is specifically tailored to meet the requirements of a self-adaptive data protection aware Video Analytics Pipeline. Hence, we also identified research in the domain of self-adaptive (distributed) AI-assisted VAPs. As stated in section III, the inference tasks of an AI-assisted VAP is computationally expensive. Hence, as well as in other domains than VAPs, many authors propose offloading those computationally expensive tasks to the cloud. However, executing inference in the cloud, especially for real-time video analysis, often incurs high bandwidth consumption, high latency, reliability issues, and privacy concerns. Therefore, many researchers follow the edge computing paradigm, i.e., processing data closer to the data source. For example Liu *et al.* proposed EdgeEye, which enables developers to transform models trained with popular deep learning frameworks to deployable components with minimal effort [34]. In [35], they introduced a controller that dynamically picks the best configurations for existing Neural-Network-based Video Analytics Pipeline. Their aim is to achieve higher accuracy with the same amount of resources, or achieve the same accuracy but utilizing less of the available resources. Zhang *et al.* propose a flexible serverless-based approach to facilitate fine-grained and adaptive partitioning of cloud-edge workloads for multiple concurrent video query pipelines. Their goal is to achieve real-time responses given a highly dynamic input workload. In contrast to our approach, those papers focus solely on improving one specific performance aspect (either accuracy or latency) of an AI-assisted VAP.

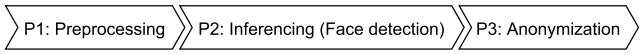

FIGURE 1. Parts of an exemplary face-anonymization application.

To summarize, to the best of our knowledge there is no approach in literature that allows for data protection focused run-time adaptations in the domain of AI-assisted Video Analytics Pipelines, that also takes performance, QoS and energy consumption into account.

III. PROBLEM STATEMENT

In this section we first give a broad overview on AI-assisted VAPs in general. Then, we describe the challenges for data protection and performance for each major tasks a VAP comprises. The process of AI-assisted video data analysis is typically composed of three major tasks. First, the video data has to be prepared for an AI-application to be able to work with such data. Second, one or more features have to be detected in a video frame (basically an image). Third, post-processing actions take place. A real-world application for a data protection-centered VAP is face-anonymization, i.e,. detected faces on each frame of the input video have to be made unrecognizable in each frame of the output video. Fig. 1 illustrates these three steps (P1-P3) based on the face-anonymization example.

Performance in context of a VAP is not only related to computational capabilities of the system. It also covers qualitative aspects of the analysis part, i.e., the accuracy with which desired features can be inferred from video data. Accuracy is typically determined by the used AI-model and respective framework. The computational performance of each task is tightly coupled to the hardware capabilities of the device executing those tasks. Running all of the three major tasks on a single computing device can have a significant impact on the overall performance of a VAP. If performance on a single device becomes an issue, a common approach is to decouple the three main tasks of the face-blurring pipeline and execute them separately on different devices. However, this pragmatic approach may often pose a non-trivial challenge to edge computing usecases, considering distributed applications running in a heterogeneous hardware and software environment. Therefore, in [36], we got a better understanding of the main drivers decreasing performance in each of the three major tasks of a VAP, in order to develop adaptation strategies to enhance performance and/or data protection quality. Hence, the approach presented in this paper builds on top of that, enabling us to incorporate the important aspects into the proposed solution.

A. MOTIVATING EXAMPLE

To better illustrate the problems and challenges of a VAP we describe a real world use case from a FogProtect partner,

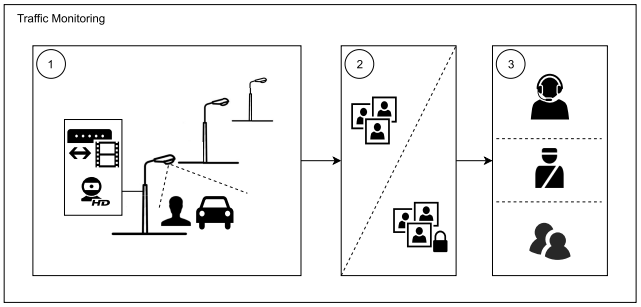

FIGURE 2. Smart lampposts recording video data that potentially contains PII. Depending on a users access rights, either anonymized or raw video is transmitted.

motivating our approach. The use case takes place in the smart cities domain and featuring a modern urban monitoring system.[2] In general, the use case scenario can be considered as a multi-tenant systems with a highly dynamic device setup. Smart lampposts, equipped with computing nodes and various sensors, are installed and distributed across the city. The heterogeneous capabilities between nodes stem from the fact that (in our specific real word use case) in an urban environment smart lamppost can be manufactured, equipped and mounted by different vendors. These smart lampposts sense and process data and share this data across the network. Data may be transferred from one lamppost to another node or directly to the cloud for further processing and/or storage. Traffic monitoring is one concrete exemplary scenario in our use case. A smart lamppost records or streams video data from a mounted camera to a computing node, where the data is further processed, streamed or stored. Various parties may access this data for further analysis. For example, this enables traffic controllers to identify and analyze potential incidents and problems regarding traffic, such as a traffic jam or a car accident, which then inform potential first responders to act accordingly. Imagine a smart lamppost records a car incident and the data gets transmitted to a traffic controller. Such data flow, as well as its content, is prone to many security and privacy threats, such as wiretapping and personal data leakage. It may not be beneficial or desired for a traffic controller to see potentially recorded faces of people nearby the accident or even licence plates of cars involved in the accident or also nearby. However, a first responder, like law enforcement, may be interested to actually identify people (as potential witnesses) or licence plates of cars (in cases of e.g., hit and run scenario). Hence, depending on e.g., access rights, specific adaptations on nodes have to be triggered. A concrete example of such an adaptation is to activate an object blurring algorithm manipulating a recorded video stream. Fig. 2 shows an overview of the scenario.

The box with label 1 displays smart lampposts, distributed across the city, each equipped with a camera and a computing node that acts as a video gateway. A lamppost records it's environment, e.g., people and cars on the street, and transmits the video to another computing node, where it can be further processed, or viewed by a user. However, as depicted in the box with label 2, depending on the user's access rights, the video is either anonymized or not upon request. The box with label 3 displays exemplary roles of users, a traffic operator, a law enforcement officer and a normal end user, each with different access rights. While the law enforcement officer would have access to the raw video, the other two users would only be allowed to watch the anonymized version of the video.

B. CHALLENGES TO DATA PROTECTION AND PERFORMANCE

From a high level perspective, an AI-assisted VAP faces similar data protection challenges as any other distributed application that stores, processes and transmits sensitive data. Sticking to the information security principle of the Parkerian Hexad [37], such systems have to be protected against security breaches affecting one or more of the fundamental attributes of information, namely: Confidentiality, Possession or Control, Integrity, Authenticity, Availability and Utility. Identifying the risks associated with those security breaches and how to mitigate them is typically a design-time activity carried out by information security experts following standards like the ISO27000 [38]. A concrete implementation of the scenario described in Section III-A makes heavy usage of AI-based tasks in order to provide additional services like automatic licence plate detection, but also enable or enhance data protection mechanisms. A simple, yet not easy task to ensure and respect the privacy of people recorded by a system like we described, is the anonymization of personally identifiable features like faces of people or licence plates of cars. Furthermore, the actual physical location where (personal) data is processed plays an important role in regulations like e.g., the GDPR in Europe. Additionally, a distributed VAP needs to take care of secure data transmission as well. A secure data transmission may not only cover encrypted communication, but could also leverage cutting edge AI-techniques like model splitting to further enhance privacy. Basically, each phase of the VAP faces specific challenges to either performance and/or data protection. In the first phase (P1), the system is typically concerned with video pre-processing tasks, such as encoding or up/downsampling. Such tasks are commonly done if the camera records with different parameters (e.g., framerate or resolution) than the desired output video, i.e., the video data that will be transmitted to, and analyzed by, a component of P2. However, transforming video data to a specific format is a heavy computational task, hence significantly affecting the performance of P1. Therefore, manufactures like Nvidia integrate dedicated chips into their hardware and offer dedicated SDKs to facilitate this task.[3] The performance of the second task of the pipeline is heavily dependent on the used AI-framework as well as on the underlying infrastructure it operates on. In the context of AI-based inference

[2] See:https://urbanplatform.city/

[3] https://developer.nvidia.com/nvidia-video-codec-sdk

tasks, like stated previously, performance is not only related to processing speed but also to the accuracy with that e.g., an object was detected in an image. A well pre-trained model embedded in modern sophisticated frameworks like e.g., *TensorFlow* or *YOLO* will generally allow for higher inference speed with high accuracy. However, such frameworks are not necessarily available and/or optimized for each system architecture like e.g., ARM, x86, x64, but developers constantly aim to port a framework to another architecture or provide lightweight alternatives such as *TensorFlow Lite* or *YOLOv3-tiny*. Furthermore, differences regarding performance can also stem from the usage of legacy versions of such machine learning frameworks. Executing this second major task of the pipeline on devices featuring hardware like a GPU or even dedicated AI-hardware like Tensor Processing Units (TPU, Google) or Neural Processing Units (NPU, Microsoft) do also heavily contribute to performance gains. This stems from the fact that inference tasks are parallelizable to a high degree and TPUs or NPUs are custom ASIC chip-designed from the ground up for machine learning workloads. The aforementioned heterogeneity of the underlying hardware infrastructure makes deployment, migration or offloading inference based tasks or components increasingly complex. The third task involves potentially multiple post-processing activities. Regarding the example of face-anonymization, this would be the anonymization process of recognized face in a video frame, i.e., a graphic overlay is drawn over the face in the image. Although it is well researched that making a face unrecognizable in visual data does not fully guarantee the anonymity of a person, it still facilitates such anonymization task to a high degree, mostly by making the de-anonymization process significantly harder for an attacker. However, other typical post-processing steps could include again transcoding or encoding tasks, sampling rate conversions, resolution alterations, etc. to e.g., facilitate and optimize for streaming the output video.

Another important aspect of a VAP is Quality of Service (QoS). Due to hardware or software constraints of the node executing (parts of) a VAP, the processing of data protection mechanisms of the application may not keep up with a reasonable output video quality, i.e., the frames per second (FPS) of the output video are far too low compared to the input video. Hence, a person viewing the output video would experience a significant loss in QoS using the application.

IV. BACKGROUND INFORMATION

In this section we provide background information that is needed to understand the approach and additionally builds the foundation of our work. First, we will give a short description on the term data protection from a legal perspective according to the GDPR. Our approach is not limited to that kind of data only (i.e., personal data, as explained later), but is rather capable of operating with any kind of sensitive data. Second, we give a brief overview on previous work, i.e., the main components from RADAR [24], which we extended, updated and enhanced for our approach. Third, we explain what kind of adaptations are possible within our approach.

A. DATA PROTECTION

The term data protection is not very well defined from a global perspective. It is related to information security but not the same. The ISO 27000:2018 standard describes information security as the "preservation of confidentiality, integrity and availability of information" [38]. Furthermore, it states that other information security aspects, such as authenticity, accountability, non-repudiation, and reliability, should also be considered. According to the GDPR, data protection mechanisms should prevent personal data breaches [39]. Personal data means "any information relating to an identified or identifiable natural person," e.g., name, address or location data. A personal data breach means "a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed." Hence, data protection includes information security aspects such as confidentiality or integrity concerning personal data. However, data protection goes beyond information security, for example by defining specific roles as well as their rights and obligations related to personal data. Our approach is able to handle the following roles according to the GDPR:

- Data Subject: A data subject is "an identified or identifiable natural person ... who can be identified ... by reference to an identifier such as a name ... or to one or more factors specific to the ... identity of that natural person."
- Data Controller: A data controller "determines the purpose and means of the processing of personal data."
- Data Processor: A data processor "processes personal data on behalf of the data controller."
- Third Party: A third party is authorised to process personal data under the direct authority of the controller or processor.

From a legal point of view, data protection must be ensured in an edge computing system processing personal data. Therefore, several information security aspects need to be taken into account when operating a system that processes sensitive data, hence adhere to the GDPR. In order to model and operate such systems with RADAR, we decided to enrich the basic approach with information security aspects from the Parkerian Hexad [37]. The Parkerian Hexad adds three additional attributes to the traditional security attributes of the CIA triad (confidentiality, integrity, availability).

This enables our updated approach to cover the following aspects of information security:

- Confidentiality
- Possession or Control
- Integrity
- Authenticity
- Availability
- Utility

B. FOUNDATION OF OUR APPROACH

This work is based on the so called RADAR (Run-time Adaptations for DAta pRotection) approach presented in [24]. RADAR aims at ensuring data protection in dynamically changing cloud-based systems and other related system concepts such as edge or fog computing. RADAR is deployed as a central control unit that manages the self-adaptive system. It is assumed that both monitoring of the system and execution of adaptations in the system are carried out by the system itself or components between the system and RADAR. The Eclipse Modelling Framework (EMF)[4] combined with Henshin[5] is used to enable model-based runtime adaptations, specifically optimized towards data protection. As shown in Fig. 3, RADAR consists of multiple components that are described now.

1) META-MODEL

RADAR includes a *meta-model* that defines the modelling constructs that can appear in the *run-time model*, a language to specify *problematic configuration patterns* (PCPs), and *adaptation rules* to mitigate problematic configurations. Thus, the meta-model ensures compatibility between the problematic configuration patterns, adaptation rules, and the run-time model. Previous research has shown that using established modelling languages from the area of security, such as UMLsec or SysML-Sec, is not sufficient to model neither edge computing systems nor data protection aspects related to edge computing systems [23]. Therefore, an independent framework is needed.

The meta-model is created by using the Eclipse Modelling Framework (EMF) and its graphical editor. In general, the meta-model is similar to UML class diagrams. Nodes are represented as classes, edges are represented as relations between classes. Properties of nodes are represented as attributes and object-oriented concepts like inheritance are supported. The meta-model extends the well established TOSCA standard, a modeling language defined by the Organization for the Advancement of Structured Information Standards (OASIS).[6] In the RADAR meta-model nodes can be any kind of entity of a system, like a compute node or a GDPR role as described in section IV-A. From a high level perspective, a compute node typically comprises various software components. These components are also modeled as nodes. Which hardware and software components belong to a certain type of compute node is modeled via relations.

2) PROBLEMATIC CONFIGURATION PATTERNS

A *Problematic Configuration Pattern* (PCP) describes a pattern that exists in the *run-time model* if the system is in a problematic state. A run-time model represents relevant aspects of the system and its environment in the style of an UML object model. It is fully based on the meta-model and gets continuously updated by a system monitoring component at run-time. PCPs are defined at design-time and specifically focus on configurations that threaten data protection or QoS aspects of the system or lead to high overall costs. By cost, we mean, for example, the cost of cloud rental that IaaS providers incur. At run-time, instances of PCPs can be detected by using RADARs pattern-based algorithms. The goal is to avoid as many PCP instances inside of the run-time model as possible. To design PCPs a graphical modeling language, called PCP Language, was created. Similar to the run-time model it is in the style of UML object diagrams. To create PCPs the PCP Language is mapped to Henshin transformation rules. Henshin also enables graphical modelling by providing a GUI tool.

3) ADAPTATIONS

Whenever the run-time model changes, an algorithm is checking whether *instances of PCPs* can be found in that model (see *problematic configuration identification* in Fig. 3). If this is the case, RADAR uses *adaptation rules* to identify possible solutions to mitigate the PCP instance (see *Reconfiguration* in Fig. 3). Adaptations rules are created at design time. They are associated with the PCP that they should mitigate. For each PCP multiple adaptations can exist. Each adaptation rule captures a potential type of adaptation in a well-formed manner. An adaptation involves adding or removing objects and relations as well as changing attribute values according to the specific rule. Each adaptation rule is split into three parts, namely (i) a PCP and its instance that have to exist in the run-time model, (ii) a precondition that adds further constraints on the run-time model, and (iii) the adaptation action that describes changes to the run-time model. A graphical Adaptation Language was defined to design adaptation rules and was mapped to Henshin transformation rules as well.

If RADAR cannot find any PCP instances, it tries to detect improvement / optimization adaptations that increase the amount working functions (see *Functionality analysis* in Fig. 3) and lower the overall costs (see *Cost analysis* in Fig. 3) without creating new threats to data protection. After RADAR has found the optimal adaptation (in the best case all PCPs are mitigated, the amount of available functions is at its highest and the overall costs at their lowest) the adaptation will be executed on the system. However, a predefined prioritization defines whether the amount of available functions or the overall costs are more important.

It has to be noted that there may be multiple PCP instances in the run-time model, there may be multiple adaptations to mitigate a given PCP instance, and an adaptation to mitigate a PCP instance may also mitigate or create other PCP instances. Hence, a sequence of adaptations may be needed to mitigate all PCP instances. In this sequence, the order of adaptations may be important because an adaptation may become applicable only after another adaptation was carried out.

In [36] we investigated the adaptation space of AI-Assisted data protection for resource constrained VAPs. These findings build the foundation of the modelled adaptions designed,

[4]https://www.eclipse.org/modeling/emf/
[5]https://www.eclipse.org/henshin/
[6]https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

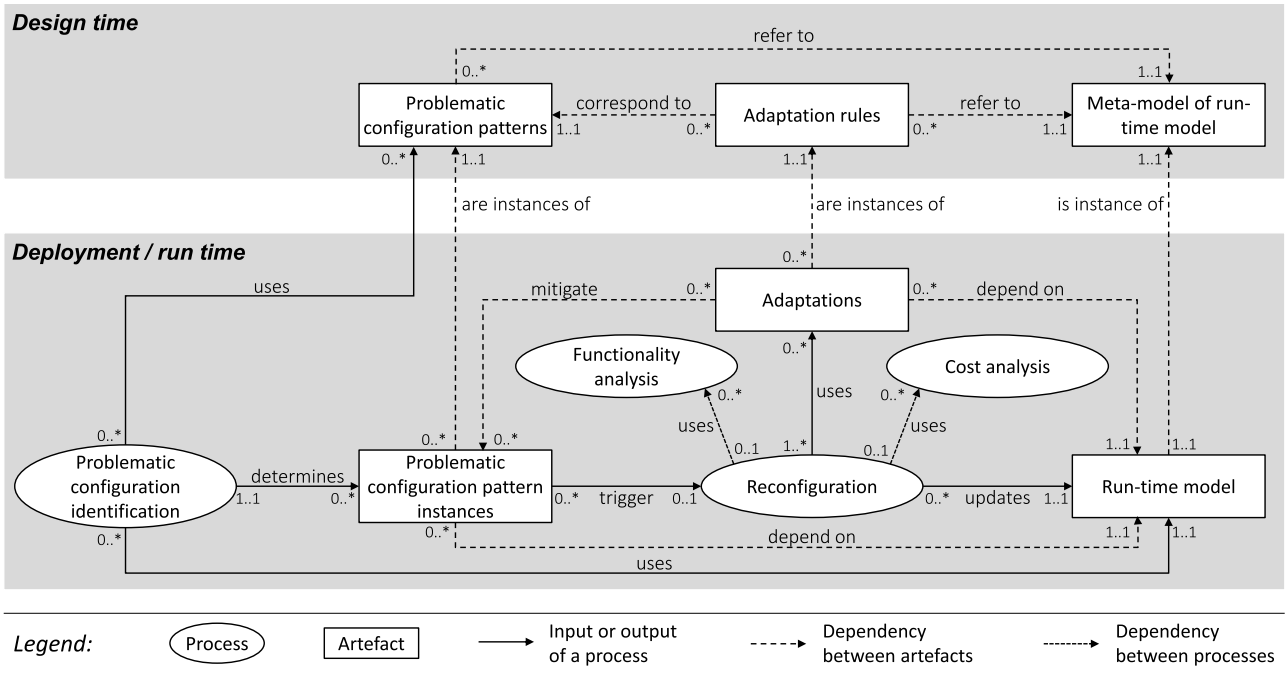

FIGURE 3. Overview of the RADAR approach.

implemented and evaluated in this paper and described in the following sections.

V. ADAPTATION ENGINE EXTENSIONS

In this section, we describe the basic functionality of our proposed data protection focused adaptation engine. Furthermore, we explain how the underlying concepts of previous research were improved in order to tackle the challenges faced by an edge computing based distributed VAP, as described in section III.

The meta-model, as described in section IV, was neither able to tackle detailed infrastructural aspects of an edge computing based system, nor was it capable of modeling fine-granular security and privacy controls that are vital for many systems, e.g., Video Analytics Pipelines, that need to adhere to some sort of data protection regulation such as the GDPR. Moreover, it was not possible to consider additional metrics like energy consumption or performance of a system while finding the optimal adaptation. In addition, it was also not possible to change the order of the prioritization of such metrics while solving data protection concerns in the first place. Therefore, we made several extensions and enhancements to our previous work.

A. CONCEPTUAL EXTENSION

We started by developing conceptual extensions and enhancements of the RADAR meta-model to address the gaps mentioned above. The extensions made to the meta-model can be seen in Fig. 4. It has to be noted that this figure only shows newly added classes and relations. A full representation of the meta-model in the style of an UML class diagram can be found online.[7] In the following, the changes and newly added model constructs are explained in detail.

1) INFRASTRUCTURE AND APPLICATION ASPECTS

To enable the modeling of infrastructural details we enhanced the RADAR meta-model by adding new nodes, relations and attributes. Especially the concepts of the so called "compute" and "software component" node were thoroughly revised. First, we now employ inheritance to distinguish between different types of compute nodes based on the three layers of edge computing architecture. The new types are called "IoTDevice," "FogCompute" and "CloudCompute." This enables modelling more precise PCPs and adaptation rules while also keeping the possibility to model generic compute nodes.

Second, each compute node can now comprise hardware components. This concept enables modeling of detailed hardware information that may be relevant when the engine is trying to identify PCP instances or trying to find the optimal adaptation, especially with regards to performance and energy consumption variables as described in section V-B. Again, inheritance is used to represent different types of hardware components, namely "CPU," "GPU," "AcceleratingComponent," "HardDrive" and "RAM." They differ on the basis of certain attributes like e.g., "clockspeed" or "capacityInGB."

[7] Seehttps://git.uni-due.de/fogprotect/vap-adaptation-engine

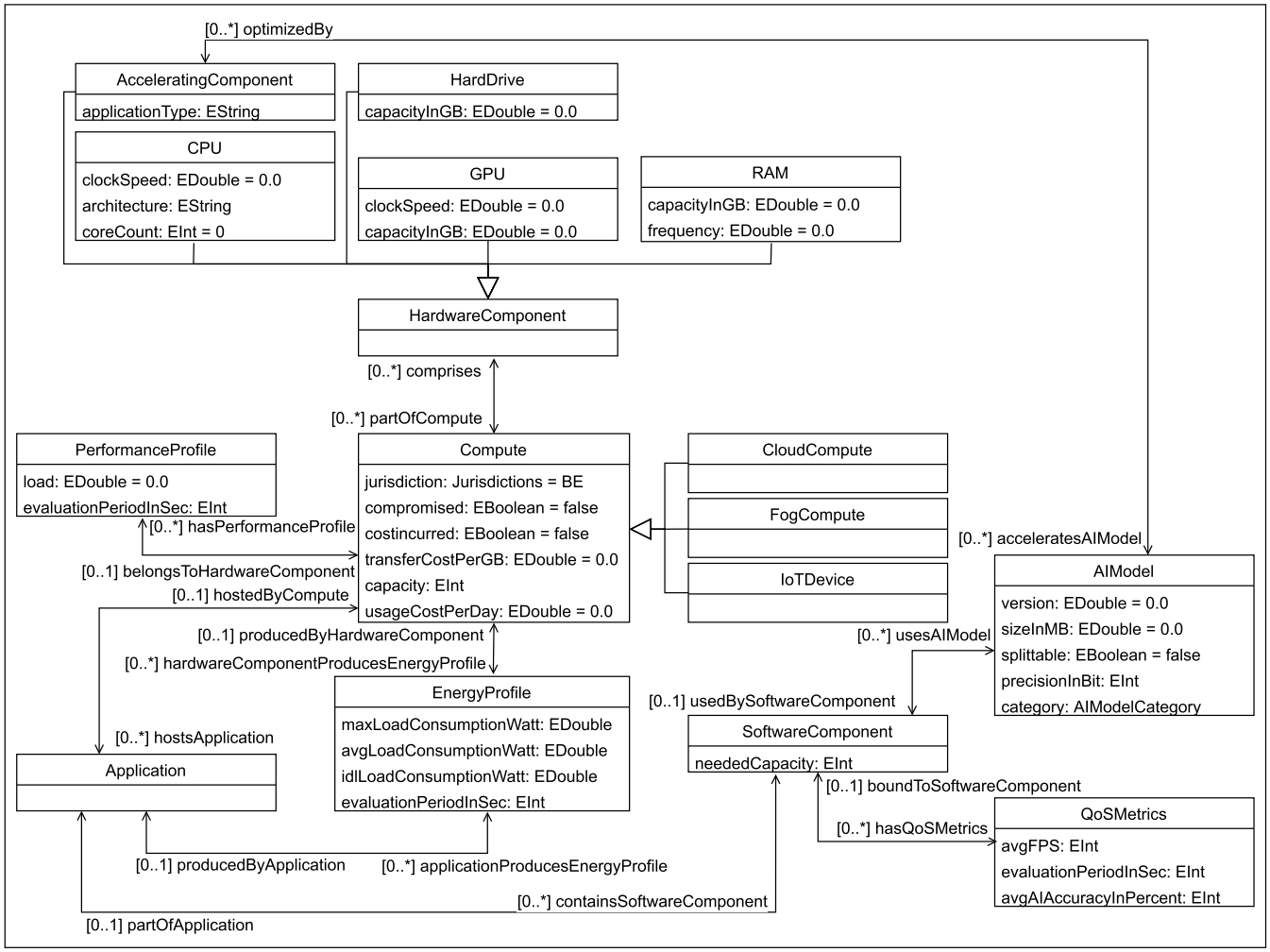

FIGURE 4. The reworked and extended part of the RADAR meta-model.

Third, each hardware component may have a performance and energy profile assigned. We introduced those profiles to the meta-model to allow energy consumption and performance to be considered when running the adaptation algorithm. We assume that at runtime a monitoring component provides the required information as average values for a selected evaluation period.

Lastly, we also rethought the concept of software components. Prior to our changes, each compute node was able to host multiple software components. Now, we added a node called ''application'' that is hosted by a compute node, and a software component describes a dedicated part of an application, for example an API or a processing algorithm. Therefore, software components take over specific tasks like communication. To represent the communication between software components we are using the following chain of nodes: software component - data flow (consisting of one or multiple data records) - software component. Thus, an application does not replace a software component, it rather comprises and clusters multiple software components. Additionally, an application can also be linked to an energy profile, similar to hardware components, to e.g., identify and handle the main drivers of total energy consumption of a node.

To store additional details that can be used to evaluate the overall performance of the managed system when searching for adaptations, each software component is related to the nodes called ''QoSMetrics'' and ''AIModel.'' Currently, QoS metrics only consider metrics that are relevant for VAPs (average frames per second, average AI accuracy in percentage). However, further metrics can easily be added when using the engine in other environments. The AI model that may be used by a software component is represented by a new node storing information like the precision, the AI-model category (e.g. face detection or dedicated object detection) and whether the model is splittable or not. Furthermore, an AI task, leveraging a specific AI-model, can be accelerated by an accelerating hardware component, such as a GPU or TPU. An example run-time model consisting of all newly added or reworked concepts can be found online in our git repository.

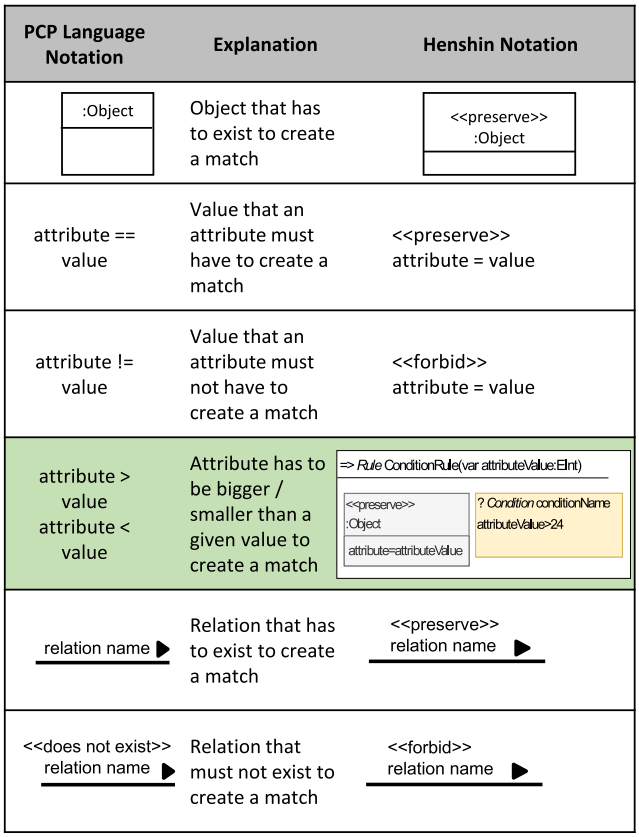

FIGURE 5. Extensions to the PCP language and mapping to Henshin.

The run-time model is based on the use case described in section III-A and will be discussed in detail in section VI.

2) DATA PROTECTION ASPECTS

We also extended the meta-model to enable modeling of security features. A security feature represents the bridge between a data record that needs to be stored or transferred in a secure way, and hardware or software components that are securing the data record. Our model covers confidentiality, integrity, availability, possession control, authenticity and utility as types of a security feature based on the Parkerian Hexad described in section IV.

In addition, we enhanced the PCP language. Prior to our extensions, it was only possible to check whether an attribute value is equal to a reference value that has to be defined at design time. Now, PCPs can also check whether an attribute value is lesser or greater than a reference value. This enables the coverage of specific VAP data protection aspects and QoS metrics, such as meeting the minimum requirements of FPS or AI model accuracy. The changes made to the PCP language and the mapping to the Henshin language are shown in Fig. 5 and are highlighted in green. It can be seen that a Henshin condition has to be used to compare an attribute to a given value.

Furthermore, reference values can now be changed at run-time which allows reacting to changing expectations from the environment. For example, additional software could be used to calculate a minimum FPS value that needs to be met by an object blurring software component to ensure that data protection and QoS requirements are fulfilled.

B. ALGORITHMIC IMPROVEMENTS

The main goal of each adaptation is to ensure adequate data protection. However, as stated in section IV multiple solutions to mitigate data protection risks and therefore ensuring data protection may exist. These adaptations may have a different impact on metrics such as energy consumption, performance, costs, and available functionality. In order to adapt the system in the best possible way, it is therefore necessary to include both data protection and above-mentioned metrics in the adaptation planning algorithm. However, because protecting personal data is a priority, it is not possible to balance data protection and system metrics. By improving the algorithms of RADAR, we enable the evaluation of further metrics, namely energy consumption and performance. Additionally, we reworked the way available functions and total costs are calculated for a run-time model or proposed adaptation model. Lastly, the order of the metrics is no longer predefined but dynamically changeable at runtime by exposing a dedicated API.

First, functionalities can now be modeled similar to PCPs by using the same notation as the PCP language and the same modeling tool provided by Henshin. Therefore, a functionality can, for example, be modelled by a system architect who defines what a functionality in the given system represents. Therefore, a functionality is now represented as a sub-graph and will be identified by using graph-pattern-matching. The amount of available functions can be understood as a QoS metric. The overall goal is to opt for a high amount of available functions. It should be noted that the calculation of the amount of available functions, costs, energy consumption and performance follows a global approach. A run-time model may consist of context nodes that are not part of the managed system and thus should not be considered when calculating global values. Hence, we are only considering values to be part of the calculation if they are part of a functionality instance that was found by our graph-pattern-matching algorithm.

Second, by using the information from the newly added nodes "EnergyProfile," "PerformanceProfile," "AIModel" and "QoSMetrics" it is now possible to calculate the total energy consumption and the performance value of a model. To calculate total energy consumption, all *EnergyProfile* nodes that are part of a functionality instance are considered. The average load consumption in Watt of each energy profile is used to calculate the overall sum. To calculate a performance value, we are using an equation that results in a numeric value. It combines different attribute values that use different scales like FPS (natural numbers) or AI model accuracy (percentage). Therefore, those different attribute values need to be normalized first. However, considering FPS as an example, a linear transformation to normalize values

between 0 FPS and 60 FPS would not be feasible and accurate in order to compare two nodes. For the human eye every video with a frame rate around 24 FPS is perceived as fluid. Differences in frame rates above 24 FPS are significantly harder to notice, than the differences below that value. In terms of QoS a human may not see the difference between 50 and 60 FPS. However, the perceived difference between 14 and 24 FPS is significant. Internally our engine expects such metrics as scalar values between 0 and 1 to combine them with other performance related metrics to calculate an overall performance value. Therefore, we used a logistic function to normalize the FPS value where the slope grows steep for values between 0 and 24, but flattens out for values above 24. Equation 1 represents the mathematical notation of the logistic function.

To calculate the global performance value, we first identify all *QoSMetric* objects and *AIModel* objects that are in relation to a software component that is part of an identified function. Second, we normalize the FPS value stored in the QoSMetrics objects. Afterwards, the normalized FPS value and the percentage value of the AI model accuracy are summed up in a distribution of $\alpha = 50\%$ as shown in Equation 2. However, our approach can be extended to use any kind of function that is appropriate for a specific QoS metric, as long as the output values are between 0 and 1. The newly added performance metric has been added to Equation 2 and α needs to be adjusted to achieve the desired *weighting* for overall QoS.

$$f(x) = \left(\frac{(a-b)}{(1+e^{-c*(x-d))}} + b\right) = A \tag{1}$$

and

$$P = A * \alpha + B * (1-\alpha) \tag{2}$$

where:

a = the curve's top asymptote value
b = the curve's bottom asymptote value
c = the logistic growth rate of the curve
d = the x value of the sigmoid's midpoint
x = input FPS value
A = normalized FPS
B = AI model accuracy in percent
P = performance value
α = performance metric weight

Algorithm 1 shows the model analysis algorithm. The algorithm is given a model M as an input to perform the model analysis (line 1). At first, two empty sets to store the PCPs and available functionalities found in M are created (line 2-3). Afterwards, placeholders to store the global numeric cost, energy consumption and performance value are created (line 4-6). By using graph-pattern matching, the engine is searching for PCP instances in M using a set of all PCPs designed at design time. Whenever a PCP instance is found, it is stored in PCP_M (line 7-12). The same procedure is used to identify instances of predefined functionality patterns. Whenever an available functionality is found, it is stored in F_M. By using the functionality instance costs, energy consumption, and performance are calculated and the result is added to the global placeholders C_M, E_M, P_M. (line 13-21). Finally, the PCP instances (PCP_M), the functionality instances (F_M), and the the calculated system metrics are stored in M (line 22-24).

Algorithm 1 Model Analysis Algorithm.

```
1:  Input: M                                   ▷ model to be analysed
2:  PCP_M ← {}                                 ▷ set of PCP instances in M
3:  F_M ← {}                                   ▷ set of available functions in M
4:  C_M ← 0                                    ▷ global costs of M
5:  E_M ← 0                                    ▷ global energy consumption of M
6:  P_M ← 0                                    ▷ global performance value of M
7:  PCP_A ← getPCPs( )                         ▷ set of all designed PCPs
8:  for each pcp ∈ PCP_A do
9:      if pcp exists in M then
10:         add pcp to PCP_M
11:     end if
12: end for
13: F_A ← getFunctions( )                      ▷ set of all designed functions
14: for each f ∈ F_A do
15:     if f exists in M then
16:         add f to F_M
17:         C_M += calcCosts(M, f)
18:         E_M += calcEnergyConsumption(M, f)
19:         P_M += calcPerformance(M, f)
20:     end if
21: end for
22: M.setPCPs(PCP_M)
23: M.setFunctions(F_M)
24: M.setMetrics(PCP_M.length, F_M.length, C_M, E_M, P_M)
```

To determine whether a possible adaptation is better than another, we are comparing models. As stated in section IV, the engine is searching for the best possible system configuration by evaluating sequences of adaptations that are reachable from the current run-time model. To do so, a search tree is constructed. A child node represents the run-time model obtained from the run-time model of the parent node through an adaptation. The search tree is built up during the search, by iteratively adding unexplored child nodes to the nodes already visited. An evaluation of different strategies has shown that a best-first search approach leads to the best results [24]. The model comparison takes the prioritization of the comparison metrics into account. By using an API, a human operator can decide which metric should be used first to optimize the VAP after ensuring adequate data protection.

Algorithm 2 explains how the best model is determined. First, the algorithm keeps track of the best solution and the path from the root node to the respective solution (line 1-2). Moreover, the current prioritization order of the metrics list is queried (line 3). As long as unexplored nodes exist (set of nodes S is not empty, starting with the current run-time model M^{RT}, line 4-5), the algorithm is searching for the best node. While evaluating nodes from S, a selected node M out of S is

first analysed with regard to the amount of PCP instances, the amount of available functions, the overall energy consumption, the overall performance value and the total costs (line 6-7). To do so, algorithm 1 is used. Afterwards, M is compared to the actual best node by comparing both models. Whenever two models are compared, the algorithm picks a metric in a defined order from the list of metrics (line 8-9). As long as there are further comparison metrics (line 10), the selected comparison metric is used to compare M to the best solution (line 11). If the calculated values for a metric are equal the next metric is used for comparison, otherwise M is either better or worse than the best solution (line 12-21). If M is better than the best solution in regard to a specific comparison metric, the best solution is replaced by M, the path to the best solution is stored, and the comparison ends (line 12-16). The comparison also ends if M is worse than the best solution (line 17-19). It is important to state that the amount of PCP instances will always be the first metric that will be used for comparison (line 9). In the end it should be possible to state which of the two models is optimal. After the current node has been evaluated, it is removed from S and the children of the node are added to S instead (line 22-23). To avoid an endless search in the potentially unbounded space of possible solutions, the search is aborted when a predefined termination criterion is met (lines 24-26). Such a termination criterion could be for instance a maximum allowed time budget for the execution of the algorithm. Finally, the algorithm returns the adaptation sequence leading to the best found solution (line 28), which should then get executed in the real system.

As already stated, the algorithm is capable of using different prioritization orders for the list of comparison metrics. The order of the metrics can be changed at runtime by using an API while the engine keeps the amount of PCP instances (i.e., threats to data protection) always as the metric with the highest priority to optimize towards.

Algorithm 2 Model Comparison Algorithm.

```
1:  best ← NULL                                   ▷ best solution found so far
2:  bestPath ← NULL                               ▷ path from root to best solution
3:  ML ← getListOfMetrics( )                      ▷ sorted list of metrics
4:  S ← {M^RT}                                    ▷ set of nodes that are being explored
5:  while S ≠ ∅ do
6:      M ← selectNode(S)
7:      AnalyseModel(M)                           ▷ see algorithm 1
8:      i ← 0
9:      CM ← ML(i)                                ▷ comparison metric from ML
10:     while CM ≠ ∅ do
11:         CompareModels(M, best, CM)
12:         if M is better than best in regard to CM then
13:             best ← M
14:             bestPath ← path from M^RT to M
15:             break
16:         end if
17:         if M is worse than best in regard to CM then
18:             break
19:         end if
20:         i++                                   ▷ M and best are equal in regard to CM
21:     end while
22:     T ← generateChildren(M)
23:     S ← S \ {M} ∪ T
24:     if termination criterion then
25:         break
26:     end if
27: end while
28: return bestPath
```

C. SUMMARY

To summarize, we highlight the most important additions and upgrades to RADAR in order to fulfill the needs of a data protection focused edge computing system, particularly for a distributed AI-assisted VAP:

- Added security and privacy features to the meta-model according to the Parkerian Hexad model.
- Added machine learning entities to the meta-model to enable the modelling of AI-based systems.
- Enhanced the meta-model to allow for fine grained modeling of infrastructural aspects of a system.
- Added energy consumption and performance as part of optimization goals.
- PCPs can now handle set operators, thereby enabling the algorithm to dynamically compare either monitoring variables or static variables. This is for example needed to enable software compatibility or version checks.
- The adaptation algorithm is now able to change the prioritization of different metrics namely costs, QoS (available functions), energy consumption, performance, while always aiming for the lowest amount of PCP instances.

VI. EVALUATION

In this section we demonstrate how the engine behaves with different configurations and correctly solves PCPs, based on a simplified example implementation of the traffic monitoring use case as described in section III. This is followed by a discussion on the limitations of our approach and its general applicability to other use cases. The runtime behavior of RADAR has been evaluated in previous work [24]. Additional information on the evaluation of the pattern-matching algorithm can be found on the Henshin website.[8]

A. EVALUATION OF THE CONSIDERED USE CASE

In order to evaluate our approach we modelled a system configuration comprising five heterogeneous compute nodes and defined a scenario typical for the considered traffic monitoring use case. The system configuration can be seen as a part of the traffic monitoring system, e.g., a large crossing with five smart lampposts. Initially, the system is configured so that low energy consumption is prioritized. As long as only trusted

[8] https://www.eclipse.org/henshin/publications.php

actors access the monitoring component, an unanonymized video is transferred.

Fig. 6 displays a descriptive simplified model representation that reflects our considered system configuration and use case scenario. It can be seen that the different edge nodes differ in their video processing performance (frames per second), their AI model accuracy (object detection accuracy in per cent), their average energy consumption (in watt), and their security features (trusted / untrusted). The nodes model real world edge computing nodes, highlighted by their factory naming, e.g., a RaspberryPi 4B, and their respective capabilities. Solid lines represent the initial situation. Dotted lines represent possible offloading solutions that are not taken into account due to data protection and optimization reasons. Dashed lines represent possible offloading solutions depending on the prioritization of performance vs. energy consumption. In the following, the different scenarios are described in detail.

In the use case scenario, we handle a request from an untrusted user wanting to access video data that is recorded by a smart lamppost. This actor is only allowed to access anonymized video data. The access of the untrusted actor is detected by our engine as a PCP instance, highlighted in Fig. 6 with a thunderbolt. Therefore, the goal is to mitigate this PCP instance while maintaining low energy consumption. As described in section V, the algorithm combines several possible adaptations and compares different system configurations.

The best solutions consists of two adaptations that need to be both carried out subsequently. First, an anonymization software component needs to be activated, and a data flow transferring video data with blurred faces and number plates needs to replace the data flow transferring the unanonymized video. The associated adaptation rule modelled with Henshin is shown in Fig. 7. Concretely, it shows that whenever a *personal record* is transferred via a *data flow* to a *software component* that is controlled by an *untrusted data controller* (precondition), the particular data flow is removed and a new *data flow* transferring a new *non-personal video* from a *software component* called ''Anonymization'' to the *software component* used to access the video is created (adaptation action). The corresponding PCP is not represented separately because it is already part of the adaptation rule as its precondition.

It has to be noted that another solution would be to adapt the system in a way in which the video stream would be stopped. However, this adaptation implies a drastic decrease in available functionality and should therefore only be used as a last resort.

In this particular use case, the adaptation that activates video anonymization is the best choice. However, another PCP instance is created when enabling video anonymization at *Node1* because *Node1* is not capable of processing the video fast enough (output video FPS < 25) to ensure QoS at an acceptable level. Hence, a second adaptation that offloads the video processing from *Node1* to another node has to be carried out. Fig. 8 shows the corresponding adaptation rule. Again, the PCP instance is already part of the adaptation rule as its precondition: Whenever the *avgFPS* of an *QoSMetrics* object is below 25 and a *software component* called ''Anonymization'' is transferring video data to another *software component* that is not part of the same *application* a PCP instance exists. In this case, the adaptation rule stipulates the following adaptation action: reallocate all existing *data flows*, starting at an *IoT device* and ending at *software components* that are not part of the edge nodes *application*. When reallocating, switch from each *software components* used before to *software components* that have the same name and are hosted on another edge nodes *application*.

Four different system configurations have to be compared by the engines' algorithm, each related to one of the four existing nodes shown in Fig. 6. All four nodes are capable of processing the video adequately, allowing for an continuous video anonymization. However, in this particular use case only *Node3* and *Node4* are reasonable possibilities because an offloading to these nodes mitigates all PCP instances and optimizes the system configuration based on the desired metric prioritization.

When prioritizing optimal energy consumption, *Node2* should be selected due to its low energy consumption. However, the algorithm always considers data protection the top most priority before optimizing the system configuration towards a QoS goal. Reallocating the video processing from *Node1* to *Node2* leads to a new PCP instance because *Node2* is untrusted. Therefore, *Node2* is not taken into consideration when comparing possible system configurations in terms of improving the system configuration.

A reallocation to *Node3*, *Node4*, or *Node5* does not lead to a new PCP instance. When comparing the remaining possible system configuration options in terms of energy consumption, *Node3* represents the best choice. Therefore, both adaptations –activating object blurring and reallocating the video processing– are carried out at once to enable video access to an untrusted actor. This is represented by the dashed line from *Node1* to *Node3* and from *Node3* to the *Monitoring Component* in Fig. 6.

When changing the prioritization from low energy consumption first to high performance first, our engine detects a possible improvement adaptation by comparing different system configurations that are created by using the adaptation rule described in Fig. 8. Three out of four possible system configuration changes lead to newly arising PCP instances or worse performance. Only the reallocation of the video processing to *Node4* does improve the overall system performance because *Node4* can process the video with the highest amount of FPS combined with a high AI model accuracy. Therefore, this adaptation is carried out. The dashed line from *Node4* to the *Monitoring Component* in Fig. 6 shows this scenario.

Should the untrusted actor quit accessing to the monitoring component, object blurring is not longer needed. The engine detects a possible improvement and uses the adaptation rule

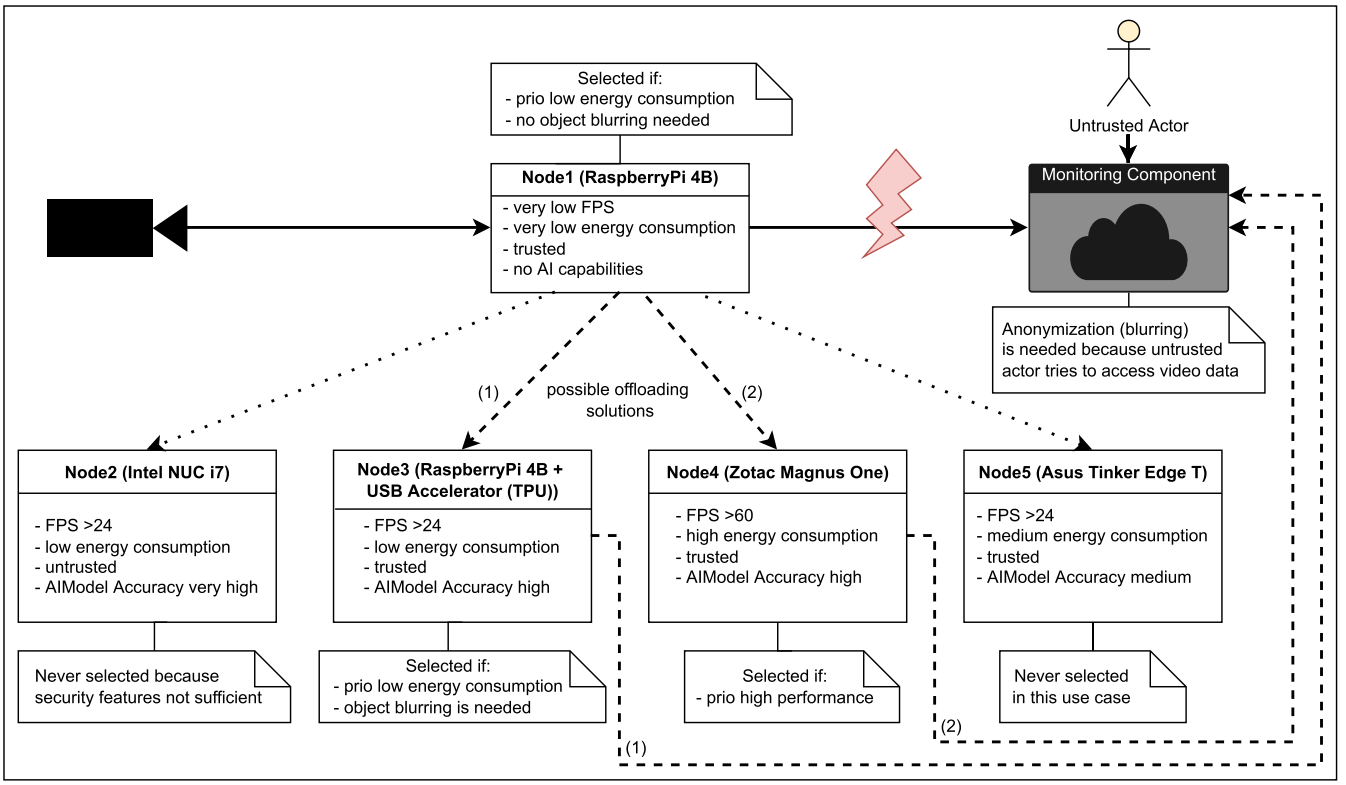

FIGURE 6. Descriptive problem representation of the traffic monitoring use case model.

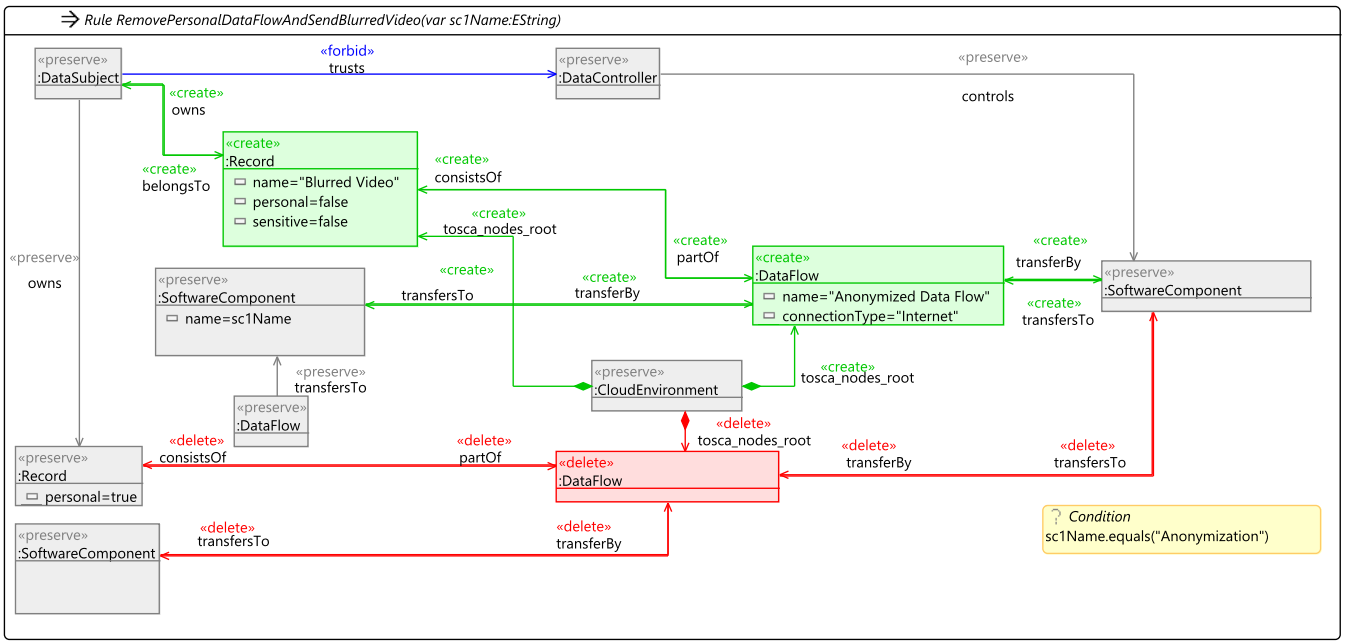

FIGURE 7. Adaptation rule that activates video anonymization.

shown in Fig. 9 to change the system configuration accordingly. The respective rule does the opposite of the rule shown in Fig. 7, i.e., removing the *data flow* transferring an *anonymized video record* and reactivating a *data flow* transferring an *unanonymized video record*. Depending on the selected prioritization order, the video processing remains at *node4* (highest performance) or an adaptation based on the adaptation rule shown in Fig. 8 (lowest energy consumption

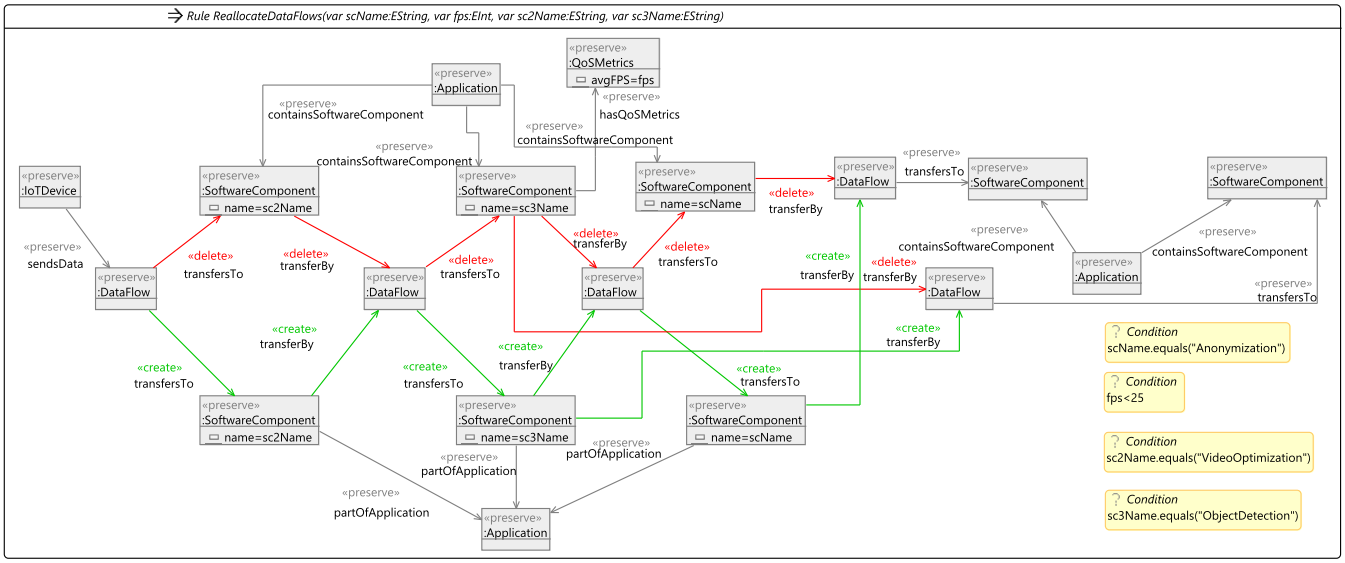

FIGURE 8. Adaptation rule that reallocates data flows.

when no object blurring is needed) is carried out, reallocating the video processing to *Node1*.

In Fig. 10 the impact of the selected adaptations is shown. $S1$ represents the initial situation where video blurring is deactivated and energy consumption is prioritized. The VAP system consumes a low amount of energy while the performance is sufficient. $S2$ represents the system metric values after video blurring was activated. Due to offloading video processing from *Node1* to *Node3*, an increase in performance and energy consumption can be seen. However, due to the activation of video blurring, the amount of available functionality has decreased, because seeing an unanonymized video was designed as part of a functionality pattern. After changing the prioritization from *low energy consumption first* to *high performance first*, both performance and energy consumption increase substantially. This is due to the migration from *Node3* to *Node4*. *Node4* does support video processing and AI-Task offloading to a dedicated video card, which affects both performance and energy consumption. The system metrics in this situation are represented by $S3$. When video blurring is no longer needed and therefore deactivated, the amount of available functions increases (see $S4$) again. If, in addition, the prioritization is changed back to low energy consumption first, then the original situation $S1$ is adopted due to the migration from *Node4* to *Node1*.

A detailed version of all described scenario steps in the style of a graphical run-time model (similar to an UML object diagram) as well as graphical representations of PCPs and adaptation rules can be found online.[9] Moreover, the code base and an instruction how to run this example use case are included there.

[9]Seehttps://git.uni-due.de/fogprotect/vap-adaptation-engine

B. LIMITATIONS

In previous work, scalability experiments have shown that the engine is capable of handling models with up to 200 nodes in a given time frame of 10 seconds [24]. In our considered use case scenario the engine is not limited by the size of the run-time model. Thus, the engine is capable of always finding the best solution in minimal time. To test how long it takes for the engine to determine the best solution in this kind of scenario we measured the time between the change of the run-time model using the monitoring API until the engine found the best solution. The tests were executed on a typical edge node equipped with an Intel Core i5-4690K processor with 3.5GHz clock frequency and with 16 GB DDR3 memory. The node was running the Windows 10 OS and JDK14.0.1 as the Java environment. After hundred rounds of testing, an average evaluation time of round about 1.4 seconds was measured. It should be mentioned that this measured time does not equal the interval between a real world system change causing a PCP instance and the final system change caused by the call to execute the optimal adaptation by our engine. We tested our engine independent from the managed self-adaptive system, as this system could be a limiting factor that we cannot influence.

Thus, our solution is mainly limited by monitoring and adaptation execution capabilities of the self-adaptive system. On the one hand, the possible frequency of monitoring reports as well as the quality and level of detail of the monitoring reports will influence our proposed approach. For example, if details are missing, our engine may not detect a PCP instance or possible adaptation. If the report frequency is too low, it takes longer for a PCP instance to be detected and thus until the self-adaptive systems system configuration is adapted. Furthermore, the adaptation execution capabilities of the self-adaptive system may limit our approach because

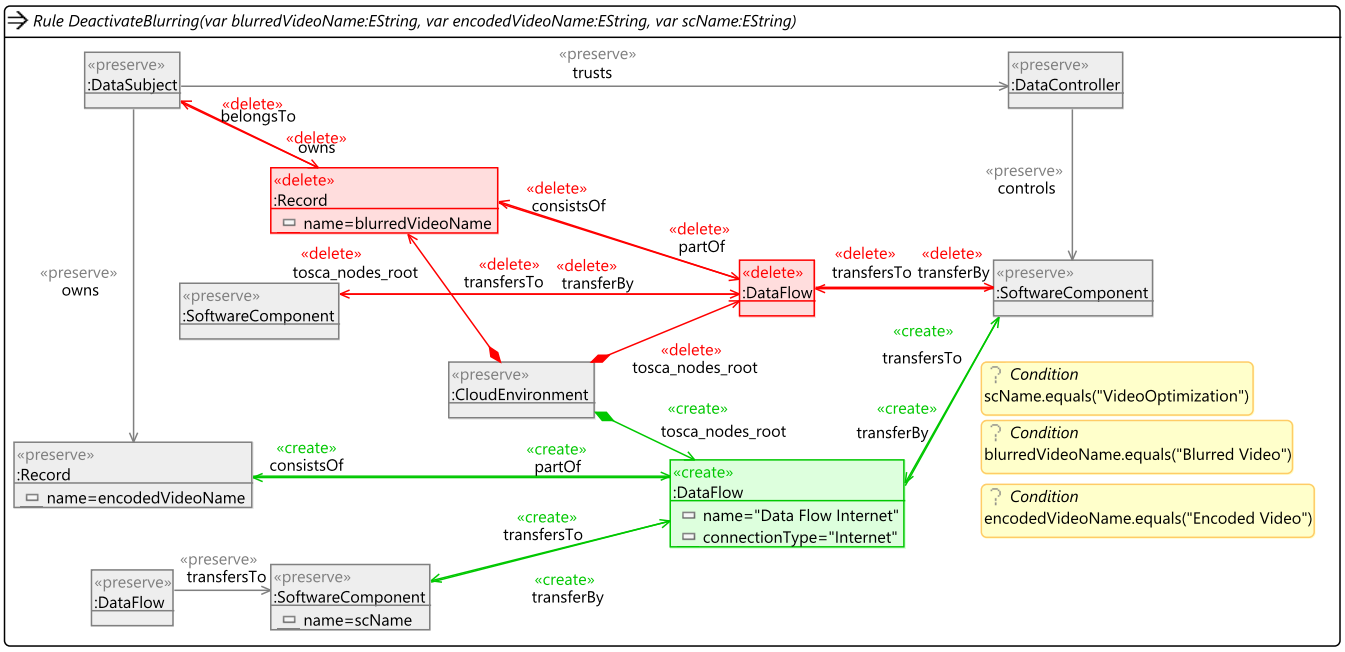

FIGURE 9. Adaptation rule that deactivates video anonymization.

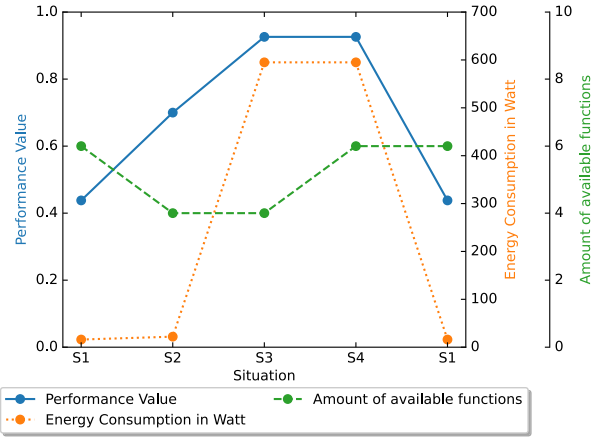

FIGURE 10. Changes in the metric values after the described adaptations.

only supported system adaptations can be taken into account when then engine is searching for adaptations. Moreover, an adaptation selected for execution may not be carried out successfully. In this case, our engine can only try to carry out the proposed adaptation again or try to carry out the next best adaptation.

C. APPLICABILITY

Our evaluation has shown that the adaptation engine is capable of solving data protection risks in Video Analytics Pipelines while also finding the optimal system configuration regarding global energy consumption, system performance and available functionalities. However, our approach is not limited to VAPs. Systems that face similar challenges, such as surveillance applications, can also implement our meta-model, PCP language, and adaptation language. If extensions to the meta-model are needed, this changes do not affect existing run-time models, PCPs, or adaptations. Therefore, our approach is transferable to many edge systems dealing with data protection and optimization problems at runtime. Additionally, our previous research [24] has shown that data protection risks related to location and jurisdiction restrictions are also covered by our approach. Therefore, the range of different data protection problems this approach can handle is not limited to the content of the data but also handles geospatial or environmental constraints, such as data locality requirements.

VII. CONCLUSION

Operating a distributed Video Analytics Pipeline (VAP) comes with many associated challenges. Adhering to data protection regulations, dealing with changes in computational load, targeting low energy consumption or facing a heterogeneous hardware and software environment are prominent examples of those challenges. In order to provide an adequate QoS and comply to data protection policies, a VAP has to react and adapt to face those challenges. While there is state of the art literature dealing with either performance or data protection related adaptations, most of them solely focus on increasing certain security or privacy aspects of a system, leaving previously mentioned performance characteristics out of scope or the other way round. To the best of our knowledge, there is no solution that covers data protection, computational performance and energy consumption aspects.

In this paper, we presented a data protection focused adaptation engine that leverages the application- and infrastructure based adaptation space of a distributed VAP. The engine employs a system model and adaptation rules that are based on previous research. The model was specifically extended and enhanced to meet the requirements of AI-assisted VAPs at the Edge. Furthermore, the engine features an optimization algorithm to improve performance, energy consumption and data protection of a distributed VAP and its functionalities. Using a traffic monitoring use case as running example, we demonstrated how the engine behaves with different configurations and correctly solves problematic configurations during design- and runtime.

Since our approach is extendable by adjusting the meta-model and by adding further PCPs and corresponding adaptations, future work will focus on applying the adaptation engine to related environments. Based on the respective system, additional metrics, such as business metrics, will be taken into account. Due to the nature of edge systems, decentralizing our approach and deploying it multiple times in the edge network of a self-adaptive system may enable better performance and faster responses to PCPs. However, further research is needed to find a suitable way of coordinating multiple adaptation engines.

ACKNOWLEDGMENT

The authors would like to thank the useful discussions with project partners in FogProtect.

REFERENCES

[1] O. Vermesan and P. Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers, 2013.
[2] A. Dearle, "Software deployment, past, present and future," in *Proc. Future Softw. Eng. (FOSE)*, May 2007, pp. 269–284.
[3] A. Brogi, S. Forti, and A. Ibrahim, "Predictive analysis to support fog application deployment," in *Fog and Edge Computing: Principles and Paradigms*. 2019, pp. 191–222.
[4] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to place your apps in the fog: State of the art and open challenges," *Softw., Pract. Exper.*, vol. 50, no. 5, pp. 719–740, May 2020.
[5] N. Dong, H. Jonker, and J. Pang, "Challenges in eHealth: From enabling to enforcing privacy," in *Proc. Int. Symp. Found. Health Informat. Eng. Syst.* Springer, 2011, pp. 195–206.
[6] C. Lachner, T. Rausch, and S. Dustdar, "Context-aware enforcement of privacy policies in edge computing," in *Proc. IEEE Int. Congr. Big Data (BigDataCongress)*, Jul. 2019, pp. 1–6.
[7] J. Daubert, A. Wiesmaier, and P. Kikiras, "A view on privacy & trust in IoT," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 2665–2670.
[8] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of Internet of Things (IoT)," *Int. J. Comput. Appl.*, vol. 111, no. 7, pp. 1–6, Feb. 2015.
[9] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of Internet of Things (IoT)," *Int. J. Comput. Appl.*, vol. 111, no. 7, pp. 1–6, Feb. 2015.
[10] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the Internet of Things: A review," in *Proc. IEEE Int. Conf. Comput. Sci. Electron. Eng. (ICCSEE)*, vol. 3, Mar. 2012, pp. 648–651.
[11] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. North Chelmsford, MA, USA: Courier Corporation, 2012.
[12] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4388–4400, Jun. 2019.
[13] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Commun.*, vol. 14, no. 11, pp. 59–68, Nov. 2017.
[14] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.
[15] S. Kim, "New application task offloading algorithms for edge, fog, and cloud computing paradigms," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–14, Oct. 2020.
[16] J. Bürger, D. Strüber, S. Gärtner, T. Ruhroth, J. Jürjens, and K. Schneider, "A framework for semi-automated co-evolution of security knowledge and system models," *J. Syst. Softw.*, vol. 139, pp. 142–160, May 2018.
[17] J. Bürger, S. Gärtner, T. Ruhroth, J. Zweihoff, J. Jürjens, and K. Schneider, "Restoring security of long-living systems by co-evolution," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2015, pp. 153–158.
[18] C. Tsigkanos, L. Pasquale, C. Ghezzi, and B. Nuseibeh, "On the interplay between cyber and physical spaces for adaptive security," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 3, pp. 466–480, May/Jun. 2016.
[19] D. Ayed, E. Jaho, C. Lachner, Z. A. Mann, R. Seidl, and M. Surridge, "FogProtect: Protecting sensitive data in the computing continuum," in *Proc. Int. Workshops ESOCC*, 2021, pp. 179–184.
[20] A. Palm, Z. A. Mann, and A. Metzger, "Modeling data protection vulnerabilities of cloud systems using risk patterns," in *Proc. Int. Conf. Syst. Anal. Modeling*. Springer, 2018, pp. 1–19.
[21] A. Alebrahim, D. Hatebur, S. Fassbender, L. Goeke, and I. Côté, "A pattern-based and tool-supported risk analysis method compliant to ISO 27001 for cloud systems," *Int. J. Secure Softw. Eng.*, vol. 6, no. 1, pp. 24–46, Jan. 2015.
[22] L. Pasquale, S. Hanvey, M. Mcgloin, and B. Nuseibeh, "Adaptive evidence collection in the cloud using attack scenarios," *Comput. Secur.*, vol. 59, pp. 236–254, Jun. 2016.
[23] J. Laufer, Z. A. Mann, and A. Metzger, "Modelling data protection in fog computing systems using UMLsec and SysML-sec," in *Proc. ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. Companion (MODELS-C)*, Oct. 2021, pp. 777–786.
[24] Z. A. Mann, F. Kunz, J. Laufer, J. Bellendorf, A. Metzger, and K. Pohl, "RADAR: Data protection in cloud-based computer systems at run time," *IEEE Access*, vol. 9, pp. 70816–70842, 2021.
[25] G. Gonzalez-Granadillo, E. Alvarez, A. Motzek, M. Merialdo, J. Garcia-Alfaro, and H. Debar, "Towards an automated and dynamic risk management response system," in *Proc. Nordic Conf. Secure IT Syst.* Springer, 2016, pp. 37–53.
[26] G. Gonzalez-Granadillo, S. Dubus, A. Motzek, J. Garcia-Alfaro, E. Alvarez, M. Merialdo, S. Papillon, and H. Debar, "Dynamic risk management response system to handle cyber threats," *Future Gener. Comput. Syst.*, vol. 83, pp. 535–552, Jun. 2018.
[27] K. Kritikos, M. Papoutsakis, S. Ioannidis, and K. Magoutis, "Towards configurable cloud application security," in *Proc. 19th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2019, pp. 684–689.
[28] N. Nostro, A. Ceccarelli, A. Bondavalli, and F. Brancati, "Insider threat assessment: A model-based methodology," *ACM SIGOPS Operating Syst. Rev.*, vol. 48, no. 2, pp. 3–12, Dec. 2014.
[29] M. Nguyen, P. Samanta, and S. Debroy, "Analyzing moving target defense for resilient campus private cloud," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 114–121.
[30] M. Boyle, C. Neustaedter, and S. Greenberg, "Privacy factors in video-based media spaces," in *Media Space 20+ Years Mediated Life*. Springer, 2009, pp. 97–122.
[31] A. Senior, *Protecting Privacy in Video Surveillance*, vol. 1. Springer, 2009.
[32] M. A. Uddin, A. Alam, N. A. Tu, M. S. Islam, and Y.-K. Lee, "SIAT: A distributed video analytics framework for intelligent video surveillance," *Symmetry*, vol. 11, no. 7, p. 911, Jul. 2019.
[33] A. B. Sada, M. A. Bouras, J. Ma, H. Runhe, and H. Ning, "A distributed video analytics architecture based on edge-computing and federated learning," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2019, pp. 215–220.
[34] P. Liu, B. Qi, and S. Banerjee, "EdgeEye: An edge service framework for real-time intelligent video analytics," in *Proc. 1st Int. Workshop Edge Syst., Anal. Netw.*, 2018, pp. 1–6.
[35] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2018, pp. 253–266.

[36] C. Lachner, Z. A. Mann, and S. Dustdar, "Towards understanding the adaptation space of AI-assisted data protection for video analytics at the edge," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jul. 2021, pp. 7–12.
[37] D. B. Parker, "Toward a new framework for information security?" in *Computer Security Handbook*. Wiley Online Library, 2012, pp. 1–3.
[38] R. Bhaskar and B. Kapoor, "Information technology security management," in *Computer and Information Security Handbook*. Amsterdam, The Netherlands: Elsevier, 2013, pp. 449–458.
[39] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC*, Gen. Data Protection Regulation (GDPR), Off. J. Eur. Union (EU), Brussels, Belgium, 2016, p. L119.

CLEMENS LACHNER (Member, IEEE) is currently a Research Scientist with the Research Division of Distributed Systems, TU Wien, Austria. His research interests include privacy and security mechanisms for resource constraint devices and AI-based data protection in the computing continuum.

JAN LAUFER is currently a Research Associate at paluno—The Ruhr Institute for Software Technology, University of Duisburg-Essen, Germany. His research interests include exploring design time activities, such as modeling data protection in fog computing systems and finding solutions to automatically address data protection threats in fog computing systems at runtime.

SCHAHRAM DUSTDAR (Fellow, IEEE) is currently a Full Professor in computer science heading the Research Division of Distributed Systems, TU Wien, Austria. He is also the Founding Co-Editor-in-Chief of *ACM Transactions on Internet of Things* (ACM TIoT) and the Editor-in-Chief of *Computing* (Springer). He is an Associate Editor of IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*, and on the Editorial Board of IEEE INTERNET COMPUTING and IEEE Computer Society. He was a recipient of the ACM Distinguished Scientist Award, in 2009, the ACM Distinguished Speaker Ward, in 2021, the IBM Faculty Award, in 2012, and an Elected Member of the Academia Europaea: The Academyof Europe, where he is the Chairperson of the Informatics Section.

KLAUS POHL (Member, IEEE) is currently a Full Professor in software systems engineering with the University of Duisburg-Essen. He is also the Director of paluno—The Ruhr Institute for Software Technology. He was the Scientific Founding Director of Lero—The Irish Software Engineering Centre. He is a member of the NESSI Board and a member of several steering and advisory committees. His research interests include proactive adaptive systems, digitization/big data, cloud computing, cyber-physical systems, service orientation, variability management, and requirements engineering.