Architectural Design Decisions for Data Communication of Cyber-Physical Systems

1st Amirali Amiri, 2nd Evangelos Ntentos, 3rd Uwe Zdun
University of Vienna, Faculty of Computer Science,
Research Group Software Architecture
Vienna, Austria
firstname.lastname@univie.ac.at

***Abstract*—Designing Cyber-Physical Systems (CPS) is a complex task involving integrating physical and digital components to achieve specific objectives. This process consolidates data from various Internet of Things (IoT) devices and sources to generate meaningful insights and actionable outcomes. IoT-cloud data communication comprises multiple stages, e.g., data collection, processing, analysis, and visualization. Adopting a comprehensive approach that considers physical and digital aspects is essential to ensure effective data communication in CPS. As a result, architectural design choices are crucial in determining CPS functionality and runtime qualities, e.g., performance, security, and reliability. While numerous CPS architectural patterns and practices have been proposed, much of the relevant knowledge remains scattered across various sources, such as practitioner blogs and system documentation. These sources are often based on personal experiences and lack consistency. To address this gap, our study presents the outcomes of an in-depth qualitative investigation into practitioners' descriptions of the best practices and patterns in CPS architecture. We have developed a formal architectural decision model using a model-based qualitative research method. We aim to bridge the division between scientific understanding and practical use cases, enhance comprehension of practitioners' approaches to CPS, and provide decision-making support for designing CPS applications.**

***Index Terms*—Architectural Design Decisions, Cyber-Physical Systems, Edge-Cloud Data Communication, Software Architecture, Grounded Theory, Gray Literature, CPS Practitioners**

I. Introduction

Various authors have tried documenting patterns and best practices concerning CPS data communication in their works [8], [12], [14], [19]. Nonetheless, these works primarily focus on applying published patterns to CPS or patterns derived from scientific research. Conversely, most well-established practices in the industry are found in the grey literature, including practitioner blogs, experience reports, and system documentation. While these sources may provide insights into specific patterns, they lack systematic architectural guidance. The practices reported in these sources often exhibit inconsistencies and rely heavily on personal experiences, resulting in considerable uncertainty and risk in CPS architectural design. To mitigate this uncertainty and risk, architects need substantial personal experience or extensively study various knowledge sources. We aim to offer a more comprehensive and consistent perspective on current industrial data-communication practices, complementing existing knowledge sources.

We conducted an in-depth qualitative study focusing on practitioners' CPS descriptions to accomplish our goal. These descriptions encompassed informal information about established practices and patterns within the CPS data-communication domain. Our study utilized the model-based qualitative research method described in [23]. We regarded practitioner sources as relatively unbiased knowledge repositories and systematically analyzed them using established coding and constant comparison methods [5], along with precise software modeling techniques. Through this approach, we developed a well-defined software model that captures established practices, patterns, and their interrelationships. Hence, we aimed to address the following research questions:

RQ1 *What are the patterns and practices currently used by practitioners for supporting data communication in CPS architectures?*

RQ2 *How are the current data-communication patterns and practices related? In particular, which Architectural Design Decisions (ADDs) are relevant when architecting data communication in CPS?*

RQ3 *What are the influencing factors, i.e., decision drivers, in architecting data communication in CPS in the eye of the practitioner today?*

Our study brings forth noteworthy contributions. Firstly, we conducted a qualitative study on CPS architectures involving an extensive collection of 37 knowledge sources. This effort aimed to gather comprehensive information about established industrial practices, patterns, interrelationships, and the driving factors behind decision-making. Secondly, we contributed by codifying this knowledge into a reusable ADD model. The decisions were formally modeled using a UML[1] metamodel, providing a structured representation of the decision-making process. Our model encompasses 5 decisions, 27 options, and 17 decision drivers or forces. Lastly, we evaluated our model's level of detail and completeness. By adopting a practitioner's perspective, we gained valuable insights into the design of cyber-physical systems, further supporting our claim.

This work was supported by FFG (Austrian Research Promotion Agency) project MODIS, no. FO999895431.

[1] http://www.uml.org/

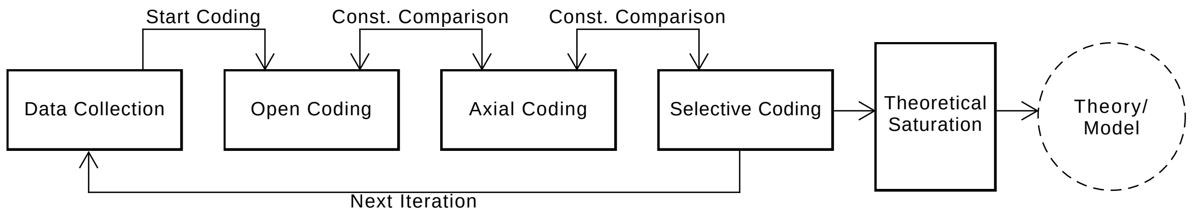


Fig. 1: Research Method

The rest of this paper is structured as follows: In Section II, we compare our study to the related work. Section III explains our applied research methods and summarizes the knowledge sources. Section IV describes our reusable ADD model on CPS data communication. Section V evaluates and discusses our results. Finally, Section VI considers the threats to the validity of our study, and Section VII summarizes our research and concludes the paper.

II. Related Work

Several studies have explored CPS patterns and practices. Jamaludin et al. [12] provide a comprehensive overview of CPS, emphasizing the significance of understanding CPS characteristics and architectures for designing reliable and adaptable systems that meet diverse application requirements. Henneke et al. [8] focus on analyzing communication patterns in CPS, such as discovery, request-response, and publish/subscribe. Reinfurt et al. [17] identify five patterns derived from examining multiple production-ready IoT solutions, addressing recurring problems and proven solution principles. Washizaki et al. [21] conduct a systematic literature review and extract 143 IoT architecture and design patterns from 32 papers, analyzing them based on various characteristics and providing insights for improving the publication and adoption of IoT patterns. Pontes et al. [16] introduce the Pattern-Based IoT Testing approach, which simplifies and organizes the testing process for IoT ecosystems by targeting common patterns of behavior known as IoT Test Patterns. Ghosh et al. [4] evaluate the current state of IoT research, revealing limitations in existing studies and employing a thorough qualitative approach to analyze the grey literature on CPS systematically, offering a comprehensive analysis.

Various decision documentation-related approaches have been explored in different domains, such as service-oriented solutions [24], service-based platform integration [13], REST vs. SOAP [15], and big data repositories [6]. However, these approaches do not specifically address CPS architectures. Warnett and Zdun [20] introduce a Grounded Theory-based approach to examine the current understanding of practitioners and architectural concepts regarding ML solution deployment. They formulate seven ADDs and establish various relations between them. Their modeling efforts include twenty-six decision options and forty-four decision drivers specific to ML deployment. Other authors have previously combined decision models with formal view models [7]. We enhance these techniques by incorporating a formal modeling approach derived from qualitative research methodology.

In our research, we investigate the approaches and techniques practitioners employ to address the disparity between theory and practice in CPS data communication. Through our study, we develop a formal model encompassing ADDs, decision options, practices, drivers, and their interrelationships. Our objective is to offer valuable insights that empower practitioners to make informed decisions regarding the data communication of CPS.

III. Approach Overview

This section discusses the research method followed in this study and the metamodel of our reusable ADDs.

A. Research Method

In this study, we employ a systematic approach to investigate the established practices in architecting data communication within CPS architectures. Our research methodology is based on the model-based qualitative research method, as described in [23]. This method combines elements of Grounded Theory (GT) [5], pattern mining [3], and GT with pattern mining [9] to analyze established practices. We begin by utilizing the authors' experiences as a foundation and then search for relevant sources in the "grey literature", such as practitioner reports, system documentation, and practitioner blogs. These selected sources serve as descriptions of established practices for further analysis. Similar to GT, our method comprehensively examines each knowledge source. We derive a model through coding and constant comparison, starting with an initial research question, drawing inspiration from Charmaz's constructivist GT [2]. Unlike traditional GT, our method translates textual codes into formal software models at the outset rather than relying solely on textual analysis.

The knowledge-mining procedure in our study consists of multiple iterations, as shown in Fig. 1. We searched for new knowledge sources and applied open, axial, and selective coding techniques [5] to identify potential categories for model elements and decision drivers. In the open-coding stage, we developed concepts based on our gray-literature sources presented in Table I. This step involves asking specific questions regarding the data and creating model concepts with minimal assumptions. Axial coding is the development of categories to link the data, concepts, categories, and properties. In the

TABLE I: Knowledge Sources Included in the Study

ID	Description	Reference
S1	How to Build an Industrial IoT Project Without the Cloud	https://bit.ly/3KqLsYd
S2	Understand the Azure IoT Edge runtime and its architecture	https://bit.ly/3XTSJ5C
S3	Connecting IoT devices to the cloud	https://thght.works/3KvnivM
S4	Real-time Data Streaming in IoT: Why and How	https://bit.ly/3kek9Wp
S5	Edge to Twin: A scalable edge to cloud architecture for digital twins	https://go.aws/3xIhSFR
S6	Understanding edge computing for manufacturing	https://red.ht/3XTy2qw
S7	Husarnet: Connected Things Without a Cloud	https://bit.ly/3XN0hHu
S8	How to use digital twins for IoT device configurations	https://bit.ly/3kodBEz
S9	Mainflux 0.11 — Digital Twin, MQTT Proxy And More	https://bit.ly/3xLbEoU
S10	Connecting OPC UA Publisher to Amazon AWS IoT with MQTT	https://bit.ly/3klcDJi
S11	IoT Telemetry Collection using Google Protocol Buffers, Google Cloud Functions, Cloud Pub/Sub, and MongoDB Atlas	https://bit.ly/3Zb1h9A
S12	Gathering system health telemetry data from AWS IoT Greengrass core devices	https://go.aws/3YZBmlC
S13	Digital Twins: Components, Use Cases, and Implementation Tips	https://bit.ly/3lZNyUH
S14	If You Build Products, You Should Be Using Digital Twins	https://bit.ly/3Sj8r9v
S15	Choose a device communication protocol	https://bit.ly/3SnEqW2
S16	Through edge-to-cloud integration framework	https://bit.ly/3Zs1iFI
S17	Send cloud-to-device messages from an IoT hub	https://bit.ly/3lSGRUl
S18	Stream Processing with IoT Data: Challenges, Best Practices, and Techniques	https://bit.ly/3ILxtLd
S19	Intelligence at the Edge Part 3: Edge Node Communication	https://bit.ly/3ZesxUI
S20	7 patterns for IoT data ingestion and visualization- How to decide what works best for your use case	https://go.aws/3YUNMLg
S21	How does a digital twin work?	https://ibm.co/3ZaZxgy
S22	Cloud Edge Computing: Beyond the Data Center	https://bit.ly/3Inl92j
S23	Understand Azure IoT Edge modules	https://bit.ly/3Ew9sFz
S24	Understand and use device twins in IoT Hub	https://bit.ly/3KqHWwU
S25	Understand and use module twins in IoT Hub	https://bit.ly/3xJCYDP
S26	How a Cloud Integration Platform Can Help Your Business	https://bit.ly/3nmHwy1
S27	Edge-to-cloud communication	https://bit.ly/3khYPiL
S28	Device connectivity	https://ibm.co/41vHgfZ
S29	How the IoT is creating today's hottest tech job: Edge analytics	https://bit.ly/3lZe8xh
S30	Edge Computing Architecture	https://bit.ly/3xTdwvz
S31	The Hark Platform	https://bit.ly/3xKFfik
S32	IoT Gateway User Guide	https://bit.ly/3InyJTx
S33	How to structure data ingestion and aggregation pipelines	https://bit.ly/3StSSMb
S34	What Is Streaming Data Integration?	https://bit.ly/3IIFPDp
S35	Plan your IoT real-time data streaming process	https://bit.ly/3EumGSZ
S36	What Is an Integration Platform? Do I Need One?	https://ibm.co/3kU52Sh
S37	What is Data Streaming?	https://bit.ly/3yrJrDI

selective-coding stage, we integrated and grouped the categories into a central core model. We continuously compared the newly generated codes with the existing model throughout the process to enhance it gradually. A crucial aspect of qualitative methods is determining when to conclude this process. For this purpose, we adopted the concept of theoretical saturation [5], which is widely recognized in the field. We concluded our analysis when twelve additional knowledge sources did not contribute new insights to our understanding of the research topic. Although our approach to theoretical saturation was conservative compared to typical qualitative research, our study had already reached a point of convergence after examining twenty-five knowledge sources. The details of the sources included in our study can be found in Table I. Our search for sources was based on our experience, i.e., tools, methods, patterns, and practices we have access to, worked with, or studied before. We also used major search engines (e.g., Google, Bing) and topic portals (e.g., InfoQ) to find more sources.

B. *Modelling Tool Implementation*

To create our decision model, we used our existing modeling tool Codeable Models[2]. This tool is a Python implementation that precisely specifies metamodels, models, and model instances in code. We specified metamodels for components, activities, and deployments as outlined above. In addition, we realized automated constraint checkers and PlantUML[3] code generators to generate initial graphical visualizations of all metamodel and model instances.

C. *Metatmodel*

Fig. 2 shows our metamodel in data communication of CPS architectures. *Model Element* is our root concept. A *Category* groups several *Decisions* with *Contexts* used. *Decisions* are colored white and *Contexts* green in our ADDs for better

[2]https://github.com/uzdun/CodeableModels
[3]https://plantuml.com

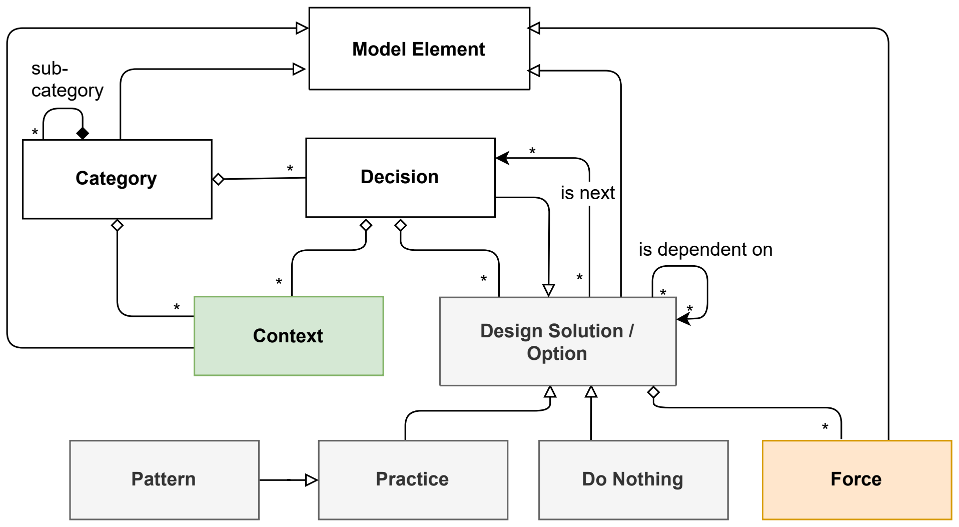


Fig. 2: Metamodel of the Reusable ADDs

readability. A *Design Solution / Option* has known use cases and can be different *Patterns* and *Practices*. These are colored gray in our models. A special option is *Do Nothing*, encountered in practice repeatedly. There are multiple *Decision Drivers (Forces)* for design solutions. These forces help the architects choose different options of each decision. We colored the forces in orange in our ADD models.

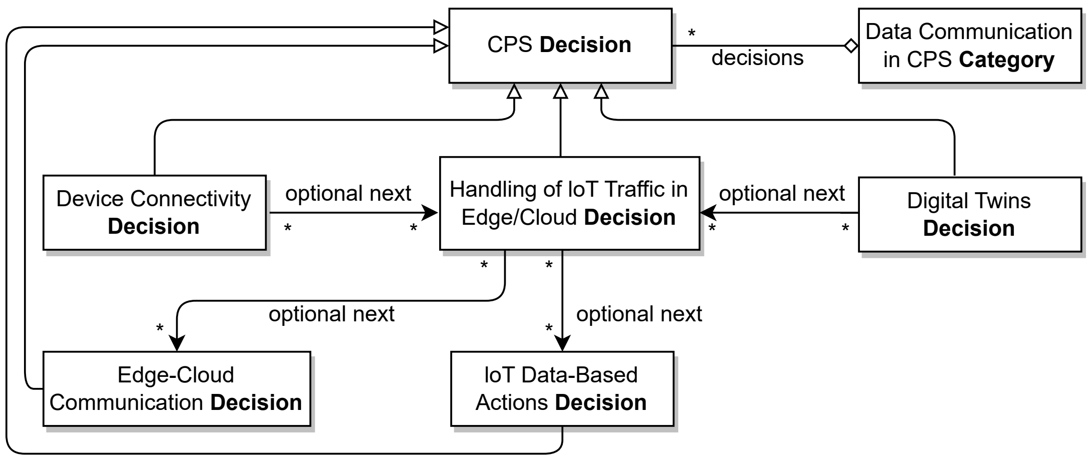
IV. Reusable ADD Model for Data Communication in CPS Architectures

This section presents the reusable ADD model based on our study. The model consists of a single decision category, the *Data Communication in CPS Category*, which comprises five top-level decisions, as depicted in Fig. 3. These ADDs are: *Digital Twins Decision*, *Device Connectivity Decision*, *Handling of IoT Traffic in Edge-Cloud Decision*, *Edge-Cloud Communication Decision*, and *IoT Data-Based Actions Decision*. We explain each ADD separately.

A. Digital Twins Decision

Digital Twins refer to a virtual representation or simulation of a physical object, system, or process. The sources mostly related to this decision are the following in Table I: S5, S8, S9, S13, S14, S21, S24, and S25. Digital twins encompass the essential attributes, behavior, and characteristics of the physical entity. This process creates a dynamic and interactive virtual counterpart, closely mirroring the real-world object in (near) real-time integrating data, sensor information, and computational models. Digital twins act as a vital link between the physical and digital realms, enabling analysis, monitoring, and optimization of the physical entity's performance, behavior, and maintenance throughout its entire lifecycle. Fig. 4 presents the details of this ADD. There are two options: To use *Digital Twin* and *No Digital Twin*.

Utilization of digital twins can benefit the following attributes: *Flexibility* refers to the ability of a system to adjust and respond to evolving circumstances, changing requirements,


CPS Decision
decisions
Data Communication in CPS Category
Device Connectivity Decision
optional next
Handling of IoT Traffic in Edge/Cloud Decision
optional next
Digital Twins Decision
optional next
optional next
Edge-Cloud Communication Decision
IoT Data-Based Actions Decision
Fig. 3: Overview of the Reusable ADD Model for CPS Data Communication

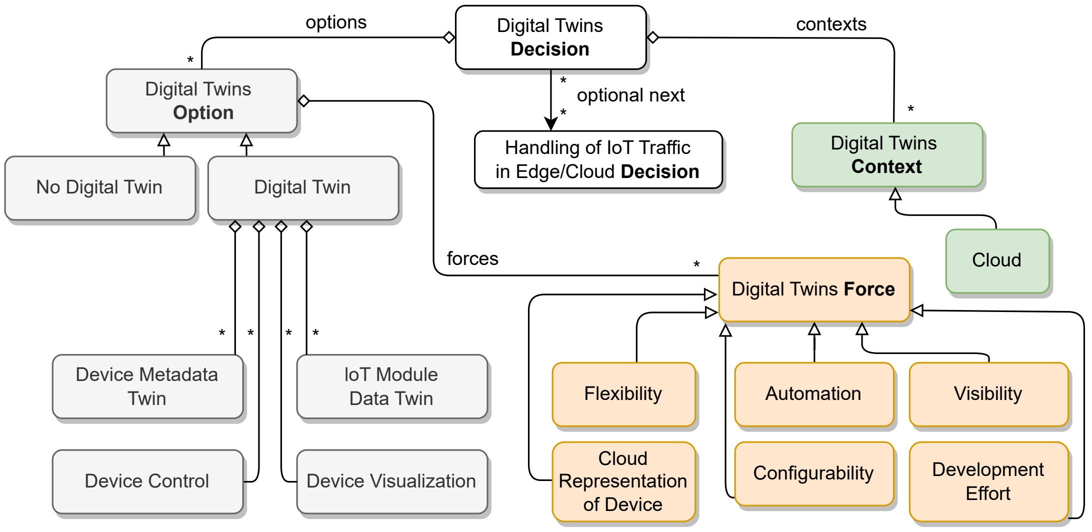


Fig. 4: ADD: Digital Twins Decision

or the needs of its users. *Automation* refers to leveraging technology to execute tasks or processes that require human involvement. *Visibility* refers to the capacity of a system to deliver pertinent and timely information regarding its status, performance, and behavior to its users or operators. *Cloud Representation of Device* refers to a cloud-hosted platform encompassing a virtualized physical device representation. *Configurability* refers to the extent to which a system can be tailored or adjusted to fulfill its users' unique requirements or preferences, and *Development Effort* refers to the investment of time, resources, and expertise needed to develop a system.

There are multiple digital twin tasks. *Device Metadata Twin* and *IoT Module Data Twin*, where the data of IoT devices are communicated over the cloud. Moreover, *Device Control* and *Device Visualization* refers to the integration of digital twins. This decision can be decided in each *Digital Twins Context*, which is integrated into the *Cloud*. Device Configuration, Device Metadata Twin and IoT Module Data Twin benefit the *flexibility* to modify and adjust its characteristics and behavior to effectively respond to novel requirements or alterations in the physical system it represents, and *automation* to automate tasks and processes. Device Visualization positively impacts *visibility* to monitor and visualize the performance and behavior of physical systems in a digital form.

B. Device Connectivity Decision

Device Connectivity refers to the ability of IoT devices to establish and maintain network connections with other devices or systems. The sources mostly related to this decision are the following in Table I: S1, S7, S15, S19, S23, S27, and S28. Device connectivity seamlessly integrates various devices, sensors, actuators, and other components into a

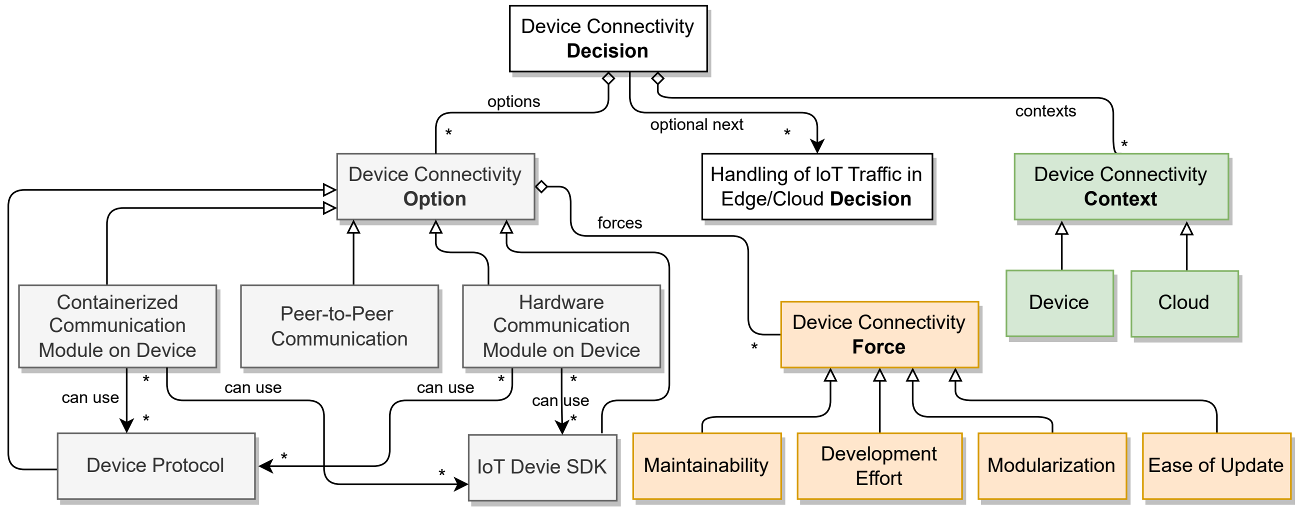

Device Connectivity Decision
options
optional next
contexts
Device Connectivity Option
Handling of IoT Traffic in Edge/Cloud Decision
Device Connectivity Context
forces
Containerized Communication Module on Device
Peer-to-Peer Communication
Hardware Communication Module on Device
Device Connectivity Force
Device
Cloud
can use
can use
can use
can use
Device Protocol
IoT Devie SDK
Maintainability
Development Effort
Modularization
Ease of Update

Fig. 5: ADD: Device Connectivity Decision

unified network, allowing them to communicate and exchange data. This decision enables the IoT ecosystem to function by facilitating data transmission, remote control, and collaboration among interconnected devices, both locally and over the Internet. It involves wireless communication protocols, network infrastructure, and Application Programming Interfaces (APIs) that enable reliable and secure connectivity between IoT devices and between devices and IoT platforms or applications. Fig. 5 gives the details of this ADD. Different options exist: Using *Hardware Communication Module on Device* or *Containerized Communication Module on Device*, where devices communicate through the IoT integration platform. These modules can use *Device Protocol* and *IoT Device SDK* to communicate information. Alternatively, a *Peer-to-Peer Communication* eliminates the need for the IoT platform and enables direct communication between devices.

There are different forces and deciding factors such as *Maintainability*, *Development Effort*, *Modularization*, and *Ease of Update*. All these forces are improved when there is a containerized module on a device. However, a hardware module on a device or a device using direct communication with other devices negatively affects maintainability and ease of update. This decision can be made in the context of a *Device* and the *Cloud* when using an integration platform.

C. Handling of IoT Traffic in Edge-Cloud Decision

As detailed in Fig. 6, this decision concerns further processing of IoT traffic, e.g., for analytics or runtime metrics assurance, such as quality of service requirements. The sources mostly related to this decision are the following in Table I: S2, S3, S9, S10, S11, S12, S17, S29, and S32. We explain two variants of this decision in our gray literature sources. The S2 source uses MQTT/AMQP-based messaging or a *Device Protocol* for handling the IoT traffic on the edge when integrating using the Microsoft Azure IoT Edge platform. This decision must be made in a *Device* and an *Edge* context. We can select one of the options for each device/edge component combination. Two options are possible: A *Direct Connection with Messaging* if messaging is supported on the device, or a *Device Protocol* for the connection and then integrating via the *Messaging API Gateway* pattern [10], [18]. An alternative option would be to *Directly Connect with Device Protocol*, but this is not recommended.

The S3 source models a more complex variant of the same decision.The two options are a *Multi-Protocol API Gateway* or the *Messaging API Gateway* pattern, both API Gateway patterns. As alternatives, the *Direct Connection with Device Protocol* and the *Direct Connection with Messaging* were considered. Here, the combination of any *Device* and the *Cloud* needs to be considered as the decision context, as this case is about direct device-to-cloud integration. Four forces are modeled, which are possible drivers for this decision: *Maintainability*, *Development Effort*, *Coupling*, and *Communication Efficiency*. The direct communication options affect all forces negatively. However, API gateways positively influence coupling, communication efficiency, and maintainability but increase development efforts. Finally, two next decision relations are modeled: *Edge-Cloud Communication* and *IoT Data-Based Actions*.

D. Edge-Cloud Communication Decision

Edge-Cloud Communication involves edge devices sending data to cloud services for storage, processing, and analysis. The sources mostly related to this decision are the following in Table I: S4, S9, S10, S11, S17, S18, S35, and S37. There is a difference in the focus of this decision compared to the previous decision. In this ADD, we model different patterns for communicating the IoT data to the cloud as explained in the following: This decision is commonly used in IoT scenarios where edge devices, such as sensors or controllers, collect data and transmit it to the cloud for analysis and decision-making. Cloud-based services can provide advanced analytics, visualization, and reporting capabilities that are not feasible at the edge due to limited processing and storage resources. IoT devices can communicate with back-end services in the cloud

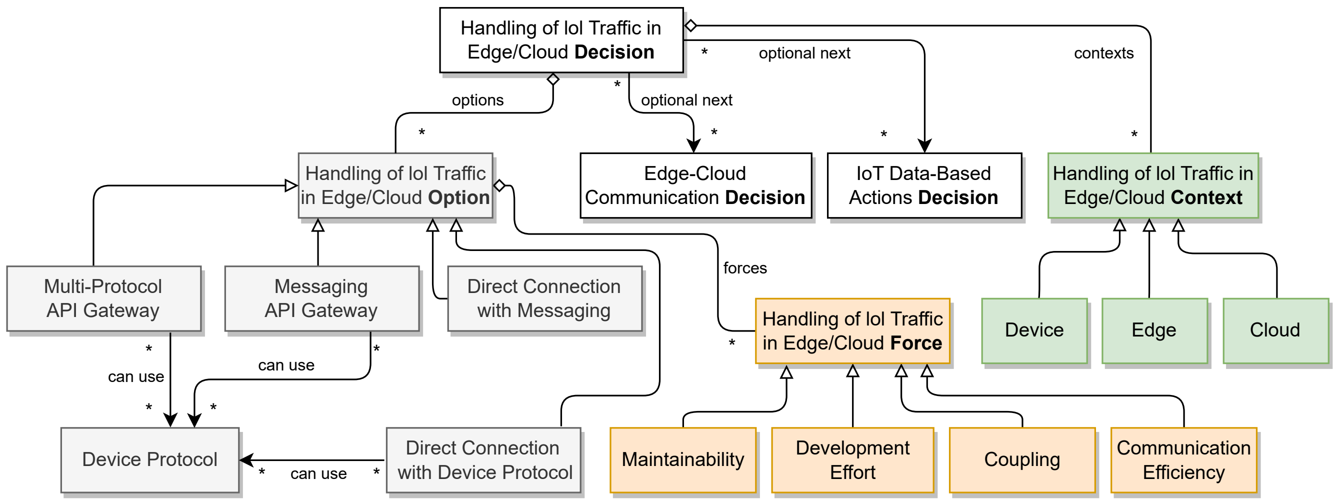

Handling of IoI Traffic in Edge/Cloud **Decision**
options
optional next
optional next
contexts
Handling of IoI Traffic in Edge/Cloud **Option**
Edge-Cloud Communication **Decision**
IoT Data-Based Actions **Decision**
Handling of IoI Traffic in Edge/Cloud **Context**
Multi-Protocol API Gateway
Messaging API Gateway
Direct Connection with Messaging
forces
Handling of IoI Traffic in Edge/Cloud **Force**
Device
Edge
Cloud
can use
can use
can use
Device Protocol
Direct Connection with Device Protocol
Maintainability
Development Effort
Coupling
Communication Efficiency

Fig. 6: ADD: Handling of IoT Traffic in Edge-Cloud Decision

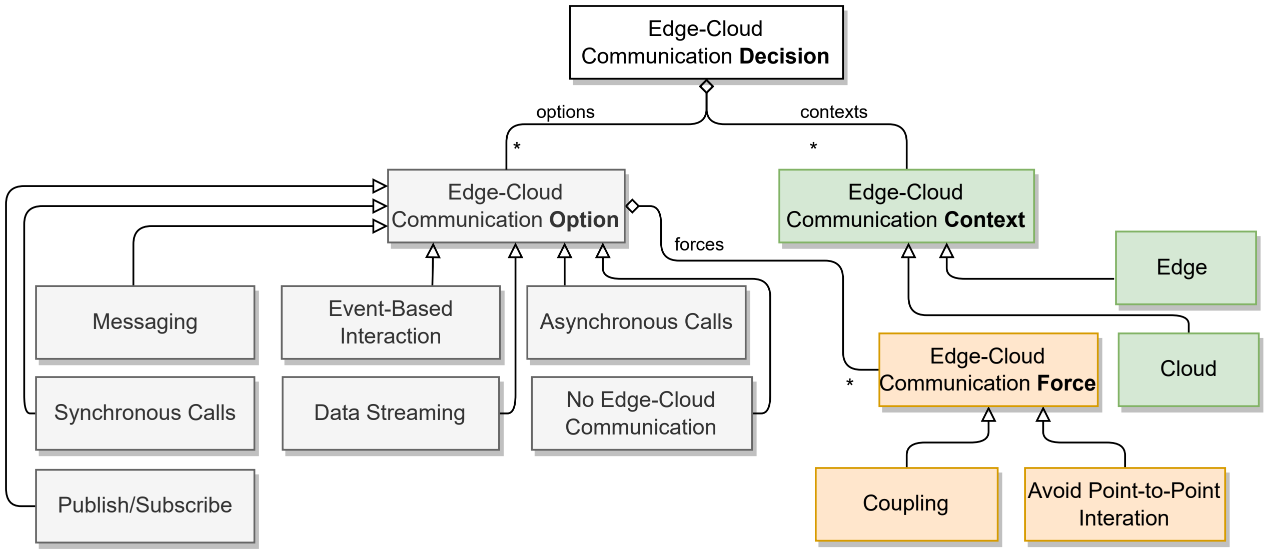

Edge-Cloud Communication Decision
options
contexts
*
*
Edge-Cloud Communication Option
Edge-Cloud Communication Context
forces
Edge
Messaging
Event-Based Interaction
Asynchronous Calls
Edge-Cloud Communication Force
Cloud
*
Synchronous Calls
Data Streaming
No Edge-Cloud Communication
Coupling
Avoid Point-to-Point Interation
Publish/Subscribe

Fig. 7: ADD: Edge-Cloud Communication Decision

through telemetry and command messages. These devices have unique characteristics, such as being embedded systems, having limited power and processing resources, and being located in remote areas with intermittent network connectivity.

Fig. 7 shows the details of this ADD. There are various communication patterns and practices. The simplest option is *No Edge-Cloud Communication*. Another pattern is *Messaging*, which is a lightweight transport protocol ideal for connecting remote devices; *Publish/Subscribe* pattern uses topics to identify messages and route them to publishing and subscribing clients; *Synchronous Calls* offers a straightforward means of exchanging data between edge and cloud. This pattern [11] involves sending a request, processing it, and returning a response message. A similar pattern is the *Asynchronous Calls* which implements the request/response asynchronously. The pattern *Data Streaming* refers to a continuous flow of data that is generated, transmitted, and received in real-time. *Event-based Interaction* is also an asynchronous pattern that ensures that all changes to the state are stored as a sequence of events. The outcome of this decision depends on several criteria. If a system utilizes the patterns associated with asynchronous communication, it can lead to more loosely coupled interactions, as it eliminates the need for point-to-point interactions. This decision can be made in the context of *Edge* and *Cloud*.

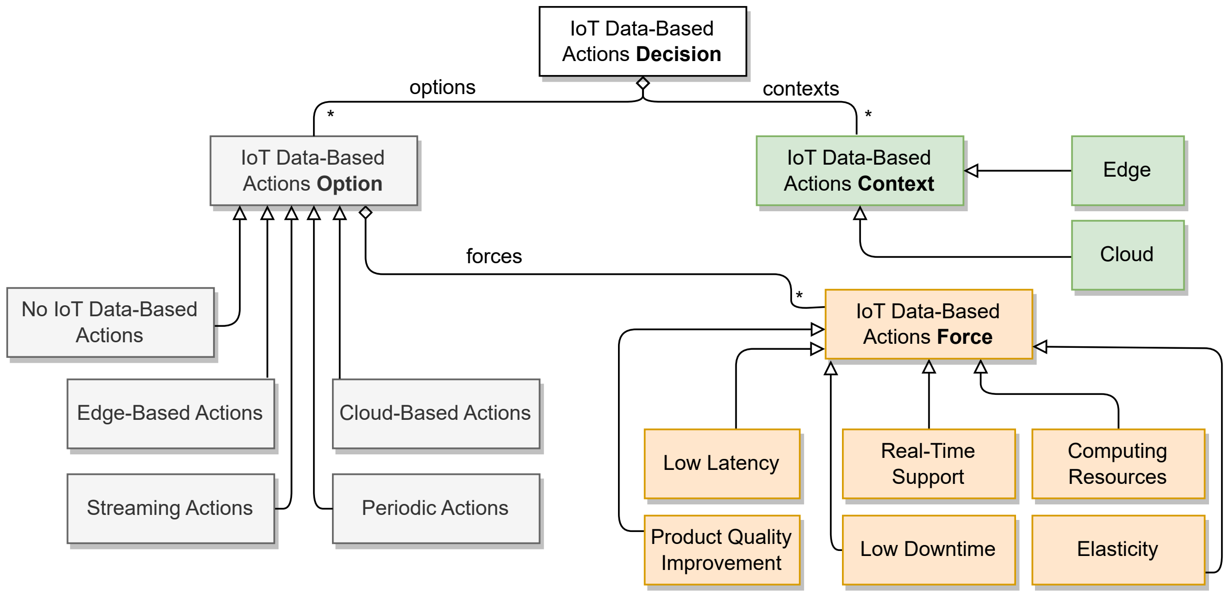


Fig. 8: ADD: IoT Data-Based Actions Decision

E. IoT Data-Based Actions Decision

This decision concerns the cloud-based actions triggered by an IoT device and the IoT data integration in the cloud. The sources mostly related to this decision are the following in Table I: S11, S12, S16, S17, S18, S26, and S33 to S37. An example action is using sensor data from a smart home security system to trigger automated actions. For instance, if a motion sensor detects movement in a specific area of the house while the occupants are away, the system can automatically alert the homeowner and activate surveillance cameras to start recording. Additionally, it can lock all doors and turn on the alarm system to ensure the security of the premises. These actions are based on real-time data collected from IoT devices, enabling immediate responses to potential security threats. As shown in Fig. 8, this decision has different options: One option is *No IoT Data-Based Actions*. Other options include *Edge-Based Actions* and *Cloud-Based Actions*, which consider where to process the IoT device data. Additionally, *Streaming Actions* and *Periodic Actions* consider how the data is transmitted to the cloud.

Multiple forces exist: *Low Latency, Real-Time Support, Computing Resources, Product Quality Improvement, Low Downtime* and *Elasticity*. On the one hand, the *Edge-Based Actions* have lower latency than cloud-based actions, but these edge devices also have lower computing resources. On the other hand, *Cloud-Based Actions* offer real-time support, elasticity, and low downtime, improving product quality. This decision can be made in the context of *Edge* and *Cloud*.

V. Discussion

This section presents the evaluation of our study and discusses our findings in the context of the research questions stated in Section I.

A. Evaluation

We created a reusable ADD model, presented in Fig. 3, based on the gray-literature sources following the sequence in Table I. We named the ADD model elements using the terminology from the respective sources and generated generic type names based on these element names. Whenever a new type name arose, we compared it against the existing names and determined whether the new type name was required. For example, one decision driver (force) of *Digital Twins Decision* was given as *Design and Implementation Effort*. This force is repeatedly used in the *Device Connectivity Decision* and the *Handling of IoT Traffic in Edge-Cloud Decision* as *Development Effort*. Therefore, we renamed the force for consistency.

As illustrated in Fig. 9, the theoretical saturation point was attained after incorporating 25 sources. In the initial thirteen sources, we had to modify the designated type names frequently. However, in the following twelve sources, such changes were less frequent. No further modifications and additions were necessary for the remaining sources.

B. Our Findings regarding the Research Questions

RQ1: After analyzing 37 practitioner knowledge sources, we discovered evidence for 27 patterns and practices currently used by practitioners for supporting data communication in CPS architectures. These patterns and practices are associated with ADDs and were found to be largely independent of each other. An exception is the options of the following two

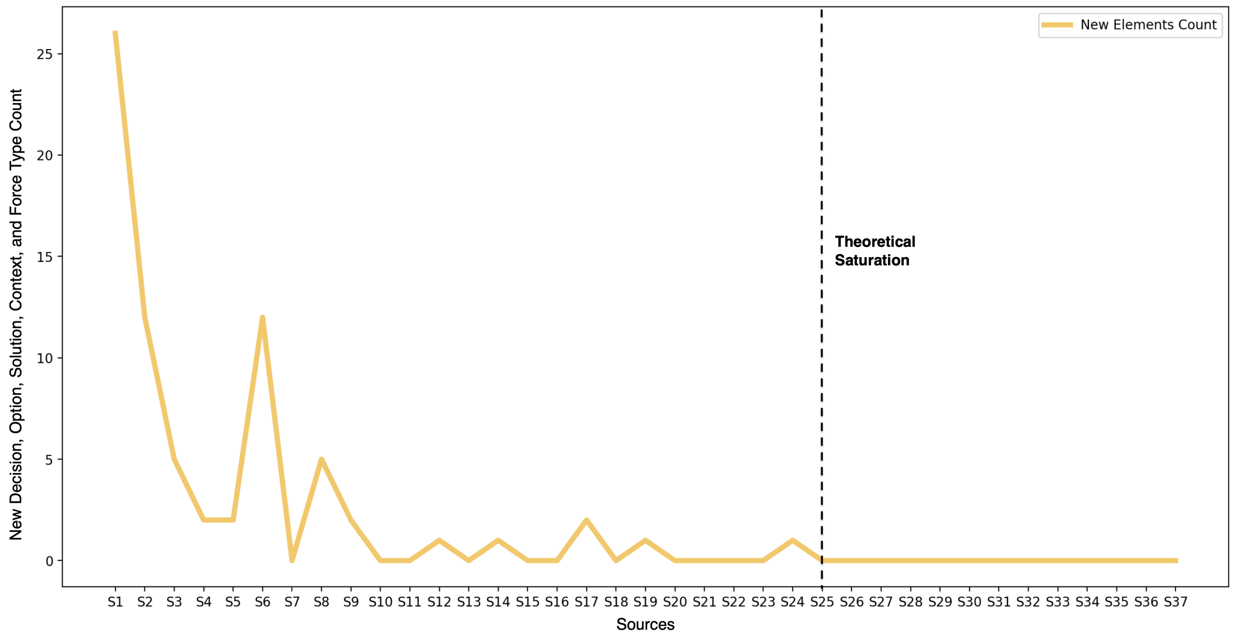


Fig. 9: Number of Elements in Newly-Added Sources

decisions: *Edge-Cloud Communication Decision* and *Handling of IoT Traffic in Edge-Cloud Decision*. These two decisions are closely related to each other. The former focuses on the patterns concerning the sending of IoT data to the cloud. The latter focuses on the analytics and quality-of-service requirements when processing IoT data. Practitioners can choose to make these decisions in the cloud or on the edge device according to the need of an application.

RQ2: As shown in Fig. 3, we identified the *Data Communication in CPS Category* that incorporates 5 top-level ADDs supporting data communication in CPS architectures. Our research revealed that the *Handling of IoT Traffic in Edge-Cloud* decision is a central ADD. Other decisions either have this ADD as an optional next, e.g., the *Device Connectivity Decision* and *Digital Twins Decision*, or are a direct optional next of this decision, e.g., the *IoT Data-Based Actions* decision. Therefore, we recommend that practitioners consider this decision early in the CPS architectural planning, as changing the architectural choice for this decision might have severe impacts in the later stages of CPS-application development. For example, practitioners can decide on a *Messaging API Gateway* or a *Multi-Protocol API Gateway* pattern as early as possible (see Fig. 6).

RQ3: Our research helped us discover 17 influencing factors (forces) when architecting CPSs in data communication from the practitioners' perspective. We found that these forces were generally fairly specific to the individual ADDs and decision options but identified some common to multiple ADDs and their options.

- *Maintainability* and *Development Effort* apply to the following three ADDs: *Device Connectivity Decision*, *Handling of IoT Traffic in Edge-Cloud Decision* and *Digital Twins Decision*.
- *Coupling* applies to decision options for both the *Handling of IoT Traffic in Edge-Cloud Decision* and the *Edge-Cloud Communication Decision*.

We recommend that the practitioners consider these forces early in the architectural planning of a CPS system. This is because the above forces are central to multiple ADDs and their respective options, as explained above.

VI. Threats to Validity

We discuss the threats to validity based on the threat types by Wohlin et al. [22].

A. Construct Validity

This threat concerns the accurate representation of the intended construct by a measurement. The experience and search-based procedure for finding knowledge sources may have introduced bias. However, this threat is mitigated largely by the chosen research method, which requires additional sources corresponding to the inclusion and exclusion criteria, not a specific distribution of sources. In this regard, our procedure is similar to how interview partners are typically found in qualitative research studies in software engineering. The threat remains that our procedures introduced the unconscious exclusion of certain sources. We mitigated this threat by assembling an author team with many years of experience in the field and performing general and broad searches.

B. Internal Validity

It concerns factors that affect the independent variables concerning causality. To increase internal validity, we used practitioner reports produced independently of our study. This avoids bias, for example, compared to interviews in which the practitioners would be aware that their answers would be used in a study. This introduces the internal validity threat that some important information might be missing in the reports, which could have been revealed in an interview. We tried to mitigate this threat by looking at many more sources than needed for theoretical saturation, as all different sources are unlikely to miss the same essential information.

C. External Validity

It concerns threats that limit the ability to generalize the results beyond the experiment. Due to the many included sources, our results can be generalized to many kinds of architecture requiring data integration in CPS architectures. However, the threat to external validity remains that our results only apply to similar CPS architectures. The generalization to novel CPS architectures might not be possible without modification of our models.

D. Conclusion Validity

This threat concerns factors that affect the ability to conclude the relations between treatments and study outcomes. We studied many gray-literature sources and reached theoretical saturation after studying 25 sources. Moreover, we continued studying more sources to ensure that the gray literature does not introduce new concepts in the data-communication domain for CPS architectures not covered in our ADD model.

VII. Conclusion and Future Work

We conducted a comprehensive study using Grounded Theory (GT) and analyzed the grey literature to develop a model for data integration in CPS architectures. This model encompasses Architecture Design Decisions (ADDs), decision options, relations, and decision drivers. Through our research, we identified interconnected design choices crucial for effective data integration in CPS architectures, such as *Device Connectivity, Digital Twins, Handling of IoT Traffic in Edge/Cloud, Edge/Cloud Communication* and *IoT Data-Based Actions*. Each of these design choices offers multiple options with varying implications.

To support practitioners in their decision-making process, we created a UML-based model that enables the evaluation of specific decision drivers for each option. By providing a structured framework, our model assists practitioners in making informed decisions while contributing to advancing scientific knowledge in this domain. Applying our ADDs can simplify the decision-making process for architects and engineers working on CPS architectures. It allows for the evaluation of architectural practices, the identification of adherence to established guidelines, and the detection of potential anti-patterns.

Our research lays the groundwork for further exploration and advancements in data communication of CPS architectures. For our future work, we plan to apply our reusable ADD model presented in Fig. 3 when architecting CPS applications. Moreover, we plan to define a framework to detect anti-patterns automatically and give recommendations on how to follow the established design patterns. For this purpose, we plan to use machine-learning techniques. We have already published a tool paper with such a recommender tool [1] in the CPS data-communication domain.

REFERENCES

[1] Amiri, A., Ntetnos, E., Zdun, U., Geiger, S.: Tool support for learning architectural guidance models and pattern mining from architectural design decision models. In: Proceedings of the 27th European Conference on Pattern Languages of Programs (2023)

[2] Charmaz, K.: Constructing grounded theory. Sage (2014)

[3] Coplien, J.: Software Patterns: Management Briefings. SIGS, New York (1996)

[4] Ghosh, A., Edwards, D., Hosseini, M.R.: Patterns and trends in internet of things (iot) research: future applications in the construction industry. Engineering, Construction and Architectural Management (08 2020). https://doi.org/10.1108/ECAM-04-2020-0271

[5] Glaser, B.G., Strauss, A.L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. de Gruyter (1967)

[6] Gorton, I., Klein, J., Nurgaliev, A.: Architecture knowledge for evaluating scalable databases. In: Proc. of the 12th Working IEEE/IFIP Conference on Software Architecture. pp. 95–104 (2015)

[7] van Heesch, U., Avgeriou, P., Hilliard, R.: A documentation framework for architecture decisions. J. Syst. Softw. **85**(4), 795 – 820 (2012)

[8] Henneke, D., Elattar, M., Jasperneite, J.: Communication patterns for cyber-physical systems. In: 2015 IEEE 20th Conference on Emerging Technologies and Factory Automation (ETFA). pp. 1–4 (2015). https://doi.org/10.1109/ETFA.2015.7301623

[9] Hentrich, C., Zdun, U., Hlupic, V., Dotsika, F.: An Approach for Pattern Mining Through Grounded Theory Techniques and Its Applications to Process-driven SOA Patterns. In: Proc. of the 18th European Conference on Pattern Languages of Program. pp. 9:1–9:16 (2015)

[10] Hohpe, G., Woolf, B.: Enterprise Integration Patterns. Addison-Wesley (2003)

[11] Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley (2003)

[12] Jamaludin, J., Rohani, J.M.: Cyber-physical system (cps): State of the art. In: 2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube). pp. 1–5 (2018). https://doi.org/10.1109/ICECUBE.2018.8610996

[13] Lytra, I., Sobernig, S., Zdun, U.: Architectural Decision Making for Service-Based Platform Integration: A Qualitative Multi-Method Study. In: Proc. of WICSA/ECSA (2012)

[14] Musil, A., Musil, J., Weyns, D., Bures, T., Muccini, H., Sharaf, M.: Patterns for self-adaptation in cyber-physical systems. Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects pp. 331–368 (2017)

[15] Pautasso, C., Zimmermann, O., Leymann, F.: RESTful Web Services vs. Big Web Services: Making the right architectural decision. In: Proc. of the 17th World Wide Web Conference. pp. 805–814 (2008)

[16] Pontes, P., Lima, B., Faria, J.: Test patterns for iot. pp. 63–66 (11 2018). https://doi.org/10.1145/3278186.3278196

[17] Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., Riegg, A.: Internet of things patterns. pp. 1–21 (07 2016). https://doi.org/10.1145/3011784.3011789

[18] Richardson, C.: A pattern language for microservices. http://microservices.io/patterns/index.html (2017)

[19] Sha, L., Meseguer, J.: Design of Complex Cyber Physical Systems with Formalized Architectural Patterns, pp. 92–100. Springer (2008). https://doi.org/10.1007/978-3-540-89437-7_5

[20] Warnett, S.J., Zdun, U.: Architectural design decisions for machine learning deployment. In: 19th IEEE International Conference on Software Architecture (ICSA 2022) (2022), http://eprints.cs.univie.ac.at/7270/

[21] Washizaki, H., Ogata, S., Hazeyama, A., Okubo, T., Fernández, E., Yoshioka, N.: Landscape of architecture and design patterns for iot systems. IEEE Internet of Things Journal **PP**, 1–1 (06 2020). https://doi.org/10.1109/JIOT.2020.3003528

[22] Wohlin, C., Runeson, P., Hoest, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering. Springer (2012)

[23] Zdun, U., Stocker, M., Zimmermann, O., Pautasso, C., Lübke, D.: Supporting Architectural Decision Making on Quality Aspects of Microservice APIs. In: 16th International Conference on Service-Oriented Computing. Springer (2018)

[24] Zimmermann, O., Koehler, J., Leymann, F., Polley, R., Schuster, N.: Managing architectural decision models with dependency relations, integrity constraints, and production rules. J. Syst. Softw. **82**(8), 1249–1267 (2009)