Decision-Making Support for Data Integration in Cyber-Physical-System Architectures

Evangelos Ntentos[1], Amirali Amiri[1], Stephen Warnett[1], and Uwe Zdun[2]

[1] University of Vienna, Faculty of Computer Science, Software Architecture Group, Doctoral School Computer Science, Vienna, Austria
`firstname.lastname@univie.ac.at`

[2] University of Vienna, Faculty of Computer Science, Software Architecture Group, Vienna, Austria
`firstname.lastname@univie.ac.at`

Abstract. Cyber-Physical Systems (CPS) design is a complex challenge involving physical and digital components working together to accomplish a specific goal. Integrating such systems involves combining data from various distributed Internet of Things (IoT) devices and cloud services to create meaningful insights and actions. Service-based IoT data integration involves several steps: collection, processing, analysis, and visualization. Adopting a holistic approach that considers physical and digital aspects is crucial when designing data integration in distributed CPS. Architectural design decisions are vital in shaping a CPS' functionality and system qualities, such as performance, security, and reliability. Although several patterns and practices for CPS architecture have been proposed, much of the knowledge in this area is informally discussed in the grey literature, e.g., in practitioner blogs and system documentation. As a result, this architectural knowledge is dispersed across many sources that are often inconsistent and based on personal experience. In this study, we present the results of a qualitative, in-depth study of the best practices and patterns of distributed CPS architecture as described by practitioners. We have developed a formal architecture decision model using a model-based qualitative research method. We aim to bridge the science-practice gap, enhance comprehension of practitioners' CPS approaches, and provide decision-making support.

Keywords: Architectural Design Decisions, Cyber-Physical Systems, Data Integration, Software Architecture, Grounded Theory

1 Introduction

Several authors have attempted to document patterns and best practices related to distributed CPS [6,8,10,14]. However, these works focus on applying published patterns or scientific results. In contrast, established industry practices are primarily found in grey literature like blogs, experience reports, and system documentation. While these sources offer some understanding of existing practices, they lack systematic architectural guidance. The reported practices vary and rely on personal experience, creating uncertainty and risk in CPS design. One needs extensive experience or a comprehensive study of knowledge sources to address this. We aim to provide a more complete and consistent view of current industrial practices, complementing existing knowledge.

We conducted an in-depth qualitative study of CPS descriptions provided by practitioners. These descriptions contain informal information about established practices and patterns in distributed CPS. Following a model-based qualitative research method [18], we systematically analyzed the practitioner sources using coding and constant comparison methods [3], followed by precise software modeling. This allowed us to develop a detailed software model of established practices, patterns, and their relationships. This paper aims to study the following research questions:

– **RQ1** What are the patterns and practices currently used by practitioners for supporting data integration in CPS architectures?
– **RQ2** How are the current data integration patterns and practices related? In particular, which architectural design decisions (ADDs) are relevant when architecting data integration in CPS?
– **RQ3** What are the influencing factors (i.e., decision drivers) in architecting data integration in CPS in the eye of the practitioner today?

This paper has three key contributions. Firstly, we conducted a qualitative study on CPS architectures, analyzing 37 knowledge sources to identify established industrial practices, patterns, relationships, and decision drivers. Secondly, we created a formal architectural design decision (ADD) model. The model encompasses *7* decisions, *31* decision options, and *22* decision drivers. Lastly, we evaluated the model's level of detail and completeness, demonstrating that our research method provides a more comprehensive examination of established practices than informal pattern mining. Our approach, derived from practitioners' perspectives, offers valuable insights into distributed CPS design.

The rest of this paper is structured as follows: In Section 2, we compare our work to the related work. Section 3 explains the research methods we have applied in our study and summarizes the knowledge sources. Section 4 describes our reusable ADD model on CPS. Section 5 evaluates and Section 6 discusses our results. Finally, Section 7 considers the threats to the validity of our study, and Section 8 summarizes our findings.

2 Related Work

Several approaches that study CPS patterns and practices exist: Jamaludin et al. [8] present a comprehensive overview of CPS state of the art and highlight the importance of understanding CPS characteristics and architectures. This knowledge is crucial for designing and implementing CPS systems that can meet the requirements of various applications and ensure their reliability and adaptability. Henneke et al. [6] focus on analyzing communication patterns for CPS, such as discovery, request-response, and publish/subscribe. Reinfurt et al. [13] present five patterns that address various problems derived by examining numerous production-ready IoT offerings to identify recurring proven solution principles. Washizaki et al. [16] conducted a systematic literature review, identifying 32 papers from which 143 IoT architecture and design patterns were extracted. These patterns were analyzed based on various characteristics, and directions for improvements in publishing and adopting IoT patterns were outlined. Pontes et al.[12] introduced the Pattern-Based IoT Testing approach to simplify and organize the testing process for IoT ecosystems. This approach uses testing tactics that target common behavior patterns in the system, referred to as “IoT Test Patterns.” Ghosh et al. [2] evaluated the current state of IoT research and discovered that existing studies were limited, biased, and subjective. Their study utilized a thorough qualitative approach to systematically analyze the grey literature on CPS to tackle this issue, providing the first comprehensive analysis.

There are several decision documentation-related approaches (e.g., for service-oriented solutions [19], service-based platform integration [9], REST vs. SOAP [11], and big data repositories [4]). However, this kind of research does not yet encompass CPS architectures. Warnett and Zdun [15] present a Grounded Theory-based approach to current practitioner understanding and architectural concepts of ML solution deployment. They formulated seven ADDs along with various relations between them. In particular, they modeled twenty-six decision options and forty-four decision drivers in ML deployment. Other authors have combined decision models with formal view models [5]. We improve these techniques with a formal modeling approach derived from qualitative research methodology.

Our study analyzes practitioner methods and techniques to bridge the gap between theory and practice in CPS data integration. Our formal model includes ADDs, decision options, practices, drivers, and relationships and aims to provide insights to help practitioners make informed data integration decisions in CPS.

3 Research Method

This section discusses the research method followed in this study and the modeling tool we used to create and visualize the decision model.

This paper aims to systematically study established practices in data integration architecture within CPS architectures. We utilize a model-based qualitative research method described in [18], which combines Grounded Theory (GT) [3] with pattern mining techniques (e.g., [1]) and their integration with GT [7]. This approach involves iterative steps of data interpretation to construct a theory based on the collected data. Data analysis is performed concurrently with data collection rather than afterward.

Constant comparison is a crucial aspect of GT, where researchers continuously compare existing and new data, identifying abstract concepts. These concepts are organized into categories and linked with properties and relationships. This iterative process guides subsequent research iterations.

Our knowledge-mining procedure involved searching for new sources, applying coding techniques to identify model elements and decision drivers, and continuously comparing codes with the existing model to improve it. We stopped the analysis using the concept of theoretical saturation, where additional sources did not contribute anything new. Our study had already converged after twenty-five sources. The sources used are summarized in Table 1, and our search relied on our experience with relevant tools, methods, patterns, and practices.

We employed three types of coding in our methodology:

- Open coding, which involves developing concepts based on the data, asking specific and consistent questions, precise and consistent coding, and memo writing with minimal assumptions.
- Axial coding, which entails developing categories and linking data, concepts, categories, and properties.
- Selective coding, which focuses on integrating developed categories and grouping them around a central core category.

Figure 1 illustrates our research method steps. To gather practitioner sources, we used popular search engines like Google, StartPage, and DuckDuckGo, along with topic portals like InfoQ and DZone. Initial search terms aligned with our focus, such as "CPS data integration." GT coding practices and constant comparison were employed iteratively to identify concepts, categories, properties, and relationships. The decision model was developed using the CodeableModels tool, a Python-based modeling tool[3] that enabled precise definition of meta-models, models, and instances in code.

Subsequent iterations involved searches using relevant terms based on identified topics from previous iterations, focusing on areas requiring coding and their potential contributions to the model. Practitioner articles were selected based on relevance to the topic and not primarily promotional in nature, with both authors reviewing and approving the source selection.

During the coding process, open coding transformed conceptual details into labels, while axial coding identified recurring and related concepts. Each source was carefully examined line by line, with memos documenting thought processes, interpretations, and reasoning for traceability. Selective coding extracted main ideas and refined previous sources. Formal UML-based modeling was employed for axial and selective coding, resulting in a precise and consistent theory represented as a UML model. Theoretical saturation was reached when approximately twelve additional sources no longer significantly contributed to our model. A summary of our knowledge sources can be found in Table 1.

[3] `https://github.com/uzdun/CodeableModels`

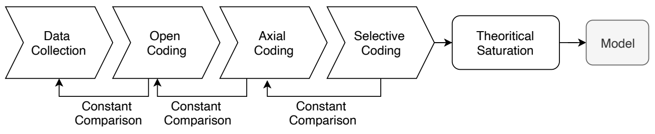

Fig. 1. Research Method Steps

4 Reusable ADD Model for Data Integration in CPS Architectures

This section presents the reusable ADD model we derived based on our study (see the data[4]). Figure 2 depicts the meta-model for the ADD models. This model encompasses the Decisions within an ADD model. Each Decision is associated with a Context, represented by a domain object that signifies the specific system part or aspect to which the decision applies. Decisions consist of Options and all options are categorized as Solutions. Each option is accompanied by Forces, which may have an impact on the decision. Furthermore, Decisions, Solutions, and Options can have Relations among them. A Solution can be linked to another Solution, with the condition that either the source or target of the relation must be an Option. It is essential for all Solutions in the model to be directly linked to a Decision, either through the Decision itself or through other Options. Decisions and Options can also have next-decision relations, indicating their sequential order or dependency.

The reusable ADD model consists of a single decision category, the *Data Integration in CPS Category*, which comprises seven top-level decisions, as depicted in Fig. 3. It is worth noting that all elements of our model are instances of a meta-model, with meta-classes such as *Decision*, *Category*, etc., which are also included in the model descriptions below. Also note that our model consists of concepts representing decisions, decision options, practices, patterns, and forces arising from our sources while applying our research methodology. These emergent concepts, appropriately named in our model, may be traced back to the referenced sources.

IoT Data Stream Integration and IoT Data Stream Integration Tasks (Fig. 4). *IoT data stream integration* combines and processes data from multiple devices or sensors to extract meaningful insights and enable better decision-making [S2, S3, S4, S33, S34, S35, S37]. It involves several steps, including data acquisition, prepossessing, analysis, and visualization. Several practices and patterns exist. *Edge-based IoT Data Stream Integration* is a decentralized practice for processing and analyzing IoT data. The data is processed closer to the source rather than transmitted to a central server or cloud-based service [S22, S3, S33]. Alternatively, *Cloud-based IoT Data Stream Integration* is a centralized option for processing and analyzing IoT data, where the data is transmitted to

[4] `https://doi.org/10.5281/zenodo.8367400`

Table 1. 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`

a remote server or cloud-based service for processing and analysis [S2, S26, S3, S33]. Another option is *Peer-to-peer (P2P) based IoT Integration*, which is a decentralized practice of connecting IoT devices and integrating their data [S2, S3, S4, S33, S34, S35]. In a P2P network, devices communicate directly with

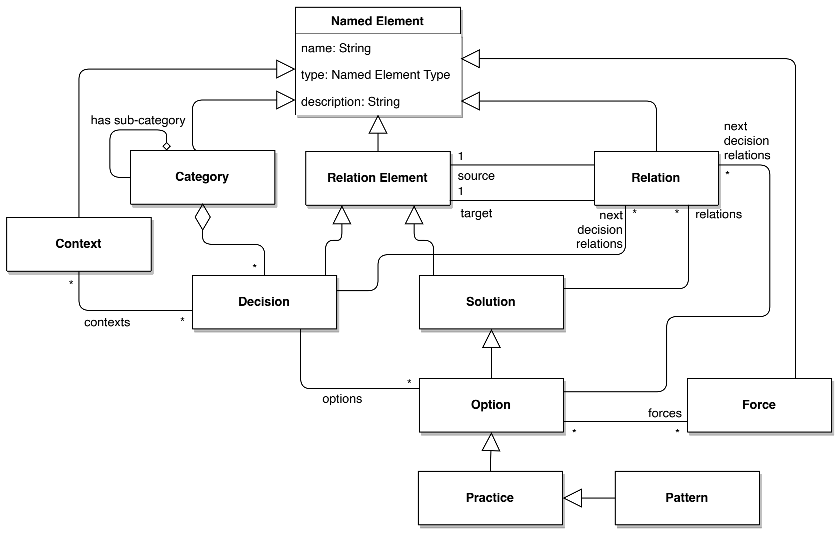

Fig. 2. Meta-model for ADD Models

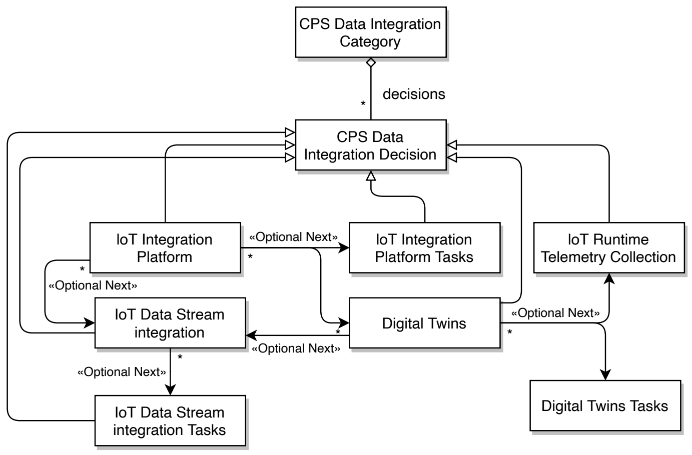
CPS Data Integration Category
decisions
CPS Data Integration Decision
IoT Integration Platform
IoT Integration Platform Tasks
IoT Runtime Telemetry Collection
IoT Data Stream integration
Digital Twins
IoT Data Stream integration Tasks
Digital Twins Tasks
«Optional Next»
*

Fig. 3. Reusable ADD Model on Data Integration in CPS Architectures: Overview

each other without the need for a central server or cloud-based service. Finally, some systems require *No IoT Data Stream Integration*.

Integrating the copious amounts of data IoT devices generate into a larger system can prove daunting [S1, S3, S4, S18, S33, S34, S35, S37]. Data stream integration involves several essential tasks, including *Data Gathering*, which involves collecting data from various sources to gain insights, make informed decisions, or optimize business operations [S1, S3, S4, S18, S20, S35]. *Data Normalization* arranges data in a database to reduce redundancy and enhance consistency [S1, S3, S4, S18, S20]. *Data Filtering* involves selecting a subset of data from a larger dataset based on specific criteria. *Data Aggregation* merges data from multiple sources or groups into a concise summary view [S1, S3, S4, S18, S20]. Lastly, *Data Anomaly Detection* entails identifying patterns or data points within a dataset that deviates from expected behavior [S18, S4, S20]. The decision context is the *IoT Data Stream*, i.e., the decision has to be taken for every IoT data stream separately.

Several factors influence the decision outcome. For instance, P2P and edge-based IoT data stream integration offer advantages such as shorter *development time*, *resilience*, as well as increased *data security* and *privacy* [S33, S34, S35]. Cloud-based IoT data stream integration may have higher network *latency* and may be more susceptible to *data security*, *integrity* and *privacy* concerns. However, *scalability* is an advantage of this practice [S2, S3, S4, S34, S35, S37].

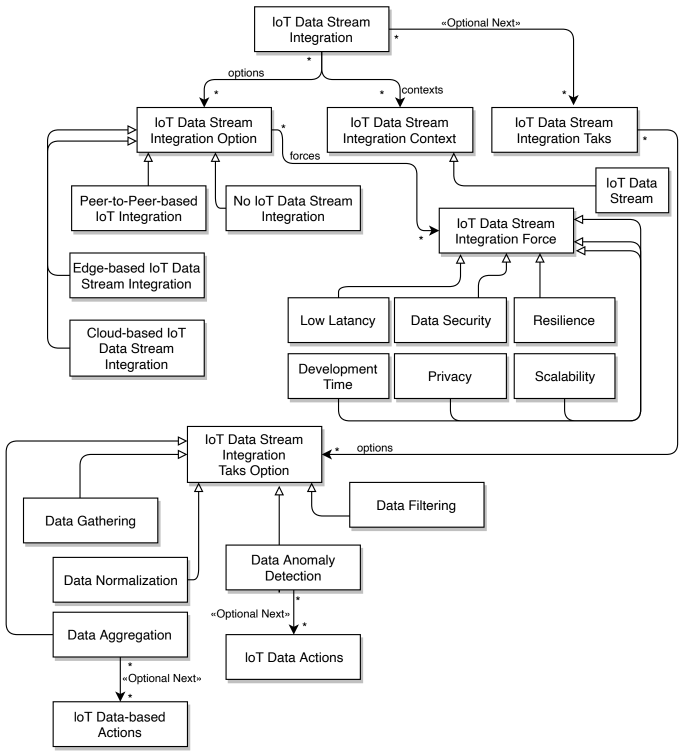
IoT Data Stream Integration
«Optional Next»
options
contexts
forces
IoT Data Stream Integration Option
IoT Data Stream Integration Context
IoT Data Stream Integration Taks
Peer-to-Peer-based IoT Integration
No IoT Data Stream Integration
IoT Data Stream
IoT Data Stream Integration Force
Edge-based IoT Data Stream Integration
Cloud-based IoT Data Stream Integration
Low Latancy
Data Security
Resilience
Development Time
Privacy
Scalability
IoT Data Stream Integration Taks Option
options
Data Gathering
Data Filtering
Data Normalization
Data Anomaly Detection
Data Aggregation
«Optional Next»
IoT Data Actions
«Optional Next»
IoT Data-based Actions

Fig. 4. IoT Data Stream Integration and IoT Data Stream Integration Tasks Decision

IoT Integration Platform and IoT Integration Platform Tasks (Fig. 5). An *IoT integration platform* is a software solution enabling connecting and communicating between different devices and systems in an IoT ecosystem [S1, S16, S20, S26, S30, S36]. It provides a central hub for managing and controlling IoT devices, data, and applications [S16, S20, S21, S26, S30, S36]. IoT integration platforms typically offer a range of features and functionalities, such as device management, data analytics, security and authentication, communication protocols, and integration with third-party systems and applications [S36, S20]. Different patterns and practices are used for connecting and integrating systems, applications, and services. One option is to use a *Cloud Integration Platform From Cloud Vendor* that seamlessly connects cloud-based applications and services [S1, S26, S30, S36]. Alternatively, *Edge Integration Platform from a Cloud Vendor* can integrate edge devices and systems with their cloud-based applications and services [S2, S30, S23]. Another practice is *Open/Standardized Cloud Integration Platform* that provides standard protocols, interfaces, and tools to connect and integrate different systems, applications, and services [S2, S30, S26]. Similarly, the *Open/Standardized Edge Integration Platform* simplifies the integration process by offering a common framework that can be used across edge devices, vendors, and domains [S1, S2, S23, S26, S30, S36].

When deciding on an *IoT integration platform*, the following tasks should be considered. One possible platform task is *Install and Update Device Workloads*, which involves deploying new software or updates to devices in a system [S20, S23, S26, S30, S36]. Additionally, updating device workloads requires careful planning and testing to ensure that updates are applied smoothly and do not cause downtime. *Establish Security* is another practice that involves implementing measures to protect against unauthorized access, data breaches, and other potential security threats [S36, S20]. Another practice is *Monitoring* referring to the ongoing observation and analysis of a system's performance and behavior. *Health Checking* evaluates the health status of a system, service, or application [S2, S20, S23]. The practice *Managing Device Communication* involves ensuring that devices can communicate with each other effectively and securely [S23, S3, S28]. *Edge/Cloud Platform Integration* involves integrating edge devices with cloud platforms to enable seamless data exchange, processing, and analysis between edge and cloud [S6, S9, S23]. Both decisions are made in the context of *IoT Edge* and *Cloud Computing.*

According to sources [S26, S30, S36], the Cloud Integration Platform and Edge Integration Platform provided by the cloud vendor affect system *evolvability.* They also impact *vendor lock-in* [S2, S23, S30]. Conversely, Open/Standardized Cloud Integration Platform and Open/Standardized Edge Integration Platform have a positive influence on *interoperability* [S1, S16, S26] and *configuration effect*, ensuring effective implementation of system configuration changes without negatively affecting performance or stability [S16, S26, S30, S36]. Moreover, Installing and Updating Device Workloads and Edge/Cloud Platform Integration contribute to *compatibility.* Monitoring and health checking ensure data *integrity* [S30, S31, S36]. Lastly, Managing Device Communication can impact the *security.*

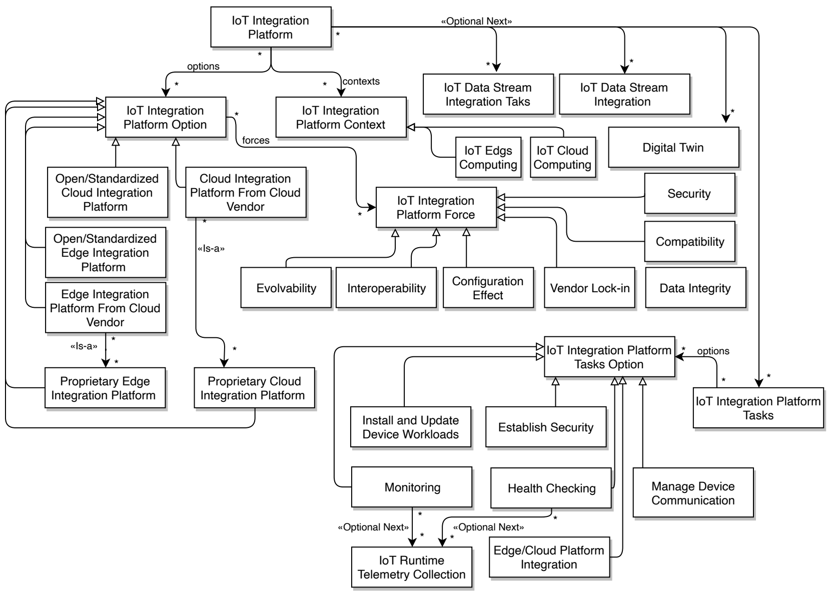
IoT Integration Platform
«Optional Next»
options
contexts
IoT Data Stream Integration Taks
IoT Data Stream Integration
IoT Integration Platform Option
IoT Integration Platform Context
forces
IoT Edgs Computing
IoT Cloud Computing
Digital Twin
Open/Standardized Cloud Integration Platform
Cloud Integration Platform From Cloud Vendor
IoT Integration Platform Force
Security
Compatibility
«Is-a»
Open/Standardized Edge Integration Platform
Evolvability
Interoperability
Configuration Effect
Vendor Lock-in
Data Integrity
Edge Integration Platform From Cloud Vendor
«Is-a»
IoT Integration Platform Tasks Option
options
Proprietary Edge Integration Platform
Proprietary Cloud Integration Platform
IoT Integration Platform Tasks
Install and Update Device Workloads
Establish Security
Monitoring
Health Checking
Manage Device Communication
«Optional Next»
«Optional Next»
IoT Runtime Telemetry Collection
Edge/Cloud Platform Integration

Fig. 5. IoT Integration Platform and IoT Integration Platform Taks Decision

Digital Twins and Digital Twins Tasks (Fig. 6). *Digital Twins* refer to virtual representations of physical objects, systems, or processes [S9, S21]. They are created using data from sensors, IoT devices, and other sources that collect data on the object or system in question [S21]. The digital twin mimics the physical object or system in real-time, allowing for better monitoring, analysis, and optimization [S5, S8, S9, S13, S14, S21, S24, S25]. Digital Twins enable remote monitoring, predictive maintenance, and provide insights into performance. The relation between digital twins and CPS data integration is that digital twins are an integral part of the data integration process in CPS. There are two options regarding this decision; one is to use *Digital Twin* and *No Digital Twin*. This decision can be decided in each *IoT Data Stream* context. If *Digital Twin* is chosen, the *Digital Twin Tasks* decision is an important follow-up decision on the tasks the twin shall fulfill [S5, S8, S9, S13, S14, S20, S21]. A *Device Metadata Twin* is a type of digital twin that reflects a physical device's metadata and configuration information, providing a virtual representation of the device for monitoring, management, and maintenance purposes [S5, S14, S21]. The *IoT Module Data Twin* practice involves creating a digital twin that mimics the behavior and data of an IoT module [S13, S14, S21]. *Device Visualization* transforms physical devices into digital representations displayed on dashboards for easier monitoring, management, and interaction [S14, S20, S21]. *Device Control* enables remote management and operation of physical devices through a digital interface, including functions like power control, settings adjustment, and other necessary operations [S14, S24, S25]. *Device Configuration* involves the setup and customization of physical devices [S21, S24, S25]. This decision can be made for each *IoT Data Stream* where a digital twin is applied.

Device Metadata Twin and IoT Module Data Twin benefit the *flexibility* for adapting and changing in response to new requirements or changes in the physical system it represents, as well as *automation* for automating tasks and processes [S5, S8, S9, S14, S20, S21]. Device Visualization positively impacts *visibility* for monitoring and visualizing the performance and behavior of physical systems in a digital form [S20, S21]. Device Configuration and Device Control practices benefits *Configurability.*

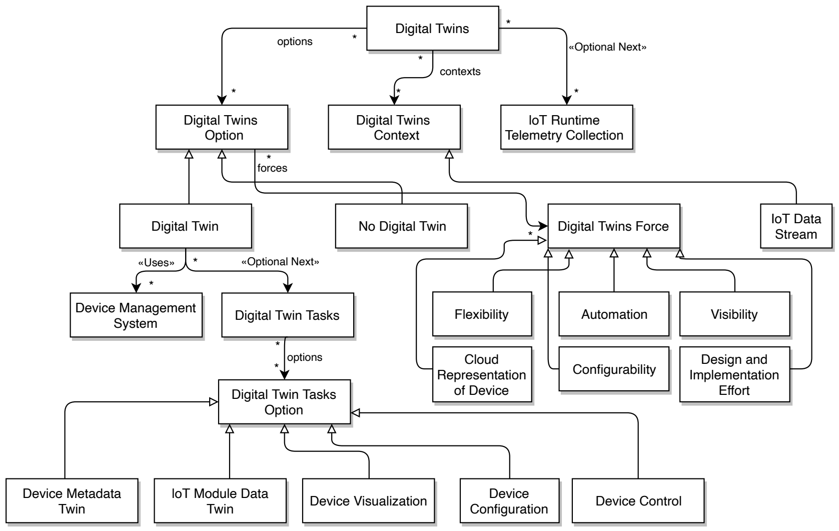

Fig. 6. Digital Twins and Digital Twins Tasks Decision

IoT Runtime Telemetry Collection (Fig. 7). IoT runtime telemetry collection involves collecting and analyzing data related to the performance and behavior of IoT devices and systems while in operation [S2, S11, S12, S15, S17, S19, S27, S28]. The telemetry data includes device status, network connectivity, sensor readings, and other performance indicators. Options for implementation include not collecting telemetry, using cloud-based or edge-based runtime telemetry collection. The option *No Runtime Telemetry Collection* is the most straightforward. *Cloud-based Runtime Telemetry Collection* gathers data and metrics from cloud-based applications and systems for analysis, collected in a central platform [S2, S11, S12, S15, S27]. On the other hand, *Edge-based Runtime Telemetry Collection* [S2, S11, S12, S15, S27] is a practice that involves the collection of runtime data from various devices and systems located at the edge of a network. This practice can use *Device Configuration* to set up and customize a device's settings to meet specific requirements [S21, S24, S25]. This decision can be made for each *IoT Data Stream*.

Cloud-based Runtime Telemetry Collection and Edge-based Runtime Telemetry Collection can benefit *product quality improvement* [S12, S15, S27]. Furthermore, it enables *monitoring* of connected devices to detect anomalies and prevent downtime while providing insights into device performance and usage to optimize operations and improve *efficiency*. Additionally, by analyzing telemetry data, organizations can predict when maintenance is needed to improve *reliability*.

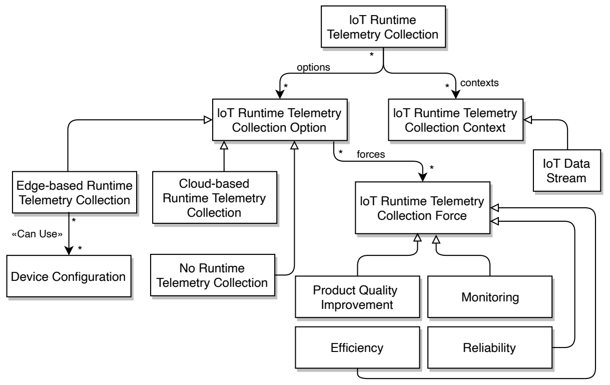
IoT Runtime Telemetry Collection
options
contexts
IoT Runtime Telemetry Collection Option
IoT Runtime Telemetry Collection Context
IoT Data Stream
forces
Edge-based Runtime Telemetry Collection
Cloud-based Runtime Telemetry Collection
IoT Runtime Telemetry Collection Force
«Can Use»
Device Configuration
No Runtime Telemetry Collection
Product Quality Improvement
Monitoring
Efficiency
Reliability

Fig. 7. IoT Runtime Telemetry Collection

5 Evaluation

We meticulously constructed an ADD model based on the chosen sources following the sequence presented in Table 1. 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. As illustrated in Figure 8, the theoretical saturation point was attained after incorporating twenty-five 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.

6 Discussion

This section discusses our findings for the research questions from Section 1.
RQ1: After analyzing *37* practitioner knowledge sources, we discovered evidence for *31* patterns and practices currently used by practitioners for supporting data

integration in CPS architectures, which we modeled as ADD decision options. These patterns and practices are associated with ADDs and were found to be largely independent of each other. An exception is the *Edge-based Runtime Telemetry Collection* practice, which can use the *Device Configuration* practice. Another commonality is that the *IoT Data Stream Integration*, *IoT Integration Platform*, *IoT Integration Platform Tasks* and *IoT Runtime Telemetry Collection* decisions all offer decision edge-based and cloud-based decision options. Depending on the specific needs, the practitioner may wish to mix and match these edge and cloud-based practices.

During our research, we discovered a subtle aspect that the practitioner should take note of. Both the decisions for *Digital Twins Tasks* and*IoT Runtime Telemetry Collection*include*Device Configuration* as an option, while the decision for *IoT Data Stream Integration* offers *P2P-based IoT Integration*. It is important to highlight that the latter decision option, despite its different description, may still involve device configuration implicitly. Therefore, when designing CPS, the practitioner should consider this potential overlap.

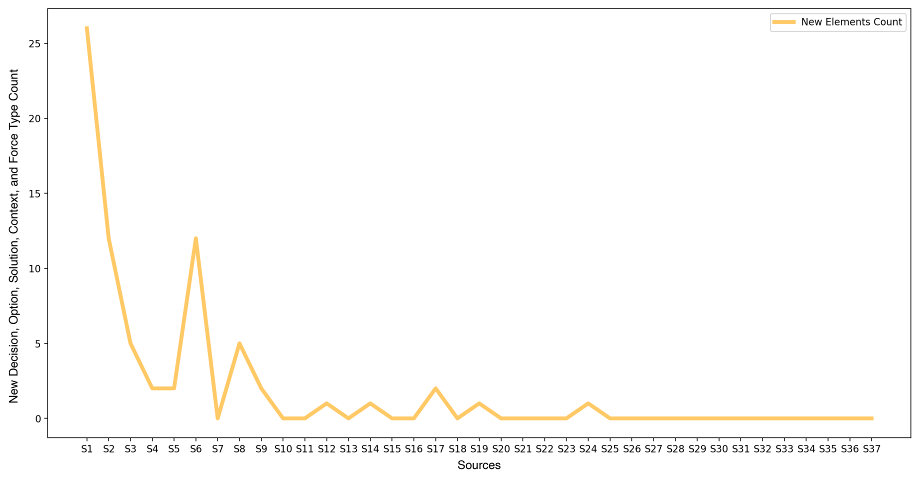

Fig. 8. Number of Elements of Newly-Added Sources

RQ2: Given the central *CPS Data Integration* decision, we identified *7* top-level ADDs for supporting data integration in CPS architectures. Our research revealed subtle relations between ADDs and decision options, which may not be immediately apparent. For instance, the seemingly loosely-related *Data Anomaly Detection*, *Digital Twins*, *IoT Runtime Telemetry Collection*, *Cloud-based IoT Data Stream Integration* decisions all are applied in the *IoT Data Stream* context, which is an important consideration during the planning of a CPS architecture.
RQ3: Our research helped us discover *22* influencing factors (forces) when architecting CPS in the context of data integration 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. For example, *Flexibility*, *Automation*, *Visibility* apply to the *Digital Twins* decision and, assuming the *Digital Twin* decision option is selected, then also the decision options for the *Digital Twin Tasks* decision; *Device Configuration* applies to decision options for both the *Digital Twins Tasks* decision and the *IoT Runtime Telemetry Collection* decision; *Security* applies especially to decision options associated with *IoT Data Stream Integration*, *IoT Data Stream Integration Tasks* and *IoT Integration Platform Tasks* decisions, but must be considered throughout.

Since the above forces are central to multiple ADDs and their respective decision options, the practitioner may wish to consider the significance of these forces early in the architectural planning of a system and be guided accordingly.

7 Threats to Validity

We discuss the threats to validity based on the threat types by Wohlin et al. [17].

To enhance internal validity, we used independent practitioner reports instead of interviews to avoid bias. However, interviews could have revealed important information that might be missing in reports. To address this, we extensively examined diverse sources, exceeding what was necessary.

To minimize researcher bias, different team members cross-checked all models independently. Yet, a potential threat to internal validity remains due to possible biases within the research team. This applies to our coding procedure and formal modeling as well, where different researchers might have used different approaches.

The experience and search-based procedure for knowledge sources may introduce bias. However, our research method primarily relied on additional sources adhering to specific criteria, mitigating this threat. Nonetheless, there is still a potential threat of unconsciously excluding certain sources, which we addressed by assembling an experienced author team and conducting comprehensive searches.

Our results can likely be generalized to various types of architectures involving data integration in CPS. However, there is a threat to external validity, indicating that our findings are applicable only to similar CPS architectures. Generalizing to novel or unconventional architectures may require modifications to our models.

8 Conclusion

We conducted a GT-based grey literature study to create a model for data integration in CPS architectures that included ADDs, decision options, relations, and decision drivers. Our research focused on supporting data integration in CPS architectures and addressed three research questions. For RQ1, we analyzed 37 practitioner knowledge sources and identified 31 patterns and practices used by practitioners in data integration. These patterns and practices were modeled as ADD decision options and were found to be largely independent. We also highlighted the relationships between certain practices and the flexibility of mixing edge-based and cloud-based approaches. RQ2 explored the top-level ADDs for data integration in CPS architectures. Based on the central CPS Data Integration decision, we identified seven top-level ADDs and revealed subtle relationships between them. Understanding the shared contexts and dependencies among these ADDs is crucial during CPS architectural planning. In RQ3, we identified 22 influencing factors (forces) that impact CPS architecture design in the context of data integration. These forces varied across individual ADDs and their options.

This paper proposes a promising approach that systematically and impartially studies multiple sources and integrates findings through formal modeling. By following this methodology, potential issues can be mitigated, and a rigorous and unbiased understanding of current practices in specific fields, like data integration in CSP architecture, can be obtained.

9 Acknowledgements

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

References

1. Coplien, J.: Software Patterns: Management Briefings. SIGS, New York (1996)
2. 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 **ahead-of-print** (08 2020)
3. Glaser, B.G., Strauss, A.L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. de Gruyter (1967)
4. 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)
5. van Heesch, U., Avgeriou, P., Hilliard, R.: A documentation framework for architecture decisions. J. Syst. Softw. **85**(4), 795 – 820 (2012)
6. 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)
7. 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)
8. 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
9. 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)

10. 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)
11. 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)
12. Pontes, P., Lima, B., Faria, J.: Test patterns for iot. pp. 63–66 (11 2018)
13. 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
14. Sha, L., Meseguer, J.: Design of Complex Cyber Physical Systems with Formalized Architectural Patterns, pp. 92–100. Springer (2008)
15. 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/`
16. 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
17. Wohlin, C., Runeson, P., Hoest, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering. Springer (2012)
18. 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)
19. 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)