EXECUTING DATA QUALITY MANAGEMENT PROCESSES ON IOT SOFTWARE SYSTEMS USING A COLLECTION OF SOFTWARE COMPONENTS

Gabriel Rodrigues Caldas de Aquino

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Prof. Claudio Miceli de Farias

Rio de Janeiro
Março de 2025

EXECUTING DATA QUALITY MANAGEMENT PROCESSES ON IOT SOFTWARE SYSTEMS USING A COLLECTION OF SOFTWARE COMPONENTS

Gabriel Rodrigues Caldas de Aquino

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Prof. Claudio Miceli de Farias

Aprovada por: Prof. Guilherme Horta Travassos
Prof. Rafael Maiani de Mello
Prof. Antonio Augusto de Aragão Rocha
Prof. Daniel Sadoc Menasché

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2025

Rodrigues Caldas de Aquino, Gabriel

Executing Data Quality Management Processes on IoT Software Systems Using a Collection of Software Components/Gabriel Rodrigues Caldas de Aquino. – Rio de Janeiro: UFRJ/COPPE, 2025.

XV, 150 p.: il.; 29, 7cm.

Orientador: Prof. Claudio Miceli de Farias

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 115 – 125.

1. Internet of Things (IoT). 2. Data Quality. 3. IoT Smart Spaces. 4. IoT Software Systems. I. Miceli de Farias, Prof. Claudio. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico esta tese à minha avó
Nancy, que sinto muita saudade.

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado as oportunidades que eu tive e por ter encontrado as pessoas que eu encontrei pelo meu caminho. Agradeço à minha avó Nancy, a quem eu dedico esta tese, que infelizmente não está mais aqui, mas que foi fundamental para que eu chegasse onde eu cheguei e pudesse ser quem eu sou hoje, muito obrigado! Sinto muito a sua falta. Agradeço à toda a minha família, principalmente aos meus pais Luiz e Lúcia por tudo, vocês são parte fundamental e sem vocês eu jamais teria chegado onde eu cheguei. Agradeço à minha irmã Mariana, ao Pedro, ao Benjamin e à Clara por estarem sempre ali para quando eu precisava conversar, me distrair e relaxar um pouco a cabeça.

Agradeço ao meu amigo Gabriel Martins, que sempre esteve lá para me ajudar, debater conceitos do meu trabalho e ajudar a enriquecer vários dos conceitos dessa tese. As horas intermináveis de discussão e debates sobre inúmeros conceitos ajudou a solidificar bastante este trabalho. Agradeço também a toda sua família, em especial à Noelle, Joaquim e a Beth, que sempre foram um ponto seguro. Agradeço por estarem sempre ali de portas abertas. Muito obrigado!

Agradeço ao meu orientador Claudio Miceli de Farias, que esteve comigo desde o meu TCC até hoje, sempre acreditou no meu trabalho, me incentivou e é fundamental para que essa tese exista. Muito obrigado mesmo! Sem a sua orientação nada disso seria possível. Agradeço à todos do LabNet, em especial a professora Luci, que abriu as portas do laboratório para mim. Agradeço à todos os amigos de laboratório que sempre estavam disponíveis para conversar sobre a tese, principalmente ao Marcos, Álvaro e Kopp, todos as conversas e debates foram muito importantes. Agradeço ao grupo de Engenharia de Software Experimental do PESC/COPPE e a todos do Lens que sempre me ajudaram no meu trabalho. Um agradecimento especial professor Guilherme Horta Travassos, que abriu as portas da COPPE.

Agradeço à SG-TIC. Em particular, agradeço à Ana Maria de Almeida Ribeiro, superintendente da SG-TIC, minha chefe quando eu fui diretor da Divisão de Segurança da Informação, que me ajudou e me incentivou no momento mais importante do meu doutorado. Gostaria de agradecer ao Nimai, que foi meu chefe quando estava na Divisão de Infraestrutura e Redes, e me possibilitou estudar ainda lá no início das minhas pesquisas. Agradeço ao Felipe Ribas, chefe da Divisão de Segurança da

Informação e que possibilitou e permitiu os meus estudos no doutorado. Agradeço à todos os funcionários da SEGTIC, e, em especial, e de coração, agradeço à Patrícia e Lilian que sempre me ajudaram nesta jornada.

Por fim, esta tese encerra um ciclo na minha vida iniciado lá em 2007/2 quando entrei para o curso de Ciência da Computação na UFRJ. Agradeço a todas as pessoas que cruzaram o meu caminho. Em especial aos professores do IC, do PPGI e do PESC que me ajudaram nesta jornada, durante a graduação e o mestrado. Agradeço aos profissionais do NCE que me ajudaram quando era estagiário. Muito obrigado!

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

EXECUTING DATA QUALITY MANAGEMENT PROCESSES ON IOT SOFTWARE SYSTEMS USING A COLLECTION OF SOFTWARE COMPONENTS

Gabriel Rodrigues Caldas de Aquino

Março/2025

Orientador: Prof. Claudio Miceli de Farias

Programa: Engenharia de Sistemas e Computação

Esta tese apresenta uma arquitetura de software para a gestão da qualidade de dados em sistemas de software IoT operando em IoT *smart spaces* utilizando processos de gestão de qualidade de dados. Para operacionalizar essa gestão na IoT, introduzimos um conjunto de componentes de software projetados para executar os processos de qualidade de dados de forma autônoma, minimizando a intervenção humana. Além disso, as técnicas propostas incluem o pré-processamento de dados na camada de dispositivos IoT, onde os dados são coletados e pré-processados para evitar a propagação de problemas de qualidade de dados. Também introduzimos uma técnica para garantir a conformidade da qualidade dos dados na camada de sistemas de software IoT, assegurando que os dados atendam aos requisitos de qualidade antes de serem utilizados. Neste trabalho apresentamos diretrizes para a implementação da solução em *IoT smart spaces*, cobrindo todas as fases, desde a definição dos requisitos até a implantação e configuração dos componentes. Para ilustrar a aplicabilidade da abordagem proposta, apresentamos uma prova de conceito na qual os processos foram instanciados em um cenário real de *smart grid*, demonstrando sua operacionalização e eficácia. Além disso, avaliamos as técnicas propostas, e os resultados confirmam sua eficácia na melhoria da qualidade dos dados e no aprimoramento do desempenho das funcionalidades do sistema de software IoT. Embora essa solução enfatize a importância da gestão dinâmica e adaptativa da qualidade de dados, ela também abre caminhos para pesquisas futuras, incluindo o escalonamento dinâmico de tarefas, o desenvolvimento de novas técnicas e estudos de escalabilidade. Este trabalho contribui para o avanço de soluções práticas e escaláveis para a gestão da qualidade de dados em *IoT smart spaces*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

EXECUTING DATA QUALITY MANAGEMENT PROCESSES ON IOT SOFTWARE SYSTEMS USING A COLLECTION OF SOFTWARE COMPONENTS

Gabriel Rodrigues Caldas de Aquino

March/2025

Advisor: Prof. Claudio Miceli de Farias

Department: Systems Engineering and Computer Science

This thesis presents a software architecture to data quality management in IoT software systems operating within IoT smart spaces. To operationalize data quality management in IoT, we introduce a set of software components designed to execute data quality processes autonomously, minimizing human intervention. Additionally, we propose techniques to be deployed within these components. The proposed techniques include data preprocessing at the IoT device layer, where data is collected and preprocessed to prevent the propagation of data quality issues. Furthermore, we introduce techniques for enforcing data quality compliance at the IoT software system layer, ensuring that data meets the system's quality requirements before being utilized. We also present implementation guidelines to facilitate the deployment of the solution in IoT smart spaces, covering all phases from requirements definition to component deployment and configuration. To illustrate the applicability of the proposed approach, we present a proof of concept where the processes were instantiated in a real-world smart grid scenario, demonstrating their operationalization and effectiveness. Additionally, we evaluate the proposed techniques, and the results confirm their effectiveness in improving data quality and enhancing the performance of IoT software system functionalities. While this solution underscores the importance of dynamic and adaptive data quality management, it also opens avenues for future research, including dynamic task scheduling, the development of new techniques, and scalability studies. This work contributes to the advancement of practical and scalable solutions for data quality management in IoT smart spaces.

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP 3rd Generation Partnership Project.

6LoWPAN IPv6 over Low Power Wireless Personal Area Networks.

AMQP Advanced Message Queuing Protocol.

BM Battery Monitoring.

CN Collector Nodes.

CoAP Constrained Application Protocol.

DAG Directed Acyclic Graph.

DPN Data Processing Node.

DQAM Data Quality Assessment Module.

DQEM Data Quality Enhancement Module.

EMO Energy Management and Optimization.

Exp-1 Using Data Quality Preprocessing Task.

Exp-2 Do Not Using Data Quality Preprocessing Task.

FN False Negatives.

FP False Positives.

GDPR General Data Protection Regulation.

HTTP Hypertext Transfer Protocol.

IoT Internet of Things.

IP Internet Protocol.

ITU International Telecommunication Union.

LGPD Lei Geral de Proteção de Dados Pessoais.

LTE Long-Term Evolution.

MAF Moving Average Filter.

MQTT Message Queuing Telemetry Transport.

NFC Near Field Communication.

OPLM Overhead Power Line Monitoring.

PoC Proof of Concept.

QoS Quality of Service.

RAM Random Access Memory.

REST Representational State Transfer.

RFID Radio Frequency Identification.

ROM Read-Only Memory.

RPL Routing Protocol for Low Power and Lossy Networks.

TN True Negatives.

TP True Positives.

VoIP Voice over Internet Protocol.

WHO World Health Organization.

Z-Wave Z-Wave Wireless Protocol.

ZigBee Zonal Intercommunication Global-standard for Energy Efficiency.

Chapter 1

Introduction

This chapter describes the motivation of this thesis. In section 1.1 we state the problem definition. The research questions are defined in section 1.2. The research goals are detailed in section 1.3. Finally, the thesis outline is provided in section 1.4.

1.1 Context and Motivation

Technological advances enabled the realization of the Internet of Things (IoT) paradigm [2]. IoT refers to a network of devices (e.g. vehicles, home appliances, etc.) embedded with electronics, software, sensors, and actuators with Internet connectivity. The adoption of IoT was accelerated due to the social distancing measures implemented during the COVID-19 pandemic, which was declared a global pandemic by the World Health Organization (WHO) in the beginning of 2020 [3] [4]. **IoT smart spaces**, such as *Smart Cities*, *Smart Hospitals* [5], and *Industry 4.0 smart factories* [6], enable the continuation of essential services without requiring direct human-to-human interaction [2]. The interconnection of these devices promotes data exchange and seamless communication [7] [2] [8] [9].

IoT smart spaces can be composed of a small number or have a plethora of devices embedded with sensors (e.g. temperature, humidity, Radio Frequency Identification (RFID) and others)[10] [11] [12] [7] [13]. These devices can collect, process data locally and send it to other devices. Usually, they use low-power wireless communication technologies to save energy on data transmission. Some data processing can also occur on edge devices, which are nodes with greater processing capabilities and fewer power constraints. Finally, the data is forwarded to an IoT software system.

The IoT Software Systems are designed to receive data from one or more IoT device, process it [14] and acuate on the environment according to predefined rules. A smart space combines software and hardware components to work together, enabling seamless functionality [15]. For example, a smart-home with an air conditioning IoT

system might collect temperature data from a network of sensors, process it and apply a predefined rule [16] such as: 'If the temperature exceeds a threshold, turn off the air conditioner.' This results in an actuation on the environment, such as turning off the air conditioner. The correct actuation is the one in which the system's intended goal is achieved given the real state of an environment. IoT Software Systems bridges the gap between sensor collected data and actions in real-world environments.

IoT devices face challenges that can affect the functionality of IoT Software Systems. Those challenges have two main causes: (i) hardware and (ii) environmental. Common issues arise from **hardware challenges**, such as limited memory, processing power, and energy constraints, which can lead to problems like packet loss or energy depletion [17]. Additionally, **hostile environments** present another layer of complexity. Sensors deployed outdoors may be exposed to extreme temperatures, humidity, or physical damage caused by vandalism or accidental impacts [18]. These factors can result in damaged sensors and data measurements with quality issues [19], ultimately compromising the operations of the IoT Software System. For example, a soil moisture sensor [20] may degrade over time, providing *inaccurate* data to the IoT Software System and potentially leading to an incorrect actuation in the environment.

IoT data is inherently tied to environmental phenomena rather than to specific use applications [1]. A phenomenon refers to any situation occurring within the smart space, such as overheating in a specific region, which generates data within a particular range [1]. This data may or may not have direct relation to an event of interest for a specific IoT Software System. To monitor a given phenomenon, it is important to consider its data requirements, including the types of data, appropriate sampling rates, and its tasks, all of which depends on the characteristics of the phenomena on the smart space. Moreover, IoT smart spaces can be unpredictable, where environmental changes and unforeseen conditions can impact the characteristics of data collected. This variability presents unique challenges for IoT Software Systems in processing and interpreting the data accurately for meaningful actuation.

IoT data quality is evaluated through various dimensions, each representing a specific attribute that reflects the fitness of data for its intended purpose. There is a myriad of data quality dimensions for IoT systems, but **accuracy**, **completeness**, and **timeliness** are the ones most frequently discussed, as they address key aspects of data quality [21][19][19][22][23][24][25][26][27][28]. **Accuracy** means the degree of the data to correctly represents the real-world phenomenon it describes. **Completeness** ensures that no critical data is missing, while **timeliness** assesses whether data is available when needed. These dimensions, when measured appropriately, provide an overall assessment of data quality. However, the definition and

importance of specific dimensions can vary depending on the phenomena of interest.

For the IoT Software Systems, it is crucial to ensure that the collected data satisfies the quality requirements **inherent** to the system [29]. This involves managing data quality to verify that it is fit for purpose and can effectively support the system's intended operations. Data quality management is a non-functional requirement of an IoT Software System because data quality management is not about addressing the core functionalities of the software system, it is about ensuring that IoT data fits the application's data quality requirements. Non-functional requirements are often more critical than individual functional requirements. Failing to meet a non-functional requirement can render the entire system unusable [30]. In the context of IoT, this can lead to serious problems, as these systems often operate without direct human intervention. A failure to meet data quality requirements could result in incorrect actuation or unresponsive behavior, potentially causing operational disruptions or unsafe conditions.

Therefore, poor data quality in IoT Software Systems can lead to severe consequences, especially in smart spaces where human safety is involved. Erroneous or incomplete data can trigger automated decisions that result in accidents or **life-threatening situations** [31]. For instance, in smart hospitals [5], inaccurate sensor readings could cause false alarms, wasting critical resources and delaying responses to genuine emergencies [31]. In safety-critical applications such as smart vehicles, poor data quality could prevent the detection of pedestrians or objects, resulting in serious accidents or **loss of life** [32]. Similarly, in industrial environments, undetected anomalies due to faulty data could lead to catastrophic failures, such as fires or explosions, posing risks to both human life and infrastructure [8].

The impacts extend beyond immediate safety concerns. Poor data quality can also result in financial losses, potentially **causing the finances of an organization to crash**. For example, unnecessary sprinkler activations in smart buildings due to inaccurate fire detection data could cause significant property damage and high cleanup costs [33]. In smart agriculture, inaccurate data from sensors could lead to over-watering or under-watering, reducing crop yields and profitability [20]. Additionally, damage to reputation and legal challenges are a risk if these failures compromise violate regulations such as General Data Protection Regulation (GDPR)[34] or Lei Geral de Proteção de Dados Pessoais (LGPD) [35].

1.2 Problem Definition and Research Question

IoT smart spaces present the following challenge: On one hand, IoT devices are resource-constrained and often operate in unreliable or hostile environments. On the other hand, IoT Software Systems are expected to enable the seamless operation

of real-world processes with minimal human intervention. The success depends on ensuring that IoT software systems function as intended, even under adverse conditions. In this context, the data quality becomes a critical factor, as it directly influences the ability of IoT services to fulfill their intended purposes. Data Quality Management, defined as the process of ensuring the "*fitness of data for a given purpose*" plays a vital role in addressing this challenge. Reliable IoT services depend on the management and evaluation of data quality to ensure real-world operations.

The task of ensuring that data is suitable for its intended purpose is not new and has been extensively addressed in traditional applications. For example, multimedia applications such as Voice over Internet Protocol (VoIP) rely on Quality of Service (QoS) mechanisms to guarantee reliable service delivery over potentially unreliable networks. These mechanisms assess and manage network performance using metrics such as packet loss, delay, and jitter [36]. Solutions for such challenges are often standardized through initiatives like International Telecommunication Union (ITU) and 3rd Generation Partnership Project (3GPP), and *off-the-shelf* tools are widely available to support these applications [37]. Commercial devices, such as *Cisco Catalyst*, implement these QoS mechanisms to maintain specific service quality levels [38]. However, unlike traditional applications, IoT software systems operate in smart spaces with resource-constrained devices, with unpredictable environment and often lack the stability and standardization seen in traditional multimedia systems, requiring novel approaches to ensure data quality for IoT software systems.

IoT software systems were designed to be agnostic of data quality requirements. However, the rapid evolution in this area and the increasing of the complexity of sensors and applications involved, the need for specific data quality solutions have raised.

So in this research we propose a software architecture to data quality management within IoT smart spaces, addressing the roles of different **actors**: (i) **IoT devices**, (ii) **network edge** and (iii) **the IoT Software System**. **Sensors** are the primary source of data, they must be monitored and managed to ensure data quality on data capture. **The network edge** refers to the intermediate point between IoT devices and the software system, where data can be processed and routed. Also, **The IoT Software System** must ensure data fitness for its operations, it plays a critical role in processing and validating this data against predefined quality criteria.

Additionally, due to the distributed nature of the IoT, data is generated across numerous devices rather than centralized repositories. In traditional data centers, data is typically static or changes at a relatively slow rate, whereas in IoT environments, data is constantly being generated by phenomena in real-time. This continuous and dynamic data flow often produces noisy, incomplete, or inconsistent

data, complicating the application of traditional data quality processes. Moreover, IoT devices frequently operate in hostile conditions, such as extreme temperatures or physical tampering, further impacting data generation. These unique characteristics of IoT systems highlight the need for adapted or alternative approaches to data quality management that address these challenges directly.

Therefore, we need a software architecture for data quality management that specifically addresses the unique data quality challenges on IoT smart spaces. Such a solution must consider the hardware challenges and the hostile environments. Additionally, the software architecture should be capable of accommodating the diverse and often conflicting requirements of multiple IoT applications. It should be scalable so support a plethora of devices [39] and decentralized. It should employ an adaptive mechanism to dynamically adjust to varying conditions and requirements in real time [1]. Furthermore, real-time assessment of data quality is essential to ensure that the system can respond promptly to issues and maintain its operations.

This thesis addresses the problem of managing data quality for IoT Software Systems, taking into account their dynamic characteristics and unique challenges. Therefore, the main research question of this thesis proposal is formulated as follows:

- How can a software architecture for data quality management be designed and implemented to address the unique challenges faced by IoT Software Systems?

1.3 Research Goals

The main objective of this work is to provide a software architecture for data quality management in IoT smart spaces. The proposed software architecture enables the execution of data quality management tasks tailored to the IoT. These tasks basically execute:

- **Data Quality Assessment**: Techniques designed to evaluate whether sensor-collected data meets defined data quality dimensions such as timeliness, accuracy, and completeness [19]. These techniques have the objective to measure how well data represents the events or phenomena that generated it in face of the requirements of the applications that will consume this data.

- **Data Quality Enhancement**: Techniques focused on improving the quality of data, such as cleaning or transforming it to meet application-specific requirements. These tasks have the objective to execute mechanisms to make data conform the requirements of the application that will consume this data.

To achieve this, the proposed software architecture uses data quality management tasks deployed across the key actors of an IoT smart space. These functionalities will

be carried out by software components instantiated on IoT devices, at the network edge, and within the IoT Software System. Each actor will host tasks tailored to its specific role and capabilities within the ecosystem:

- **IoT Devices** operate in the physical environment, collecting data that may be impacted by hardware challenges and hostile conditions such as extreme temperatures or vandalism. These devices provide data for multiple applications, often independently of specific application requirements. At the point of data generation, sensors can perform preliminary tasks such as detecting outliers, identifying unusual events (e.g. fires), and checking for missing data. However, sensors are detached from the specific quality requirements of the applications that the IoT Software System will ultimately support.
- The **network edge** serves as the intermediary between the sensors and the IoT Software System, facilitating data transmission and ensuring efficient communication. It can act as a broker, such as an Message Queuing Telemetry Transport (MQTT), to manage data flows between devices. Additionally, the network edge can perform tasks such as packet prioritization, implementing priority queues, and providing QoS to meet the varying requirements of different applications.
- The **IoT Software System**, in contrast, directly aligns with the requirements of the applications it serves. It has access to the data quality requirements defined as non-functional system requirements, such as accuracy, completeness, and timeliness. This allows the system to evaluate the data it receives against these quality dimensions and ensure that the data aligns with the operational goals of the applications.

Therefore, this work proposes a software architecture for data quality management on IoT smart spaces. The data quality tasks of the proposed software architecture are designed to be distributed among the actors of the IoT smart spaces: IoT devices, the network edge, and the IoT Software System. By leveraging these actors to perform data quality assessment and enhancement tasks, the software architecture aims to address the challenges of IoT systems while ensuring data quality is delivered to according to the data quality requirements.

1.4 Thesis Outline

The remaining of this thesis is divided as the following: **Chapter 2** presents **Basic Concepts**, providing the theoretical foundation for this work. It introduces key concepts related to data quality management, IoT system architectures, and relevant

systems engineering principles. **Chapter 3** reviews **Related Works**, positioning this research within the existing literature. It discusses prior studies on data quality management in IoT environments and identifies the limitations and gaps that this work aims to address. **Chapter 4** details the **Proposal**, outlining the proposed software architecture for data quality management in IoT smart spaces. **Chapter 5** presents a **Proof of Concept (PoC)** with an instantiation in an Smart Grid [11] Monitoring Scenario. This chapter demonstrates how the software architecture can be instantiated in a real-world IoT scenario. **Chapter 6** presents the **Evaluations and Results**, of an instance of the proposed software architecture for data quality management through a Proof of Concept. We assess its software components in a smart grid scenario by executing the data quality tasks. **Chapter 7** provides **Practical Guidelines for the Implementation of the Proposed Data Quality Management in IoT Smart Spaces**. It describes a step-by-step methodology for deploying the proposed solution, defining requirements, configuring components, and tasks for enabling continuous monitoring and adaptation. Finally, **Chapter 8** provides the **Conclusion**, summarizing the contributions of this work, discussing its implications, and highlighting potential directions for future research.

Chapter 2

Basic Concepts

This chapter introduces the basic concepts of this work. In section 2.1 we presents the concepts related to the Internet of Things (IoT): In subsection 2.1.1 there is a presentation about the characteristics of IoT data. Subsection 2.1.2 presents different on IoT Smart Spaces and their challenges. Section 2.2 presents data quality, its dimensions, and its importance. Subsection 2.2.1 addresses data quality in IoT. Subsection 2.2.2 discusses relevant data quality dimensions and subsection 2.2.3 discusses data quality management for the IoT.

2.1 Internet of Things

The **Internet of Things (IoT)** is a global system that interconnects physical and virtual "things" using the Internet Protocol (IP) [40] [41]. A **smart space** is represented in figure 2.1, it is a physical space equipped with networked devices capable of collecting and processing data generated by the phenomena occurring in the environment, with a network edge that receives the data and sends it to an IoT software system that has the objective of processing, analyzing, and managing the collected data to enable decision-making, automation, and meaningful insights for the people.

An IoT smart space comprises key actors that enable its functionality: (i) **IoT devices**, (ii) **network edge**, and (iii) **IoT Software Systems**. Building on this perspective, each actor plays a distinct role in facilitating the seamless integration of IoT smart spaces:

- **IoT devices** are capable of collecting data from the physical world through communication, sensing, and actuation capabilities [42]. They are physical devices or virtual abstractions responsible for gathering and uploading information to the communication network [43]. These sensors interact directly with the environment, collecting data that represents various phenomena [1],

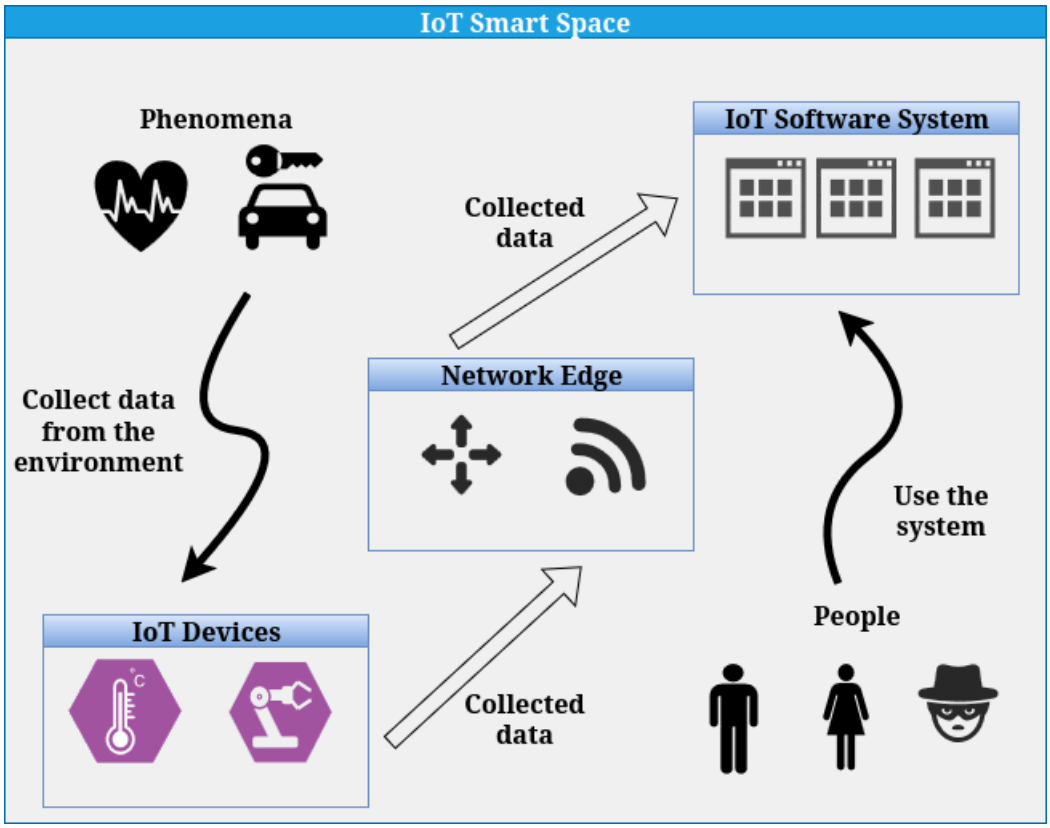

Figure 2.1: IoT Smart Space actors, phenomena and the people

such as temperature or humidity. Sensors form the foundational layer of IoT systems by enabling the transformation of physical-world inputs into data that is sent to the edge of the sensing network [44].

- **Network edge** acts as an intermediary, managing data transmission between devices and systems. It bridges the gap between sensors and IoT Software Systems, handling tasks such as data transmission, preprocessing, and prioritization. The network edge can execute wide range of protocols and technologies. Examples include Wi-Fi, Bluetooth, Ethernet, 3G, 4G-Long-Term Evolution (LTE), HTTP, Zonal Intercommunication Global-standard for Energy Efficiency (ZigBee), Z-Wave Wireless Protocol (Z-Wave), Near Field Communication (NFC), IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN), Routing Protocol for Low Power and Lossy Networks (RPL), Constrained Application Protocol (CoAP), and RFID [41]. Furthermore, brokers like MQTT can be on the network edge to manage data flows while supporting scalability and real-time requirements.

- **IoT Software Systems** process, analyze, and utilize the data generated by sensors and transmitted through the network edge to enable application-specific operations. They represent the operational layer where IoT applica-

tions reside. These systems process and analyze the data received from sensors via the network edge, aligning it with application-specific requirements. IoT Software Systems enable decision-making and actuation, closing the loop between the physical and informational worlds and interacting with the environment it is monitoring.

From picture 2.1 it can be observed the IoT smart space and its characteristics. In an IoT smart space, different phenomena may occur. A phenomenon denotes any situation on the monitored area (e.g. an overheating on a certain part of the area) that generates data in a specific range that may or may not have correlation to an event of interest for an application in the real world [1]. A phenomenon generates data within a defined range. These phenomena can include events such as a fire outbreak, a light turning on, or a person entering a room. Each of these events produces data that is measured, such as temperature readings for fire detection or occupancy data for space utilization. The people within the IoT smart space interact with these phenomena and with the IoT-enabled services. The flow of information follows a structured path: IoT devices collect data from the environment and transmit it to the IoT Software System via the edge. Additionally, IoT devices can act upon the IoT smart space, enabling real-time responses, such as activating alarms, adjusting lighting, or triggering air conditioner systems to enhance the experience of the people within the space.

2.1.1 IoT data

Traditionally in well-known conventional internet web applications, data come generally from people using their computers (e.g. to interact with each other on social networks) and are used generally to provide services for these same people. In contrast, in IoT, the main task is to provide ubiquitous services, without human intervention by exchanging data between things embedded with processing, sensing and communicating technologies. Hence, things will produce the majority of data and will also be their main consumer. In this scenario data if of paramount importance as the main communication medium in this paradigm: IoT devices should communicate and collaborate to autonomously provide ubiquitous services.

Data are a valuable asset in the IoT because they give insights about a given phenomenon, person or entity. Those in-sights are used by applications to provide intelligent services in a ubiquitous manner. If data are inaccurate, extracted knowledge and action based on it will probably be unsound.

IoT sensors monitors an aspect (e.g. temperature, humidity, etc.) of the physical world. Moreover, the environments in which the harvesting of data occurs can rapidly change and can be volatile. As a result, many characteristics are usually

associated with data in the IoT. While some of these characteristics might be considered omnipresent (i.e. uncertain, erroneous, noisy, distributed and voluminous) [19], other characteristics are not general and highly depend on the context and the monitored phenomena (i.e. smooth variation, continuous, correlation, periodicity and Markovian behavior) [19]. Below is a summary of these characteristics:

- **Uncertain, erroneous and noisy** : Considering the numerous factors endangering the quality of data, generated data in the IoT are considered to be inherently uncertain and erroneous.
- **Voluminous and distributed**: In the IoT, sources of data are millions of devices scattered all over the world. The generation rate is enormous and easily overwhelms its human-generated counterpart.
- **Smooth variation**: Many physical monitored variables of interest (e.g. ambient temperature) exhibit smooth variations, i.e. a small variation (or none) occurs between two consecutive time stamps.
- **Continuous**: Data produced from monitoring many physical phenomenon is continuous (e.g. temperature, etc.) even when a sampling strategy is adopted. This is a direct result of the smooth variations. Sampling is used primarily to achieve energy-efficiency because the monitored phenomenon does not often change in a sudden.
- **Correlation**: Generated sensor value dataset has often an underlying correlation. The data are either temporally correlated, spatially correlated or both.
- **Periodicity**: Dataset related to many phenomenon may present an inherent periodic pattern where the same values occur at specific intervals.
- **Markovian behavior**: The sensor value at a given time stamp t is only function of the previous sensor value at the previous timestamp *t-1*.

2.1.2 IoT Smart Spaces

An **IoT smart space** can be characterized as an environment equipped with various networked devices capable of processing and sensing, assisting users in performing their tasks more efficiently. Recently, numerous smart environments have been developed to enhance user comfort and safety. However, different smart environments come with distinct requirements based on their applications. Clearly, the significance of a temperature control system differs from that of a fire alarm system, just as a cardiac monitoring system in a hospital holds more importance than lighting

control. Even the same application may have varying requirements in different environments. For instance, refrigeration control is less critical in a home compared to a hospital.

The term **smart building**, according to [45], refers to buildings equipped with intelligent devices installed to minimize energy consumption without compromising user comfort and safety. Improving the energy efficiency of buildings is a critical step toward a more sustainable lifestyle. Although recent advances in materials science have reduced energy consumption directly within building structures, challenges remain [46]. One such challenge is the significant energy consumption of equipment like temperature control and lighting systems in both residential and commercial buildings. Common **smart building applications** cited in the literature [9, 47] include heating, ventilation, and air conditioning (HVAC) control, lighting systems, shading applications, air quality and window control, device shutdown systems, home appliances (e.g., TVs, washing machines), security (access control), and device safety. These applications often rely on similar types of environmental data, such as light, vibration, temperature, presence, chemicals (e.g., smoke or gases), and voltage.

In the context of smart buildings, recent proposals [47–49] suggest monitoring and control solutions leveraging wireless sensor networks (WSNs). In smart grid environments, however, high costs have limited monitoring solutions. Installing and maintaining communication cables in such environments is expensive, making wired monitoring systems less common. This highlights the urgent need for low-cost wireless monitoring and diagnostic systems that improve system reliability and efficiency by enhancing the management of energy generation and transmission systems.

Structural Health Monitoring (SHM) systems enable the prediction of damage (e.g., fractures), allowing for timely repairs and preventing accidents. In traditional SHM applications, sensing devices capture measurements related to the structure and external events affecting it, sending this data to a centralized collection entity. Decisions about structural damage are typically made centrally. However, transferring part of the decision-making process into the sensors themselves creates an **intelligent SHM system**. This decentralized approach can accurately and reliably detect and predict damage or device malfunctions within the network. For example, employing intelligent SHM in structures connected to a wind power plant within a smart grid context enhances the reliability and cost-effectiveness of energy generation.

The term **smart grid**, according to [50], is defined as a bidirectional electricity transmission network (from generator to consumer and vice versa) that utilizes ICTs in generation, transmission, distribution, and end-use energy systems to improve efficiency, reliability, and security. A notable advantage of smart grids is their capacity

to integrate renewable, non-polluting energy sources such as solar and wind power. The bidirectional nature of smart grids enables real-time monitoring of wires, cables, transformers, and even specific devices in factories, offices, or homes. This capability facilitates device control and the use of variable-rate tariffs based on system load, time of day, or season, encouraging energy conservation and reducing peak demand.

Studies conducted by the Electric Power Research Institute (EPRI) [9] predict that smart grids could reduce greenhouse gas emissions by 60 to 211 million tons of CO_2 in the United States by 2030. Communication within smart grids can occur through optical fibers or using medium- and low-voltage power lines as IP communication channels. Another prominent application is **smart farming** [51], where sensor-controlled farms assist farmers in improving crop and livestock productivity. Common applications in smart farming include soil moisture measurement, temperature control, irrigation, and evapotranspiration assessment. Recent studies [48, 52] employ predictive algorithms to analyze how climatic and soil factors influence production, adjusting irrigation and input decisions accordingly. For instance, fuzzy logic is used in [52] to model environmental variables and adapt irrigation systems in real time. Similarly, [48] introduces a data fusion model that integrates weather station data with farm sensor data for irrigation control, adjusting weights based on data accuracy. The concept of **smart greenhouses** aims to eliminate environmental uncertainties by isolating crops in sensor-controlled greenhouses [48], where temperature and lighting control are critical.

Smart hospitals [53] utilize sensors and cameras to monitor patient conditions and hospital facilities. While some applications overlap with smart homes, unique applications include monitoring infections and contamination. Chemical sensors play a vital role, and multimedia sensors like cameras are crucial for predicting patient falls and tracking movements. Another category involves monitoring patients' vital signs through **body sensor networks** (BSNs). Vital sign monitoring is complex and prone to false alarms, which can lead to alert fatigue—where hospital staff dismiss alarms assuming they are false. Data fusion techniques in BSNs must aim to minimize false alarms to ensure timely responses to emergencies. Military applications also utilize data fusion for real-time soldier health monitoring and autonomous vehicle navigation [54]. As observed, each environment has unique challenges and requirements that must be addressed carefully. The next section introduces various techniques developed to tackle these challenges, albeit partially.

2.2 Data Quality

2.2.1 Data quality for IoT

The data quality definition follows the quality definition of [55], which is: *"the degree to which a set of inherent characteristics of an object fulfills requirements"*. In this case, the object in question is data. In this scenario, the requirements are the data requirements of the data consumers - which are the applications that consume data. Following this, the definition of data quality can be *"how well data meet the requirements of data consumers"* [19] or as *"data that are fit for use by data consumer"* [21]. Hence, the concept of *"fitness for use"* emphasizes the importance of taking a consumer viewpoint of quality because ultimately it is the consumer who will judge whether or not a product is fit for use [21].

Data is an important asset in the IoT paradigm. Data is the source of information for context-based decision making. In this context, quality is a deemed as a critical requirement for decision making mechanisms, applications and services on the IoT. Data quality has different meanings depending on the area in which it is applied. In the field of IoT, data quality means how suitable the data collected from IoT devices are for IoT applications [56]. The data quality definition gives a broader conceptualization of data quality by focusing on how consumers - IoT applications - perceive quality, rather than the perception of specialized professional limited to intrinsic level and accuracy dimension, considering that data have become a product, the fitness of which is judged by its user. This means that data quality would hardly be seen in the same way by different users. In fact, each data consumer requires the used data to fulfill certain criteria which he presumes essential for his own tasks at hand [19].

2.2.2 IoT Data quality dimensions

It is demanded for data consumers that the consumed data meets certain quality criteria. These criteria, aspects or attributes of data quality are known as data quality *dimensions*. There exist a myriad of data quality dimensions in literature due to the fact that data are a representation of various aspects of the real world phenomena. Also, there is no standardized definition for each data quality dimension. In fact, a single dimension could present more than one different definitions depending on the context in which a given dimension is inserted [19]. Also, the ISO international standard data quality model identifies several data quality characteristics in the context of Software Engineering [57].

Given the enormous quantity of data quality dimensions, it is usual to divide the set of data quality dimensions into groups according to common characteristics.

The work [19] provides four categories for data quality dimensions: (i) Intrinsic; (ii) Contextual; (iii) Representational and (iv) Accessibility [21].

- *Intrinsic data quality semantic aspect* is related to the quality of the data in relation to itself. As examples of dimensions there are: accuracy, objectivity, believability and reputation;
- *Contextual data quality semantic* aspect is related to how appropriate the data is for its usage. As examples of dimensions can be relevancy, value-added, timeliness, completeness and amount of data;
- *Representational data quality semantic* aspect describes how understandable and representative of the environment the data is. Examples of dimensions are interoperability, ease of understanding, concise representation and consistent representation;
- *Accessibility data quality semantic* dimensions describe how accessible the data is for data consumers. Examples can be accessibility and access security.

Data quality challenges refers to difficulties affecting any data quality dimension. Such challenges can cause data to become entirely or partially unusable, since it may not meet the requirements of data consumers. As we discussed, data quality does not relate to problems only related to data accuracy. Instead, data quality problems can surpass the accuracy dimension to other dimensions such as the aforementioned.

Several works present data quality dimensions for several types of data in IoT context, such as wireless sensor data, IoT enabled devices data and data from pervasive environments [56]. For wireless sensor network data, the data quality dimensions are proposed for evaluating sensor data according to timeliness, reliability and accuracy [58]. Also, for sensor data, the automated data quality management can be performed by extending the data stream with data quality metadata, by taking advantage of five data quality dimensions: accuracy , confidence, completeness, data volume and timeliness [59]. For pervasive environments, the use of dimensions regarding currency, availability and validity are proposed [60].

Regarding IoT context, the work of Karkouch et al. [19] surveys and highlights the following data quality dimensions for WSN and RFID – enabled data: accuracy, confidence, completeness, data volume, timeliness; Also, Additional data quality dimensions and IoT Domain-specific data quality dimensions are presented: ease of access , assess security, interpretability, duplicates and availability.

- Accuracy: *Intrinsic* data quality dimension defined as the maximal absolute systematic error such that the real values belong to an error interval.

- Confidence: *Intrinsic* data quality dimension defined as the statistical error such that an interval contains the real value with a confidence probability of p.
- Completeness: *Contextual* data quality dimension defined as the ratio of non-interpolated items to all available (i.e. both non-interpolated and interpolated) data in the considered stream window.
- Data volume: *Contextual* data quality dimension defined as the number of raw data items (values) available for use to compute a result data item (in a stream query or sub-query).
- Timeliness: *Contextual* data quality dimension defined as the difference between the current timestamp and the recording timestamp.
- Ease of access: *Accessibility* data quality dimension defined as the availability and easiness of retrieving data.
- Access security: *Accessibility* data quality dimension defined as the securing of data in order to protect its privacy and confidentiality.
- Interpretability: *Representational* data quality dimension defined as how Data is clear in meaning and format.

A diverse set of IoT applications are domain dependent, such applications prompts IoT decision makers to adopt a different approach or worldview depending on the domain in which the application depends. Looking at the scope of the IoT and the numerous applications that could be built using its paradigm, the work [19] presents the domain-specific DQ dimensions:

- Duplicates: data quality dimension regarding one of two or more identical things. In e-health applications' domain, the *duplicates* data quality dimension can indicate the healthcare records duplication's rate in the patients' databases. Also, in smart-grid applications domain, duplicates can indicate if a reading is not being received and stored more than once.
- Availability: data quality dimension regarding if the data is able to be used or obtained. In e-health applications' domain, the *availability* data quality dimension can indicate if electronic health records are available for a secondary use in epidemiological research.

Data quality management is the process of dealing or controlling data quality. Data quality management process is related to defining data quality goals, defining

an strategy for performing the data collection, it relates the quality of the individual data sets to data quality goals through data quality dimensions and interprets it to provide as result the control of the data quality [61].

2.2.3 Data Quality management for the IoT

Organizations can adopt a data quality management process to prevent issues and deliver data application services in line with requirements, enhancing user satisfaction. Here, data is a business asset with management responsibilities [62]. Traditionally, organizations deploy applications in controlled data centers with resources supporting projected workloads. Application hardware requirements shape provisioning decisions, spanning physical/virtual servers, network infrastructure, and databases. The database lifecycle begins with designing a logical database's global schema, allocating data across networks, and defining local database management system schemes [63]. Applications use acquired data, performing tasks like customer billing or decision-making routines, which contextualize and interpret data. Poor data quality often stems from process, system, policy, or data design problems [64]. Thus, grasping data-generating, using, and storing processes is vital for understanding data quality.

However, the Internet of Things (IoT) embodies a paradigm where ordinary objects (e.g., light bulbs or coffee machines) interconnect via the Internet to exchange data [65] [66]. The IoT infrastructure comprises diverse devices with varying computational capabilities. Generally, its data collection setup consists of cost-effective, battery-operated devices with limited processing and radio communication abilities.

The scope of IoT applications is vast, encompassing domains like smart cities, health, homes, and industry 4.0 [67]. These applications typically acquire data from IoT sensors, employ data fusion methods for processing, and act based on predefined rules [16]. IoT devices are strategically positioned to gather data for various applications [1], often in challenging or harsh environments. These devices are susceptible to malfunctions, interference, and deliberate sabotage [19].

These challenges within the IoT infrastructure can compromise data quality, potentially impacting IoT users. A data quality issue's manifestation indicates a corresponding challenge [19]. Data anomalies degrade data quality, with outliers being one such manifestation [19]. Moreover, problems with data quality may present as bias, drifts, or missing values.

Similar to how organizations adopt data quality management processes, IoT applications can implement analogous procedures. This helps detect symptoms of data quality problems and prevent the use of inadequate data that doesn't meet IoT application standards, ensuring accurate and valuable services for users. However,

the contextual aspects of IoT application environments should be considered when implementing these processes.

Contrary to traditional applications that adhere to specific hardware requirements for provisioning servers, network infrastructure, and databases, IoT applications rely on an unreliable infrastructure. Given that IoT devices possess limited resources, the IoT infrastructure must effectively manage their power consumption and handle potential link failures. Thus, the data quality management strategy needs customization to function optimally within this uncertain environment.

For IoT applications, data quality functions as a non-functional requirement, dictating the necessary data quality dimensions for its intended purpose. To facilitate the execution of data quality management within an unstable infrastructure, resources must be allocated within the network. One viable approach is designing a data quality system that employs relevant techniques and protocols to mitigate the potential impact of low-quality IoT data on application performance.

Chapter 3

Related Works

This chapter reviews existing research and approaches relevant to the topic addressed in this thesis, providing a contextualization of the current state of the art in the literature. Section 3.1 discusses previous surveys and systematic literature reviews, highlighting key contributions, trends, and gaps in the field. Section 3.2 focuses on environment-related data quality issues in IoT, its challenges such as data uncertainties, missing data, and environmental impacts. Section 3.3 addresses IoT quality management, analyzing approaches and solutions proposed in the literature and positioning our proposed solution within this context. Section 3.4 presents the conclusions, summarizing the findings and identifying gaps that motivate this thesis.

3.1 Previous Surveys and Systematic Reviews

This section focuses on previous studies that have conducted systematic literature reviews (SLR) or surveys. These works provide an overview of the existing knowledge, identify trends, and highlight challenges or gaps in the field of data quality for IoT. By examining these contributions, we aim to position our work within the context of these broader analyses and demonstrate its relevance and distinction.

The study presented in [27] conducts a systematic literature review (SLR) focused exclusively on three key aspects of data quality within IoT, Cyber-Physical Systems (CPS), and Industry 4.0. These aspects include: (i) identifying data quality issues, their sources, application domains, data types; (ii) exploring techniques to address these issues, such as metrics for monitoring data quality, approaches for data repair and cleaning, and evaluations of these methods; and (iii) investigating software engineering solutions for improving data quality, including system architectures, programming languages, and data storage strategies. This work is important for this thesis as it presents types of IoT data, their issues and techniques that can be employed as tasks and components for a software architecture, like the one presented in this thesis.

The work [28] conducts a Systematic Literature Review (SLR) to examine **how context is considered in recent proposals for data quality (DQ) management**. The study provides extensive evidence that DQ assessment is inherently context-dependent, emphasizing the need for DQ models, dimensions, and metrics to be tailored to specific contexts. However, it highlights a lack of consensus on what precisely makes these DQ concepts contextual and on the most suitable DQ dimensions and metrics for a given DQ project. The work [28] is particularly important for this thesis as it is a recently published work that shows that the lack of agreement underscores the necessity of domain experts to define appropriate dimensions and metrics based on the data context of each project. Furthermore, the study [28] is important for this thesis, as it implicitly agrees that DQ requirements should guide all DQ management tasks. These findings are particularly relevant for IoT scenarios, where context plays a critical role in DQ management decision-making. The reference architecture proposed in this thesis is defined by its components and tasks, which are crucial DQ management in IoT smart spaces, ensuring that tasks are aligned with the specific requirements of IoT systems.

Authors of [22] provide a comprehensive classification of time series data cleaning techniques and an in-depth review of state-of-the-art methods for each category. The study also **summarizes various data cleaning tools, systems, and evaluation criteria from both research and industry**. Furthermore, [22] identifies four key challenges in time series data cleaning: (i) managing large volumes of data with high error rates; (ii) addressing ambiguous causes of errors; and (iii) handling the continuous nature of time series data. The authors highlight the need for further research to refine the analysis of broadly defined error types, address the scarcity of multivariate cleaning algorithms, and explore the potential of machine learning in advancing data cleaning processes. These considerations are particularly relevant in IoT scenarios, where managing large volumes of data continuously is crucial. In this context, the proposed software architecture is designed for data quality management and address these challenges and operate in dynamic, data-intensive environments.

The authors of [24] conducted a systematic literature review (SLR) on data quality in IoT. The review focuses on **data quality issues, dimensions, and measures, establishing connections between data quality dimensions, the manifestations of data quality problems, and the methods used to measure data quality**. The study highlights potential areas for future research, including: (i) developing guidelines for defining specific data quality dimensions for IoT data, (ii) addressing data quality challenges specific to different IoT layers, and (iii) constructing comprehensive data quality frameworks tailored to the IoT context. This work is particularly important for this thesis as it presents data quality issues and its manifestations. For the operation of our proposal its particularly important

to have a foundation on the data quality issues for the instantiation of the software architecture.

The work [25] compares **methodologies and frameworks for data quality (DQ) management** and **international standards**, analyzing them in terms of data types, definitions, dimensions, and metrics. The study explores processes of definition, assessment, and improvement in DQ management and their alignment with international standards. However, the diversity among frameworks and the varying definitions of DQ dimensions and metrics posed challenges for comparability. The survey highlights the need for a more user-friendly methodology for data quality assessment, leveraging existing generic frameworks to enhance usability and consistency in IoT data quality management. The work [25] is particularly significant to this thesis, as it introduces a data quality management solution that incorporates processes defined by ISO standards, including ISO 8000-61, which employs the PDCA (Plan-Do-Check-Act) cycle. While the work [25] highlights other existing methodologies and standards, such as Total Data Quality Management (TDQM), Total Information Quality Management (TIQM), AIMQ, and the observe–orient–decide–act (OODA) framework and the standards ISO/IEC 25012 and ISO/IEC 25024 standards. This proposal aims to address the challenges highlighted in [25] by providing a software architecture for DQ management on IoT smart spaces.

The work [23] presents a systematic literature review (SLR) on sensor data quality problems. The review identifies common sensor data errors, including **missing data and faults such as outliers, bias, and drift**, focusing on the classification of sensor data errors, methods for their detection and correction, and the domains where these approaches are applied. This is particularly relevant to this thesis, as it introduces data quality management mechanism capable of addressing sensor data problems, including **missing data and faults such as outliers, bias, and drift**. Our proposal also ensures that these issues are managed in alignment with the specific quality requirements of applications, providing the components and tasks to address the challenges identified in [23].

The authors of [26] conducted a SLR to investigate data quality challenges in smart cities as large-scale CPSs and to identify the most common techniques used to address these challenges. The study reveals that data quality issues in large-scale CPSs arise from factors such as **sensor malfunctions, calibration problems, poor sensor node quality, environmental effects, external noise, and network or communication errors**. The review highlights the **need for a practical, comprehensive data quality management framework to detect sensor node measurement errors aligned with key data quality dimensions such as accuracy, timeliness, completeness, and consistency**. This is particularly relevant to this thesis, as it provides foundations for data quality dimensions and

the requirements of specific applications, such as **accuracy, timeliness, completeness, and consistency**. By providing a software architecture, our solution provides data quality management components and tasks to overcome the challenges noted in [26].

Table 3.1: Overview of Related Works

Reference	Year Published	Type of Work	Relation to This Thesis
[19]	2016	Survey	Introduces foundational concepts of data quality management.
[22]	2019	Systematic Review	Highlights techniques for time series data cleaning, important for IoT.
[23]	2020	Systematic Review	Focuses on sensor data issues and corrections, aligning with our solution.
[24]	2020	Systematic Review	Explores DQ dimensions, problems, and measures for IoT systems.
[25]	2021	Survey	Analyzes frameworks and standards.
[26]	2022	Systematic Review	Highlights the need for comprehensive DQ frameworks for sensor errors.
[27]	2023	Systematic Review	Explores DQ issues, techniques, and software solutions for IoT.
[28]	2024	Systematic Review	Validates the context-dependency of DQ.

3.2 Environment-related Data Quality issues

There are several works in literature that deal with IoT data quality challenges: [68], [19], [56]. A survey on IoT data quality enumerates IoT environment-related factors and analyze their impact on various data quality dimensions [19]. Also, the forms in which data quality problems are manifested (e.g. bias or outliers) are identified and associated with symptoms that indicate a data quality problem [19]. Some other work presents an introduction to dynamic data quality challenges [68] and mobile sensing sensing devices [56], which relates to contextual changes on IoT infrastructure in terms of data and structure dynamics. Also, mobile crowd-sensing (MCS) IoT applications are proposed to take advantage of personal sensing devices (e.g. smartphones, wearables, cars, etc.) to overlays a cross-validating approach to ratify sensor data and adds credibility to the IoT collected data [69]. In the same

way as the aforementioned approaches as we provide a solution capable of detecting data quality problems [19] and to process and actuate to mitigate these problems [68] [56]. However, our work differs from these works as we are not focusing on a subset of mobile and crowd-sensing applications and we are not only providing a data quality problem detection mechanism. Our proposal is a solution for managing data quality on IoT infrastructure in which all IoT applications can take advantage from and benefits from data quality management.

On the objective to mitigate uncertainty on IoT data, several works deal with missing data problem on IoT [70], [71], [72], [73]. A new ensemble of neural network tools was developed for improving the accuracy of solving prediction tasks of the missing IoT data recovery is addressed by some works [70], [71]. As uncertainty can be manifested in the form of incomplete information [71], two successive General Regression Neural Network (GRNN) networks and one neural-like structure of the Successive Geometric Transformation Model (SGTM) are proposed to improve the accuracy of solving the task of completing omissions in the data collected by Internet of Things devices [70]. Also, an approach to mitigate missing data imputation for large gaps in univariate time-series data is proposed as an iterative framework using multiple segmented gap iteration that segments the gap into several pieces, then initializes the missing value imputation process and then, it iteratively run gap reconstruction and gap concatenation to obtain the final imputation results [72]. In this way, a novel ST-correlated proximate missing data imputation model was proposed to deal with missing data problem in IoT [73]. In a similar way to our work, these proposed solutions focus on enhancing the overall quality of the IoT data by mitigating the occurrence of missing data [70], [71], [72], [73]. However, while it focus on missing data and provides an algorithmic implementation of an ensemble method along with a flowchart of the operation, our work presents a software architecture to manage data quality on IoT.

3.3 IoT quality management

Research on IoT data quality management has proposed various approaches, each addressing specific aspects of the problem. The work in [74] introduces an additional data testing layer as a requirement for IoT data quality assurance. This layer is designed to transform raw IoT data into suitable abstractions and verify its quality before integration with generic big data applications. While our proposal shares the goal of managing IoT data quality, our approach addresses data quality at multiple stages—during data production, transportation, and consumption—whereas [74] focuses only on the data consumer's perspective through a quality assurance layer on top of the data collection layer.

Another study [75] proposes managing data quality using traditional approaches and best practices to improve data quality in smart environments. The authors highlight challenges specific to IoT data quality management that traditional systems fail to address. While their work is grounded in adapting traditional frameworks, our research develops a tailored software architecture for IoT, addressing its unique context, such as data transportation, application-specific requirements, and the complex interactions within IoT ecosystems. Unlike [75], our software architecture introduces metrics and processes explicitly designed for IoT, moving beyond the limitations of traditional approaches.

The work in [76] examines the correlation between data quality dimensions and IoT infrastructural aspects. It proposes a systematic framework using the House of Quality technique for information quality management. While it shares similarities with our work by addressing the infrastructural impact on data quality dimensions, our approach further incorporates IoT application requirements and emphasizes both the assessment and enhancement of IoT data quality to align with application needs. This dual focus ensures that data not only meets infrastructural requirements but also fulfills application-specific expectations.

The data stream cleaning system (DSCS) presented in [77] is another significant contribution to IoT quality management. It operates on edge and cloud layers to clean heterogeneous data streams. The DSCS consists of a dynamic protocol interpreter, a structure parser, and a cleaning model activator, enabling data preprocessing and cleaning on edge nodes supported by cloud servers. While [77] focuses on data cleaning tasks using an edge-cloud architecture, our work goes beyond cleaning by proposing a broader range of data quality management tasks, such as data quality assessment and enhancement. Our software architecture incorporates task scheduling and orchestration mechanisms to align these tasks with the roles and capabilities of IoT actors (sensors, network edge, and IoT Software Systems), ensuring decentralized, context-aware, and real-time data quality management.

Lastly, research on stream reasoning and semantic technologies, such as [78], has proposed frameworks for maintaining data stream consistency by applying semantic rules. The primary goal is to enhance stream processing systems like C-SPARQL [79] [80] to support continuous time-aware reasoning. While such works focus on semantic consistency and temporal characteristics, our approach focuses on the comprehensive management of data quality dimensions (e.g., accuracy, timeliness, and completeness) and aligning data streams with application-specific requirements in IoT environments.

3.4 Conclusions

The review of related works demonstrates the diversity of approaches to managing data quality in IoT systems, ranging from traditional methodologies to advanced semantic frameworks and data cleaning mechanisms. These works address key challenges, such as missing data, environmental factors, and infrastructural constraints, and provide valuable insights into specific aspects of data quality management. However, many of these approaches are limited in scope, focusing on particular issues, such as mobile sensing, crowd-sourcing, or semantic consistency, without addressing the broader, systematic needs of IoT ecosystems.

In contrast, this thesis proposes a software architecture for IoT-specific contexts. By integrating data quality management tasks across the IoT ecosystem—encompassing data production, transportation, and consumption stages—our solution addresses a wider range of challenges than traditional methods. It introduces tasks for data quality assessment and enhancement, distributed across IoT actors such as sensors, the network edge, and IoT Software Systems, while incorporating task orchestration to align these components and tasks with the dynamic requirements of IoT systems.

Despite the related works reviewed in this chapter, several limitations must be acknowledged regarding their impact on this research and the design of our proposed data quality management solution:

- **Scope of the works**: The related works analyzed in this chapter focus on specific aspects of IoT data quality, such as missing data or outliers. While these contributions provide valuable insights, their focus may limit their applicability to the broader and more complex IoT ecosystem addressed by our solution. The diversity of IoT scenarios and application requirements might pose challenges when generalizing the findings from these works to the data quality management approach proposed in this thesis.

- **Lack of consensus**: The related works often adopt differing definitions of data quality dimensions, metrics, and methods. The lack of consensus among these studies may influence the selection of dimensions and processes incorporated into our solution. This could lead to gaps or misalignments between the data quality metrics defined in our approach and the specific requirements of IoT applications, potentially impacting how solutions might be described and used.

- **Integration and compatibility**: Many of the related works reviewed focus on isolated aspects of data quality management, such as detecting outliers or imputing missing data, without addressing how these tasks integrate into

a larger, cohesive system. This fragmented perspective might introduce challenges when synthesizing these contributions into a comprehensive, end-to-end solution. Any inconsistencies in integrating these tasks could undermine the reliability of the proposed approach.

- **Methodological biases**: The selection of related works may introduce biases, as the studies chosen for review emphasize certain aspects of IoT data quality while potentially overlooking others. For instance, while some works prioritize semantic consistency and data cleaning, others address task orchestration or context-aware management. This reliance on existing literature might skew the design priorities of our solution and overlook critical challenges.

- **Scalability**: Many related works reviewed, particularly those focusing on data cleaning or missing data, are not explicitly designed to handle the scale and performance requirements of IoT environments. Our solution aims to address these limitations by incorporating real-time task orchestration across IoT actors. However, the scalability and performance of these mechanisms in real-world IoT scenarios, especially under high data volume and dynamic conditions, remain potential risks that require further validation.

Chapter 4

Proposal: Executing data quality management processes on IoT software systems using a collection of software components

In this chapter, we present our proposal: a software architecture for data quality management, with focus on their practical implementation in IoT smart spaces. This proposal is structured around a **collection of software components** that execute **data quality management processes**, ensuring that data used by IoT software systems meets predefined quality requirements. Managing data quality in IoT smart spaces requires addressing multiple aspects, including the definition of processes, the design of software components, and the implementation of techniques tailored to IoT-specific challenges.

4.1 Overview

We propose a software architecture for data quality management. This software architecture is modeled following a **tiered approach** (Figure 4.1): first we present in section 4.2 the **data quality management processes** that are meant to be executed by the actors of the IoT smart space. Then, in section 4.3 we present a collection of **software components** that operationalize the execution of the data quality management processes presented in the previous section of this chapter. Finally, in section 4.4 we propose two **data quality management tasks** that are meant to be **instantiated** in two of the proposed data quality management components inline with the concepts we present in this thesis.

1. **Data Quality Management Processes**: Defines the overarching data qual-

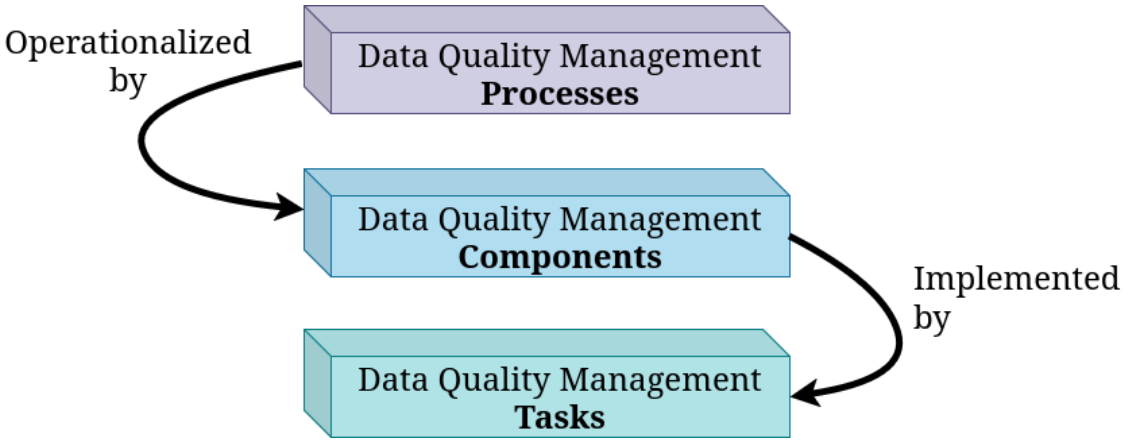

Figure 4.1: Tiers

ity management strategy, ensuring that quality control, compliance, and improvement actions are continuously executed. The proposed processes aim to address the characteristics of IoT environments: distributed nature, minimal human intervention, and real-time requirements.

2. **Data Quality Management Components**: Represents the distributed software elements responsible for operationalizing data quality management within the IoT infrastructure. We propose a set of components along with its interfaces and databases that are designed to enable the execution of data quality management tasks within IoT smart spaces. These components will be instantiated across the actors of a smart space: IoT devices, the network edge, and the IoT Software System. The components we propose aim to support data quality management by aligning tasks with the specific roles and characteristics of each key actor within the context of IoT smart spaces.

3. **Data Quality Management Tasks**: Implements specific data quality techniques within the components to address key data quality challenges in real-world IoT environments. We propose specific data quality techniques to address data quality challenges at different levels. We propose two techniques inline with the concepts we propose in this work: (i) One technique is for data quality preprocessing to address issues such as noisy, incomplete, or inaccurate data while data is being collected; (ii) the second technique is for data quality compliance for the IoT Software System. Each **software component** in this proposal plays a distinct role in executing the **data quality management processes**. These components are deployed across different actors of an IoT smart space — **IoT Devices, Edge, and IoT Software System**. Additionally, we propose two **data quality techniques** to be executed within these components:

- **Data Preprocessing Technique**: Deployed on IoT devices to enhance data quality at the source by detecting and handling anomalies before transmission.
- **Data Quality Compliance Technique**: Executed within the IoT Software System to assess incoming data against predefined quality requirements and trigger adaptation mechanisms if needed.

4.2 Data quality management processes

This section propose data quality management processes for IoT. These processes aim to address the challenges associated with IoT data quality and mitigate their impact. The processes aim to adhere to specific requirements of IoT environments: such as resource constraints, distributed and heterogeneous nature, minimal human intervention and real-time requirements.

The proposed processes cover data creation to consumption, as well as the management and enhancement of its quality. These processes are designed to function within a distributed architecture that aligns with the three key actors of smart spaces: IoT devices, the network edge, and the IoT Software System. Components deployed across these actors communicate and coordinate their actions through message-passing mechanisms. Each actor plays a distinct role in the architecture:

- **IoT Device:** An IoT device that operates within the sensing infrastructure, serving as the gateway for data collection. Its primary role is to collect and preprocess data, performing tasks such as data cleaning, data completion and tagging. The IoT device then sends this data to the IoT Software System via the network edge.

- **Network Edge:** The network edge is responsible for the data transmission between IoT devices and the IoT Software System. It is responsible for maintaining data integrity during transit and implementing Quality of Service (QoS) mechanisms to prioritize critical data based on data quality requirements. After performing these tasks, the network edge forwards the processed data to the IoT Software System for further analysis and utilization.

- **IoT Software System:** The IoT Software System represents the layer where data quality management and application logic converge. It ensures that the data received through the network edge aligns with the specific data quality requirements needed for its operations. If the data does not meet the required quality standards, the IoT Software System may request adjustments or interventions to address these issues.

The idea is to manage data quality by continuously monitoring and proactively addressing any kind of problem to maintain the desired data quality, even in the face of potential challenges. These processes follow the four core concepts [81]:

1. **Data Quality Planning:** Establishes strategies and action plans to make IoT data meets the quality requirements. This includes identifying potential challenges that can endanger IoT data within the IoT smart space and mechanisms to address and mitigate the impact of those challenges. In this step it should define quality metrics, specify techniques for data cleaning, data quality assessment, enhancement and prioritization if its needed.

2. **Data Quality Control:** This process focuses on keep the data quality management processes going on in the IoT smart space. The objective is to ensure that the processes are taking place correctly and IoT data conforms to established requirements. It operates the devices on the IoT smart space to execute the data quality management control: the monitoring of devices and data, executing fallback mechanisms and applying corrective actions to respond to issues (e.g. rerouting data).

3. **Data Quality Assurance:** In data quality assurance, the objective is to assure that IoT data conforms the expected data quality defined in data quality planning. This is done by executing the data quality tasks observing the expected data quality outcome. In this case it means: executing data quality assessment and enhancement task with the predefined parameters or configurations that will lead to the desired data quality outcome.

4. **Data Quality Improvement:** Addresses the causes of data quality issues and implements changes to prevent future occurrences. This includes refining processes, optimizing resource usage, and enhancing system resilience to infrastructural challenges.

The integration of these processes within an IoT smart space aims to ensure that data is managed and delivered at the desired quality levels, even under the challenging conditions faced by the IoT Software System. In the following sections, we propose how the processes operate with the key actors of the IoT smart space, defining their roles and interactions to achieve effective data quality management.

4.2.1 Data Quality Planning

Data Quality Planning is a critical process responsible for ensuring that IoT data meet the specific quality requirements of applications. It involves receiving the data quality requirements (such as timeliness or completeness requirements) and

creating an action plan to align the IoT data with these requirements. During this process, the relationships between data quality requirements should be analyzed, determining, for example, whether they are complementary or redundant, and use this information to create the data quality management plan.

The Data Quality Planning process comprise four steps to ensure that Data Quality Planning not only meets the specific requirements of the IoT software system but also considers its diverse objectives and functionalities. The four steps are:

1. **Requirements Management:** This step involves receiving the data quality requirements of the diverse objectives and functionalities that an IoT software system might have. It should ensure that these requirements are clearly defined and aligned with the intended use.

2. **Data Quality Strategy Management:** In this step, a strategy is developed to achieve the defined quality requirements. This includes determining the methods, resources, and priorities needed to address specific data quality challenges.

3. **Data Quality Procedures Management:** This step focuses on defining the specific procedures and workflows required to manage and improve data quality. These procedures are tailored to the unique characteristics of the IoT.

4. **Data Quality Implementation Planning:** The final step involves developing a implementation plan that outlines the actions to be taken across the network. This includes scheduling tasks, allocating resources, and specifying how it will implemented according to the roles of IoT smart space actors.

1- Requirements Management

The purpose of the quality *requirements management* is to establish the basis for creating or refining a data quality strategy that aligns with the needs and expectations of the diverse objectives and functionalities that an IoT software system might have. This includes collecting, structuring, and classifying data quality requirements. The quality requirements are analyzed to determine how they can be expressed in terms of data quality rules. The result is a defined list of data quality requirements, aligned with the specific needs of the IoT software system.

For IoT, technological feasibility is if a given technique can be deployed, its cost to be deployed and how to schedule the deployment of a given technique. For each IoT software system objective, the requirements are prioritized and validated. In this step, the collected requirements are modeled as a Directed Acyclic Graph (DAG) [82] where each node of a DAG is a requirement (i.e. maximum delay of

two seconds) and the edges represent the relationship between the requirements (i.e. maximum delay of two seconds and completeness minimum of 90%). Hence, the needs and expectations are refined into data requirements DAG. To maintain the requirements relationship, we combine the DAGs with common tasks into a new DAG without damage the integrity of each requirement graph. Once the DAGs are combined, the action plan is elaborated based on DAG' s composition.

A **Data Quality Requirements Model** defines the structured representation of data quality needs within an IoT smart space for an IoT Software System. An example is shown in Table 4.1. This model formalizes the expected data characteristics, such as accuracy, timeliness, completeness, and consistency, ensuring that collected data meets application-specific constraints. It provides a systematic approach to specifying, verifying, and enforcing data quality rules through structured representations like tables, diagrams, or computational models. By clearly defining these requirements, the model supports the effective implementation of data quality management strategies, aligning data processing with the operational needs of IoT applications.

Table 4.1: Data Quality Requirements for Environmental Monitoring (Example)

Dimension	**Requirement**	**Applicable Rule**	**Application**	**Verification**
Timeliness	Low latency in data transmission	The time between collection and transmission must be less than 5 seconds.	Wildfire monitoring	Compare data timestamp with arrival time.
Accuracy	Reliable readings	The variation between nearby sensors must not exceed 1°C.	Smart building climate control	Cross-check data from redundant sensors.
Completeness	No essential data can be missing	At least 95% of expected readings must be available every hour.	Urban traffic management	Count received packets compared to expected total.
Consistency	No contradictory data	There should be no abrupt variations (> 5°C in 1 min) without justification.	IoT weather stations	Apply smoothing filters such as *Kalman Filters*.

2- Data Quality Strategy Management

A data quality strategy is developed based following the *Requirements Management* step. The purpose of *Data Quality Strategy Management* is to develop strategies to meet the defined data quality requirements by outlining the tasks, techniques, methods, resources, and priorities needed for implementation. This step focuses

on how these strategies will be executed, establishing the foundation for creating policies, standards, procedures, and implementation plans. The resulting strategy must be tailored to address diverse data quality challenges while aligning with the overall objectives of data quality management across the network and supporting strategic goals for data quality.

3- Data Quality Procedures Management

The purpose of *Data Quality Procedures Management* is to define standardized workflows and procedures for evaluating and enhancing data quality across the network. This step focuses on capturing and specifying the rules and actions required to consistently perform data quality management tasks in alignment with the predefined strategy. Policies are defined to guide the network in implementing appropriate actions that ensure IoT data meets the data quality requirements of the IoT Software System. These procedures include defining how specific data quality assessment and enhancement techniques should be deployed on the IoT smart space, as well as determining actions to maintain or improve data quality (e.g., adjusting device operations or modifying data collecting rates) [49] [1]. The outcome is standardized procedures for data quality management.

4- Data Quality Implementation Planning

The purpose of *Data Quality Implementation Planning* is to establish a comprehensive plan for deploying and executing data quality management techniques across the IoT smart space. This step focuses on identifying where and how the required techniques will be provisioned, ensuring that data quality objectives are met.

Task scheduling plays a critical role in this process, determining the order and priority of tasks to address specific data quality requirements. By organizing and prioritizing tasks, scheduling ensures the creation of a cohesive data quality plan that aligns with the capabilities and roles of the IoT smart space actors (e.g., CPU, memory, power) [67]. This step also involves defining the allocation and management of resources necessary to execute the data quality plan [82]. Resource allocation must balance the demands of data quality management with the available computational, energy, and network capacities of IoT devices, the network edge, and the IoT Software System. The resulting plan aims to ensure that IoT data meets the required quality levels while maintaining efficient and scalable operation throughout the smart space.

4.2.2 Data quality control

The objective of Data Quality Control is to execute the tasks and procedures defined during the Data Quality Planning phase, ensuring that data meets the required standards throughout the IoT smart space. This process focuses on continuously managing both the key actors of the IoT smart space—IoT devices, the network edge, and the IoT Software System—and the data itself. It involves overseeing physical devices and how data is collected, transported, and consumed.

The process is guided by the implementation plan established in the Data Quality Planning phase, ensuring that data aligns with the requirements of the IoT Software System. It continuously monitors the health of the IoT infrastructure [83], evaluating both the devices and the data they generate. This includes creating, using, and updating data according to predefined work instructions while verifying its quality to ensure compliance with the established standards.

During its operation, Data Quality Control may activate fallback mechanisms and implement corrective actions to address issues as they arise. This requires identifying problems, diagnosing their root causes, and applying preventive measures. If any discrepancies are identified, corrective measures are executed to ensure compliance. These measures may involve verifying collected data for issues [18], cleaning it [77] as needed, transporting it through the network, and prioritizing its transmission when necessary [36]. Upon reaching the IoT Software System, the data undergoes further evaluation to confirm it meets the required quality standards.

For example, if a sensor exhibits drift, the system can detect the problem, perform cross-validation with neighboring devices to confirm the drift, and schedule recalibration for the affected sensor. Also, another example can be if a requirement specifies monitoring a sensor's deterioration, the system can assess whether the frequency of outliers increases over time. In such cases, an outlier detector is deployed on the IoT device responsible for collecting the data. Additionally, if a data latency requirement is identified, the nodes responsible for transmission may implement traffic prioritization mechanisms using Quality of Service (QoS) [36] techniques. The system also verifies whether data arrives within the necessary time frame to remain relevant for its intended purpose. Data that is generated too far in advance of its reception may no longer be suitable for its application, highlighting the importance of maintaining strict control over quality and timeliness.

4.2.3 Data quality assurance

The objective of the Data Quality Assurance process is to measure the quality of IoT-collected data by evaluating its characteristics against specific data quality dimensions requirements. This measurement involves directly assessing the data

to determine its alignment with defined quality requirements. While Data Quality Control oversees the broader data quality management process by managing both devices and data within the IoT smart space, Data Quality Assurance focuses on executing specific data quality assessment techniques. These techniques are applied to the various actors within the IoT smart space, such as IoT devices and the IoT Software System. By assessing data at different levels, the assurance process ensures that the data meets the required standards for its intended use.

Data Quality Assurance encompasses two subprocesses. These two subprocesses assess data quality within IoT smart spaces, addressing both general preprocessing tasks and specific compliance requirements:

- **Data Quality Preprocessing:** This subprocess has the objective of guarantee that basic data quality issues [23] are addressed as early as possible on IoT devices, operating close to the data collection source. Its primary function is to preprocess data and ensure a baseline level of quality without considering the specific data quality requirements of the IoT software system that will consume the data. Tasks performed during this subprocess include **detecting anomalous events** [31], distinguishing between **legitimate occurrences** and **erroneous** readings, **identifying and flagging** missing data or outliers [84] [85] and **addressing** errors or anomalies directly at the data source [86].

- **Data Quality Compliance:** This subprocess has the objective to check if the data fits the requirements. This subprocess is executed on the IoT Software System and focuses on evaluating whether the collected data conforms to the specific data quality requirements defined [24]. This means that it checks if IoT data adheres to the data quality requirements of the IoT software system operations by measuring and matching IoT data against predefined requirements [28], such as timeliness or accuracy. Its tasks include (i) *Measuring Compliance*, which is the evaluation of whether the data meets specific quality requirements, such as arriving on time or maintaining the required accuracy; (ii) *Providing Feedback*, which is the communication of the results of the compliance assessment to the IoT Software System, enabling adjustments to be made as needed; and (iii) *Handling Non-Compliant Data*, in which it forwards data that fails to meet quality requirements to the Data Quality Enhancement subsystem for further processing.

Data Quality Preprocessing

Data quality preprocessing is shown in figure 4.2. This subprocess receives data from an actor in the IoT smart space. Then it executes the data anomaly verification or the data completeness verification, depending on the desired task, which is defined by the data quality plan.

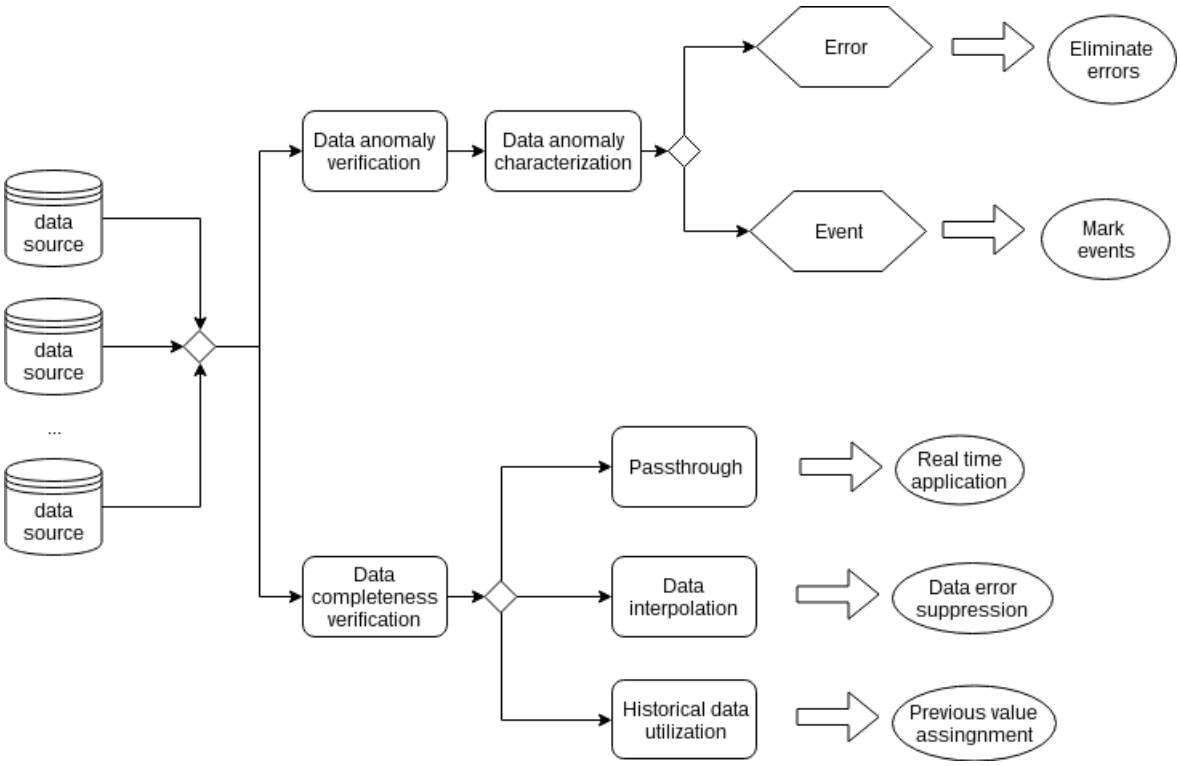

Figure 4.2: Data Quality Preprocessing

Data anomaly verification is the task that evaluates the data to find anomaly behavior, which are data that deviates from what is the expected behavior [87]. Anomaly data can be product of error or events. Hence, a data anomaly characterization is the process responsible for characterizing whether the data is from a legit event (e.g. an occurrence of fire) or from an error event (e.g. a sensor malfunctioning) [1]. In this case, data errors should be eliminated and data from an event should be marked.

For example, a soil moisture sensor [20] may degrade over time, which may lead to the generation of data that does reflect the real condition of the environment, but, instead, are product of sensor failure . The input of those data in the IoT software system may lead to a wrong actuation (e.g., it may actuate to water the soil while it is already wet) [52]. The objective of the data quality preprocessing is to detect outliers and indicate potential sensor deterioration, which might trigger an alert for maintenance. A challenge is to evaluate if the

outlier [88] is product of sensor malfunctions or an environmental phenomena (an actual event that is taking place in the smart space). To do this, data from other sensors can be used to infer the actual soil moisture level and mitigate the impact of the faulty readings.

Data completeness verification is the task that verifies if there are data missing problems (e.g. occurrence of packet collision on data transmission might lead to the loos of data). After verifying if there is any data missing, a countermeasure should be applied producing the following outcomes:

- *Data interpolation* is the outcome that interpolates the received data values to recompose the missing values [89].
- *Historical data utilization* is the outcome that uses previously received data when a data value is missing.
- *Passthrough* is the outcome that just passes the received data to the applications by informing the occurrence of missing values. It is applied when the application of any other outcome imply in delay on data delivery to IoT Software System.

Data Quality Compliance

Figure 4.3 shows the components of the **data quality compliance** subprocess. Its objective is to evaluate whether the data meets the specific data quality requirements of the IoT software system application [90]. Basically, it receives as input the data from IoT data sources and the requirements of the IoT software system and measures the data according to the requirements. As output, it returns if the data fits or if the data does not fit the requirements of the IoT software system. If the data meets the requirements, then the data might be consumed by the IoT software system. Else, data can be discarded by the IoT software system or be processed with data from another data source to be used by the IoT software system.

An example of this subprocess in action is addressing timeliness requirements for a specific data consumption functionality within an IoT Software System. Delayed data can lead to inaccurate conclusions about the environment. If this subprocess detects that data is not arriving on time, the Data Quality Compliance mechanism may choose to disregard the delayed data or forward it for further processing alongside information from other sensors, enabling a more accurate inference of the current state.

Another example scenario involves a fire detection functionality within an IoT Software System that identifies a delay in temperature sensor data. To main-

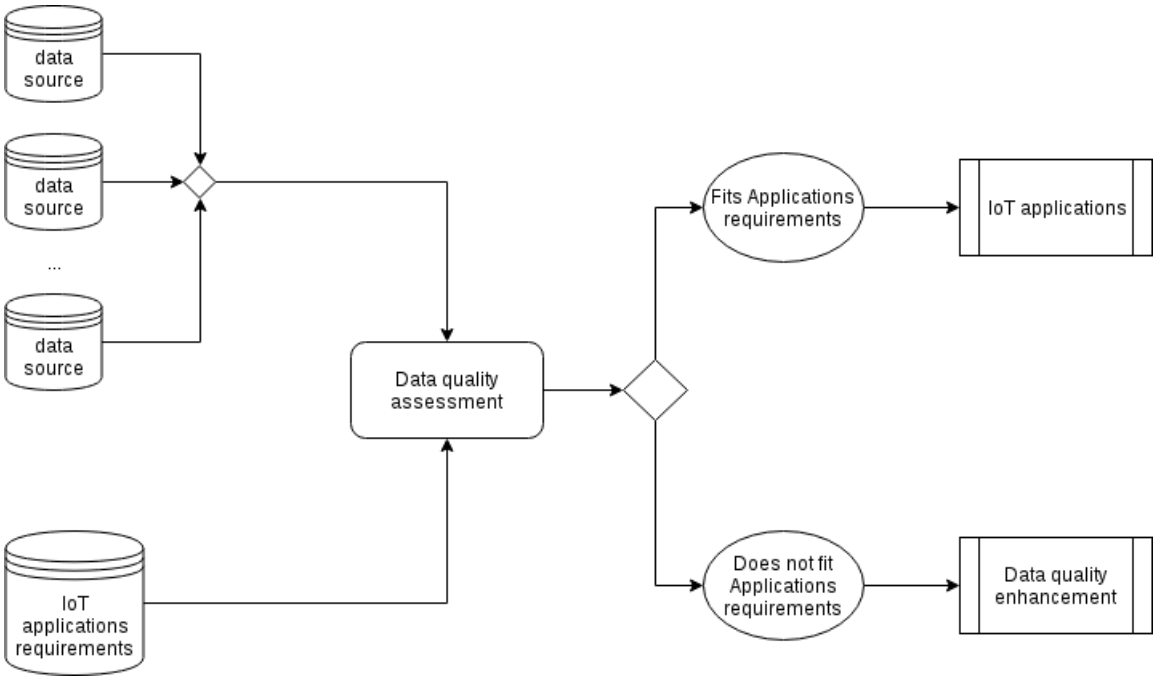

Figure 4.3: Data quality compliance

tain the accuracy of its analysis, the subsystem could prioritize using more recent sensor data, assigning it greater importance in the evaluation. By combining this data with the delayed temperature readings, the subsystem provides a more reliable understanding of the current environmental conditions.

4.2.4 Data quality improvement

The objective of Data Quality Improvement is to analyze the causes of data quality issues within the IoT smart space and modify the data quality management processes accordingly. This ensures that data quality aligns with the requirements of the IoT Software System. The objective of this process is to actively identify the root causes of data quality issues and adjusts the data quality process to address how data quality is managed to prevent future occurrences.

This process is structured around two key functionalities: **root cause analysis** and **process transformations**. These functionalities operate in a complementary manner. First it diagnoses the underlying factors affecting data quality. Then it address the necessary modifications in the data quality management approach to prevent the issues [19].

– **Root Cause Analysis:** This step involves identifying the underlying causes of persistent data quality issues. The analysis is conducted through

a series of measures, such as: **continuous monitoring**, **anomaly detection** and correlation with environmental and infrastructure-related factors [86]. Basically, it continuously logs sensor readings and tracks key data quality metrics such as *accuracy*, *timeliness*, and *completeness* to detect patterns of degradation [27]. An historical data analysis can be executed alongside data from other devices on the system can to detect correlations between data degradation and specific external influences (such as extreme heat) [23]. Also, automated integrity checks can be executed on sensor readings and communication links, verifying whether devices need recalibration, replacement, or alternative data processing techniques. Another issue can be from network delays, when it is affecting data timeliness, routing optimizations can be deployed. If a sensor's accuracy declines over time, recalibration tasks can be scheduled.

– **Process Transformations:** This step focuses on modifying the tasks deployed across the IoT smart space actors and adapting the technologies used. For example, changes could include replacing a routing protocol, switching from HTTP to MQTT for communication [91], applying QoS [36] recalibrating sensors, or adopting new data processing techniques. These adjustments aim to optimize the data quality management process and ensure better alignment with the requirements of the IoT Software System.

In Figure 4.4, the data quality improvement process is illustrated. This process receives the data quality requirements specified by the IoT Software System, which define the desired quality levels for its functionalities, along with the collected data. The process evaluates whether the data quality can be enhanced to better align with these requirements. If improvement is feasible, it acts upon the actors of the IoT smart space to make the necessary adjustments, ensuring that the data meets the specific quality standards required for the effective operation of the IoT Software System. Essentially, this process reviews and modifies how the IoT smart space's actors operate to ensure they can deliver data that satisfies the IoT Software System's quality demands.

For example, if data arrives with a delay, an action might involve implementing QoS mechanisms during the data's transmission through the IoT smart space. Another scenario could involve addressing a malfunctioning sensor, such as a soil moisture sensor that becomes uncalibrated or deteriorates over time due to prolonged exposure to soil. In this case, a potential action might include recalibrating or replacing the faulty sensor to restore its functionality.

If improving the data quality is not feasible, the process can still forward the

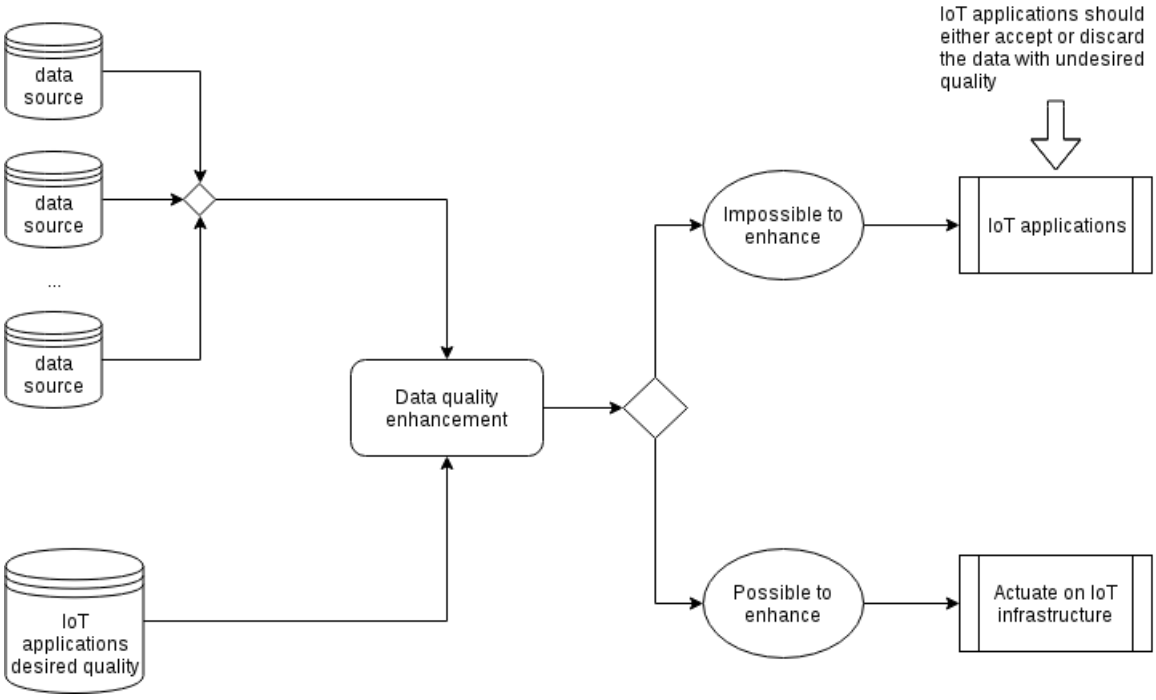

Figure 4.4: Data Quality Enhancement Subsystem

data to the IoT Software System. However, the data is flagged to indicate that it does not meet the required quality standards. This enables the IoT Software System to determine whether to use or disregard the data. Should the system decide to utilize the data, it can account for the associated uncertainty due to the lower quality and adjust its operations or decisions accordingly.

4.2.5 Conclusions regarding the proposed IoT data quality processes

The dynamic nature of IoT highlights that IoT requires these processes to operate autonomously or with minimal human intervention. Verification and control must be applied not only to the data but also to the infrastructure itself, requiring simultaneous execution of these processes. In IoT, control processes must not only manage the operation but also ensure that the infrastructure is functioning according to the data quality management plan. At the same time, assurance processes verify whether the data meets the quality objectives that were previously defined. If the infrastructure is found to be inadequate during control, the current data quality management processes must be adjusted to ensure that data that meets the required data quality requirements.

We conclude this section by emphasizing the necessity of operationalizing data quality management in an autonomous manner. This approach should minimize or eliminate the need for human intervention. So, we envision a **set of software components** to execute data quality management processes within the IoT smart space [29]. To do so, we propose components aligned with the roles of its actors: For **IoT Devices**, components for data collection and preprocessing (detecting anomalies and generating metadata regarding data quality); For the **Edge**, components to deal with the data transmission and routing mechanisms that prioritize data flows based on quality indicators (which are out of scope of this work); and for the **IoT Software System**, components for data consumption data while assessing compliance with quality requirements. Also, for a continuous monitoring and improvement we envision a **Data quality management actor** to orchestrate the entire data quality management process. It should be responsible for dynamically deploying or adjusting tasks as needed to maintain system reliability.

4.3 Data quality management components

In this section, we present the operationalization of data quality management within an IoT smart space. We present a set of components designed to perform data quality management within an IoT smart space. These components are tailored to the specific roles of the actors in the environment: **IoT devices**, the **Edge**, and the **IoT Software System**.

In this section we present a set of components for enabling the execution of the data quality management processes in a way that overcomes these challenges. Traditional data quality management assumes manual oversight, while IoT requires automated mechanisms for data validation, error detection, and self-adaptive corrections. Thus, we integrate the proposed components into the IoT infrastructure. We aim at implementing data quality planning, control, assurance, and improvement dynamically. The objective is to ensure that the system can adapt to environmental changes and infrastructure variations without manual intervention.

The goal is to manage IoT data quality within an IoT smart space where an IoT software system must accommodate multiple objectives, which represent the various functionalities it needs to fulfill within an IoT smart space. Each objective is associated with specific data quality requirements that must be satisfied to ensure proper operation. Since the components operate within IoT smart space devices, they are designed to be lightweight and compatible with

the resource constraints of IoT environments.

The proposed components aim to manage data quality across the entire data lifecycle—from collection to consumption—ensuring that data processing meets the distinct quality requirements of the IoT software system's functionalities. On **IoT devices**, the components are responsible for executing data quality management processes close to the data source, enabling preprocessing directly at the data collection. **At the Edge**, the components focus on managing data quality during transmission, incorporating functionalities such as routing and ensuring QoS. Meanwhile, the components within the **IoT Software System** are designed to verify compliance with data quality requirements, ensuring that the delivered data meets the system's expected standards. Together, these components work across the IoT smart space to manage and maintain data quality effectively.

There is also the task of **scheduling the various data quality management activities** according to the roles of the actors within the IoT smart space. This is done with the aim of implementing the data quality plan and taking actions to improve data quality management when necessary. This task is also performed by components that must be instantiated within the IoT smart space. Initially, we envision that these components can be installed on devices with processing capabilities that are available within the IoT smart space.

4.3.1 Role-Based components allocation

To present the components and how they will be instantiated within the IoT smart space, this subsection introduces a layered approach to allocate data quality management tasks based on the roles of each actor within the IoT smart space: the IoT devices, the edge, and the IoT software system.

In this way, we propose a three-layer approach for the operation of data quality and a transversal management layer that serves all three operational layers (figure 4.5). The three operational layers are as follows: the first layer consists of IoT devices, the second layer is the Edge layer, and the third layer is the IoT software system layer. The transversal layer is dedicated to data quality management, ensuring coordination and oversight across all operational layers.

- The first layer is the *IoT device layer*, where devices communicate directly with each other, forming networks in an ad-hoc manner. This layer is characterized by its energy constraints, as IoT devices are often designed to operate in environments with limited energy resources.

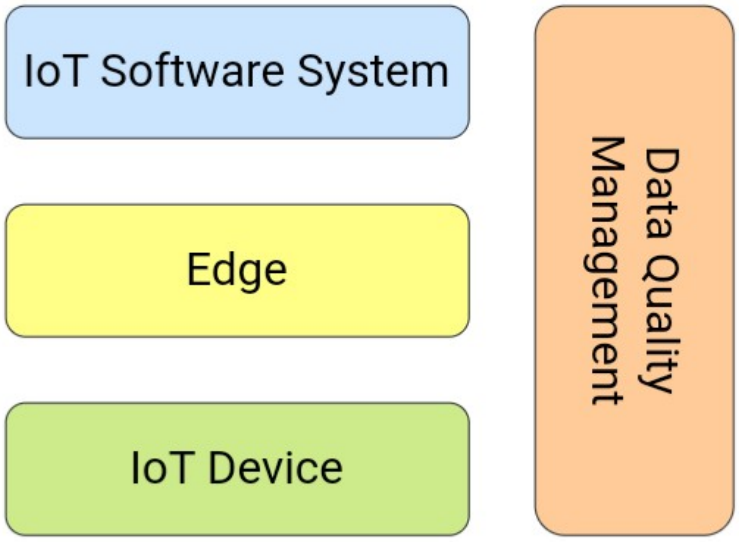

Figure 4.5: Layered approach

To address this, devices in this layer typically implement "sleeping" and "waking-up" mechanisms to conserve energy effectively. Various technologies support the connectivity of IoT devices in this layer, enabling devices to communicate through different wired or wireless protocols, such as ZigBee, Bluetooth, or Ethernet. Managing data quality at the devices within this layer is crucial because it operates close to the source of data generation and the phenomena occurring in the environment. IoT devices within this layer are responsible for capturing, preprocessing, and communicating the assessed condition of the collected data to the IoT software system. This proximity to the data source enables real-time actions, such as identifying anomalies, tagging missing or incomplete data, and ensuring a baseline level of data quality before it is transmitted to the subsequent layers of the IoT smart space. By addressing data quality issues at the source, these devices help reduce the propagation of low-quality data.

– The second layer is the *Edge layer*, which acts as an intermediary between the IoT devices and the IoT software system. The primary role of this layer is to manage data quality during the transmission and routing of data across the network. Devices in this layer are often equipped with greater processing capabilities compared to IoT devices, allowing them to prioritize data traffic by managing QoS. The Edge layer is essential for providing a bridge between data collection and data consumption, addressing potential quality issues that may arise during transit. This layer supports various communication protocols, such as MQTT and CoAP,

and ensures interoperability across heterogeneous IoT devices.

- The third layer is the *IoT Software System layer*, which is responsible for utilizing the data collected from IoT devices. It is within this layer that data quality management is aligned with the specific requirements and objectives of the IoT software system. The IoT Software System layer is characterized by its ability to handle diverse functionalities. Managing data quality at the IoT Software System layer is crucial because it is the final point where data quality compliance is verified before being utilized. Components within this layer evaluate the data against the inherent quality requirements of the IoT software system, identifying non-compliant data and either enhancing its quality or flagging it for further review. This layer can also provide feedback to the Edge and IoT device layers, enabling a continuous improvement loop in data quality management.
- The *Data Quality Management layer* serves as a transversal layer responsible for overseeing the allocation of data quality management tasks across the three previously mentioned layers: the IoT device layer, the Edge layer, and the IoT Software System layer. This layer is designed to distribute and orchestrate data quality tasks to make the data quality management operations aligned with the data quality planning. The primary role of this transversal layer is to maintain the compliance of data quality management processes with the established plan while also enabling adaptive improvements when necessary, even in the face of changes in IoT devices, data conditions, or environmental phenomena. In addition to task allocation, the Data Quality Management layer continuously monitors the performance of the quality management processes, identifying areas that require adjustments or enhancements. This proactive approach ains to address issues and acting as the coordinating layer.

4.3.2 Data Quality Management Components

For the proposed layers, the structure includes the following components and associated databases (illustrated in Figure 4.6):

- **IoT Device Layer:** This layer consists of the *IoT Sensor*, which captures data from the environment, and the *Data Producer Manager*, responsible for preprocessing and tagging the collected data. The associated databases in this layer are *Environment Collected Data*, which stores raw and preprocessed data from the sensors, and *Infrastructure Parameters*, which track the operational state and settings of IoT devices.

- **Edge Layer:** The primary component in this layer is the *Data Routing Manager*, which oversees the efficient and reliable transmission of data through the network. The database associated with this layer is *Routing Rules*, which defines the prioritization and routing strategies to maintain data quality during transmission.
- **IoT Software System Layer:** This layer comprises the *IoT Application*, which processes and utilizes the collected data for specific system functionalities, and the *Data Consumer Manager*, which ensures that the data meets the quality requirements of the IoT software system's functionalities. The databases in this layer include *Service Rules*, which define the quality criteria for different functionalities, and *Application Parameters*, which store specific configuration settings for IoT software system operations.
- **Data Quality Management Layer:** As a transversal layer, it includes the *Data Quality Manager*, which oversees the implementation of data quality processes, and the *Data Quality Task Scheduler*, responsible for distributing and orchestrating data quality tasks across the smart space. The associated databases are *IoT Devices*, containing information about all devices in the smart space, *Data Quality Assessment Tasks*, which store predefined assessment criteria, *Data Quality Enhancement Tasks*, which define improvement actions, and *Scheduling Rules*, which guide the allocation and prioritization of tasks.

4.3.3 Components, databases and interfaces

This subsection describe the components, databases and interfaces (Figure 4.6). The components on its layers are shown in figure 4.7.

IoT Device Layer

The *IoT Sensor* is the **component** that represents the sensors located in the IoT device layer. These devices are responsible for collecting environmental data. The *IoT Sensor* component provides the collected data to the *Data Producer Manager* through the *IoTData* **interface**.

The *IoT Sensor* component interacts with the *environment collected data* **database**, which stores unstructured environmental data. This database contains information related to monitored phenomena, often not tied to specific IoT software system functionalities.

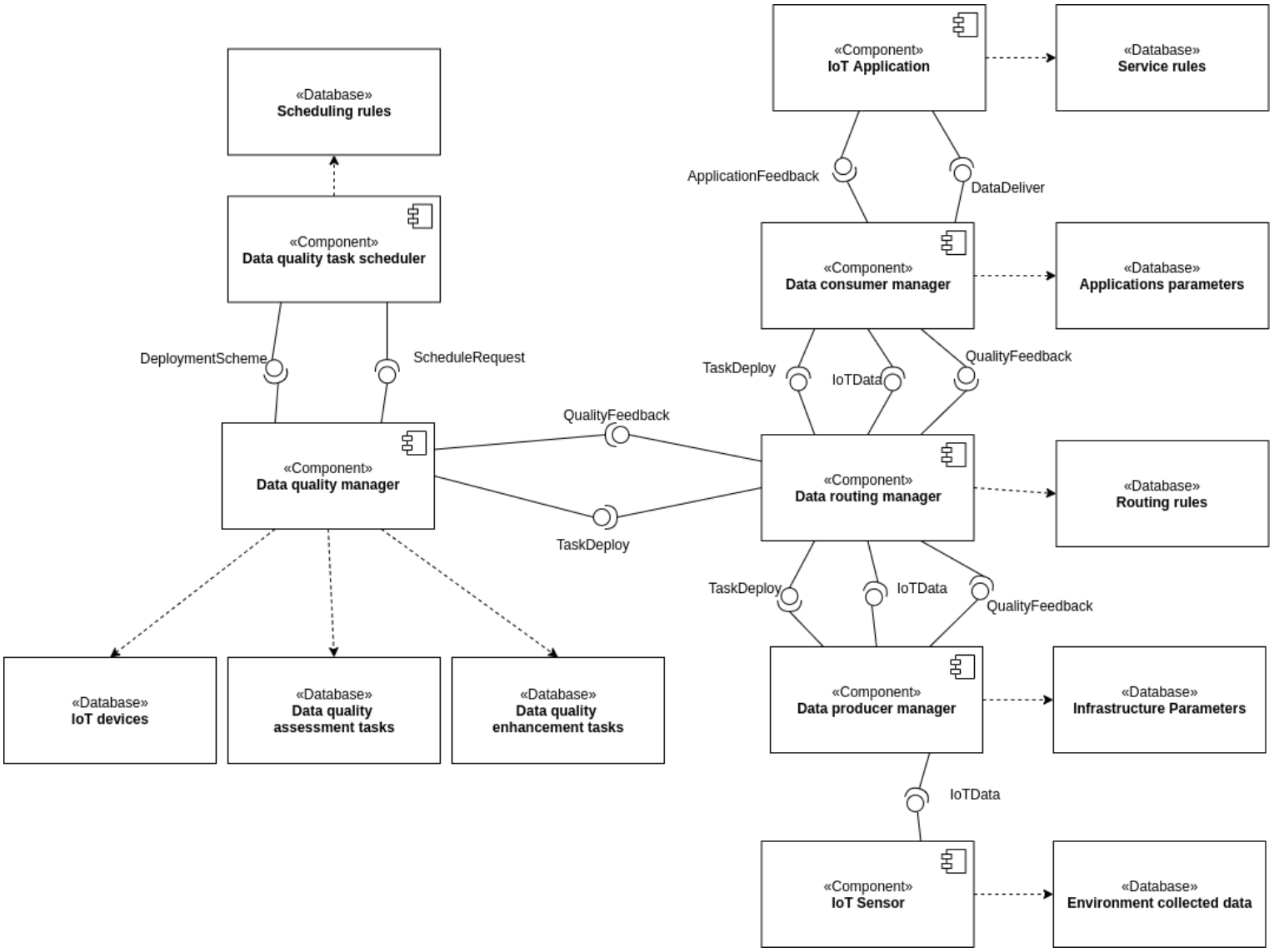

Figure 4.6: Components

The *Data Producer Manager* is the **component** responsible for managing data quality within the device layer. This component applies data quality management tasks for data quality preprocessing.

The *Data Producer Manager* consumes the *IoTData* **interface** provided by the *IoT Sensor* component to access the collected environmental data. Additionally, it consumes the *TaskDeploy* **interface** provided by the *Data Routing Manager* to receive data quality tasks for execution. The *Data Producer Manager* component provides two interfaces: the *IoTData* **interface**, used to forward the processed data, and the *QualityFeedback* **interface**, used to transmit metadata related to the quality of the data. These two interfaces are used for the transmission of the Iot data and the data quality management

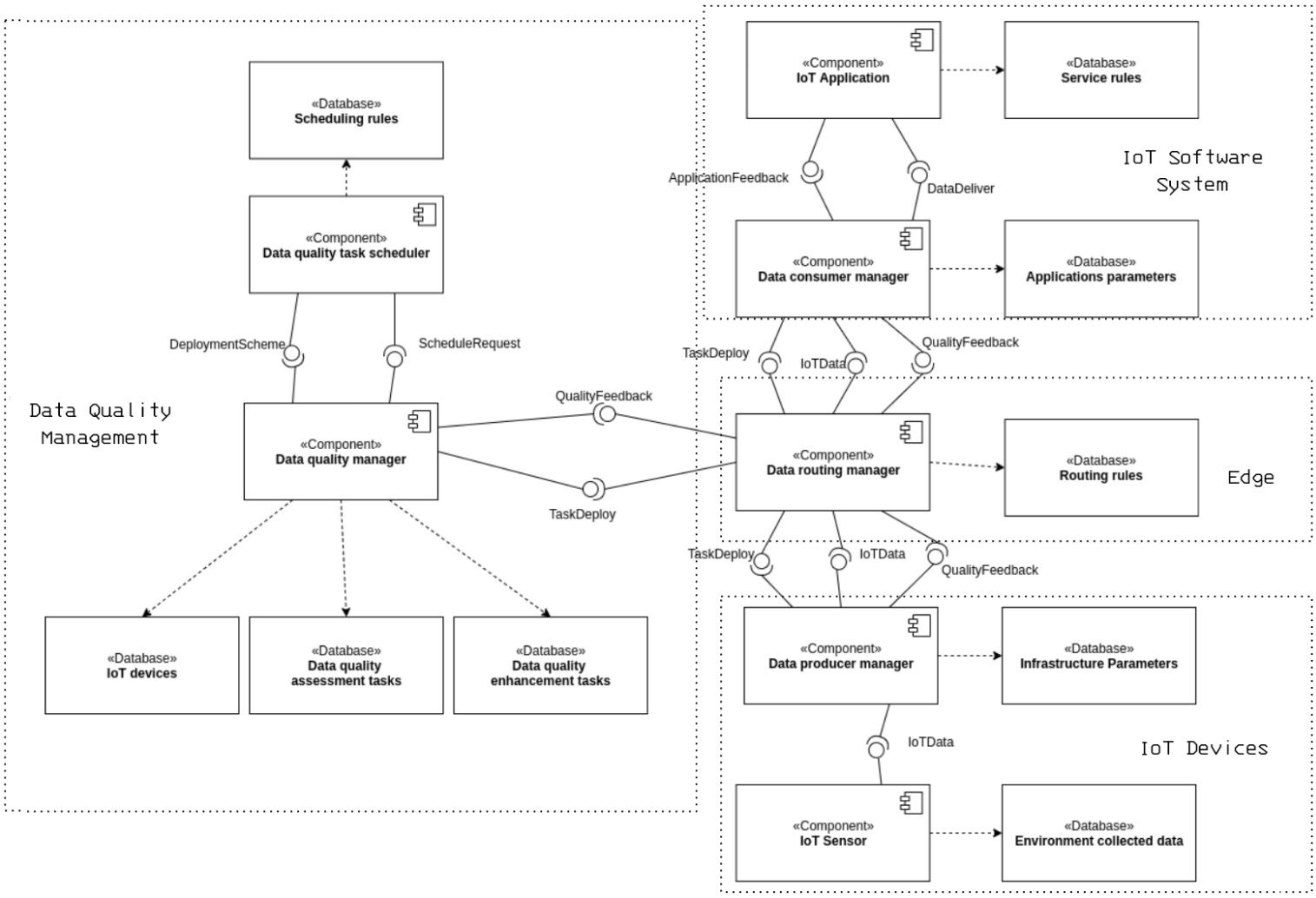

Figure 4.7: Components on its layers

results executed in the IoT device layer.

The *Data Producer Manager* also interacts with the *infrastructure parameters* **database**. This database stores parameters specific to the execution of data quality tasks on IoT devices, ensuring that each task operates according to its required configuration.

Edge Layer

The *Data Routing Manager* is the **component** responsible for routing IoT data and control messages. This means that it serves a dual purpose: managing the traffic of IoT data and handling control messages related to the processes of data quality management. By performing these functions, it connects IoT devices (in the lower layer), the IoT Software System (in the upper layer), and the data quality management process (transversal layer). Additionally,

the *Data Routing Manager* uses the *routing rules* **database**, which acts as a routing table. This table can be populated either statically or dynamically.

The *Data Routing Manager* provides the following interfaces to enable its functionalities:

- *TaskDeploy* **interface**: This interface is provided to the *Data Consumer Manager* and *Data Producer Manager* components for deploying data quality tasks.
- *IoTData* **interface**: This interface is provided to the *Data Consumer Manager* to forward IoT data efficiently from the IoT devices to the IoT software system.
- *QualityFeedback* **interface**: This interface is provided to the *Data Quality Manager* to transmit metadata related to data quality operations.

To execute its tasks, the *Data Routing Manager* consumes the following interfaces:

- *TaskDeploy* **interface**: Provided by the *Data Quality Manager* to receive scheduled data quality tasks to be deployed across the IoT smart space.
- *QualityFeedback* **interface**: Provided by the *Data Producer Manager* and *Data Consumer Manager* to collect feedback on the performance of the data quality management process in their respective layers.
- *IoTData* **interface**: Provided by the *Data Producer Manager* to gather IoT data collected at the IoT device layer for routing to its final destination.

Data Transmission Functionality: As the *Data Routing Manager* is responsible for transferring both IoT data and control messages, it performs the following tasks:

- Transferring IoT Data: The *Data Routing Manager* plays a pivotal role in forwarding IoT data through the *IoTData* interface. It receives data from the *Data Producer Manager* in the lower layer and routes it to the *Data Consumer Manager* in the upper layer.
- Transferring Control Messages: For data quality control messages, the *Data Routing Manager* uses two primary interfaces: (i) *TaskDeploy* interface: To receive data quality tasks from the *Data Quality Manager* and deploy them to the appropriate components based on the target layer. (ii)

QualityFeedback interface: To collect feedback on data quality management from the *Data Producer Manager* and *Data Consumer Manager* components.

Collaboration with the Data Quality Manager: The *Data Routing Manager* works closely with the *Data Quality Manager* to make task deployment and feedback sharing. It provides the *QualityFeedback* interface to send metadata related to data quality operations and consumes the *TaskDeploy* interface to execute scheduled tasks defined by the *Data Quality Manager*.

Depending on the target of the task, the *Data Routing Manager* routes the tasks to the appropriate components:

- For tasks aimed at the IoT device layer, it forwards them to the *Data Producer Manager*.
- For tasks related to the IoT software system, it routes them to the *Data Consumer Manager*.
- For tasks involving routing or Edge-level adjustments, the *Data Routing Manager* executes these tasks directly within its own operations.

IoT Software System Layer

The *IoT Software System Layer* includes two **components**: the *Data Consumer Manager* and the *IoT Application*. These components work in together to ensure data quality for the IoT software system. The *Data Consumer Manager* **component** is responsible for managing data quality within the IoT software system layer. Its main task is to analyze the IoT data it receives against the quality requirements of the IoT software system's functionalities. It performs two key data quality management processes: data quality assessment and data quality enhancement.

- Data Quality Assessment: This process evaluates whether the received IoT data aligns with the predefined quality requirements of the IoT software system. If the data meets the quality standards, it is delivered to the *IoT application* **component**; otherwise, it undergoes the enhancement process.
- Data Quality Enhancement: This process transforms IoT data that fails to meet quality requirements. If the transformed data satisfies the quality requirements, it is delivered to the *IoT application* **component**. However, if the data still does not meet the requirements after transformation, it is discarded.

The *Data Consumer Manager* interacts with various components through the following interfaces:

- *DataDelivery Interface*: Provided to the *IoT Application* **component** to deliver IoT data that has been analyzed and enhanced.
- *ApplicationFeedback Interface*: Consumed from the *IoT Application* **component** to receive feedback on the data used by the application.
- *QualityFeedback Interface*: Provided to the *Data Routing Manager* **component** to send metadata about the results of data quality procedures.
- *IoTData Interface*: Consumed from the *Data Routing Manager* **component** to receive IoT data.
- *TaskDeploy Interface*: Consumed from the *Data Routing Manager* **component** to receive deployed data quality tasks.

Additionally, the *Data Consumer Manager* **component** utilizes the *Application Parameters Database*: This database stores the data quality requirements for the IoT software system's functionalities. Each requirement includes an identifier and quality parameters specifying the desired data quality level.

The *IoT Application* **component** represents the functionalities of the IoT software system within the IoT software system layer. It receives IoT data, process it, and makes decisions based on predefined rules. These decisions drive actions in the IoT smart space. However, the decision-making process can produce unintended outcomes if the input data lacks the required quality.

The *IoT Application* **component** interacts with the *Data Consumer Manager* **component** and utilizes the following resources:

- *ApplicationFeedback Interface*: Provided to the *Data Consumer Manager* **component** to send feedback on the quality and usability of the data.
- *DataDelivery Interface*: Consumed from the *Data Consumer Manager* **component** to receive IoT data that meets the quality requirements.
- *Service Rules Database*: This database stores the decision-making rules in a structured format, enabling the IoT application to process data and perform actions based on these rules.

Data Quality Management Layer

The Management Capabilities Layer is responsible for orchestrating the data quality management process. It has two components: (i) the *Data Quality*

Manager **component** ensures that tasks are planned, monitored, and adapted to meet quality objectives; while the (ii) *Data Quality Task Scheduler* **component** executes the scheduling process to deploy tasks to execute the data quality plan. Together, these components provide a way for managing data quality across the IoT smart space, by coordinating the data quality process on the three layers and adapting the management dynamically according to operating conditions.

The **Data Quality Management Layer** encompasses the components responsible for overseeing and coordinating the entire data quality management process across the IoT smart space. This layer ensures that data quality tasks are effectively scheduled, executed, monitored, and adapted to maintain the desired quality levels in IoT data.

The *Data Quality Manager* is the central **component** in this layer, responsible for coordinating all activities related to data quality management. It plans the data quality process by scheduling tasks to be executed across the device, edge, and IoT software system layers. Once deployed, these tasks are executed by their respective components, and feedback on their operations is provided to the *Data Quality Manager* **component**, enabling it to verify the effectiveness of the process. The *Data Quality Manager* is also tasked with adapting the data quality management strategy in response to any operational issues or unexpected challenges.

The *Data Quality Manager* provides the following interfaces:

- *TaskDeploy Interface*: Is provided by this **component** to the *Data Routing Manager* **component**, this interface is used to deploy data quality tasks to the components in the device, edge, and IoT software system layers.
- *ScheduleRequest Interface*: Is provided by this **component** to the *Data Quality Task Scheduler* **component**, this interface is used to request a deployment scheme that includes data quality tasks and their intended deployment locations.

The *Data Quality Manager* consumes the following interfaces:

- *QualityFeedback Interface*: This interface is provided by the *Data Routing Manager* **component**, it is used to deliver feedback on the data quality management procedures executed by the *Data Producer Manager*, *Data Routing Manager*, and *Data Consumer Manager* **components**.

– *DeploymentScheme Interface*: This interface is provided by the *Data Quality Task Scheduler* **component**, this interface to deliver deployment schemes specifying which tasks need to be executed and their target locations.

The *Data Quality Manager* utilizes the following databases to support its operations:

– *IoT Devices Database*: Stores information about each IoT device, including its unique identifier, location, and assigned layer (device, edge, or IoT software system).
– *Data Quality Assessment Tasks Database*: Contains executable tasks designed to evaluate IoT data quality, such as anomaly detection or outlier identification mechanisms.
– *Data Quality Enhancement Tasks Database*: Stores tasks aimed at improving IoT data quality, such as data interpolation methods for mitigating missing data or historical data usage in fusion windows.

The *Data Quality Task Scheduler* is a specialized **component** that offloads the scheduling responsibilities from the *Data Quality Manager* **component**. Its primary function is to generate deployment schemes by applying scheduling rules to task, device, and objective information. By automating the scheduling process, this component ensures that data quality tasks are efficiently allocated and executed across the IoT smart space. The *Data Quality Task Scheduler* **component** utilizes the *Scheduling Rules Database* to store the rules used to generate deployment schemes, including information about task priorities, execution frequencies, and resource constraints.

4.4 Data quality management tasks

In this section, we introduce two techniques for executing data quality management tasks at both ends of the data pipeline within an IoT smart space: the point of data collection by IoT devices in the **lower layer** and the point of data quality compliance within the IoT Software System in the **upper layer**.

This section is structured as follows: we first present the data preprocessing technique designed for deployment in the IoT devices of the lower layer. Following this, we propose a data quality compliance technique designed for deployment in the IoT Software System layer, ensuring that the data delivered to the IoT Software System adheres to its quality requirements.

4.4.1 IoT Devices: Data Quality Preprocessing Technique

In this section, we present a data preprocessing technique designed to operate on IoT devices, aligning with the concepts proposed in this work. The technique has the objective to assess data quality at the source before entering the data pipeline. This approach aims to enhance data quality during collection, addressin–g challenges that arise in dynamic IoT environments.

The proposed preprocessing technique focuses on detecting outliers. Outliers are defined as data points that deviate significantly from normal behavior and may result from issues such as malfunctioning sensors or legitimate environmental events [19] [68] [21]. For instance, a sudden rise in temperature detected by a single sensor or a small group of sensors could indicate an early-stage fire. Distinguishing between these scenarios is crucial to accurately represent the environmental conditions and ensure reliable data for decision-making processes.

Anomalies are a common challenge in IoT systems that can compromise data quality. There are numerous solutions in the literature for detecting and eliminating outliers and anomalies. However, for the context of this work—where the goal is to preprocess data in a decentralized manner and send it to the IoT Software System—it is necessary to propose a technique whose behavior aligns with the specific objectives and requirements of this work: The technique must align with the decentralized nature of the preprocessing process and support the constant improvement of the data quality management workflow. An objective is to be lightweight design in terms of imposed memory overhead and have small executable size.

Detecting outliers at the source is particularly important because IoT environments are dynamic, with various events occurring in the monitored area. Each event generates data within specific ranges, which may differ significantly from other events, increasing the complexity of data analysis. By assessing data quality in real time, IoT devices can provide essential insights to the IoT software system regarding environmental conditions of the IoT smart space.

However, an IoT software system receiving the data may have multiple functionalities (e.g., controlling air conditioning systems or detecting fires), so this task should not eliminate outliers, it should enable real-time data quality control for any functionality that depends on the collected data.

The proposed data cleaning technique enables IoT software systems to detect and address errors in collected data by identifying outliers and determining

whether they result from legitimate events or IoT challenges that compromise data quality. The technique is designed to be simple, imposing low energy consumption, minimal memory usage, and negligible delays, making it well-suited for resource-constrained IoT devices.

Overview

In an overview, the task is to evaluate whether a newly received data from a Data Source is an outlier or not. The basic idea is to use the historical data and check if its neighboring sources also detected an outlier. If an outlier was detected by its neighbors, then the data is characterized as an event outlier (a legit event in the monitored area). On the other hand, if an outlier was not detected by its neighbors, it should be informed that it is an error outlier.

We propose the execution in two steps: the outlier detection step and the outlier characterizing step. During the **outlier detection step**, it uses the **Interquartile range** statistical approach to detect outliers since it is a well-known statistical outlier detection approach that presents low computational overhead and well suits the constrained devices on IoT.

Then, in the **outlier characterizing step**, IoT devices send the data, already labeled as outlier or non-outlier, to an aggregating node. This node is responsible for receiving the data and verifying if the same outliers were detected by neighboring sensors. If the outliers were identified by multiple sensors, they are classified as a legitimate environmental event rather than an infrastructure error.

Outlier detection step

Here we present the outlier detection step. Ou solution is designed to: (i) be lightweight in terms of the imposed memory overhead; (ii) impose low communication overhead to be transmitted; and (iii) do not impose a significant delay. The features (i) and (ii) are justified since sensor devices present restricted amount of memory and energy resources [92]. The feature (iii) is justified since delays in while performing its tasks degrade the timeliness of the data for applications that require data in real time, or near real time, to make decisions [19].

In outlier detection step data is received and it is performed the evaluation of the Interquartile range of the data, which takes constant time to be performed and directly access data stored in the Random Access Memory (RAM) of the network nodes.

Also, the outcome is a single bit of information whether the data is "event outlier" or "non-outlier" (to impose low communication overhead). After, this information is given as input to the **outlier characterizing step**.

Outlier detection algorithm Data Structures: The outlier detection step (Algorithm 1) has two main data structures: *RecData* and *RW*. *RecData* is the data structure responsible for storing the data that is going to be analyzed. *RW* is the data structure that stores the historical data received from the Data Source (DS).

The other data structures used are the following: *P25* is responsible for storing the percentile 25 of the data in RW. *P75* is responsible for storing the percentile 75 of the data in RW. *IRQ* is responsible for storing the *inter quartile range* of the data stored in RW. In *Uboundary* is stored the upper boundary of the data analysis and in *Lboundary* is stored the lower boundary of the data analysis. In *Qualif* is stored the DQ assessment.

Pseudocode: The pseudocode is the algorithm 1, which has two phases: the **Setup Phase** and the **Operation Phase**. The Setup Phase is responsible for populating the data structures (Alg. 1, line 1) as setting up the size of the RW data structure (Alg. 1, line 2). The Operation phase is responsible for executing the task (Alg. 1, lines 4 to 28).

The Operation Phase is the main phase and begins by receiving a data input (Alg. 1, line 4), storing it in RecData data structure (Alg. 1, line 5) and then, appending the data to the RW data structure to be further analyzed (Alg. 1, line 6). It evaluates if there is historical data by evaluating if the size of the RW data structure is greater than one (Alg. 1, line 7). If there is no historical data, it waits for new data.

If there is historical data in RW, it sorts the data in RW (Alg. 1, line 8) to calculate the 25 and 75 percentiles. The 25 and 75 percentiles are stored in the P25 and P75 data structures, respectively (Alg. 1, lines 9 and 10). It uses the values in P25 and P75 to calculate the inter quartile range by simply executing the difference between the data in P75 and the data in P25. The inter quartile range is stored in the IQR data structure (Alg. 1, line 12).

After calculating the inter quartile range, it calculates the lower boundary and the upper boundary and stores them in the Lboundary (Alg. 1, line 14) and Uboundary (Alg. 1, line 16) data structures, respectively. Since a Normal Distribution presents its peak of data within its quartiles [89], outliers in this work are defined as observations that fall below Q1 - 1.5 IQR or above Q3 + 1.5 IQR. Hence, it uses a factor of 1.5 for multiplying the data on IQR .

Algorithm 1 The outlier detection step

Input: The data structures to be set up
Setup Phase :
1: Populate the data structures
2: Allocate memory space for RW
Operation Phase :
Input: RecData
Output: (RecData, Qualif)
3: **loop**
4: Receive a piece of data to analyze
5: RecData = received piece of data
6: RW.append(RecData)
7: **if** (RW.size() > 1) **then**
8: RW.sort()
9: p25 = percentile(RW, 0.25)
10: p75 = percentile(RW, 0.75)
11: IQR = p75 - p25
12: Uboundary = p75 + 1.5*IRQ
13: Lboundary = p25 - 1.5*IRQ
14: **if** ((RecData > Uboundary) **or** (RecData < Lboundary)) **then**
15: RW.delete(RecData)
16: Qualif = 'outlier'
17: **send** (RecData, Qualif)
18: **else**
19: Qualif = 'non-outlier'
20: **send** (RecData, Qualif)
21: **end if**
22: **end if**
23: **if** (RW.isFull() **then**
24: RW.deleteOldest()
25: **continue**
26: **else**
27: **continue**
28: **end if**
29: **end loop**=0

After setting the lower and upper boundaries, it analyzes if the received data stored in RecData is greater than the upper boundary or smaller than the lower boundary (Alg. 1, line 14). If the data received is lower or greater than the boundaries, it is removed from the RW data structure (Alg. 1, line 15) and is considered as outlier (Alg. 1, line 17). Else it is considered as non-outlier (Alg. 1, line 20).

Then, it verifies if the RW data structure is full (Alg. 1, line 23). If RW is full, it deletes the oldest data in its data structure (Alg. 1, line 24) and continues its operations (Alg. 1, line 25). This step is important, for updating it as the environment changes. By always deleting the oldest data, it is aways renewing its understanding about the environment. If RW is not full, it just continues its operation (Alg. 1, line 27).

Outlier characterizing algorithm

Data Structures: After executing the **outlier detection step**, the data is sent to the aggregating node that executes the **outlier characterizing step**. The outlier characterizing step is an heuristics that evaluates if more than one outlier is detected within the same time by two or more nodes. If an outlier is detected by more than one node, those outliers are labeled as product of a legit event. Else, the outlier is labeled as product of errors due to IoT infrastructural challenges. The outlier data and its label are sent to the IoT software system.

In this subsection, we present the data structures used in the pseudocode for the outlier characterization algorithm based on neighbor states (Algorithm 2). This algorithm builds upon the results of the outlier detection algorithm and characterizes whether an outlier is the result of a legitimate event or an infrastructure error.

The following data structures are used:

- **RecOutlier**: This data structure stores the outlier data and its corresponding flag (‘OutlierFlag‘) as identified by the outlier detection step. The flag indicates whether the data was classified as an outlier (1) or not (0).
- **NeighborOutlierFlags**: This structure stores the flags received from neighboring sensors. It contains the outlier status (‘OutlierFlag‘) reported by each neighbor for the same time interval.
- **EventFlag**: This binary flag indicates whether the outlier is the result of a legitimate environmental event (1) or an infrastructure error (0).

These data structures are critical for enabling the characterization process to analyze and categorize the outlier based on spatial correlation with neighboring sensors.

Pseudocode: Here, we present and describe the pseudocode for the outlier characterization algorithm (Algorithm 2). The algorithm has two main phases: the Setup Phase and the Operation Phase.

The **Setup Phase** initializes the necessary data structures and allocates memory for storing neighbor flags (Alg. 2, lines 1–2).

The **Operation Phase** is the main phase where the algorithm processes outliers identified by the detection step, collects feedback from neighboring sensors, and determines whether the outlier represents a legitimate event or an error (Alg. 2, lines 4–17).

Algorithm 2 Outlier Characterization Based on Neighbor States

Input: RecOutlier, NeighborOutlierFlags
Output: EventFlag

Setup Phase:
1: Initialize the data structures: NeighborOutlierFlags.
2: Allocate memory for NeighborOutlierFlags.
Operation Phase:
3: **loop**
4: Receive RecOutlier (data and its outlier flag) from the outlier detection algorithm.
5: **if** RecOutlier.OutlierFlag = 1 **then**
6: Broadcast OutlierFlag to Neighbor Sensors.
7: Receive NeighborOutlierFlags from Neighbor Sensors.
8: Count the number of neighbors with OutlierFlag = 1.
9: **if** Count > 1 **then**
10: EventFlag = 1. {Legitimate event detected.}
11: **else**
12: EventFlag = 0. {Error detected.}
13: **end if**
14: **send** (RecOutlier.Data, EventFlag) to the IoT Software System.
15: **else**
16: **continue**. {No outlier detected, continue monitoring.}
17: **end if**
18: **end loop**=0

The **Operation Phase** begins by receiving the ‘RecOutlier‘ data structure from the outlier detection algorithm (Alg. 2, line 4). If the data is flagged as an outlier (‘RecOutlier.OutlierFlag = 1‘), the algorithm broadcasts this outlier flag to neighboring sensors (Alg. 2, line 6) and subsequently collects the outlier flags from its neighbors (Alg. 2, line 7).

The algorithm then counts the number of neighbors that also flagged the data as an outlier (Alg. 2, line 8). If more than one neighbor flagged the same data as an outlier, the algorithm categorizes the outlier as a legitimate event ('EventFlag = 1') (Alg. 2, line 10). Otherwise, the outlier is categorized as an error resulting from an IoT infrastructure challenge ('EventFlag = 0') (Alg. 2, line 12).

Finally, the algorithm sends the original data along with its 'EventFlag' to the IoT Software System for further analysis or decision-making (Alg. 2, line 14). If the data is not flagged as an outlier, the algorithm simply continues monitoring (Alg. 2, line 16).

4.4.2 IoT Software System: Data Quality Compliance Technique

In this chapter, we present a data quality compliance technique designed for use by the IoT software systems. The primary objective of this technique is to assess the quality of data to ensure it meets the specific requirements of the IoT software system. To achieve this, it is necessary to select and define a specific data quality dimension as the focus of the technique. For this proposal, the chosen data quality dimension is the **confidence dimension**.

Confidence can be **defined** as how much trust it is put in data. According to [19], the confidence requirement can be defined as a statistical error ϵ such that an interval $[X - \epsilon, X + \epsilon]$ contains the real value X with a confidence probability of p.

The confidence dimension is an intrinsic data quality semantic aspect [19]. The different objectives and functionalities of an IoT software system may present different confidence probability p values and statistical error ϵ.

Processing the IoT data for improving the representation of one or more phenomena that take place in a monitored area is a challenge. The improvement of the data representation implies in an improvement in the data quality aspects of the data for a given application, since the better is the data representation, the greater is the confidence about the correct representation of the data for one or more phenomena in a monitored area [19] [68] [21] [1].

Overview

This technique has two modules: (i) Data Quality Assessment Module (DQAM) that assess the sensor collected data quality to verify if it meets

the data quality requirements of the IoT applications and (ii)Data Quality Enhancement Module (DQEM) that process the data to meet the confidence requirements of the IoT applications. Both DQAM and DQEM can run independently in different nodes or in a single node.

In a previous work we proposed an information fusion technique capable of inferring features about the different phenomena that take place in the monitored area. In this work, we use the concepts on [1] to process data with the objective to increase the confidence on the collected data.

The basic idea is to make a peak analysis on the data to infer about the set of phenomena that are taking place on the IoT smart space. The occurrence of multiple phenomena can affect how one phenomena is represented on data, thus affecting the confidence on the data [1]. By executing a peak analysis, this task aims at increasing the data confidence by separating the data from multiple different phenomena.

In general, this solution receives the different p confidence probabilities and ϵ statistical error for each application. Then, evaluates the data quality aspects of the collected data to verify if it meets the required confidence probability of each application. And, depending on the result of the data analysis, if the desired confidence is not met, it migh execute a peak analysis to divide the data for the multiple phenomena on the monitored area.

To enhance the data quality of the IoT data, it uses the mean and skewness [89] [93] [94] of the collected data to perform its data quality enhancement. We propose the use of these statistics due to its simplicity to be implemented on a sensor node and their low computational cost.

Basic Operation

We propose a cyclic operation: receiving data and application requirements, assessing the data quality and enhancing data quality (as shown in Figure 4.8). Prior to starting its operations a Set-Up phase takes place to populate the data structures with default values and clean the memory space.

An execution cycle starts by receiving data and application requirements. Then, the DQAM executes an heuristics that evaluates if a given data meets the quality requirements of the IoT applications. In this work the DQAM is proposed to evaluate the data quality confidence dimension. The data quality confidence requirement is defined as a statistical error ϵ such that an interval $[X - \epsilon, X + \epsilon]$ contains the real value X with a confidence probability of p [19]. In general words, DQAM receives from each α IoT application a tuple

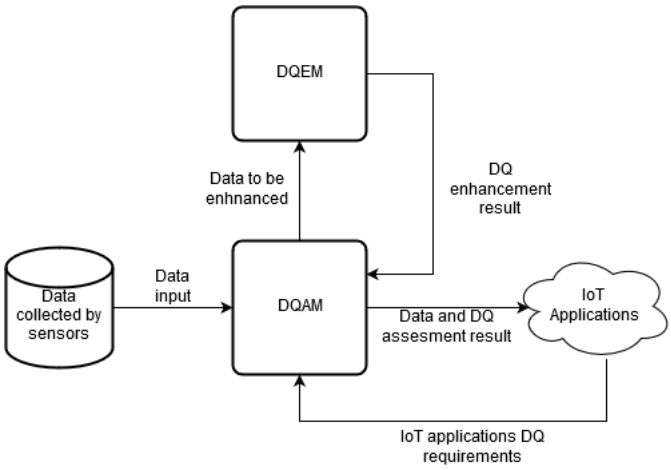

Figure 4.8: The cyclic operation

$\tau_\alpha = (\epsilon, p)$ for the data quality requirement of the α IoT application. Then DQAM applies an heuristics on the data to evaluate if the data meets the τ_α requirements. If the data meets the data quality requirements, it is provided to the applications flagged as data with desired quality. Else, it executes the DQEM.

The DQEM receives the data that does not meet the data quality requirements and tries to enhance the data quality. The DQEM integrates the collected data and infers data features. The inferred features are used to separate data generated by different phenomena. By separating the different data, DQEM aims at decreasing the range of the confidence interval of the estimated data parameter. After executing the DQEM, the generated data is given as input to the DQAM. The DQAM evaluates if the generated data meets the quality requirements. In the case of the data has been already processed by DQEM and it still does not meet the data quality requirements, it is flagged as data with undesired quality. It delivers the data to the applications and the IoT applications decide to use such data given the flagged data quality.

Data structures and functions

The following data structures are used: $\tau_\alpha = (\epsilon, p)$, DI=(RD, Pflag), QD=(*data*, Qflag), IntConf, me, sk, LD, HD. Also it implements two functions ConfidenceIntervalCalc(*data*, *probability*) and Range(*interval*).

The τ_α data structure stores the data quality requirements of an application and is composed by the ϵ data structure, which stores an statistical error and is composed by the p data structure, which stores a confidence probability. The DI data structure stores the data given as input and is composed by

the RD data structure, which stores the received data and the Pflag data structure, which stores a flag to inform if the data was processed by DQEM. The IntConf data structure stores the confidence interval that is calculated. The me data structure stores the mean of the data. The sk data structure stores the skewness of the data. The LD data structure stores the data values lower than the mean of the data. The HD data structures stores the data values higher that the mean of the data.

The ConfidenceIntervalCalc(*data*, *probability*) function calculates the confidence interval for the average of the data with a given probability. And the Range(*interval*) function calculates the range of a data interval.

DQAM

The Data Quality Assessment Module (DQAM) (Algorithm 3) starts by receiving the IoT applications requirements and the IoT data. The DQAM receives as input the IoT applications data quality requirement and stores it in τ_α. And the DQAM receives the collected data and stores it in DI data structure. The τ_α stores the statistical error in ϵ and the confidence probability in p. The DI data structure stores the received data in an internal data stricture called DI.RD and stores a Boolean value as flag (values 1 or 0) in the Pflag internal data structure. The Pflag data structure is set as default to 0, which indicates that the data has not yet passed through DQEM.

After receiving the IoT applications data quality requirement and the collected data, the DQAM calculates the confidence interval of the mean of the received data stored in DI.RD using the confidence probability in p required by the IoT application and stores it in IntConf data structure (line 1, algorithm 3). Then DQAM verifies if the range of the confidence interval is less or equal than the double of the statistical error in ϵ (line 2, algorithm 3).

If the range is less or equal than the double of the statistical error, than the data meets the data quality requirements of the IoT application. In this case, DQAM returns the data to the IoT application informing that this data meets the data quality requirements of the IoT application. This is done by returning the data using the QD data structure. The QD data structure stores the data that was stored in the DI.RD data structure along with a Boolean value of 1 stored in the Qflag internal data structure (line 3, algorithm 3).

Else, if the range is greater than the double of the statistical error, than the data does not meet the data quality requirements of the IoT application (line 4, algorithm 3). In this case DQAM should send the data to the DQEM.

Algorithm 3 Data Quality Assessment Module (DQAM)

Input: $\tau_\alpha = (\epsilon, p)$, DI=(RD, Pflag);
Output: QD=(*data*, Qflag);

```
1: IntConf = ConfidenceIntervalCalc(DI.RD, τα.p);
2: if Range(IntConf) ≤ 2 x τα.ϵ then
3:    return (DI.RD, Qflag=1);
4: else
5:    if DI.Pflag == 0 then
6:       DQEM(DI.RD);
7:    else
8:       return (DI.RD, Qflag=0);
9:    end if
10: end if
    =0
```

However, to prevent that the data was not previously sent to the DQEM, DQAM verifies if the flag in DI.Pflag of the received data in DI is set to 0 (line 5, algorithm 3). DQAM verifies if the DI.Pflag is set to zero, which is the default value of the DI.Pflag when the data is received.

If the flag in DI.Pflag of the received data in DI is set to 0, DQAM sends the data to the DQEM (line 6, algorithm 3). Else, If the flag in DI.Pflag of the received data in DI is not set to 0 (line 7, algorithm 3), the data has already been sent to the DQEM and there was no effective enhancement on the data. In this case the data does not meet the data quality requirements and DQEM was not capable of enhancing the data quality to meet the data quality requirements of the IoT application. In this case, DQAM returns the data to the IoT application informing that this data does not meet the data quality requirements of the IoT application. This is done by returning the data using the QD data structure. The QD data structure stores the data that was stored in the DI.RD data structure along with a Boolean value of 0 stored in the Qflag internal data structure (line 8, algorithm 3).

DQEM

The Data Quality Enhancement Module (DQEM) (Algorithm 4) sends the received data to the DQEM if the data does not meet the data quality requirements of the IoT application. Hence, the input of DQEM is the DI.RD data structure. The first step of DQEM is to calculate the mean of the data in DI.RD (line 1, algorithm 4). Than, DQEM calculates the skewness of the data in DI.RD (line 2, algorithm 4).When skewness is less or equal than 1 there is a single phenomenon or multiple phenomena generating data with similar

Algorithm 4 Data Quality Enhancement Module (DQEM)

Input: DI.RD;
Output: Data Quality Enhancement;
1: $me = mean(DI.RD)$;
2: $sk = skewness(DI.RD)$;
3: **if** $|sk| \leq 1$ **then**
4: DQAM(DI.RD, Pflag = 1);
5: **else**
6: LD = DI.RD $\leq me$;
7: HD = DI.RD $> me$;
8: DQAM(LD, Pflag = 0);
9: DQAM(HD, Pflag = 0);
10: **end if**
=0

data ranges in the monitored area environment [1]. Else, if skewness is greater than 1, then there are multiple phenomena occurring in the environment [1]. By separating the multiple phenomena, DEQM improves the confidence data quality semantic aspect.

If the skewness is less or equal than 1 (line 3, algorithm 4), DQEM was not capable of enhancing the data quality for the data. In such case, DQEM sends the data to the DQAM with the boolean flag value of 1 in Pflag data structure. This flag informs to the DQAM that this data has already been sent to DQEM (line 4, algorithm 4).

Else, if the skewness is greater than 1 (line 5, algorithm 4), DQEM divides the data in DI.RD data structure into two pieces of data. DQEM stores the data less or equal than the average on LD data structure (line 6, algorithm 4) and the data greater than the average on HD data structure (line 7, algorithm 4). After dividing the data in DI.RD and storing it in LD and HD data structures, DQEM sends the data in LD and HD data structures to the DQAM with the boolean flag value in Pflag set to 0 (lines 8 and 9, algorithm 4). DQEM sends the data with the Pflag set to 0 because when it divides the original data in DI.RD data structure into two, it creates two new pieces of data that should have its data quality assessed.

After executing DQAM and DQEM, it sends the data to the IoT application along with the quality assessment result. Then the applications receive the data and decide about the use of the data.

4.5 Conclusion

This chapter presented our proposal for data quality management in IoT smart spaces from its generation to its final use. The solution integrates processes, a structured set of distributed components across the IoT device layer, Edge layer, and IoT software system layer, and specific techniques designed to address key data quality challenges in alignment with the core concepts and objectives of this work: (i) The proposed **preprocessing technique** for IoT devices ensures early detection of data quality issues, while the **compliance assessment** in the IoT software system validates data according to the data quality requirements of the IoT software system, in this work we propose a task for the data quality **confidence dimension**. Our solution is designed to be both scalable and adaptable, addressing the dynamic and evolving nature of IoT smart spaces. The main innovation of this proposal lies in the distributed execution of data quality tasks and the proactive management of data quality aligned with the IoT software system's requirements.

Limitations: implementing this approach in real-world scenarios may present challenges, such as handling large-scale deployments or adapting to different IoT domains with distinct requirements. The integration of data quality assessment techniques, performance evaluations in real-world IoT environments, and enhancements in task scheduling algorithms to optimize resource use across IoT actors may be a challenge for the complete realization of this proposal.

Considerations: the proposed solution offers a promising foundation for effective data quality management in IoT smart spaces, addressing the complexities and dynamic nature of these environments.

Chapter 5

Proof of Concept: Implementing Data Quality Management in an IoT Smart Grid Scenario

In this chapter, we present a Proof of Concept (PoC) for the proposed data quality management solution presented in this thesis (chapter 4). This is an instantiation of the proposed software architecture. We implement our solution it in a Smart Grid monitoring scenario [11]. The objective is to demonstrate how the data quality management processes, components, and techniques can be applied in a real-world IoT environment. Through this PoC, we verify the feasibility of our solution.

We selected a **Smart Grid monitoring scenario** [11], where IoT devices are deployed to monitor **power transmission lines and battery storage systems** [1]. This scenario was chosen due to its complexity, dynamic behavior, and critical role in ensuring a stable and efficient energy distribution. In this scenario, sensor data is collected to monitor two essential elements for energy distribution: **Overhead Power Lines** and **Battery Storage Systems**. Overhead power lines transport electricity over long distances, and battery systems store excess energy for load balancing [29]. However, the decision-making processes in Smart Grid operations can be endangered by issues on the grid. For instance, IoT sensors deployed in the Smart Grid environment are susceptible to sensor drift [19], measurement noise, environmental interference, and network delays and other issues. These challenges can compromise the operations of these grids. Therefore, implementing a data quality management solution is essential for its operations.

The rest of this chapter is the following: In Section 5.1, is about the scenario

and the functionalities of the IoT Software System in the Smart Grid environment. Section 5.2 is related to the execution of *Data Quality Planning*. Section 5.3 is about the implementation of *Data Quality Control*. On section 5.4 the application of *Data Quality Assurance* is presented. Also, on section 5.5 is about the the process of *Data Quality Improvement*.

5.1 Scenario Description

A **Smart Grid monitoring** IoT software system requires the distribution of IoT devices to collect real-time data on the smart grid. The IoT devices collect **temperature, voltage levels, and battery charge status** data [1]. The task of ensuring the quality of this data is essential for keeping the IoT software system operational for energy distribution. Based on the collected data, the IoT software system supports multiple functionalities aimed at optimizing power transmission and storage. These functionalities include:

- **Overhead Power Line Monitoring (OPLM):** This functionality is related to a safe operation of power transmission lines. This is archived by monitoring their temperature and detecting overheating conditions. This happens because excessive temperatures can indicate **overloading, equipment degradation, or environmental factors**. High ambient heat or adverse weather conditions can endanger the equipment and detecting and addressing these issues promptly prevents damage to power lines and avoids power outages.
- **Battery Monitoring (BM):** This functionality is related to the monitoring of the status of energy storage systems. It works by measuring battery charge levels and temperature. This functionality is important to make load balancing of the energy loads to optimize battery lifespan. It ensures that batteries operate within safe temperature ranges, preventing thermal events that could compromise storage capacity and system reliability.
- **Energy Management and Optimization (EMO):** This functionality is related to the providing of insights regarding the for **grid stability and efficiency**. The objective is to analyzing historical and real-time data. It operates by checking power demand patterns, energy losses, and environmental influences. This functionality enables the system to adjust energy distribution strategies. It is important to reduce inefficiencies and improving the resilience of the grid.

5.2 Data Quality Planning

This section presents the **Data Quality Planning** process for the **Smart Grid Monitoring System**. Each subprocess is presented: (i) **Requirements Management** is presented in subsection 5.2.1; (ii) **Data Quality Strategy Management** is presented in subsection 5.2.2; (iii) **Data Quality Procedures Management** is presented in subsection 5.2.3) and (iv) **Data Quality Implementation Planning** is presented in subsection 5.2.4.

5.2.1 Requirements Management

This subsection covers the identification and definition of data quality requirements for monitoring power transmission lines and battery storage systems. IoT devices collect temperature and energy storage data to serve an IoT Software System with multiple functionalities. The requirements of each functionality are the following:

- **Overhead Power Line Monitoring (OPLM)**: For this functionality we identified the *timeliness* and *accuracy* data quality dimensions as requirements. The (i) *timeliness* data quality requirement was chosen in order to ensure that real-time temperature arrive on time to prevent overheating and damage on the overhead power line. The (ii) *accuracy* data quality dimension was chosen to in order to avoid that deviations in temperature measurements can lead to incorrect assessments of the real condition. The use of these two data quality dimensions were chosen to avoid unnecessary shutdowns and failures on the detection of overheating events.
- **Battery Monitoring (BM)**: For this functionality we identified the following data quality dimension as requirements to its correct functions: *accuracy* and *completeness*. The (i) *accuracy* data quality dimension was chosen to ensure that battery charge levels are measured correctly. This prevents either overcharging or premature depletion of the energy of the battery. By preventing these issues we avoid the degradation of energy storage systems, which is important to increase its lifespan. The (ii) *completeness* data quality dimension was chosen to ensure that the IoT devices that collects data from the battery report their data consistently. It is responsible to prevent hidden spots, which could affect energy management decisions.

– **Energy Management and Optimization (EMO)**: *Consistency* and *timeliness* are the data quality dimensions that are identified as requirements. Those dimensions were identified for the proper operation of the EMO functionality. The *Consistency* data quality dimension was chosen in order to ensure that historical power demand and consumption data remain uniform and comparable. This is important to allow effective load balancing and predictive energy distribution. The *Timeliness* data quality dimension is also important. It is important because outdated energy data readings could lead to inefficiencies in power distribution. It may cause power shortages and unnecessary grid overloading.

The collected data quality requirements should be clearly defined and aligned with each IoT software system use case. For instance, the **Overhead Power Line Monitoring (OPLM)** emphasizes *timeliness* to enable real-time detection of potential overheating risks. Also, the **Battery Monitoring (BM)** functionality prioritizes *accuracy* and *completeness*. It uses those two data quality dimensions to ensure reliable battery management and prevent inconsistencies in energy storage reporting. Meanwhile, the **Energy Management and Optimization (EMO)** focuses on *consistency* and *timeliness* to ensure effective long-term energy planning and grid stability.

5.2.2 Data Quality Strategy Management

This subsection focuses on defining strategies to ensure that the collected data meets the operational needs of the Smart Grid, including real-time monitoring, anomaly detection, and energy efficiency optimization. To meet the diverse data quality requirements of the **Smart Grid Monitoring System**, we present the following strategy aligned with the defined dimensions:

– **Timeliness**: **Message-oriented** communication protocols can be used to provide timeliness. Protocols such as *MQTT* or *Advanced Message Queuing Protocol (AMQP)* can be applied for fulfilling the timeliness data quality dimension, we have chosen those two instead of Representational State Transfer (REST) based Hypertext Transfer Protocol (HTTP) models as *HTTP* adds overhead on data transmission [91] in comparison to MQTT. Additionally, apply **traffic prioritization** mechanisms to ensure that time-sensitive data—such as real-time power line temperature readings critical for the **Overhead Power Line Monitoring (OPLM)**—is transmitted ahead of less urgent information, minimizing delays in critical decision-making.

- **Accuracy**: Implement periodic **sensor calibration** and **redundancy mechanisms** to enhance measurement reliability. Scheduled calibration of temperature sensors in power lines ensures that readings remain within acceptable margins (e.g., maintaining an error tolerance of $\pm 2°C$ to avoid false overheating alarms). Utilize **data fusion techniques** to integrate readings from multiple sensors, leveraging *complementary*, *redundant*, or *cooperative* fusion mechanisms [16] to reduce errors and improve decision-making in battery and power line monitoring.
- **Completeness**: Deploy predictive models to estimate missing data when sensor readings are unavailable due to network failures or hardware malfunctions. Mechanisms such as **machine learning algorithms**, **spatial interpolation techniques**, or **temporal interpolation methods** can be used. In this scenario, *Temporal Interpolation Methods* are the preferred choice for the **Battery Monitoring (BM)** functionality, as they provide fast and resource-efficient estimates of missing energy storage data, ensuring smooth energy balancing in real time.
- **Consistency**: Ensure uniform data reporting across different grid components by applying standardized measurement formats and validation rules. For example, if multiple energy storage units report inconsistent charge levels, the system can apply **Kalman filtering** or weighted averaging to align values based on past reliability metrics and correct inconsistencies before they propagate into energy optimization processes.

5.2.3 Data Quality Procedures Management

This subsection describes the standardized procedures for executing data quality tasks, such as temperature validation for power lines and charge level consistency checks for batteries. To define the procedures for data quality management, we establish rules and parameters that must be applied. The following procedures are defined:

- **Real-Time Data Transmission**: Configure sensors to transmit temperature and battery status data at a rate aligned with the requirements of the **Overhead Power Line Monitoring (OPLM)** and **Battery Monitoring (BM)** functionalities. For instance, updates can be sent every 1 minute to support timely load balancing and overheating detection. Data transmission is managed using MQTT due to its low latency and efficiency in resource-constrained environments [91]. To ensure that

high-priority data (e.g., power line overheating events) is not delayed due to network congestion, QoS prioritization is applied.

- **Sensor Calibration**: Establish a periodic schedule for power line temperature sensors and battery monitoring devices to undergo calibration checks (e.g., every two weeks). Define high-priority assets (such as transmission lines operating near critical thresholds) that require more frequent calibration. The calibration workflow includes automated alerts triggered by deviations in sensor accuracy, ensuring that sensors are recalibrated when anomalies are detected rather than following a fixed schedule.
- **Data Completion**: Missing data is estimated using predictive models, with missing values flagged to indicate they were inferred. For the **Battery Monitoring (BM)** functionality, *Temporal Interpolation Methods* are the preferred approach due to their computational efficiency, suitability for real-time energy storage monitoring, and ability to handle short-term data gaps without significant latency or computational overhead. This ensures that missing energy storage data does not disrupt load optimization and energy distribution decisions.

5.2.4 Data Quality Implementation Planning

This subsection outlines the implementation plan for deploying data quality tasks, ensuring that data collection, transmission, and validation align with the Smart Grid's functional requirements. To present a comprehensive implementation plan, it is necessary to specify:

- **Device Roles:** Clearly define responsibilities across the IoT network. **Data Provider Nodes** collect raw data from various sensors, such as temperature sensors installed on overhead power lines and battery monitoring sensors. These nodes apply initial tagging (e.g., marking missing values or identifying anomalies) and ensure data completeness at the source. They also execute data cleaning tasks, such as detecting and handling outliers when needed. **Data Router Nodes** implement traffic prioritization for high-priority data, manage Quality of Service (QoS) levels, and facilitate data transmission using lightweight protocols like MQTT. **Data Consumer Nodes** validate incoming data against the operational requirements of the IoT Software System, ensuring it meets criteria such as timeliness and accuracy, and perform final data processing to support energy management and grid stability decisions.

– **Resource Allocation**: Tasks should be allocated according to the roles of the devices, ensuring they align with available resources such as CPU, memory, and power. For time-sensitive applications like **Overhead Power Line Monitoring (OPLM)**, priority should be given to real-time temperature monitoring tasks. This ensures that overheating conditions are detected promptly to prevent potential failures. **Battery Monitoring (BM)** functionalities should leverage predictive models to estimate energy storage capacity and consumption trends while minimizing computational overhead. Less time-sensitive data, such as historical energy usage patterns, can be processed at the cloud level, reducing computational strain on edge devices.

– **Task Scheduling**: Establishing a clear timeline for critical processes ensures that tasks are executed efficiently within the IoT system. Sensor calibration checks should be scheduled periodically (e.g., every two weeks) to maintain accurate temperature and battery level readings, prioritizing power lines operating at high loads. The real-time data transmission interval should be defined by domain specialists, considering energy efficiency trade-offs for IoT devices. For the **OPLM** functionality, real-time monitoring should occur at high-frequency intervals (e.g., every minute), while **BM** functionalities can utilize longer intervals for battery performance tracking. Grid load analysis and predictive maintenance tasks should be scheduled daily, ensuring a stable and optimized energy distribution strategy.

– **Monitoring Mechanisms**: It is important to implement monitoring mechanisms. Dashboards can be used track of performance metrics in real time. Metrics such as **timeliness**, **accuracy**, and **consistency** can be checked by operators in a dashboard. These dashboards integrate data from both application feedback loops and infrastructure telemetry. Application feedback assesses the relevance and reliability of the data consumed, enabling adjustments based on grid performance and energy demand. Infrastructure telemetry captures critical insights such as device health, network congestion, and transmission delays, offering a detailed view of system operations. Periodic reviews of these metrics allow for optimization, ensuring the system remains aligned with dynamic application requirements and resource constraints.

Table 5.1 provides a structured overview of task scheduling in the Smart Grid scenario. The schedule ensures that essential tasks—such as **Sensor Calibration** and **Data Transmission**—are given high priority, while tasks related to

Task	Frequency	Responsible Actor	Priority
Sensor Calibration	Defined by specialist	IoT device	High
Data Transmission	Defined by specialist	IoT device	High (Real-Time)
Anomaly Detection	Continuous	IoT device	High
Grid Load Optimization	Daily	IoT software system	Medium
Predictive Battery Maintenance	Daily	IoT software system	Medium
Fault-Tolerance Monitoring	Continuous	All Nodes	High
Traffic Prioritization	Continuous	Edge node	High (Real-Time)
Data Quality Monitoring	Periodic (e.g., hourly)	All Nodes	High

Table 5.1: Example of Task Scheduling for Implementation Planning in the Smart Grid Scenario

Grid Load Optimization and **Predictive Battery Maintenance** occur at regular intervals to support energy management decisions. The prioritization of real-time **Traffic Prioritization** and **Fault-Tolerance Monitoring** prevents service degradation and ensures the reliability of power transmission systems.

5.3 Data Quality Control

In this section we present the **Data Quality Control** process for this scenario. This process is responsible for ensuring that the collected, transmitted, and processed data meets the specific requirements of the IoT software system functionalities (OPLM, EMO and BM). At this stage, the tasks identified during the **Data Quality Planning** phase are instantiated, ensuring that the planned strategies and schedules are executed to achieve the desired data quality objectives.

In this scenario, data quality tasks must ensure **data completeness** above 95% and maintain **timeliness** for real-time applications, as defined by domain specialists during the **Data Quality Planning** phase. Maintenance tasks, including sensor recalibration, should be scheduled regularly according to the guidelines established by domain experts, with priority given to high-load transmission lines and critical energy storage sites to ensure accurate and reliable data collection.

In general, **Data Quality Control** encompasses continuous and periodic reviews of critical data quality dimensions, adhering to the thresholds and standards set during the planning phase. For **Timeliness**, the delay between data collection and its availability for grid management applications should be measured, analyzed, and reported. If inconsistencies are detected, correc-

tive actions—such as adjusting transmission schedules or addressing network congestion—should be promptly implemented to meet the planned standards. **Accuracy** is monitored by assessing the error margins in temperature and battery level measurements against the tolerances defined by specialists. Any identified inaccuracies should be factored into decision-making processes to avoid misinformed actions regarding grid load and energy storage operations. Additionally, **Reliability** is evaluated based on the availability and consistency of data over time, ensuring it aligns with the criteria established in the planning phase. This includes tracking sensor uptime, detecting patterns of missing data, and identifying inconsistencies that may impact long-term energy management or real-time grid stability.

To enhance data quality and address nonconformities, the IoT system continuously monitors both the *health of the infrastructure* and the *quality of the generated data*, including:

Device Monitoring: The monitoring tasks, such as periodic calibration checks and maintenance log reviews, are executed by the **Data Provider Nodes** to ensure operational health and accuracy. For instance, if a temperature sensor on a power transmission line starts showing drift or fails to transmit data, the system generates an alert for immediate corrective actions. High-priority grid sectors, such as regions with high energy demand, are monitored first, ensuring that critical areas maintain reliable data.

Data Monitoring: Real-time validation checks, as defined in the planning phase, are performed to ensure that data aligns with predefined thresholds. For example, if the battery monitoring system detects an unusually high temperature reading, the system flags the data for further investigation. Additionally, the system tracks **data completeness** by monitoring transmission intervals, triggering notifications for missing or delayed data.

IoT devices can then execute fallback mechanisms, such as utilizing corrected values for missing data or rerouting collection tasks to nearby devices in case of failure. By implementing the tasks defined in the planning phase, **Data Quality Control** ensures that the IoT system operates in alignment with the specific requirements of its applications. Continuous monitoring and validation processes prevent the propagation of low-quality data, guaranteeing that the information delivered meets the standards necessary for informed and timely decision-making in Smart Grid operations.

5.3.1 Corrective Actions

When data fails to meet quality requirements, the system takes proactive corrective actions to maintain reliability and continuity of operations:

Identifying Issues: The root cause of the problem is traced through diagnostic processes. For instance, outlier readings from a sensor are cross-validated with neighboring sensors or historical data to determine whether they represent a legitimate grid event (e.g., an actual power line overheating) or a sensor malfunction. Real-time validation mechanisms flag anomalies for immediate investigation, ensuring that potential issues are identified promptly.

Responding to Issues: Corrective actions are implemented based on the identified root cause. These actions may include activating backup sensors to ensure data continuity, triggering missing data correction workflows that apply predictive models (e.g., temporal interpolation or machine learning algorithms) to estimate and replace missing values, and scheduling recalibration tasks for malfunctioning sensors. For severe issues, the system may temporarily reroute data collection tasks to other nearby devices to mitigate disruptions.

Example: In a high-priority transmission line, a temperature sensor starts providing inconsistent readings. The **Data Quality Control** process identifies the issue during real-time monitoring. The system cross-checks these readings with data from neighboring sensors and historical patterns, confirming that the inconsistency is due to sensor drift. A recalibration task is promptly scheduled, and predictive models are used to estimate and correct missing data during the recalibration period.

5.4 Data Quality Assurance

In this subsection we present the **Data Quality Assurance** process. This process the ensures the collected data meets the quality requirements of the IoT software system functionalities. This process is composed of two subprocesses that work together to identify, assess and address data quality issues: the **Data Cleaning** and the **Data Quality Assessment**.

The objective of the **Data Cleaning Subsystem** is detecting and removing data quality issues at the data collection. It works by preprocessing raw data collected from IoT devices. For the **Smart Grid** context, the IoT devices are deployed in transmission lines and battery storage systems. These sensors measure parameters such as **line temperature**, **battery voltage**, and **charging/discharging cycles**. Given the operational constraints of a Smart

Grid, issues such as sensor noise, drift, or network congestion can impact data reliability. This subsystem performs *data anomaly verification*, which identifies outliers (e.g., sudden spikes in line temperature) and determines whether they result from legitimate electrical events (e.g., an actual overheating condition) or sensor malfunctions. If an event is deemed legitimate, it is flagged and forwarded for further processing. However, erroneous readings caused by sensor degradation or environmental interference are filtered out to prevent misleading grid management decisions.

Another essential function of the **Data Cleaning Subsystem** is *data completeness verification*, which ensures that data gaps caused by factors such as signal interference or packet loss are detected and handled appropriately. Missing data is processed using one of several strategies: (i) passing it through with a missing-data flag if real-time operation is critical, (ii) estimating missing values using temporal interpolation techniques for applications where approximate values are acceptable, or (iii) replacing lost data with historical values when past patterns provide meaningful insights for decision-making. After cleaning, the subsystem outputs structured data with appropriate metadata annotations, allowing for further assessment by higher layers of the system.

The **Data Quality Assessment Subsystem** evaluates the cleaned data against predefined quality requirements tailored to each grid management functionality. Unlike the cleaning subsystem, which operates generically on raw data, the assessment subsystem is **application-aware**, meaning it incorporates application-specific constraints into its evaluations. It verifies whether data meets defined thresholds for **timeliness** (e.g., power line temperature readings must be delivered within a strict latency window to allow real-time monitoring), **accuracy** (e.g., voltage measurements must remain within an acceptable deviation threshold), and **completeness** (e.g., data gaps in power consumption logs should not exceed 5%). For instance, temperature data from transmission lines is assessed to ensure it adheres to safe operating limits. If a temperature measurement exceeds predefined safety thresholds but remains within expected operational variability, it is flagged for further monitoring. Conversely, if the deviation is extreme, the system triggers an alert for immediate intervention.

An illustrative scenario demonstrates how these processes interact. Suppose a transmission line sensor detects an abnormally high temperature reading while simultaneously experiencing intermittent data loss due to network congestion. The **Data Cleaning Subsystem** first analyzes the anomaly and cross-validates it with readings from neighboring sensors. If the high tempera-

ture is confirmed as a legitimate event, it is tagged as critical and forwarded to the next stage. In parallel, missing data points are interpolated using historical records from the same line segment. The cleaned dataset is then passed to the **Data Quality Assessment Subsystem**, where it is checked against the **timeliness** and **accuracy** requirements of the *OPLM* and *BM* functionalities. If the data meets these requirements, it is forwarded for decision-making; otherwise, it is flagged for additional enhancement processes.

By integrating these two subsystems, the **Data Quality Assurance** process ensures that Smart Grid applications receive **high-quality, reliable, and actionable** data, enabling informed decision-making for power transmission and storage management. This structured approach prevents grid failures due to faulty data, enhances operational efficiency, and ensures compliance with industry standards for energy system reliability.

5.5 Data Quality Improvement

In the context of a **Smart Grid**, the **Data Quality Improvement** process addresses critical issues such as missing data, sensor drift, and inconsistent measurements in power line monitoring and battery management. By leveraging advanced data enhancement techniques, the system corrects nonconformities and improves overall data reliability, ensuring that IoT applications receive accurate and actionable information for energy management.

5.5.1 Root Cause Analysis and Process Transformation

The first step in **Data Quality Improvement** involves analyzing the root causes of data quality issues identified during the **Data Quality Assessment** process. For instance, frequent gaps in transmission line temperature readings may be due to network congestion or interference in wireless communication protocols. Similarly, battery voltage sensors may exhibit gradual accuracy loss due to sensor aging, leading to inaccurate state-of-charge estimations. Additionally, data inconsistencies may arise from environmental variations affecting different power line segments. Addressing these issues requires targeted **process transformations**, such as prioritizing energy-critical data in network transmissions, scheduling automated recalibration for voltage sensors to counteract drift, and implementing redundancy mechanisms to cross-validate temperature readings across multiple sensors. These transformations strengthen the data pipeline, ensuring a robust energy monitoring system.

5.5.2 Data Quality Enhancement Techniques

To improve data quality, the system applies a range of enhancement techniques tailored to specific challenges in Smart Grid management:

- **Data Fusion:** Combines readings from multiple sensors monitoring the same power line segment or battery storage unit. If one sensor reports an anomalous temperature rise while others do not, data fusion helps differentiate between legitimate overheating events and faulty readings, ensuring reliable input for grid management decisions.
- **Predictive Modeling:** Estimates missing data points caused by network delays or sensor failures. For example, if a transmission line temperature reading is lost due to connectivity issues, historical temperature patterns combined with real-time measurements from adjacent sensors are used to interpolate missing values.
- **Recalibration Adjustments:** Ensures that sensors remain within an acceptable accuracy range by periodically correcting measurement deviations. This is particularly relevant for battery monitoring, where voltage sensors must maintain precision to prevent incorrect state-of-charge estimations that could lead to inefficient energy distribution.

By implementing these techniques, the system enhances data integrity and aligns the processed information with the operational requirements of Smart Grid applications.

5.5.3 Preventing Future Nonconformities

Preventing the recurrence of data quality issues is a key component of **Data Quality Improvement**. Several proactive strategies are deployed:

- **Dynamic Recalibration Alerts:** Sensors autonomously notify the system when drift exceeds predefined thresholds, prompting timely recalibration and reducing long-term measurement deviations.
- **Enhanced Network Monitoring:** Real-time tracking of data transmission performance enables the detection of network congestion or packet loss before they significantly impact Smart Grid operations.
- **Adaptive Learning Models:** These models analyze historical grid performance and environmental patterns to predict potential sensor malfunctions or connectivity disruptions, allowing preventive measures to be applied before issues escalate.

These preventative mechanisms create a self-regulating system that minimizes disruptions and maintains high data quality over time.

5.5.4 Outcome Example

After implementing the **Data Quality Improvement** process, the Smart Grid system achieves significant enhancements. As the root causes of data quality issues are addressed, missing data is reduced by 90% through the use of predictive modeling and optimized transmission protocols, ensuring data completeness. Automated sensor recalibration and data fusion techniques increase accuracy, aligning voltage and temperature readings with actual operating conditions. Cross-validation mechanisms improve data consistency, reducing the risk of false alarms or misinformed energy distribution decisions. These enhancements ensure that Smart Grid applications, such as **Overhead Power Line Monitoring (OPLM)** and **Battery Monitoring (BM)**, operate with higher reliability and efficiency, ultimately optimizing energy transmission and storage.

5.6 Conclusions

This proof of concept successfully demonstrates an instantiation of the proposed software architecture applied in an IoT Smart Grid environment. The results validate that the system can dynamically monitor, assess, and enhance data quality. However, further research is needed to explore scalability, integration with additional IoT applications, and real-world operational constraints before full-scale deployment.

Also, this proof of concept highlights the importance of adaptive mechanisms. The ability to dynamically deploy corrective actions, such as outlier removal and routing prioritization, ensures that the system can respond to changes in data quality conditions in real time. These findings reinforce the effectiveness of the proposed approach in improving data quality across distributed IoT smart spaces.

5.6.1 Limitations

Despite the results, we identify the following limitations that this proof of concept has and should be addressed in future research:

- **Scalability**: The current implementation focuses on a specific Smart Grid scenario with a limited number of IoT devices. Future studies should investigate the performance and feasibility of the approach in larger-scale deployments with thousands of devices operating in complex environments.
- **Edge Limitations**: While the Edge was included in the system architecture, its role was primarily limited to routing and data transmission. Further exploration is needed to implement and evaluate advanced Edge-based data quality techniques, such as traffic prioritization, distributed processing, and edge intelligence to improve real-time data handling.
- **Task Scheduling and Optimization**: The proof of concept assumes a static scheduling approach for executing data quality tasks. However, in real-world scenarios, data quality tasks may need to be dynamically scheduled based on available resources, system constraints, and evolving data quality requirements. Future research on investigating task optimization strategies for balancing performance, energy consumption, and latency remains an open research challenge should be addressed.
- **Real-World Deployment and Practitioners Validation**: This study does not include evaluations involving practitioners or domain experts responsible for IoT deployments. Involving real-world IoT system operators in future research could provide insights into usability, system adaptability, and the effectiveness of automated data quality management mechanisms in practical applications.
- **Use of more Data Quality Dimensions**: The scenario of this work focuses on selected data quality dimensions such as timeliness, accuracy, and completeness. Future research should explore additional dimensions to enhance the robustness of data quality management in heterogeneous IoT environments.

Chapter 6

Evaluations

This chapter presents the **Evaluations and Results**, of an instance of the proposed software architecture for data quality management through a Proof of Concept. We assess its software components in a smart grid scenario by executing the data quality tasks.

Thus, in this evaluation we executed a scenario to verify the following data quality tasks, data quality preprocessing and compliance. These evaluations focus on assessing the effectiveness of data quality management operations across the entire IoT smart space. To assess data quality preprocessing we search for outliers on data collection and how it affects the IoT software system behavior. To assess compliance we evaluate if our task is able to detect data outside the range according to the application data quality requirements.

The evaluations were performed in a practical setting using real sensor nodes to demonstrate the applicability of the proposed solution in real-world scenarios. This evaluation followed the same procedure presented in [95].

6.1 Implementation

To validate the proposed solution, we implemented the three operational layers defined in this thesis: **IoT Devices Layer**, **Edge Layer**, and **IoT Software System Layer**.

In the *IoT Device Layer*, the implemented components included a **IoT Sensor**, represented by a temperature sensor, and a **Data Producer Manager**, which utilized the data quality preprocessing task proposed in the solution chapter.

For the *Edge Layer*, we implemented the **Data Routing Manager**, responsible for routing data between the IoT Device Layer and the IoT Software System Layer, using the RPL Routing Protocol [96].

In the *IoT Software System Layer*, we developed an IoT software system with two functionalities, each represented by a separate **IoT Application** component: OPLM (Overhead Power Line Monitoring) and BM (Battery Monitoring). Additionally, the **Data Consumer Manager** component was implemented using the data quality compliance task described in the proposal chapter.

This implementation aimed to evaluate the operation of our solution in the IoT smart space by examining the following properties:

- **IoT Devices**: Evaluating the accuracy of the data quality preprocessing process performed at the IoT device layer.
- **IoT Software System**: Assessing the system's capability to validate and enhance data quality, improving the accuracy of its functionalities.
- **Delay**: Measuring the latency imposed on data delivery to IoT Software System due to data processing activities across the layers.
- **Resource Consumption**: Analyzing the memory usage (Random Access Memory (RAM) and Read-Only Memory (ROM)) by both IoT devices and the IoT Software System.
- **Energy Consumption**: Quantifying the energy usage incurred by the communication processes of the IoT smart space actors.

To implement this scenario, we utilized the following hardware technologies:

- The IoT devices are based on the Arduino Uno platform (Fig. 6.1), which is a resource-constrained microcontroller board. The Arduino Uno is built on the ATmega328P microcontroller [97], a low-power CMOS 8-bit microcontroller utilizing the AVR® enhanced RISC architecture. The Arduino Uno is characterized by limited resources, including 32 KB of Flash Memory (with 0.5 KB reserved for the bootloader), 2 KB of volatile Static Random-Access Memory (SRAM), and 1 KB of non-volatile Electrically Erasable Programmable Read-Only Memory (EEPROM). It operates at a clock speed of 16 MHz, making it suitable for lightweight IoT applications.
- The IoT software system was deployed on a Raspberry Pi 3 Model B+ platform (Fig. 6.2), which is a resource-richer platform compared to the Arduino. The Raspberry Pi features a 1.2 GHz quad-core ARM Cortex-A53 CPU with an ARMv8 Instruction Set and is equipped with 1 GB LPDDR2-900 SDRAM, making it well-suited for processing-intensive tasks and hosting the IoT software system.

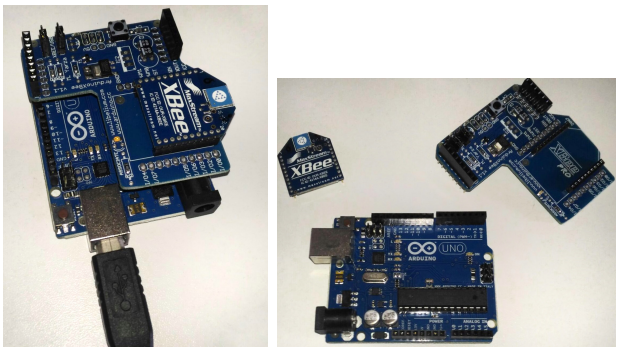

Figure 6.1: Arduino and XBee platforms used as CN

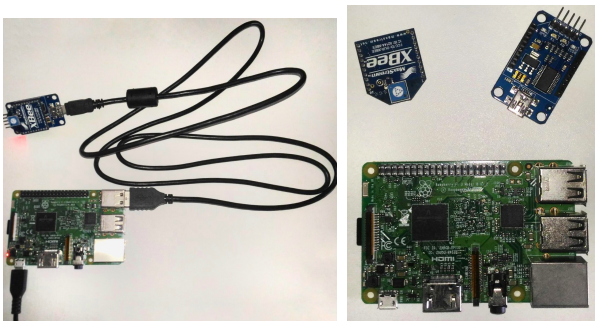

Figure 6.2: Raspberry Pi and XBee platforms used as FN

The network uses the IEEE 802.15.4 as the technical standard for the operation of low-rate wireless personal area networks (LR-WPANs). The Arduino UNO board uses an XBee shield with an XBee 802.15.4 RF module (Fig. 6.1, right) while the Raspberry Pi node uses an XBee Explorer Dongle for USB interfaces (Fig. 6.2, right). The data preprocessing message format includes three key fields: the *NodeID*, which identifies the source of the data; the *IsOutlier* flag, which indicates whether the data point has been classified as an outlier; and the *Data*, representing the collected value. Similarly, the data quality compliance message format consists of two fields: the *Data*, which contains the processed measurement, and *isComply*, a binary flag (1 or 0) that denotes whether the data meets the defined quality requirements. All tasks were developed using C programming language. All real node evaluations lasted 1 hour and each of them was repeated 30 times with a 99% confidence interval. The distance between each node was 1 meter.

6.2 Smart Grid Scenario

For the evaluation, we selected the **Smart Grid** scenario [11]. A Smart Grid can be defined as "an integrated array of grid technologies, devices, and controlling systems that provide and utilize digital information, communications,

and controls to optimize the efficiency, reliability, and security of electric power delivery" [11].

In this scenario, an IoT software system is used to monitor a transmission tower. A transmission tower supports an overhead power line, the most common means of transmitting electrical energy. In the Smart Grid context, the transmission tower also stores energy using local batteries. The IoT software system has two functionalities: **Overhead Power Line Monitoring (OLPM)** and **Battery Monitoring (BM)**. Both functionalities share the same communication and sensing infrastructure and are utilized for monitoring energy transmission towers. This scenario was chosen because it has been explored in prior works [98], [1] and demonstrates the critical role of data collected from the infrastructure in making operational decisions for the transmission lines.

The system includes five IoT devices installed on the Overhead Power Line and five more IoT devices installed on the battery - Those devices are called *Collector Nodes (CN)*. These CN collect temperature samples periodically and transmit the data to the IoT Software System - Which we call *Data Processing Node (DPN)*.

An important aspect of this scenario is that both OPLM and BM share the same type of data (temperature) but with different data ranges and operational implications. Additionally, the two functionalities are correlated, as the behavior of one functionality can influence the other. For example, a temperature of 75°C may indicate potential damage to the Overhead Power Line, jeopardizing energy transmission. Consequently, no power would be available to store in the battery. However, the same temperature of 75°C is considered normal for the BM functionality, as the typical battery temperature range is around 40°C to 144°C [1]. In contrast, the safe operational temperature range for the Overhead Power Line is 40°C to 65°C [1], meaning 75°C indicates a critical situation for OLPM.

In order to evaluate the impact of dealing with multiple objectives simultaneously, we set four time slots called T1, T2, T3 and T4. Each time slot represents a particular state of the environment along the time, a particular event to be detected. T1 represents ideal conditions for both OPLM and BM (a safe condition, where there is no need to perform preventive actions to avoid damage on the power line and on the battery). T2 represents an increase in the overhead power line temperature. T3 represents an increase in the battery temperature. T4 represents an increase in the temperature of both the battery and the overhead power line. The temperature data ranges for the time slots

are summarized in Table 6.1. Given the data ranges in Table 6.1 we calibrated the data quality requirement for OPLM as ($\epsilon = 5.0, p = 0.99$) and for BM as ($\epsilon = 10.0, p = 0.99$). We have set that the initial air temperature, battery temperature and the overhead power line temperature were respectively 25°C, 44°C and 44°C. Battery and overhead power line temperatures changed according to the thermal models presented in [99].

Table 6.1: Temperature data range for each time slot [1]

Time slots	Power line data range	Battery data range
T1	40°C to 65°C	40°C to 144°C
T2	65°C to 75°C	40°C to 144°C
T3	40°C to 65°C	Over 144°C
T4	65°C to 75°C	Over 144°C

6.3 Metrics

To evaluate the operation of the data quality management process operation on the IoT smart space, we present the following metrics [87] [90]:

- **Accuracy**: The accuracy is measured using as metrics: *False Positives (FP)*, *False Negatives (FN)*, *True Positives (TP)*, and *True Negatives (TN)*.

  Each functionality monitors a specific **item** within the Smart Grid: OPLM monitors the overhead power line, while BM monitors the battery. These metrics compare the actual state of the monitored **item** with the state inferred by the respective functionality.

  * **TP**: Counts the number of situations where the functionality correctly identifies its monitored **item** as being in ideal condition, and it is indeed in ideal condition.
  * **TN**: Counts the number of situations where the functionality correctly identifies its monitored **item** as not being in ideal condition, and it is actually not in ideal condition.
  * **FP**: Counts the number of situations where the functionality identifies its monitored **item** as being in ideal condition, but it is not.

    - ∗ **FN**: Counts the number of situations where the functionality identifies its monitored **item** as not being in ideal condition, but it is actually in ideal condition.

  True occurrences are defined as the sum of *TP* and *TN*, representing cases where it matches the actual state of the monitored item.

- **Delay**: To measure the delay imposed by our solution, we used the *elapsed time to execute the data task* as metric. This metric is measured by calculating the difference in microseconds between the time in which the data task receives data to the time in which it outputs the result. This metric is designed to evaluate if the data task imposes much delay to perform its operations and evaluating the impact of the data processing in the delay imposed to execute each data quality task.

- **Resource Consumption**: To evaluate resource consumption, we use the metric of *memory consumption of the data quality task when installed in the device*, which includes measuring both RAM and ROM usage. RAM usage represents the memory required for executing tasks, while ROM usage reflects the memory needed for installation and data storage, if necessary.

  Smart sensor nodes are resource-constrained devices with limited memory resources. Therefore, solutions that impose significant overhead on these resources are infeasible. Similarly, an IoT software system may support multiple functionalities, and if data quality management imposes excessive memory consumption, it may become impractical to implement.

  This metric evaluates memory usage (RAM and ROM) for both IoT devices and the IoT Software System.

- **Energy Consumption**:

  To quantify the energy usage of data quality management tasks, we use the *amount of transmitted bits* in the network as a metric. In IoT devices, energy consumption is primarily driven by the radio's operation for transmitting and receiving data.

  The radio's usage is an energy-intensive operation, and when a node depletes its battery, it compromises the communication and sensing coverage of the application [100]. Therefore, evaluating the energy efficiency of data quality tasks is crucial to ensuring the longevity and reliability of IoT deployments.

6.4 Results

In this section we present the results.

6.4.1 IoT Devices: Data Quality Preprocessing Technique

In this section we discuss the evaluations we conducted to for the data quality preprocessing at the IoT device layer. To evaluate the accuracy of the data quality preprocessing at the IoT device layer, we used two different scenarios: (i) Using Data Quality Preprocessing Task (Exp-1) and (i) Do Not Using Data Quality Preprocessing Task (Exp-2). In Exp-1 the data quality preprocessing *is considered* by the IoT software system to perform its decision making for the OPLM functionality. In Exp-2 the data quality preprocessing *is not considered* by the IoT software system to perform its decision making. The results for evaluating the accuracy are shown in Table 6.2.

The first line of Table 6.2 shows the accuracy result for Exp-2 scenario and the second line of Table 6.2 shows the accuracy result for Exp-1 scenario. The accuracy result for Exp-2 scenario is of 89.21% with standard deviation of 0.93% while for Exp-1 scenario the accuracy is of 97.62% with standard deviation of 1.41%. This result shows an improvement of 8.41% in the accuracy result of the Exp-1 scenario in comparison to the results of the Exp-2 scenario. This improvement is given by the fact that in Exp-2 scenario all data is provided to the IoT software system without considering the result of the data quality preprocessing task.

In both Exp-2 and Exp-1 scenarios the presence of outlier data degrades data quality. However, since in Exp-2 the IoT software system does not consider the data quality preprocessing task, it has no information supplies to whether consider or not a degraded information in its analysis. While in Exp-1 scenario the data is provided to the OPLM functionality of the IoT software system along with information about the data quality preprocessing, thus the resulting shows an improvement in the accuracy results.

The aforementioned results shows that the proposed data quality preprocessing is capable of improving the accuracy of the IoT software system. The accuracy result shows the importance of considering executing the data quality preprocessing. In this scenario, the presence of dysfunctional IoT devices degrades data quality, which can be manifested in form of outliers. Also, IoT data is a valuable asset for the IoT Software System decision making. The knowledge about the DQ aspects in Exp-1 was capable of improving the decision making in comparison to Exp-2, where the DQ aspects of the IoT data was not considered.

The results for measuring *energy consumption* according to the amount of transmitted bits are shown in tables 6.3. We can observe from Table 6.3 that when using

Table 6.2: Accuracy results

	Accuracy	Standard Deviation	Confidence Interval (99%)
Exp-2	89.21 %	σ0.93	CI:0.47
Exp-1	97.62 %	σ1.41	CI:0.71

Table 6.3: Amount of Transmitted bits

	Transmitted bits	Standard Deviation	Conf. Interval (99%)
Exp-2	23705.6 bits	σ272.19	CI:136.98
Exp-1	25187.2 bits	σ289.21	CI:145.54

the data quality preprocessing task, there is an increase of 5.88% of transmitted bits. This is due to meta information about data quality. In our scenarios the information provided is sent in a one bit flag in data frame, either it is an outlier or it is not. Therefore, there is a increase of a bit for each data frame transferred in network. In this work we do not focus on implementing an optimal solution for meta information compression. By using data compression techniques, this result might be improved.

Also, regarding *resource consumption*, the data quality preprocessing task consumes 3454 bits of program storage space. The maximum available is 32256 bits of the Arduino memory. The data quality preprocessing task presents *memory consumption* of just 10% of the memory of Arduino. This result shows that our solution does not impose memory overhead in Arduino platform. The delay imposed by our data quality solution showed that it imposes a delay of 307.07 microseconds (0.00030707 seconds) on average with standard deviation of σ11.96239318 and a Conf. Interval (99%) of 6.020013505 for executing the data quality preprocessing task.

The results for *delay* evaluation shows that the data quality preprocessing does not impose a significant overhead when it is used in comparison to when it is not used.

The evaluations for the data quality preprocessing task indicates that it is an effective task, which is capable of improving the accuracy of the IoT software system in comparison to the scenario where it is not used. Also, our evaluations indicates that our data quality preprocessing solution does not impose much overhead in the network resources (energy and memory) nor imposes a significant delay when performing its procedures.

6.4.2 IoT Software System: Data Quality Compliance Technique

This section describes the evaluations conducted with the data quality compliance task.

Table 6.4: Scenario 1 - Accuracy result

Time slots	T1	T2	T3	T4
OPLM	72.5% ±0.08	58.8% ±0.2	58.4% ±0.1	97.9% ±0.03
BM	65.9% ±0.13	71% ±0.1	99.99% ±0.01	99.99% ±0.01

Table 6.5: Scenario 2 - Accuracy result

Time slots	T1	T2	T3	T4
OPLM	52.9% ±0.13	82.2% ±0.1	50.0% ±0.2	50.1% ±0.2
BM	100%	100%	0%	0%

Evaluating accuracy

To evaluate the accuracy of the proposed data quality compliance task for an IoT software system in an IoT smart space, we executed two distinct scenarios:

- *Scenario 1*: The proposed data quality compliance task was applied to the IoT software system. Additionally, a well-known Moving Average Filter (Moving Average Filter (MAF)) [16] was applied to the received data before it was provided as input to the IoT software system.
- *Scenario 2*: No data quality compliance task was applied. In this scenario, the MAF was directly applied to the data without any prior data quality treatment before providing it as input to the IoT software system.

In both scenarios, IoT data was utilized by the IoT software system's OPLM and BM functionalities. For executing the evaluations, four distinct time slots ($T1$, $T2$, $T3$, and $T4$) were considered. For each time slot, both scenarios were tested to compare the results in terms of true occurrences ($TP + TN$). Tables 6.4 and 6.5 present the achieved accuracy results for *Scenario 1* and *Scenario 2*, respectively.

The first row of Table 6.4 presents the accuracy results for the OPLM functionality in *Scenario 1*, while the second row shows the accuracy results for the BM functionality. Similarly, the first row of Table 6.5 displays the accuracy results for the OPLM functionality in *Scenario 2*. The second row shows the accuracy results for the BM functionality when no data quality compliance task is applied.

Next, we present the results for each time slot: T1, T2, T3, and T4.

- *T1*: For the T1 time slot, both the overhead power line and the battery are in ideal conditions. As observed in Table 6.5, under *Scenario 2*, the BM functionality achieved an accuracy of 100%, while the OPLM functionality presented an accuracy of $52.9\% \pm 0.13$. This outcome is explained by the composition of the T1 dataset, as illustrated in the histogram of Figure 6.3. In this scenario, the MAF smooths the data, resulting in an average temperature that falls outside the usual and safe Overhead Power Line temperature range. However, the same average represents a normal temperature for the BM functionality.

  In *Scenario 1*, during T1, the OPLM functionality achieved an accuracy of $72.5\% \pm 0.08$, while the BM functionality achieved an accuracy of $65.9\% \pm 0.13$. In this scenario, the data quality requirements for both functionalities were assessed, and the data quality was enhanced accordingly. By applying the proposed data quality compliance task, the data was appropriately divided, ensuring that the averages accurately reflected the actual conditions of the overhead power line and the battery. These results are presented in Table 6.4.

  This result demonstrates that the use of the data quality compliance task can significantly improve the performance of the OPLM functionality. However, a drawback observed is that the BM functionality experienced a performance penalty. This penalty arises because the proposed task may not be as effective for this specific functionality, suggesting that the development of additional or alternative tasks tailored to the BM functionality might be necessary.

- *T2*: On *Scenario 2*, the BM functionality presented an accuracy of 100%, while the OPLM functionality achieved an accuracy of $82.2\% \pm 0.1$. This result is influenced by the composition of the T2 dataset, as shown in the histogram in Figure 6.4. In this time slot, the overhead power line is not in safe condition, whereas the battery remains in a normal condition. However, the data from the overhead power line and the battery are overlapped. This overlap in data ranges directly impacts the accuracy obtained with MAF, as shown in Table 6.5.

  For *Scenario 1*, the OPLM functionality presented an accuracy of $58.8\% \pm 0.2$, while the BM functionality achieved an accuracy of $71\% \pm 0.1$ (Table 6.4). The decrease in accuracy is attributed to the overlap in data ranges, which reduces confidence in the data and limits the ability to improve data quality effectively. In this case, the static nature of the scenario prevented the deployment of an alternative task that could address this limitation, resulting in suboptimal decision-making by the applications.

  This result highlights a critical consequence of unmet data quality requirements. When the data quality requirements for both applications exceed what

the data can realistically provide, it may be more effective to reconsider the requirements or adapt the approach. Otherwise, the inability to meet these stringent requirements may leave applications, such as OPLM, without sufficient information to make accurate decisions.

- *T3*: In T3 time slot the battery is overloaded while the overhead power line is in normal condition. The data composition for both sources generated a completely disjoint multi-modal dataset. In this case, the average performed by MAF does not represent either the data peaks shown in Figure 6.5 histogram. Hence, the biased average result reflects in the accuracy of $50.0\% \pm 0.2$ for the OPLM application and 0% for the BM application on Scenario 2. However, when applying our data quality compliance solution the OPLM application presents an accuracy of $58.4\% \pm 0.1$ and the BM application presents an accuracy of $99.99\% \pm 0.01$.

- *T4*: For T4 time slot, both the overhead power line and the battery present malfunctioning behavior. In this case, there is a multi-modal disjoint dataset, as shown in Figure 6.6, similar to T3 dataset (Figure 6.5). In this scenario, the result the average performed by MAF is biased and does not represent either application behavior. In such case the OPLM application presented an accuracy of $50.1\% \pm 0.2$, the BM application presented an accuracy of 0%. When applying our data quality compliance solution, the accuracy of the OPLM application is $97.9\% \pm 0.03$, the accuracy for BM application is $99.99\% \pm 0.01$.

Evaluating resource consumption and delay

This section describes the performed evaluations for scenarios 1 and 2 in terms of resource consumption and delay imposed.

In our evaluations, The delay imposed by scenario 2 was of 36.14 ± 7.61 microseconds while the delay imposed by scenario 1 was of 1201.14 ± 45.87 microseconds. This delay increased imposed by the execution of the data quality compliance task.

Regarding resource consumption: in scenario 1, it uses $99.77MB \pm 0.18$ of RAM memory while in scenario 2 it uses $99.29MB \pm 0.28$ of RAM memory. It is important to notice that these values represent the total of ram memory consumed by the raspberry pi with operating system. Nevertheless, we can conclude that the proposed data quality compliance task is feasible to be used.

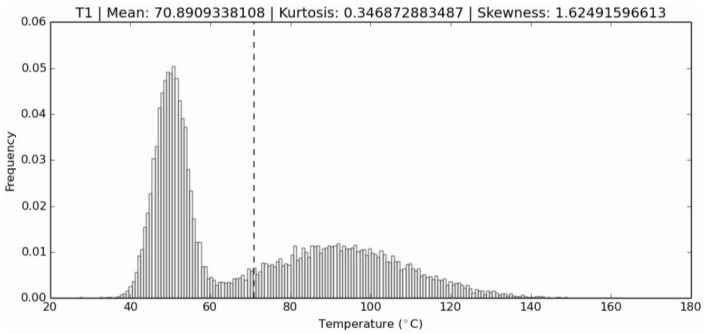

Figure 6.3: T1 dataset [1]

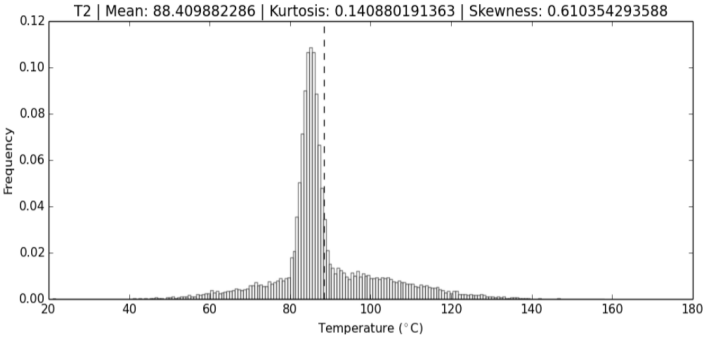

Figure 6.4: T2 dataset [1]

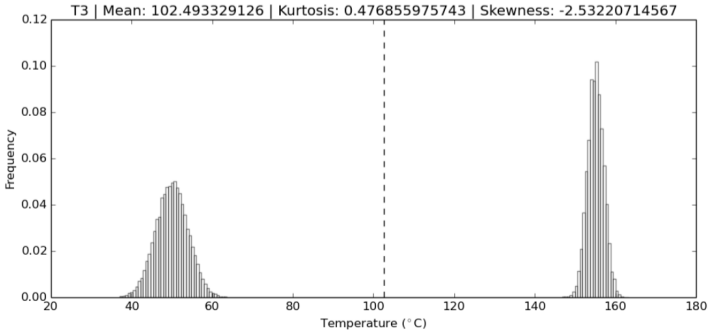

Figure 6.5: T3 dataset [1]

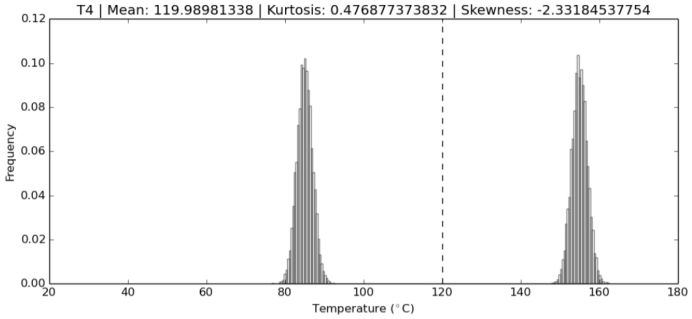

Figure 6.6: T4 dataset [1]

6.5 Conclusions

In this chapter, we conducted evaluations and obtained results that demonstrate the feasibility of data quality management for an IoT software system within an IoT smart space. The evaluations were carried out using the components proposed in this work, which instantiated the tasks designed for data quality preprocessing and compliance. These results highlight that implementing the data quality software architecture is both feasible and beneficial.

The results also emphasize the value of the proposed data quality preprocessing task, which showed significant potential in improving the overall performance of the IoT software system. By performing preprocessing directly on IoT devices, we observed improvements in data quality that positively impacted the system's functionalities. This highlights the importance of further developing tasks that align with the principle of performing preprocessing tasks at the IoT device layer before transmitting data to the IoT software system.

The results demonstrated that the proposed solution, when executed by the proposed components and tasks, successfully managed data quality to the IoT software system. The evaluations confirmed that data quality management prevents the impact data quality issues on the IoT Software System of the scenario.

The proposed data preprocessing and data quality compliance tasks were assessed in terms of their impact on data quality and system performance. The preprocessing tasks improved data quality at the source by detecting and flagging potential anomalies before transmission. Which reduces the propagation of erroneous data. Meanwhile, the compliance task ensured that data met the specific requirements the IoT software system functionality. This improves decision-making processes. However, results indicated that applying these tasks can introduce additional resource consumption and processing delays. Hence, it requires trade-offs depending on the constraints of the IoT environment.

Additionally, the proposed data quality compliance task, which operates based on the confidence dimension, achieved notable improvements in certain scenarios. However, there were cases where the compliance solution was not effective, particularly when the composition of the data did not meet the requirements of the tasks. This underscores the need for the development of additional compliance tasks to address diverse data compositions and operational contexts.

Overall, these findings demonstrate the potential of a layered approach to data quality management, where preprocessing and compliance tasks are distributed across IoT devices and the IoT software system. By further advancing preprocessing tasks and involving domain experts during system deployment, it is possible to enhance the adaptability and effectiveness of data quality management in dynamic

and complex IoT smart spaces.

The evaluations also highlighted key aspects for future refinement, particularly in the dynamic adaptation of data quality processes to evolving conditions in IoT smart spaces. While the system effectively responded to identified quality issues, further research is needed to optimize the scheduling and execution of data quality tasks, considering real-time changes in data distribution and infrastructure performance. These findings reinforce the importance of an adaptive, component-based approach to managing data quality in IoT environments.

6.5.1 Threats to Validity

In this section, we discuss potential threats to the validity of the results and evaluations presented in this chapter.

- **Hardware and environment**: We used Arduino Uno and Raspberry Pi 3 to the execute the evaluations. These platforms are resource-constrained and may not fully represent all types of IoT devices. Additionally, sensor inaccuracies, including calibration errors or environmental interference, could affect the evaluation results, particularly for data quality preprocessing. Finally, static configurations of tasks and thresholds may not fully account for dynamic changes in the operational environment.

- **Generalization**: The evaluations were conducted in a controlled environment using a specific smart grid scenario. While this scenario was chosen due to its complexity and relevance, the results may not directly apply to other IoT smart spaces, such as healthcare or industrial monitoring systems. Also, the use of only temperature sensors may limit the use of the results in different types of sensors or data.

- **Used metrics**: In this work we evaluated accuracy using true positives, true negatives, false positives, and false negatives. This approach provides insights into the accuracy performance but may not capture other aspects of data quality. Aspects such as completeness or consistency should be assessed by other metrics. Additionally, the use of energy consumption metrics based on transmitted bits might oversimplify the complexities of real-world energy usage patterns in IoT devices.

Mitigation Strategies: To address these threats, several mitigation strategies were employed. First, the evaluations included real-world hardware to ensure practical applicability. Second, the metrics used for evaluation were aligned with the objectives of the proposed solution, focusing on critical aspects such as accuracy,

resource consumption, and delay. Finally, the results were analyzed within the context of the chosen scenario, with an acknowledgment of their limitations for broader generalization.

Despite these limitations, the evaluations provide valuable insights into the effectiveness of the proposed data quality management solution, highlighting areas for future work to enhance its robustness and applicability across diverse IoT environments.

Chapter 7

Pratical Guidelines for the Implementation of the Proposed Data Quality Management Processes and Components in IoT Smart Spaces

This chapter presents a practical guide for implementing the proposed software architecture for data quality management and the components that operationalize it in IoT smart spaces. The objective is to provide a step-by-step approach for integrating the proposed components and data quality techniques into a real-world IoT software system. The implementation process follow two phases: **initial phase** and **operational phase** (both described in sections 7.1 and 7.2, respectively). Also, in section 7.3 we present three illustrative scenarios to present challenges related to data quality management and how the system dynamically adapts through its processes and components.

The **initial phase** is described in section 7.1, it has the following steps:

1. Identifying IoT Software System functionalities (subsection 7.1.1).
2. Selecting data quality dimensions (subsection 7.1.2).
3. Deployment of components (subsection 7.1.3).
4. Configuration of data quality tasks (subsection 7.1.4).

The **operational phase** is described in section 7.2, it has the following steps:

1. Components interoperation (subsection 7.2.1).
2. Process adaptation (subsection 7.2.2).

7.1 Initial phase

7.1.1 Identifying IoT System Functionalities

To implement the proposed software architecture, first a comprehensive analysis of the **IoT software system's functionalities** must be conducted. An IoT software system deployed in a smart space can serve multiple purposes, leveraging raw sensor data to support a variety of features. For instance, the system might collect temperature data not only to regulate air conditioning but also to detect potential fire hazards. Similarly, the same temperature data can simultaneously serve different objectives, such as generating historical averages for energy efficiency analysis and identifying sudden spikes indicative of emergencies. The **outcome** of this analysis is a detailed list of the system's functionalities, along with the specific types of data that need to be collected to effectively support each functionality.

7.1.2 Selecting Data Quality Dimensions

To effectively manage data quality in an IoT smart space, it is crucial to select and define the most relevant data quality dimensions that align with the IoT software system's functionalities and objectives. The starting point for this step is the comprehensive list of system functionalities and the types of data that need to be collected, as identified in the previous phase. With this information, it becomes possible to analyze which functionalities are critical and to determine the specific characteristics that the collected data must have to meet the system's operational goals. For example, certain functionalities may require data to be collected at precise intervals (e.g., every minute) or must include information from multiple regions to provide comprehensive insights (completeness).

This analysis leads to the identification of the system's non-functional requirements, which are directly related to data quality. Understanding these diverse functionalities is essential because **each use case imposes specific data quality requirements**. For instance, fire detection demands real-time and highly accurate data to promptly identify emergencies, while historical temperature analysis may prioritize data completeness and consistency over timeliness. These differing requirements highlight the need to carefully select appropriate data quality dimensions—such as confidence, accuracy, and timeliness—that align with each specific system functionality.

Once the relevant data quality dimensions are identified, it is essential to define quantitative metrics for each dimension. For example, acceptable data latency thresholds could be established for real-time monitoring functionalities, while acceptable error rates or data completeness levels might be defined for historical data

analysis.

Outcome: The result of this step is a detailed list that connects each IoT software system functionality with the types of data it requires, the relevant data quality dimensions, and the corresponding quantitative metrics.

7.1.3 Deployment of Components

After identifying the functionalities of the IoT software system, the types of data to be collected, the data quality requirements: its dimension and metrics; the next step is to select data quality tasks by identifying existing methods that can be applied within the context of data quality management in the IoT smart space.

The chosen tasks must align with the data quality plan and be distributed across the layers: the IoT Device Layer, Edge Layer, IoT Software System Layer, and the Management Layer.

On the **IoT Device Layer** it is essential to deploy lightweight tasks to preprocess data to the data quality management, this helps to early detect low-quality data and its sources. For this task the IoT devices must be configured with tasks to ensuring that anomalies are identified early without introducing significant resource overhead, as well as with data completion techniques to find sources of data loss. In this work we provide a technique to be instantiated on IoT devices for preprocessing data directly at the source.

The **Edge Layer** is the intermediary for data transmission and quality assurance before reaching the IoT software system. If it is a data quality requirement its devices should be configured to manage data transmission and execute QoS tasks to prioritize data flows based on quality indicators, to guarantee timeliness on data delivery.

On the **IoT Software System Layer** the deployed task should ensure that the data consumed by the IoT software system meets established quality requirements: tasks to carry out data quality compliance checks, validating data against predefined metrics and dimensions should be deployed on the system to enhance and process data as necessary to meet functional goals, aligned with the operational needs of the IoT software system.

The **Management Layer** orchestrates the data quality management process. Tasks are dynamically allocated to continuously adapt and optimize data quality management across the IoT smart space. At this stage, databases are populated: types of devices present in the IoT smart space; all available data quality assessment tasks that can be used; and all available data quality quality enhancement tasks that also can be used; It is fundamental to identify any supplementary tasks that may serve as alternatives to the ones that are initially deployed.

The objective of maintaining this repository of devices and task options, is to make the system equipped with the resources to modify data quality management strategies in response to emerging challenges. This is useful for the replacement or adjustment of tasks when necessary keep the data quality management inline with the requirements of the IoT software system.

Outcome: The result of this step is the structured deployment of selected data quality techniques across the IoT smart space.

7.1.4 Configuration of Data Quality Tasks

After the deployment of the components in the IoT smart space, it should take place the **configuration of data quality tasks**. This phase involves defining operational parameters for each task and aligning them with the data quality requirements of the IoT software system. The first step is **Task Parameterization**, where each data quality task must be tuned on each IoT device. This involves configuring several parameters:

- **Window Size**: Defines the amount of data to be analyzed within a given time frame. For example, setting a sliding window for outlier detection to process recent data.
- **Thresholds**: Establishes acceptable limits for data quality dimensions that trigger corrective actions when exceeded.
- **Baseline Definitions**: Defines standard data quality levels for each functionality, serving as reference points for identifying deviations and initiating corrective actions.
- **Initial Calibration**: Sets up baseline configurations for sensors and other devices, ensuring that they operate within the expected performance ranges from the start.

Additionally, it is necessary to define and configure a **Quality Protocol** that integrates data quality tasks across all layers of the IoT smart space. It should define the following aspects:

- **Feedback Types**: Specifies how feedback is generated and communicated across layers (e.g., immediate alerts, periodic reports).
- **Message Types**: Determines the structure and priority of messages exchanged between components, supporting Quality of Service (QoS) for critical data flows.

- **Communication Standards**: Ensures consistent formatting and delivery of quality-related messages.

Outcome: The outcome of this stage is the deployment and configuration of data quality tasks across IoT devices, the Edge layer, and the IoT software system. Each task is deployed and tuned according to the defined operational parameters for the alignment with the data quality management strategy.

7.2 Operational phase

After the deployment and configuration of data quality components and tasks, the system enters the **operational phase**. In this phase, the data quality management system functions autonomously, continuously executing the configured tasks across the IoT smart space. The primary responsibility of the system operators is to supervise and verify that the components are working as expected, intervening only when necessary. Human intervention may be required for actions such as sensor recalibration, hardware replacement, or reconfiguration of specific parameters in response to detected anomalies or infrastructure changes.

To ensure smooth operation and sustained data quality, this phase is divided into two critical steps: (i) **Components Interoperation** and (ii) **Monitoring and Adaptation**

7.2.1 Components Interoperation

This step involves verifying integration and interaction of all deployed components across the *IoT devices*, *Edge nodes*, and the *IoT software system* layers. It should ensure that data flows correctly from collection to processing and that feedback mechanisms are functioning properly to support continuous data quality management.

During this phase, operators may identify the need for adjustments due to unforeseen interactions between components. For instance, the data collection rate might need to be modified if it is impacting system performance, or the execution of a data quality task for one functionality of the IoT software system could unintentionally degrade the data quality required by another functionality. Such issues may not have been detected during the initial deployment and configuration phases.

Operators must monitor these interactions to detect and resolve conflicts or inefficiencies. Adjustments could include recalibrating data collection intervals, reallocating tasks, or reconfiguring data processing workflows to ensure that all data quality management process operates properly.

7.2.2 Process Adaptation

Continuous monitoring of the data quality processes are essential to detect potential problems in the data quality management process. This involves real-time tracking of task execution, using dashboards to view quality management and infrastructure health. Operators must analyze system feedback and logs to identify signs of degradation or failure. When necessary, adaptive actions should be taken, such as recalibrating sensors, updating routing rules, or deploying alternative data quality tasks. This proactive monitoring has the objective to keep data quality aligned with the IoT software system's quality requirements throughout its operation even if it cannot change it autonomously.

7.3 Illustrative Scenarios of Data Quality Management in IoT Smart Spaces

To better illustrate how the proposed data quality management solution operates within an IoT smart space, this section presents three practical scenarios. These scenarios showcase different challenges related to data quality and how the system dynamically adapts through its processes and components to ensure compliance with the IoT software system's requirements. Each scenario demonstrates the **detection** of a data quality issue, the **feedback and decision-making** process within the management layer and the **deployment** of appropriate corrective actions or the recognition that no intervention is needed.

The following scenarios are presented:

- Scenario 1 - Data Collection Error: An outlier is detected in the data collection phase, triggering a response that deploys an anomaly detection and removal technique (subsection 7.3.1).
- Scenario 2 - Behavioral Change Without Intervention: An outlier is detected but later validated as a normal environmental variation, requiring no further action (subsection 7.3.2).
- Scenario 3 - Timeliness Issue and Routing Prioritization Deployment: A delay in data delivery is detected, leading to the deployment of a routing prioritization policy in the Edge layer (subsection 7.3.3).

In the next subsections, we detail how each scenario unfolds and how the system components interact to ensure effective data quality management.

7.3.1 Scenario 1: Data Collection Error and Dynamic Adaptation of Data Quality Management

This scenario illustrates how the proposed data quality management approach reacts to an outlier detected at the data collection stage, triggering a sequence of actions that lead to the deployment of a data quality enhancement technique. This scenario is illustrated in figure 7.1.

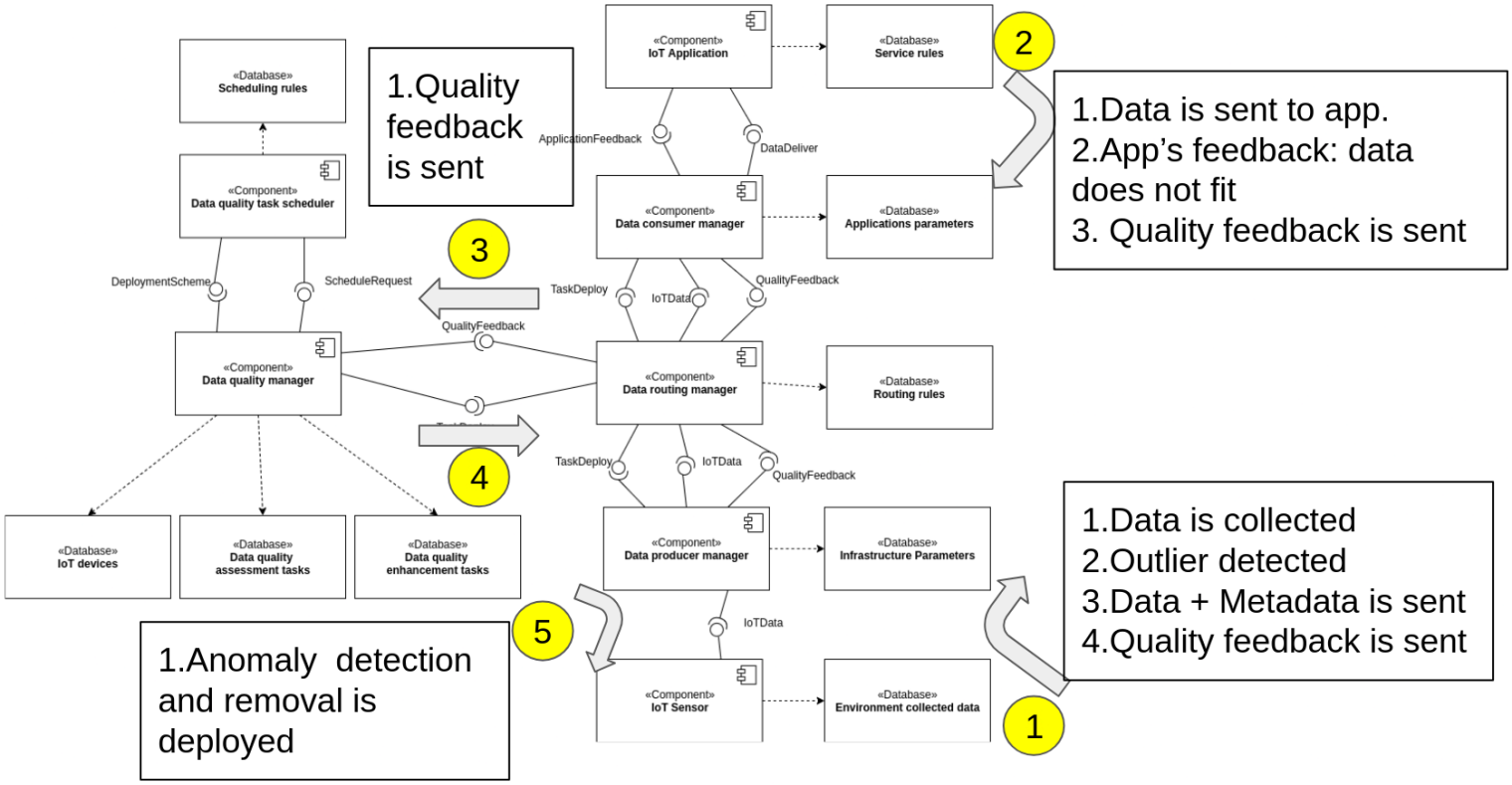

Figure 7.1: Scenario 1

Step 1: Sensor Data Collection and Outlier Detection

At the **IoT Device Layer** (figure 7.1), an *IoT Sensor* collects environmental data. During the data preprocessing stage, the *Data Producer Manager* detects an outlier in the collected data (figure 7.1). The outlier is identified as a potential anomaly, but since it is unclear whether it results from a legitimate environmental event or a sensor malfunction, the raw data and a metadata quality flag are transmitted to the Edge (figure 7.1).

Step 2: Transmission to the IoT Software System and Compliance Check

Upon reaching the **IoT Software System** (figure 7.1), the data is received by the *Data Consumer Manager*, which is responsible for assessing its compliance with the

predefined data quality requirements of the IoT software system. After evaluation, the *Data Consumer Manager* (figure 7.1) determines that the data does not meet the quality criteria for the intended functionalities. Consequently, a feedback message is generated, informing the data quality management process that the data is not fit for application use.

Step 3: Data Quality Feedback and Task Scheduling

The feedback message is transmitted via the *QualityFeedback* interface and routed back to the **Management Layer**, where the *Data Quality Manager* analyzes the situation. Since an outlier has been flagged at the **IoT Device Layer**, but it was not adequately processed before reaching the **IoT Software System Layer**, an adjustment is required. The *Data Quality Manager* determines that a data quality enhancement task—specifically, an outlier detection and removal technique—should be deployed to improve data quality at the source (figure 7.1).

Step 4: Deployment of an Anomaly Detection and Removal Technique

The *Data Quality Manager* communicates with the *Task Scheduler*, which schedules and deploys an appropriate anomaly detection and removal technique at the **IoT Device Layer**. This task is installed on the *Data Producer Manager* (figure 7.1), ensuring that future outliers can be detected and, if necessary, removed before transmission. This proactive adaptation prevents further propagation of low-quality data and enhances the overall reliability of the IoT system.

Outcome: Dynamic Improvement of Data Quality

This scenario (figure 7.1) demonstrates the ability of the proposed data quality management system to adapt dynamically to detected anomalies. Instead of allowing faulty data to propagate, the system identifies the issue, processes feedback, and deploys corrective actions autonomously, ensuring that future data is collected and processed with improved accuracy.

7.3.2 Scenario 2: Behavioral Change Without Intervention

This scenario illustrates how the proposed data quality management system differentiates between actual data anomalies and legitimate environmental changes, ensuring that unnecessary corrective actions are not deployed. This scenario is illustrated in figure 7.2.

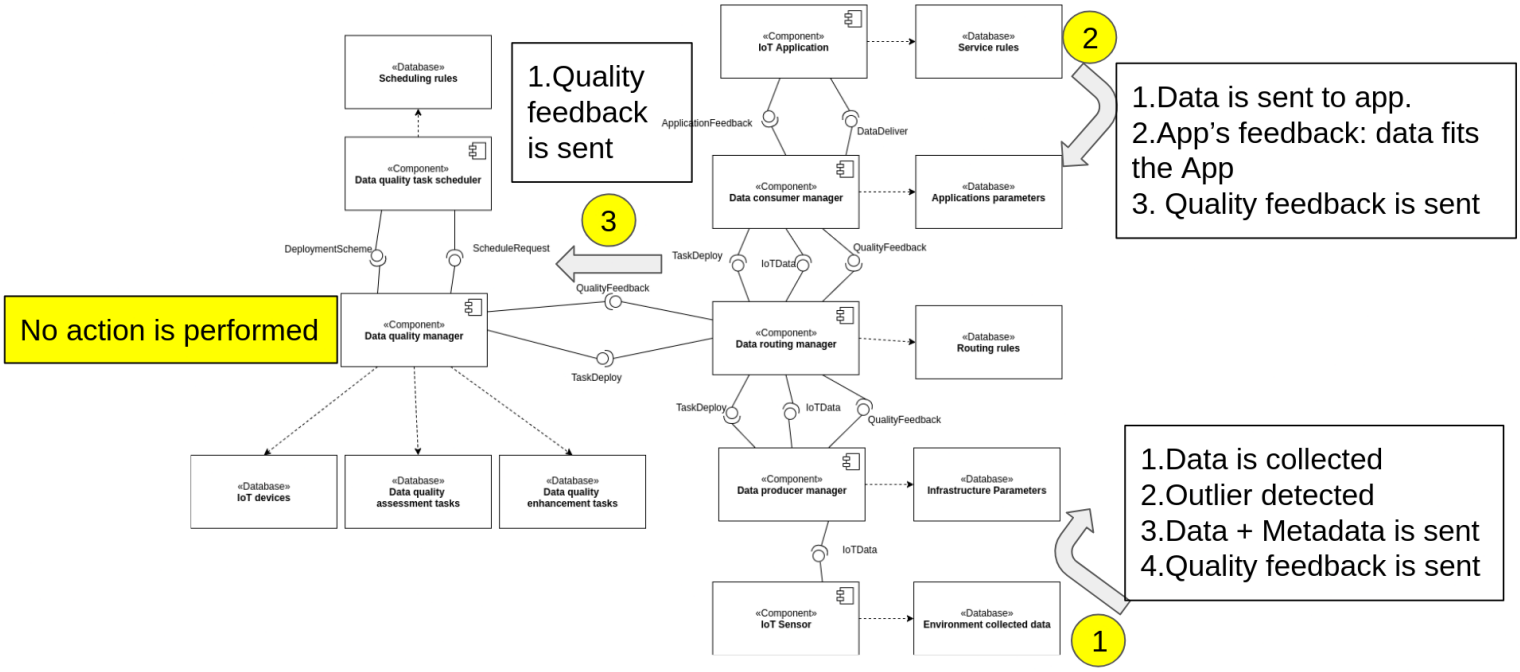

Figure 7.2: Scenario 2

Step 1: Sensor Data Collection and Outlier Detection

At the **IoT Device Layer**, an *IoT Sensor* collects environmental data (figure 7.2). During the data preprocessing stage, the *Data Producer Manager* detects an outlier in the collected data. Similar to the previous scenario, the system cannot immediately determine whether the outlier results from a sensor malfunction or a natural environmental shift. Consequently, the raw data and metadata quality flag are transmitted to the Edge layer (figure 7.2).

Step 2: Transmission to the IoT Software System and Compliance Check

Upon reaching the **IoT Software System Layer**, the data is received by the *Data Consumer Manager*, which evaluates the data against the predefined quality requirements. However, unlike the previous case, the *Data Consumer Manager* determines that the data fits within the expected operational conditions (figure 7.2). This means that the detected outlier was not an error, but rather a result of environmental changes that naturally modified the data distribution.

Step 3: Data Quality Feedback Without Corrective Action

Since the data is considered valid for the IoT software system, the *Data Consumer Manager* sends a *QualityFeedback* message to the *Data Quality Manager*, informing it that the data was successfully used by the IoT software system despite being

flagged as an outlier earlier (figure 7.2). The **Management Layer** registers this feedback but does not trigger any corrective actions, as no actual data quality issue was identified (figure 7.2).

Outcome: Adaptive Data Quality Without Unnecessary Modifications

This scenario (figure 7.2) highlights the adaptability of the proposed data quality management approach. Instead of blindly enforcing data correction for every detected outlier, the system correctly distinguishes between actual errors and natural environmental variations. By relying on feedback from the IoT software system, unnecessary interventions are avoided, ensuring that the system does not overcorrect and degrade legitimate data.

7.3.3 Scenario 3: Timeliness Issue and Routing Prioritization Deployment

This scenario demonstrates how the proposed data quality management system detects, processes, and resolves timeliness issues by dynamically adapting network routing policies. This scenario is illustrated in figure 7.3.

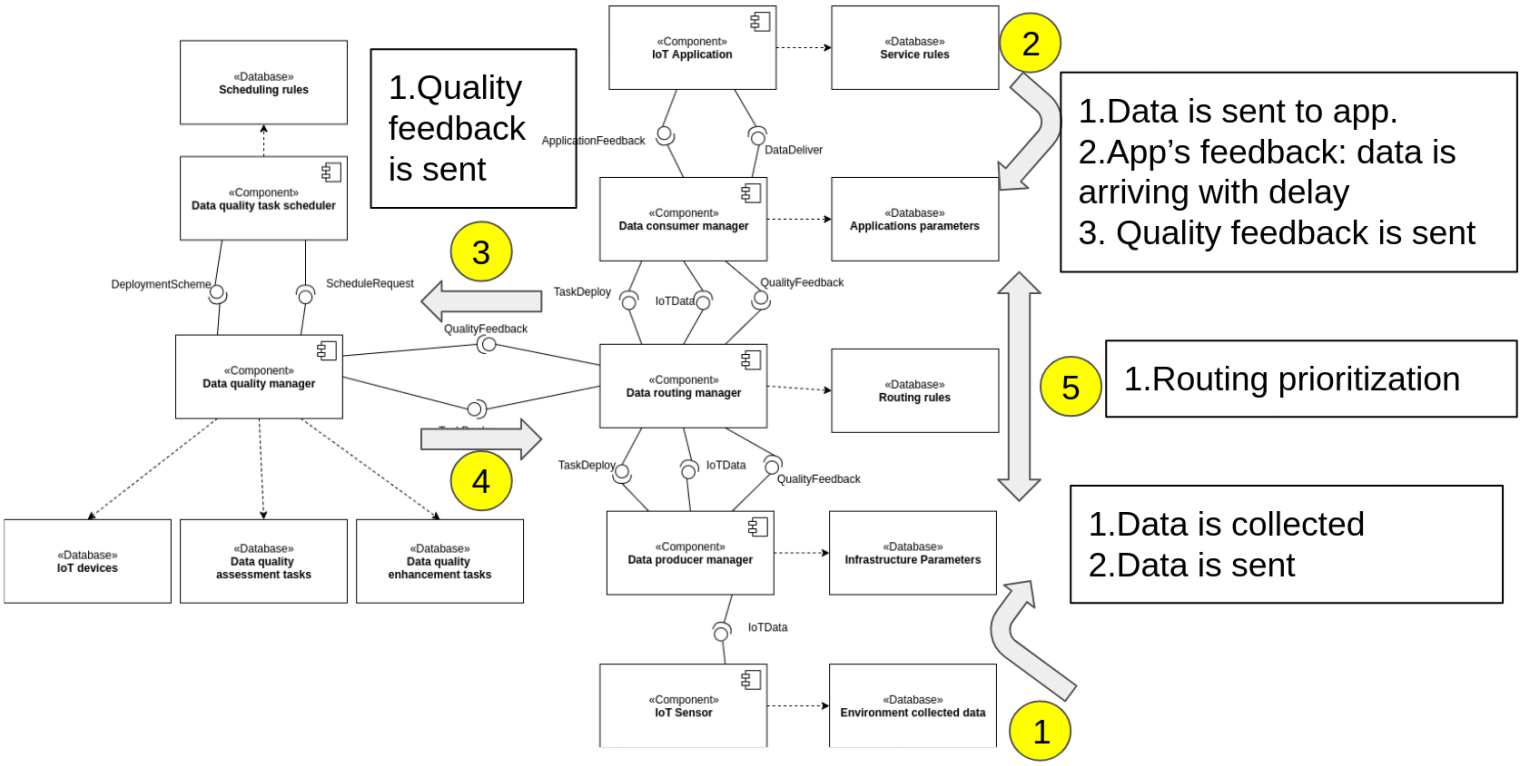

Figure 7.3: Scenario 3

Step 1: Sensor Data Collection and Transmission

At the **IoT Device Layer**, an *IoT Sensor* collects environmental data and transmits it to the Edge layer (figure 7.3). The *Data Producer Manager* performs preprocessing but does not detect anomalies in the data itself. The data is then forwarded through the network via the *Data Routing Manager* (figure 7.3).

Step 2: Late Data Arrival at the IoT Software System

Upon reaching the **IoT Software System Layer**, the *Data Consumer Manager* evaluates the received data (figure 7.3). While the data itself meets the required quality standards, its timeliness does not comply with the IoT software system's requirements. Due to network delays or congestion, the data arrives late, making it less valuable or even unusable for real-time decision-making (figure 7.3).

Step 3: Data Quality Feedback and Analysis

Recognizing the delay, the *Data Consumer Manager* sends a *QualityFeedback* message to the *Data Quality Manager* in the Management Layer (figure 7.3). The feedback specifies that the data does not meet the timeliness requirements and that if this issue persists, it could affect the overall performance of the IoT software system functionalities.

Step 4: Deploying Routing Prioritization at the Edge Layer

Upon receiving the feedback, the *Data Quality Manager* determines that network congestion or inefficient routing may be the cause of the delay. To address this issue, it interacts with the *Task Scheduler* to deploy a routing prioritization policy at the Edge Layer. The *Data Routing Manager* is reconfigured to prioritize time-sensitive data flows, ensuring that critical real-time information is transmitted with higher priority over less urgent data (figure 7.3).

Outcome: Adaptive Network Routing for Enhanced Timeliness

This scenario (figure 7.3) illustrates how the proposed data quality management approach actively adapts to transmission delays, ensuring that data quality is maintained according to different data quality dimensions. By dynamically adjusting routing strategies, the system minimizes delays and improves data timeliness.

7.4 Conclusion

This chapter presented a guide for implementing the proposed data quality management solution in IoT smart spaces. We outlined a step-by-step process, starting from the identification of data quality requirements, progressing through the deployment of distributed components, and culminating in the continuous operation and monitoring of the system.

The proposed solution integrates distributed components across the IoT smart space, with data quality management processes aligned with the objectives defined in the data quality management plan. Our approach not only addresses common data quality challenges but also enables proactive responses to infrastructure changes and evolving environmental conditions due to the dynamic nature of IoT environments. We propose a continuous monitoring of the dynamic task allocation to prevent issues on unpredictable operational scenarios.

7.4.1 Threats to Validity

Despite the structured implementation approach, certain threats to validity must be acknowledged:

- The proposed solution may face challenges in **generalizing** to diverse IoT smart space configurations. Different deployment contexts, device capabilities, and requirements could impact the effectiveness of the data quality management approach.
- **Implementation issues**, such as incorrect task configurations, miscommunication between components, or hardware limitations, could disrupt the intended functioning of the data quality management system.
- The **dynamic and decentralized** nature of IoT environments may introduce challenges in maintaining consistent performance. For instance, the rate of varying data loads or unexpected sensor failures might affect the system's ability to maintain data quality over time.

Chapter 8

Conclusion

This thesis presented a software architecture for data quality management, with focus on their practical implementation in IoT smart spaces. We proposed data quality management processes to manage data quality according to the data quality requirements of the IoT software system. Also, we introduced a set of components designed to be deployed across the key actors of the IoT smart space: **IoT devices**, **Edge**, and the **IoT Software System**. Additionally, we proposed two data quality techniques aligned with the concepts introduced in this work: a data quality preprocessing technique to enhance data quality at the point of data creation; and a data quality compliance technique to ensure data quality at the point of data consumption.

In Chapter 2, we presented the basic concepts of this work. We presented IoT related concepts as well the IoT Smart Space and its related aspects, including the fundamental characteristics of IoT systems, their architecture, and the challenges associated with data quality management. We also present data quality related concepts. We also discussed key data quality dimensions and standards. The objective of this chapter was to established the necessary background for understanding how data quality management can be effectively implemented in IoT software systems.

In Chapter 3, we reviewed related work on data quality management in IoT environments. We analyzed existing approaches, highlighting their strengths and limitations in addressing data quality challenges in IoT context. Additionally, we examined systematic literature reviews that synthesized prior research on IoT data quality, identifying common challenges and trends. Through this analysis, we observed how previous studies incorporated data quality principles and pinpointed gaps in data quality management solutions. This assessment provided the foundation for our proposal, reinforcing the need for a structured, component-based approach to executing data quality management processes in IoT Smart Spaces.

In chapter 4, we presented our proposal: **Executing data quality management processes on IoT Software Systems using a collection of software**

components. To structure the proposal, we adopted a tiered approach, which includes:

1. **Data Quality Management Processes**: We defined and structured the proposed data quality management processes. We proposed it considering the decentralized and dynamic nature of IoT smart spaces. In IoT Smart Spaces data is continuously generated and infrastructure conditions change. Our approach ensures that data quality management operates autonomously, minimizing the need for human intervention while maintaining the continuous assessment, control, assurance, and improvement of data quality.

2. **Software Components**: We proposed a structured set of software components to operationalize data quality management within the IoT smart space. We proposed them by **aligning with the roles and responsibilities of its key actors**: IoT devices, the Edge, and the IoT software system. These components were designed to autonomously execute the data quality processes. Duality management tasks are performed at the appropriate actor. We proposed it to enable the integration between distributed entities, distribute the data quality tasks among the different IoT smart space actors and being capable of changing the data quality management processes while being responsive to changes in the IoT Smart Space.

3. **Data Quality Techniques**: We proposed techniques designed to be executed within specific components. For IoT devices, we proposed a **data preprocessing technique** to enhance data quality at the source. The objective is to avoid the propagation of low quality data. For the IoT software system, we proposed a **data quality compliance technique** to ensure that consumed data meets predefined quality requirements. The objective is to avoid that the IoT Software System consumes data that is not inline with its data quality requirements.

Regarding the techniques we proposed in chapter 4 it is important to notice that they were tailored and designed with the concepts proposed in this work in mind. So we proposed it, ensuring alignment with the overall approach. Two distinct techniques were introduced: one focused on data preprocessing near the data collection source, executed at the *IoT Device* actor, to detect and address issues at the point of origin. This technique identifies and addresses issues in the data collection process that could impact the functionalities of the IoT Software System. Another technique was proposed as a data quality compliance technique aimed at ensuring that data meets the quality requirements of its functionalities. In this context, we focused on the confidence dimension of data quality. The technique we proposed

has the objective to perform data quality compliance at the *IoT Software System* actor. The data quality compliance technique ensures that data quality aligns with the requirements of the various functionalities of the IoT software system, enabling effective and accurate decision-making.

In Chapter 5, we conducted a PoC. The objective was to show the applicability of the proposed software architecture in an IoT Smart Grid scenario. We instantiate the proposed processes in a real-world scenario presented. We went from the gathering of the requirement to the deployment of data quality techniques within the components responsible for executing data quality tasks. The objective of this PoC was to demonstrate how data quality management can be effectively operationalized within an IoT software system with multiple functionalities. The PoC encompassed all stages of the object of proposal of this thesis and provided a practical view of how the proposed approach can be applied in dynamic IoT environments.

In chapter 6 we presented the **Evaluations and Results** of an instance of the proposed software architecture for data quality management through a Proof of Concept. We assess its software components in a smart grid scenario by executing the data quality tasks. In these evaluations, we implemented the proposed techniques (data quality preprocessing technique and data quality compliance technique) within the components designed to operationalize data quality management. This approach allowed us to assess the effectiveness of the proposed techniques, their integration into the concepts of this work and how the IoT data quality can be managed in an IoT smart space. The results demonstrate their ability to address data quality challenges in real-world applications and the effectiveness of the proposed data quality techniques.

The data preprocessing technique, in particular, showed **significant promise** in enhancing data quality at the source, highlighting the critical role of preprocessing capabilities in IoT devices. This technique demonstrated its ability to detect outliers, which may result from errors during data collection, and to flag them, allowing the IoT software system functionalities to make informed decisions about whether to use or discard such data. The results also indicated an improvement in the overall performance of the IoT software system applications. **However**, the evaluations revealed an increase in resource consumption and data transmission delay as a result of the data quality management processes. These trade-offs are justified by the observed improvement in application performance due to enhanced data quality. Nonetheless, in scenarios where resource limitations or latency are critical constraints, it is essential for a domain expert to assess the impact of such trade-offs on specific data quality dimensions and determine the suitability of the proposed approach for the given application context.

The data quality compliance technique was proposed based on the data quality

dimension of confidence. This choice was made to align the technique with a well-defined quality dimension that enables the evaluation of whether the data meets the quality requirements desired for the IoT software system functionalities. The dimension of confidence was selected because this work builds on prior studies that demonstrate how the dispersion and shape of the collected data curve significantly impact the data processing outcomes of IoT software system functionalities. In the evaluations conducted, the data quality compliance technique showed that managing data quality based on confidence had **mixed results**. While it improved functionality in some scenarios, in others, it adversely affected the performance of the IoT software system functionalities. This limitation suggests that the technique may not be optimal for all use cases and highlights the need for developing additional techniques that address other data quality dimensions.

Despite these challenges, the results underline the importance of implementing data quality management and selecting appropriate techniques for the specific quality dimensions required by IoT software system functionalities. The findings also emphasize that the IoT context is inherently dynamic, requiring data quality management processes and techniques to adapt continuously to changing scenarios. **However**, in this work the evaluation scenarios were static, which limited the ability to adapt or modify data quality management tasks dynamically. This highlights the need for future solutions to incorporate adaptive mechanisms that allow the data quality management system to adjust to evolving environmental conditions and infrastructural changes in IoT smart spaces.

Additionally, in Chapter 7, we presented implementation guidelines, providing a structured step-by-step approach for deploying the proposed solution in a given IoT smart space. This guide covered defining data quality requirements, deploying and configuring components across the *IoT Device Layer*, *Edge Layer*, and *IoT Software System Layer*, and enabling monitoring and adaptation mechanisms. The objective of these guidelines was to serve as a practical resource for practitioners, facilitating the adoption of the proposed approach in a variety of IoT scenarios while addressing the specific challenges of dynamic environments.

In **conclusion**, the proposed solution demonstrated significant adaptability to different scenarios. We now can revisit the **research question**, presented in the introduction: *How can a software architecture for data quality management be designed and implemented to address the unique challenges faced by IoT Software Systems?* - This research question is answered through this work. By operationalizing the proposed software architecture through components that can be utilized with various techniques, the approach provides a flexible approach for addressing diverse data quality challenges in IoT smart spaces that can endanger the use of IoT Software Systems. Furthermore, this work contributed by proposing two tailored techniques

for managing data quality: one focusing on preprocessing data at the IoT device layer and another addressing data quality compliance within the IoT software system. We put in the Appendices the published works that were product of this thesis.

8.1 Limitations

While this thesis has made significant contributions to data quality management in IoT smart spaces, it also has certain limitations that highlight areas for future research and development. This work did not explore the scheduling of data quality tasks, leaving open the opportunity to investigate how tasks can be dynamically allocated and prioritized across layers. Similarly, the development of new techniques tailored to specific IoT challenges remains an area for future exploration.

The Edge, although addressed in terms of its role in routing and data transmission, was not extensively explored for advanced solutions such as traffic prioritization, Quality of Service (QoS) mechanisms, or adaptive routing strategies. Future work could investigate how the Edge can play a more active role in optimizing data flows and ensuring data quality in resource-constrained and dynamic environments. This could include the development of algorithms and frameworks to balance processing workloads, manage traffic, and improve latency-sensitive applications.

Moreover, this thesis did not include experiments with practitioners responsible for implementing IoT software systems in IoT smart spaces. Such experiments would provide critical insights into the practical challenges faced during deployment and operation. Incorporating practitioner feedback would enhance the applicability and usability of the proposed solution, ensuring it addresses real-world constraints and scenarios effectively.

Another limitation is the static nature of the scenarios evaluated. This work did not consider dynamic scenarios where data quality tasks and requirements might change over time. Investigating dynamic adaptations of data quality management processes, including automated task reconfiguration and self-optimization mechanisms, could greatly enhance the robustness and scalability of the proposed solution.

Despite these limitations, this thesis opens a pathway for further research and practical applications in data quality management for IoT. It lays a foundation for exploring structured processes, integrated components, and tailored techniques to address data quality challenges in complex and evolving IoT environments. By addressing the gaps and limitations identified here, future research can build upon the contributions of this thesis to create more comprehensive and adaptive solutions for IoT smart spaces.

8.2 Future Works

While this thesis has provided a software architecture for data quality management in IoT smart spaces, it also opens several avenues for future research and development. The following directions are proposed to extend and complement the contributions of this work:

1. **Dynamic Task Scheduling**: This work did not explore the scheduling of data quality tasks in dynamic IoT environments. Future research could focus on developing algorithms for dynamic task allocation and prioritization, enabling tasks to adapt to changing conditions and demands. Such studies could explore scheduling across the *IoT Device Layer*, *Edge Layer*, and *IoT Software System Layer* to optimize resource usage and enhance overall system performance.

2. **Development of New Techniques**: The proposed data quality preprocessing and compliance techniques address specific scenarios and dimensions of data quality, such as confidence. However, future work could expand on this by developing new techniques tailored to other data quality dimensions, such as accuracy, completeness, and timeliness. These techniques could address a broader range of IoT challenges, further improving data quality management for diverse applications.

3. **Advancing the Edge Layer**: The Edge layer was primarily addressed in this thesis for its role in routing and data transmission. Future research could investigate advanced solutions such as Quality of Service (QoS) mechanisms, traffic prioritization, and adaptive routing strategies. These solutions could enable the Edge layer to play a more active role in optimizing data flows, reducing latency, and ensuring quality in resource-constrained environments.

4. **Human-Centered Evaluations**: This thesis did not conduct experiments with practitioners responsible for deploying IoT software systems in IoT smart spaces. Future studies could involve practitioners to gather insights into the challenges faced during deployment and operation. Understanding these practical constraints could guide the refinement of the proposed solution and make it more aligned with real-world needs.

5. **Dynamic and Adaptive Management**: The scenarios evaluated in this thesis were static, with fixed data quality tasks and requirements. Future work could investigate dynamic adaptations of data quality management processes, allowing for real-time reconfiguration of tasks and parameters. Research in

this area could explore self-optimization mechanisms to adapt to evolving environmental and infrastructural changes.

6. **Exploring New Use Cases**: While this thesis demonstrated its solution in a smart grid scenario, future studies could extend this approach to other IoT domains, such as healthcare, agriculture, and industrial IoT. Exploring different use cases would test the adaptability of the proposed solution and identify domain-specific requirements for data quality management.

7. **Integration with Machine Learning and AI**: Another promising direction is the integration of machine learning and artificial intelligence into the proposed solution. These technologies could enhance data quality management by enabling predictive analytics, anomaly detection, and automated decision-making. For example, machine learning models could predict sensor drift or identify patterns that require dynamic task reconfiguration.

8. **Scalability Studies**: Future work could explore the scalability of the proposed solution in large-scale IoT deployments with thousands of devices. This could involve stress testing the proposed components, processes, and techniques under varying loads to ensure robustness and efficiency in high-demand scenarios.

References

[1] AQUINO, G., PIRMEZ, L., DE FARIAS, C. M., et al. "Hephaestus: A multi-sensor data fusion algorithm for multiple applications on wireless sensor networks". In: *2016 19th International Conference on Information Fusion (FUSION)*, pp. 59–66, July 2016.

[2] ATZORI, L., IERA, A., MORABITO, G. "The internet of things: A survey", *Computer networks*, v. 54, n. 15, pp. 2787–2805, 2010.

[3] LI, G., DE CLERCQ, E. "Therapeutic options for the 2019 novel coronavirus (2019-nCoV)". 2020.

[4] PEERI, N. C., SHRESTHA, N., RAHMAN, M. S., et al. "The SARS, MERS and novel coronavirus (COVID-19) epidemics, the newest and biggest global health threats: what lessons have we learned?" *International journal of epidemiology*, 2020.

[5] JAVAID, M., KHAN, I. H. "Internet of Things (IoT) enabled healthcare helps to take the challenges of COVID-19 Pandemic", *Journal of Oral Biology and Craniofacial Research*, v. 11, n. 2, pp. 209–214, 2021.

[6] JAVAID, M., HALEEM, A., VAISHYA, R., et al. "Industry 4.0 technologies and their applications in fighting COVID-19 pandemic", *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, v. 14, n. 4, pp. 419–422, 2020.

[7] ARASTEH, H., HOSSEINNEZHAD, V., LOIA, V., et al. "Iot-based smart cities: A survey", *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 1–6, June 2016. doi=10.1109/EEEIC.2016.7555867.

[8] DA XU, L., HE, W., LI, S. "Internet of things in industries: A survey", *IEEE Transactions on industrial informatics*, v. 10, n. 4, pp. 2233–2243, 2014.

[9] WHITMORE, A., AGARWAL, A., DA XU, L. "The Internet of Things—A survey of topics and trends", *Information Systems Frontiers*, v. 17, n. 2, pp. 261–274, 2015.

[10] SOTRES, P., SANTANA, J. R., SÁNCHEZ, L., et al. "Practical lessons from the deployment and management of a smart city Internet-of-Things infrastructure: The smartsantander testbed case", *IEEE Access*, v. 5, pp. 14309–14322, 2017.

[11] EDRIS, A.-A., D'ANDRADE, B. W. "Transmission Grid Smart Technologies". In: *The Power Grid*, Elsevier, pp. 37–55, 2017.

[12] XIAO, Z., LIM, H. B., PONNAMBALAM, L. "Participatory Sensing for Smart Cities: A Case Study on Transport Trip Quality Measurement", *IEEE Transactions on Industrial Informatics*, v. 13, n. 2, pp. 759–770, 2017.

[13] FADEL, E., GUNGOR, V. C., NASSEF, L., et al. "A survey on wireless sensor networks for smart grid", *Computer Communications*, v. 71, pp. 22–33, 2015.

[14] MOTTA, R. C. *An Evidence-Based Roadmap to Support the Internet of Things Software Systems Engineering.* Tese de Doutorado, Université Polytechnique Hauts-de-France; Universidade federal do Rio de Janeiro, 2021.

[15] SILVA, V. M. D. "ScenarIoT: support for scenario specification of internet of things-based software systems", 2019.

[16] NAKAMURA, E. F., LOUREIRO, A. A., FRERY, A. C. "Information fusion for wireless sensor networks: Methods, models, and classifications", *ACM Computing Surveys (CSUR)*, v. 39, n. 3, pp. 9, 2007.

[17] GOUD, D. S., KUMAR, C., DEVIPRIYA, S., et al. "Low Power Design Techniques for IoT Devices". In: *2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, pp. 1–5, 2024. doi: 10.1109/ICONSTEM60960.2024.10568684.

[18] MANSOURI, T., SADEGHI MOGHADAM, M. R., MONSHIZADEH, F., et al. "IoT Data Quality Issues and Potential Solutions: A Literature Review", *The Computer Journal*, v. 66, n. 3, pp. 615–625, 2021. doi: 10.1093/comjnl/bxab183.

[19] KARKOUCH, A., MOUSANNIF, H., AL MOATASSIME, H., et al. "Data quality in internet of things: A state-of-the-art survey", *Journal of Network and Computer Applications*, v. 73, pp. 57–81, 2016.

[20] TORNESE, I., MATERA, A., RASHVAND, M., et al. "Use of Probes and Sensors in Agriculture—Current Trends and Future Prospects on Intelligent

Monitoring of Soil Moisture and Nutrients", *AgriEngineering*, v. 6, n. 4, pp. 4154–4181, 2024.

[21] WANG, R. Y., STRONG, D. M. "Beyond accuracy: What data quality means to data consumers", *Journal of management information systems*, v. 12, n. 4, pp. 5–33, 1996.

[22] WANG, X., WANG, C. "Time series data cleaning: A survey", *Ieee Access*, v. 8, pp. 1866–1881, 2019.

[23] TEH, H. Y., KEMPA-LIEHR, A. W., KEVIN, I., et al. "Sensor data quality: a systematic review", *Journal of Big Data*, v. 7, n. 1, pp. 11, 2020.

[24] LIU, C., NITSCHKE, P., WILLIAMS, S. P., et al. "Data quality and the Internet of Things", *Computing*, v. 102, n. 2, pp. 573–599, 2020.

[25] ZHANG, L., JEONG, D., LEE, S. "Data quality management in the internet of things", *Sensors*, v. 21, n. 17, pp. 5834, 2021.

[26] ALWAN, A. A., CIUPALA, M. A., BRIMICOMBE, A. J., et al. "Data quality challenges in large-scale cyber-physical systems: A systematic review", *Information Systems*, v. 105, pp. 101951, 2022.

[27] GOKNIL, A., NGUYEN, P., SEN, S., et al. "A systematic review of data quality in CPS and IoT for industry 4.0", *ACM Computing Surveys*, v. 55, n. 14s, pp. 1–38, 2023.

[28] SERRA, F., PERALTA, V., MAROTTA, A., et al. "Use of context in data quality management: a systematic literature review", *ACM Journal of Data and Information Quality*, v. 16, n. 3, pp. 1–41, 2024.

[29] DE AQUINO, G. R. C., DE FARIAS, C. M. "Asclepius: Data quality framework for iot". In: *Proceedings of the Int'l ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, pp. 69–76, 2023.

[30] SOMMERVILLE, I. "Software engineering (ed.)", *America: Pearson Education Inc*, 2011.

[31] DE MELLO DANTAS, H., DE FARIAS, C. M. "A data fusion algorithm for clinically relevant anomaly detection in remote health monitoring". In: *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–8. IEEE, 2020.

[32] LYSSENKO, M., GLADISCH, C., HEINZEMANN, C., et al. "Towards Safety-Aware Pedestrian Detection in Autonomous Systems". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 293–300, 2022. doi: 10.1109/IROS47612.2022.9981309.

[33] DIACONU, B. M. "Recent advances and emerging directions in fire detection systems based on machine learning algorithms", *Fire*, v. 6, n. 11, pp. 441, 2023.

[34] EUROPEAN PARLIAMENT AND COUNCIL. "General Data Protection Regulation (GDPR)". 2016. Disponível em: <`https://eur-lex.europa.eu/eli/reg/2016/679/oj`>. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016.

[35] PRESIDÊNCIA DA REPÚBLICA DO BRASIL. "Lei Geral de Proteção de Dados Pessoais (LGPD)". 2018. Disponível em: <`http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`>. Lei nº 13.709, de 14 de agosto de 2018.

[36] VARSHNEY, S., SANDHU, R., GUPTA, P. K. "QoS Based Resource Provisioning in Cloud Computing Environment: A Technical Survey". In: Singh, M., Gupta, P., Tyagi, V., et al. (Eds.), *Advances in Computing and Data Sciences*, pp. 711–723, Singapore, 2019. Springer Singapore. ISBN: 978-981-13-9942-8.

[37] CACHEDA, R. A., GARCÍA, D. C., CUEVAS, A., et al. "QoS requirements for multimedia services". In: *Resource management in satellite networks*, Springer, pp. 67–94, 2007.

[38] FROOM, R., FLANNAGAN, M., TUREK, K. *Cisco catalyst QoS: quality of service in campus networks*. Cisco Press, 2003.

[39] SHAH, S. H., YAQOOB, I. "A survey: Internet of Things (IOT) technologies, applications and challenges". In: *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pp. 381–385. IEEE, 2016.

[40] RECOMMENDATION, Y. "2060 «overview of internet of things»", *ITU-T, Geneva*, 2012.

[41] GUBBI, J., BUYYA, R., MARUSIC, S., et al. "Internet of Things (IoT): A vision, architectural elements, and future directions", *Future generation computer systems*, v. 29, n. 7, pp. 1645–1660, 2013.

[42] FARIAS, C., PIRMEZ, L., DELICATO, F., et al. "Multisensor data fusion in shared sensor and actuator networks". In: *Information Fusion (FUSION), 2014 17th International Conference on*, pp. 1–8. IEEE, 2014.

[43] IMAN, M., DELICATO, F. C., DE FARIAS, C. M., et al. "THESEUS: A routing system for shared sensor networks". In: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 108–115. IEEE, 2015.

[44] COSTA, E. A., PIRMEZ, L., DE FARIAS, C. M., et al. "CAMAW: A Clustering Algorithm for Multiple Applications in WSAN", *ICSNC 2015*, p. 91, 2015.

[45] LIU, Y., YANG, C., JIANG, L., et al. "Intelligent edge computing for IoT-based energy management in smart cities", *IEEE network*, v. 33, n. 2, pp. 111–117, 2019.

[46] MARCHENKOV, S. "Infrastructure multi-layer model for smart spaces middleware development". In: *Conference of Open Innovations Association, FRUCT*, n. 22, pp. 353–360. FRUCT Oy, 2018.

[47] DE FARIAS, C. M., PIRMEZ, L., FORTINO, G., et al. "A multi-sensor data fusion technique using data correlations among multiple applications", *Future generation computer systems*, v. 92, pp. 109–118, 2019.

[48] DE FARIAS, C. M., PIRMEZ, L., DELICATO, F. C., et al. "GROWN: A control and decision system for smart greenhouses using wireless sensor networks". In: *Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1–8, 2017.

[49] CALDAS, G., DE FARIAS, C. M., PIRMEZ, L., et al. "S-LEACH: A LEACH extension for Shared Sensor Networks". In: *Proceedings of the International Conference on Wireless Networks (ICWN)*, p. 121. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[50] DE FARIAS, C. M., PIRMEZ, L., DELICATO, F. C. "Density based multisensor data fusion for multiapplication wireless sensor networks". In: *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 814–821. IEEE, 2018.

[51] CULMAN, M., PORTOCARRERO, J. M., GUERRERO, C. D., et al. "PalmNET: An open-source wireless sensor network for oil palm plantations". In: *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 783–788. IEEE, 2017.

[52] CULMAN, M., DE FARIAS, C. M., BAYONA, C., et al. "Using agrometeorological data to assist irrigation management in oil palm crops: A decision support method and results from crop model simulation", *Agricultural water management*, v. 213, pp. 1047–1062, 2019.

[53] HASSAN, M. K., EL DESOUKY, A. I., ELGHAMRAWY, S. M., et al. "Big data challenges and opportunities in healthcare informatics and smart hospitals", *Security in smart cities: Models, applications, and challenges*, pp. 3–26, 2019.

[54] BLASCH, E., BORIL, J., SMRZ, V., et al. "Pilot interface considerations using high level information fusion". In: *2018 IEEE Aerospace Conference*, pp. 1–8. IEEE, 2018.

[55] ISO. *ISO 9000:2015 Quality management systems - Fundamentals and vocabulary* . Standard, International Organization for Standardization, Geneva, CH, sep 2015.

[56] FERREIRA, E., FERREIRA, D. "Towards altruistic data quality assessment for mobile sensing". In: *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pp. 464–469. ACM, 2017.

[57] FOR STANDARDIZATION/INTERNATIONAL ELECTROTECHNICAL COMMISSION, I. O., OTHERS. "Software engineering-Software product Quality Requirements and Evaluation (SQuaRe) Data quality model", *ISO/IEC*, v. 25012, pp. 1–13, 2008.

[58] QIN, Z., HAN, Q., MEHROTRA, S., et al. "Quality-aware sensor data management". In: *The Art of Wireless Sensor Networks*, Springer, pp. 429–464, 2014.

[59] KLEIN, A., LEHNER, W. "Representing data quality in sensor data streaming environments", *Journal of Data and Information Quality (JDIQ)*, v. 1, n. 2, pp. 10, 2009.

[60] LI, F., NASTIC, S., DUSTDAR, S. "Data quality observation in pervasive environments". In: *2012 IEEE 15th International Conference on Computational Science and Engineering*, pp. 602–609. IEEE, 2012.

[61] WEIDEMA, B. P., WESNAES, M. S. "Data quality management for life cycle inventories—an example of using data quality indicators", *Journal of cleaner production*, v. 4, n. 3-4, pp. 167–174, 1996.

[62] MATHIEU, R. G., KHALIL, O. "Data quality in the database systems course", *Data Quality Journal*, v. 4, n. 1, pp. 1–12, 1998.

[63] TEOREY, T., LIGHTSTONE, S., NADEAU, T., et al. "1 - Introduction". In: Teorey, T., Lightstone, S., Nadeau, T., et al. (Eds.), *Database Modeling and Design (Fifth Edition)*, The Morgan Kaufmann Series in Data Management Systems, fifth edition ed., Morgan Kaufmann, pp. 1–11, Boston, 2011. ISBN: 978-0-12-382020-4. doi: https://doi.org/10.1016/B978-0-12-382020-4.00001-X. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978012382020400001X>.

[64] CYKANA, P., PAUL, A., STERN, M. "DoD Guidelines on Data Quality Management". In: *IQ*, 1996.

[65] SINGH, A., PAYAL, A., BHARTI, S. "A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features, and open issues", *Journal of Network and Computer Applications*, v. 143, pp. 111–151, 2019. ISSN: 1084-8045. doi: https://doi.org/10.1016/j.jnca.2019.06.013. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804519302188>.

[66] KASRIN, N., BENABBAS, A., ELMAMOOZ, G., et al. "Data-sharing markets for integrating IoT data processing functionalities", *CCF Transactions on Pervasive Computing and Interaction*, v. 3, n. 1, pp. 76–93, 2021.

[67] BALAJI, S., NATHANI, K., SANTHAKUMAR, R. "IoT technology, applications and challenges: a contemporary survey", *Wireless personal communications*, v. 108, n. 1, pp. 363–388, 2019.

[68] LABOUSEUR, A. G., MATHEUS, C. C. "An introduction to dynamic data quality challenges", *Journal of Data and Information Quality (JDIQ)*, v. 8, n. 2, pp. 6, 2017.

[69] LUO, T., HUANG, J., KANHERE, S. S., et al. "Improving IoT Data Quality in Mobile Crowd Sensing: A Cross Validation Approach", *IEEE Internet of Things Journal*, 2019.

[70] TKACHENKO, R., IZONIN, I., KRYVINSKA, N., et al. "An approach towards increasing prediction accuracy for the recovery of missing IoT data based on the GRNN-SGTM ensemble", *Sensors*, 2020.

[71] IZONIN, I., TKACHENKO, R., VERHUN, V., et al. "An approach towards missing data management using improved GRNN-SGTM ensemble method", *Engineering Science and Technology, an International Journal*, 2020.

[72] LIU, Y., DILLON, T., YU, W., et al. "Missing Value Imputation for Industrial IoT Sensor Data With Large Gaps", *IEEE Internet of Things Journal*, v. 7, n. 8, pp. 6855–6867, 2020. doi: 10.1109/JIOT.2020.2970467.

[73] MARY, I. P. S., AROCKIAM, L. "Imputing the missing data in IoT based on the spatial and temporal correlation". In: *2017 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, pp. 1–4, 2017. doi: 10.1109/ICCTAC.2017.8249990.

[74] ANN, F. N., WAGH, R. "Quality assurance in big data analytics: An IoT perspective", *Telfor Journal*, v. 11, n. 2, pp. 114–118, 2019.

[75] PEREZ-CASTILLO, R., CARRETERO, A. G., RODRIGUEZ, M., et al. "Data quality best practices in IoT environments". In: *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pp. 272–275. IEEE, 2018.

[76] ALRAE, R., NASIR, Q., ABU TALIB, M. "Developing House of Information Quality framework for IoT systems", *International Journal of System Assurance Engineering and Management*, pp. 1–20, 2020.

[77] SUN, D., XUE, S., WU, H., et al. "A Data Stream Cleaning System Using Edge Intelligence for Smart City Industrial Environments", *IEEE Transactions on Industrial Informatics*, v. 18, n. 2, pp. 1165–1174, feb 2022. doi: 10.1109/tii.2021.3077865. Disponível em: <`https://doi.org/10.1109%2Ftii.2021.3077865`>.

[78] BAMGBOYE, O., LIU, X., CRUICKSHANK, P. "Semantic stream management framework for data consistency in smart spaces". In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, v. 2, pp. 85–90. IEEE, 2019.

[79] ANICIC, D., FODOR, P., RUDOLPH, S., et al. "EP-SPARQL: a unified language for event processing and stream reasoning". In: *Proceedings of the 20th international conference on World wide web*, pp. 635–644, 2011.

[80] BARBIERI, D. F., BRAGA, D., CERI, S., et al. "Querying rdf streams with c-sparql", *ACM SIGMOD Record*, v. 39, n. 1, pp. 20–26, 2010.

[81] BALDASSARRE, M. T., CABALLERO, I., CAIVANO, D., et al. "From big data to smart data: a data quality perspective". In: *Proceedings of the 1st ACM SIGSOFT International Workshop on Ensemble-Based Software Engineering*, pp. 19–24. ACM, 2018.

[82] DE FARIAS, C. M., PIRMEZ, L., DELICATO, F. C., et al. "A scheduling algorithm for shared sensor and actuator networks". In: *Information Networking (ICOIN), 2013 International Conference on*, pp. 648–653. IEEE, 2013.

[83] GUPTA, B. B., QUAMARA, M. "An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols", *Concurrency and Computation: Practice and Experience*, v. 32, n. 21, pp. e4946, 2020.

[84] KAKANAKOVA, I., STOYANOV, S. "Outlier Detection via Deep Learning Architecture". In: *Proceedings of the 18th International Conference on Computer Systems and Technologies*, pp. 73–79. ACM, 2017.

[85] ABID, A., KACHOURI, A., MAHFOUDHI, A. "Data analysis and outlier detection in smart city". In: *Smart, Monitored and Controlled Cities (SM2C), 2017 International Conference on*, pp. 1–4. IEEE, 2017.

[86] COOK, A., MISIRLI, G., FAN, Z. "Anomaly detection for IoT time-series data: A survey", *IEEE Internet of Things Journal*, 2019.

[87] DE AQUINO, G. R. C., DE FARIAS, C. M., PIRMEZ, L. "Hygieia: data quality assessment for smart sensor network". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 889–891. ACM, 2019.

[88] VINISHA, F. A., SUJIHELEN, L. "Study on Missing Values and Outlier Detection in Concurrence with Data Quality Enhancement for Efficient Data Processing". In: *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1600–1607. IEEE, 2022.

[89] DEGROOT, M. H., SCHERVISH, M. J. *Probability and statistics*. Pearson, 2012.

[90] DE AQUINO, G., FARIAS, C. M., PIRMEZ, L. "Volcanus: Using Information Fusion Concepts to Perform Data Quality Assessment and Enhancement

on IoT Data (Poster)". In: *2019 22nd International Conference on Information Fusion (FUSION) (FUSION 2019)*, Ottawa, Canada, jul. 2019.

[91] GEMIRTER, C. B., ŞENTURCA, Ç., BAYDERE, Ş. "A comparative evaluation of AMQP, MQTT and HTTP protocols using real-time public smart city data". In: *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 542–547. IEEE, 2021.

[92] ATZORI, L., IERA, A., MORABITO, G. "The Internet of Things: A survey", *Computer Networks*, v. 54, n. 15, pp. 2787 – 2805, 2010. ISSN: 1389-1286. doi: https://doi.org/10.1016/j.comnet.2010.05.010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>.

[93] JOANES, D., GILL, C. "Comparing measures of sample skewness and kurtosis", *Journal of the Royal Statistical Society: Series D (The Statistician)*, v. 47, n. 1, pp. 183–189, 1998.

[94] PEARSON, K. "Contributions to the mathematical theory of evolution. II. Skew variation in homogeneous material", *Philosophical Transactions of the Royal Society of London*, v. 186, n. Part I, pp. 343–424, 1895.

[95] PORTOCARRERO, J. M., DELICATO, F. C., PIRES, P. F., et al. "RAMSES: A new reference architecture for self-adaptive middleware in Wireless Sensor Networks", *Ad Hoc Networks*, v. 55, pp. 3–27, 2017. ISSN: 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2016.11.004. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570870516303006>. Self-organizing and Smart Protocols for Heterogeneous Ad hoc Networks.

[96] GAO, L., ZHENG, Z., HUO, M. "Improvement of RPL protocol algorithm for smart grid". In: *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, pp. 927–930. IEEE, 2018.

[97] PIYARE, R., LEE, S. R. "Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing", *arXiv preprint arXiv:1310.2095*, 2013.

[98] DE FARIAS, C. M., PIRMEZ, L., DELICATO, F. C., et al. "Information fusion techniques applied to Shared Sensor and Actuator Networks". In: *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pp. 188–191. IEEE, 2012.

[99] SCHLAPFER, M., MANCARELLA, P. "Probabilistic modeling and simulation of transmission line temperatures under fluctuating power flows", *IEEE Transactions on Power Delivery*, v. 26, n. 4, pp. 2235–2243, 2011.

[100] DJIROUN, F. Z., DJENOURI, D. "MAC Protocols With Wake-Up Radio for Wireless Sensor Networks: A Review", *IEEE Communications Surveys Tutorials*, v. 19, n. 1, pp. 587–618, Firstquarter 2017. ISSN: 1553-877X. doi: 10.1109/COMST.2016.2612644.

Appendices

Data Quality assessment and enhancement on Social and Sensor Data

Gabriel R. Caldas de Aquino, Claudio Miceli de Farias, and Luci Pirmez *
gabriel@labnet.nce.ufrj.br claudiofarias@nce.ufrj.br luci.pirmez@gmail.com

Universidade Federal do Rio de Janeiro - Programa de Pós-Graduação em Informática
UFRJ, Rio de Janeiro, Brasil

Abstract. Smartphones are key devices in the Internet of Things paradigm. Social networking services on the Internet can use smartphones applications as data providers. The data gathered from sensors and data harvested from social networking services can be used by different applications for providing context-aware services. However, the excellence of the data oriented services depends on the Data quality (DQ). DQ is critical for decision making mechanisms. We present the problem related to DQ when dealing with social and sensor data. Also, we present and explore a framework whose objective is to evaluate and control DQ aspects when dealing with social and sensor data.

Keywords: Data Quality · Social Network · Internet of Things.

1 Introduction

The Internet of Things (IoT) paradigm embraces several types of smart devices which are composed by sensors, actuators and other devices with networking capabilities [1]. In the context of IoT, smartphones are key devices embedded with different types of sensors. Smartphones are providers of large amounts of environmental data. In addition, social networking platforms use smartphone applications to enable the interaction of their users with the social networks anywhere and anytime. This creates a pervasive channel for users to record and share their personal activities on social platforms [11] (e.g. Twitter).

Huge data repositories are produced in this scenario. Services can use data fusion [7] techniques on sensor and social networking platforms data to perform environment actuations. Also data analytics procedures can use social and sensor data in a complementary manner for enriching the data analysis. This enables the use of the data gathered from sensors and data harvested from social media to create contextual integrated services [1].

However, the excellence of the aforementioned services depends on Data Quality (DQ) aspects [5] of the social and sensor data that is consumed. Quality is deemed as a critical requirement for decision making mechanisms, applications

* The authors of this research paper would like to thank Kontron for providing the private cloud server Symkloud as the basic infrastructure used in this work.

and services. Data with poor DQ aspects can lead to erroneous decisions and analysis. In this work we define DQ as *a concept that refers to how well the data corresponds to the quality necessities of data consumers* [6] [5]. An interesting fact comes from the aforementioned definition: DQ refers to the fitness of which the data is perceived by its consumer. This means that DQ would hardly be seen in the same way by different users. In fact, each data consumer requires the used data to fulfill certain criteria which he presumes essential for his own tasks at hand. DQ standardizing is the process of making the data conforms certain DQ requirements by assessing and enhancing DQ. The problem of DQ assessment is commonly addressed through DQ dimensions [4]. DQ enhancement can be done through DQ enhancement techniques [5]. In this work we present and explore a framework for social and sensor DQ standardizing. The rest of this work is divided into 3 chapters: in chapter 2 we present the Data Quality Concepts; in chapter 3 we present the Framework for Social and Sensor DQ and in chapter 4 we conclude our work.

2 Data Quality Concepts

In this chapter we briefly discuss some important Data Quality (DQ) concepts. DQ is deemed as a critical requirement for decision making mechanisms, applications and services on the IoT. There are different definitions of DQ. In this work DQ can be defined as a concept that refers to how well the data corresponds to the necessities of data processing mechanisms [6] [5]. An interesting conclusion comes from the aforementioned definition: DQ refers the fitness of which data is perceived by its consumer. This means that DQ would hardly be seen in the same way by different users. In fact, each data consumer requires the used data to fulfill certain criteria which he presumes essential for his own tasks at hand. The problem of DQ assessment is commonly addressed through data quality dimensions [4]. The ISO international standard DQ model identifies several DQ characteristics in the context of Software Engineering [8]. According to [10], the dimensions of data quality data can be categorized into four semantic aspects: (i) *intrinsic* , (ii) *accessibility*, (iii) *contextual*, and (iv) *representational*.

The (i) *intrinsic data quality semantic aspect* is related to the quality of the data in relation to itself. As examples of dimensions there are: accuracy, objectivity, believability and reputation [10]. (ii) The *accessibility data quality semantic* dimensions describe how accessible the data is for data consumers. Examples can be accessibility and access security [10]. *Contextual data quality semantic* aspect is related to how appropriate the data is for its usage. As examples of dimensions can be relevancy, value-added , timeliness, completeness and amount of data [10]. *Representational data quality semantic* aspect describes how understandable and representative of the environment the data is. Examples of dimensions are interoperability, ease of understanding, concise representation and consistent representation [10]. The DQ challenges refers to difficulties affecting any DQ dimension. Such challenges can cause data to become entirely or partially unusable, since it may not meet the requirements of data consumers.

As we discussed, DQ does not relate only to data accuracy. Instead data quality problems can surpass the accuracy dimension to other dimensions such as the aforementioned.

3 Framework for Social and Sensor Data Quality

This section presents the framework for social and sensor DQ standardizing. Standardizing DQ is making the data conforms certain DQ requirements. This framework receives data as input and transforms the input data to conform the DQ requirements of a given application or system that will receive such data as the output of the framework. The framework has two components: (i) social DQ component and (ii) sensor DQ component.

The (i) Social DQ component is responsible for standardizing social-originated data according to DQ requirements. The (ii) Sensor DQ component is responsible for standardizing sensor-originated data according to DQ requirements. Both the (i) and (ii) components present two subcomponents: a first subcomponent is responsible for DQ assessment, while the second subcomponent is responsible for DQ enhancement. DQ assessment subcomponent is responsible for the DQ evaluation according to given DQ requirements. DQ enhancement subcomponent is responsible for enhancing DQ to a given DQ requirement. The framework is illustrated at Figure 1.

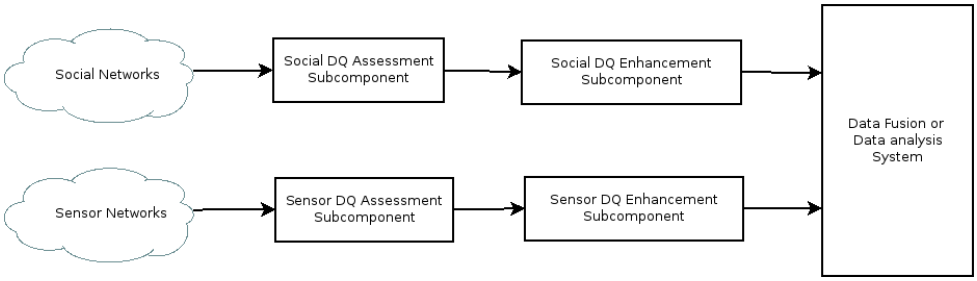

Fig. 1. The Framework

3.1 Social DQ Component

Social DQ component is responsible for social DQ standardizing. The first challenge for the Social DQ component is that social-originated data present significant differences according to the social media that originated the data. Differences can be if the data is structured or if it is unstructured [9], can be about the type of social interaction(the understanding of a message posted in forum thread is different from the understanding of a message posted in a Twitter discussion

[2]), etc. In other words, social data should be analyzed differently, according to the social platform that originated the message.

These differences influence the development of Social DQ analysis solutions. It is important to notice that different Social platforms may request a priori specialized DQ analysis solutions. The social DQ component is composed by two subcomponents: (i) Social DQ assessment subcomponent and (ii) Social DQ enhancement subcomponent.

The **Social DQ assessment subcomponent** aims at assessing the DQ from data originated from social networks. According to [2], Social DQ can be degraded by: (i) *Keyword ambiguity* and (ii) *Users Spamming*. (i) Keyword ambiguity is the addressing of a same context through different keywords. Even in some cases the correlation may not be obvious. Also, in the opposite, different contexts can be addressed by a common keyword. Since social data is generally collected through keyword searches, the Keyword ambiguity impacts the overall collected DQ. (ii) Users Spamming is the use of trending keywords to massively propagate a message. In some cases, such messages may not present a correlation with to the context in which the keyword is inserted. Instead, users can use trending keywords to leverage the visualization rate of the message. Spam messages can lead to the misunderstanding about the keyword context. According to [2], a key characteristic of Spam messages is its neutral tone (e.g., Check out this coupon).

The **Social DQ enhancement subcomponent** aims at processing social data to enhance the social DQ to correspond a DQ requirement of a data consumer. Given the result of the Social DQ assessment subcomponent, the Social DQ enhancement subcomponent performs the following actions: (i) accept the data to be given to the data consumer, (ii) enhance the data to the corresponding DQ requirement of the data consumer or (iii) discard the data.

3.2 Sensor DQ Component

Sensor DQ component is responsible for standardizing sensor-originated data. However, assessing and enhancing DQ for sensor data is not trivial. The sensory platforms are heterogeneous and resource-constrained. Sensor data can be originated from different kinds of sensor devices (e.g. smartphones, smart sensor networks, etc. [1]). Different sensor devices can present different data precision, data ranges, data units, hardware specifications, etc.

Regarding the sensed environment, sensory devices are placed in uncontrolled environments. In such case, misplacement, communication errors, power failure, sensor malfunction, human error or intentional misuse can potentially degrade sensor-originated DQ. The sensor DQ component is composed by two subcomponents: (i) Sensor DQ assessment subcomponent and (ii) Sensor DQ enhancement subcomponent.

The **Sensor DQ assessment subcomponent** has the objective of assessing DQ for data-collected by sensors. Much of the sensor DQ assessment can be performed on sensor data for assessing DQ according to the dimensions mentioned in the session 2. Particularly, the dimensions of believability (comparison

with the correct operating bounds), completeness (missing values), free-of-error (erroneous values), consistency (over time), timeliness (delay), accuracy (deviation from true value) and precision (granularity of readings) are all important aspects of high-quality sensor data [4]. It is necessary that the sensor data well represents the events that originated the data. It implies that the data collected by multiple sensors should be processed through data fusion techniques [3] while maintaining the data consistency when representing the underlying phenomenon that originated such data.

The **Sensor DQ enhancement subcomponent** aims at the enhancement of sensor-originated DQ. Since the sensor data is constantly being integrated by data fusion procedures, it is important to perform DQ enhancement on the fly, as the data is being collected and processed. The on-the-fly data processing avoid erroneous and low quality data to propagate on fusion procedures. Also, after data fusion procedures, applying well defined DQ enhancement procedures can avoid the production of low quality data. According to [5] there are five major DQ enhancement techniques: *outlier detection*, *interpolation*, *data integration*, *data deduplication* and *data cleaning*.

(i) Outlier detection helps to improve the overall quality of datasets by making them more consistent. Moreover, outlier detection is concerned about handling instances of the unreliable datasets. Metrics used in outlier detection techniques focus on enhancing the difference between data values in order to identify outliers. (ii) Interpolation consists of inferring missing values based on other (available) values. Missing values represent gaps in available data about a certain entity or phenomena of interest for the user. As knowledge deriving processes use these datasets as input, these gaps could also lead to incomplete knowledge or wrong decisions which means that missing values could lead to a decrease in DQ. (iii) Data integration is important since social and sensor data come from different sensing platforms and different environments. In order to be used, these data need to overcome their structure differences and inconsistencies to become truly beneficial for the various services. Data integration solutions mainly focus on resolving the inconsistencies between the various data streams. (iv) Data deduplication is a data compression mechanism aiming to reduce data handling's resources consumption by reducing the amount of available data through removing of duplicate data items and replacing them with a pointer to the unique remaining copy. Data deduplication is quite simply a removal process of redundant data items. (v) Data cleaning is a process composed of 3 main phases: (i) Determination of error types, (ii) Identification of potential errors and (iii) the correction of identified (potential) errors

4 Conclusion and Future Works

In this work we presented the problem related to DQ when dealing with social and sensor data standardization. In this work we defined that standardizing DQ is making the data conforms certain DQ requirements. Also, we presented and explored a framework for social and sensor DQ standardizing. The framework

we presented has the primary objective of standardizing social and sensor DQ according to an application or system DQ requirements. The proposed framework has two components: social DQ and sensor DQ components. Social DQ component is responsible for standardizing social-originated data according to DQ requirements, while Sensor DQ component is responsible for standardizing sensor-originated data according to DQ requirements. Also, each component is composed by two subcomponents: DQ assessment and DQ enhancement subcomponents. DQ assessment subcomponent is responsible for the DQ evaluation according to given DQ requirements. DQ enhancement subcomponent is responsible for enhancing DQ to a given DQ requirement.

For the realization of a framework for social and sensor DQ standardizing, a future work is to systematically study the DQ requirements for different types of applications that deal with social and sensor data. This future work aims at directing the research for solving DQ standardizing problems. Another future works that directs solutions for the problem of DQ standardizing are related to the development of techniques for assessing and enhancing sensor and social DQ aspects. We direct future works to create techniques to assess and enhance DQ considering the diverse social and sensor data inputs.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks **54**(15), 2787–2805 (2010)
2. Czernek, A.: Social measurement depends on data quantity and quality - tech report (2015)
3. Farias, C., Pirmez, L., Delicato, F., Carmo, L., Li, W., Zomaya, A.Y., de Souza, J.N.: Multisensor data fusion in shared sensor and actuator networks. In: Information Fusion (FUSION), 2014 17th International Conference on. pp. 1–8. IEEE (2014)
4. Ferreira, E., Ferreira, D.: Towards altruistic data quality assessment for mobile sensing. In: Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers. pp. 464–469. ACM (2017)
5. Karkouch, A., Mousannif, H., Al Moatassime, H., Noel, T.: Data quality in internet of things: A state-of-the-art survey. Journal of Network and Computer Applications **73**, 57–81 (2016)
6. Labouseur, A.G., Matheus, C.C.: An introduction to dynamic data quality challenges. Journal of Data and Information Quality (JDIQ) **8**(2), 6 (2017)
7. Nakamura, E.F., Loureiro, A.A., Frery, A.C.: Information fusion for wireless sensor networks: Methods, models, and classifications. ACM Computing Surveys (CSUR) **39**(3), 9 (2007)
8. for Standardization/International Electrotechnical Commission, I.O., et al.: Software engineering-software product quality requirements and evaluation (square) data quality model. ISO/IEC **25012**, 1–13 (2008)
9. Tarasconi, F., Farina, M., Mazzei, A., Bosca, A.: The role of unstructured data in real-time disaster-related social media monitoring. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 3769–3778 (Dec 2017). https://doi.org/10.1109/BigData.2017.8258377

10. Wang, R.Y., Strong, D.M.: Beyond accuracy: What data quality means to data consumers. Journal of management information systems **12**(4), 5–33 (1996)
11. Yerva, S.R., Jeung, H., Aberer, K.: Cloud based social and sensor data fusion. In: 2012 15th International Conference on Information Fusion. pp. 2494–2501 (July 2012)

Volcanus: using information fusion concepts to perform data quality assessment and enhancement on IoT data (Poster)

Gabriel Rodrigues Caldas de Aquino, Luci Pirmez, Claudio Miceli de Farias
Postgraduate Program in Informatics (PPGI)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, Brazil
gabriel@labnet.nce.ufrj.br, luci.pirmez@gmail.com, claudiofarias@nce.ufrj.br

***Abstract*—The Internet of Things (IoT) is composed of a heterogeneous set of devices connected through the Internet. In this scenario the set of different phenomena that can take place in a monitored area is unknown and the set of devices deployed in the monitored area is heterogeneous. Hence, it is not guaranteed that a real-world phenomenon is observed and interpreted in the same way by the IoT devices. This affects the way the IoT applications interpret the data from IoT devices, this affects the confidence in which the application has on IoT data. Confidence represents how much trust an application put in data and is a data quality semantic aspect. In this context, "data quality" can be defined as how well the IoT data corresponds to the requirements of IoT applications. Hence, a major challenge in this scenario is to enable IoT data to meet the data quality requirements of the IoT applications. In this work, we propose Volcanus, which is a data quality assessment and enhancement technique whose objective is to enable the IoT data to meet the confidence requirements of the IoT applications. This work uses Information Fusion concepts to perform data quality enhancement by evaluating the mean and skewness of the environment sampled data. In our experiments, we evaluated the applicability of Volcanus in a real IoT environment.**

***Index Terms*—Internet of Things, Information Fusion, Data Quality**

I. Introduction

The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances and other smart devices. Such smart devices are embedded with electronics, software, sensors, actuators, and connectivity to the Internet which enables these things to be interconnected to exchange data. A characteristics of the IoT paradigm is the ubiquitous presence of smart devices [1] [1] [2] [3]. The deployment of the IoT smart devices is on a global scale. Hence, the IoT is composed of a heterogeneous set of devices scattered globally [4].

In this scenario the set of different phenomena that can take place in a monitored area (e.g. failures, a fire occurrence or the unwanted people access) is unknown and the set of devices deployed in the monitored area is heterogeneous [5]. Thus, in this scenario, it is not guaranteed that the occurrence of a real-world phenomenon is observed and interpreted in the same way by each smart device that composes the heterogeneous set of IoT devices. A phenomenon denotes any situation on the monitored area that generates data in a specific range that may or may not have a correlation to an event of interest for an application in the real world [5] (e.g. an overheating on a certain part of the monitored area, which generates data in a specific range). As more different phenomena generate data on the monitored area, more the confidence in which the data represents a real phenomenon in the world is affected [6]. This affects the way the IoT applications receive and process the data according to its data requirements and application states. In this case, the application is prone to make wrong decisions and environment actuations.

Confidence represents how much trust an application put in data. According to [6], the confidence requirement of an IoT application can be defined as a statistical error ε such that an interval $[X-\varepsilon, X+\varepsilon]$ contains the real value X with a confidence probability of p. The different IoT applications may present different confidence probability p values and statistical error ε. In this context, *data quality* can be defined as how well the IoT data corresponds to the requirements of IoT applications [7] [6], and confidence is an intrinsic data quality semantic aspect [6]. For an IoT application, the correct interpretation of the IoT data is a key concern. The incorrect interpretation of the IoT data may lead to incorrect decisions and actuation by the IoT applications [5]. Processing the IoT data for improving the representation of one or more phenomena that take place in a monitored area is a challenge. The improvement of the data representation implies in an improvement in the data quality aspects of the data for a given application, since the better is the data representation, the greater is the confidence about the correct representation of the data for one or more phenomena in a monitored area [6] [7] [8] [9].

A major challenge in this scenario is to enable IoT data to meet the data quality requirements of the IoT applications. The work [10] proposes a framework for data quality assessment and enhancement on IoT context. In such case, the data is processed by a data quality assessment mechanism for assessing data quality according to a given data quality dimension (e.g.

completeness, timeliness, confidence, etc.) [6]. And, also, the data is processed by a data quality enhancement mechanism, whose objective is to avoid erroneous and low-quality data to propagate to fusion procedures. Hence, avoiding potential erroneous decisions to be taken by IoT applications.

In this work, we propose Volcanus, which is a data quality assessment and enhancement technique [10] whose objective is to enable the IoT data to meet the confidence requirements of the IoT applications. Volcanus has two modules: (i) a data quality assessment module that assess the sensor collected data quality to verify if it meets the data quality requirements of the IoT applications and (ii) a data quality enhancement module that process the data to meet the confidence requirements of the IoT applications.

Volcanus is a data quality assessment and enhancement technique based on information fusion concepts. The authors of [5] proposed an information fusion technique capable of inferring features about the different phenomena that take place in the monitored area. We use the concepts on [5] since it performs a peak analysis to analyse the data collected by sensors to infer about the set of phenomena that are taking place on the monitored. Since as more different phenomena generate data on the monitored area, more the confidence in which the data represents a real phenomenon in the world is affected by [5]. By executing a peak analysis, Volcanus aims at increasing the data confidence by separating the data from multiple different phenomena. In general, Volcanus receives the different p confidence probabilities and ε statistical error for each application. Then, evaluates the data quality aspects of the collected data to verify if it meets the required confidence probability of each application. And, depending on the result of the data analysis, Volcanus enhance the IoT data quality to meet the requirements of the IoT applications.

Volcanus executes a heuristics that analyses the confidence requirements and the received IoT data to assess if the data meets the confidence requirement of the IoT application. To enhance the data quality of the IoT data, Volcanus analyse the mean and skewness [11] [12] [13] of the collected data to perform its data quality enhancement. We propose the use of these statistics due to its simplicity to be implemented on a sensor node and their low computational cost.

The rest of this work is divided as: in Section II we present the data quality concepts. In section III we present the literature review. In section IV we present Volcanus. In section V we provide a discussion about a data quality management framework. In section VI we present the experiments conducted to evaluate Volcanus behaviour. And, finally, in section VII we present the conclusions and the future works.

II. Data Quality Concepts

In this chapter, we briefly discuss data quality concepts. Data is an important asset in the IoT paradigm. Data is the source of information for context-based decision making. In this context, quality is deemed as a critical requirement for decision-making mechanisms, applications and services on the IoT.

In the context of this work *data quality* can be defined as a concept that refers to how well the data corresponds to the necessities of data processing mechanisms [7]. An interesting fact comes from the aforementioned definition: data quality refers to the fitness of which data is perceived by its consumer. This means that data quality would hardly be seen in the same way by different users. In fact, each data consumer requires the used data to fulfil certain criteria which he presumes essential for his own tasks at hand. The problem of data quality assessment is commonly addressed through data quality dimensions [14].

The ISO international standard data quality model identifies several data quality characteristics in the context of Software Engineering [8]. According to [9], the dimensions of data quality data can be categorised into four semantic aspects: (i) *intrinsic* , (ii) *accessibility*, (iii) *contextual*, and (iv) *representational*.

The (i) *intrinsic data quality semantic aspect* is related to the quality of the data in relation to itself. As examples of dimensions, there are: accuracy, objectivity, believability and reputation [9]. (ii) The *accessibility data quality semantic* dimensions describe how accessible the data is for data consumers. Examples can be accessibility and access security [9]. *contextual data quality semantic* aspect is related to how appropriate the data is for its usage. As examples of dimensions can be relevancy, value-added, timeliness, completeness and amount of data [9]. *representational data quality semantic* aspect describes how understandable and representative of the environment the data is. Examples of dimensions are interoperability, ease of understanding, concise representation and consistent representation [9].

Data quality challenges refer to difficulties affecting any data quality dimension. Such challenges can cause data to become entirely or partially unusable, since they may not meet the requirements of data consumers. As we discussed, data quality does not relate to problems only related to data accuracy. Instead, data quality problems can surpass the accuracy dimension to other dimensions such as the aforementioned.

In this work, we propose Volcanus as a mechanism whose objective is to enhance the quality of the IoT data to meet the confidence requirements of the IoT applications. Confidence is an intrinsic data quality semantic aspect dimension. Confidence can be defined as a statistical error such that $[X-\varepsilon, X+\varepsilon]$, contains the real value X with a confidence probability of p [6]. As an example, the authors [6] present a dataset of values $[20, 25, 23, 19.5, 18.2, 28, 22.2, 18.4, 16.5]$, in which for a confidence probability of $p = 99\%$, the confidence ε is 3.47.

III. Literature Review

This section presents works that deal with data quality concepts. Several works in literature deal with the concepts related to data quality challenges: [7], [6], [14] and [8], which is an International Organisation for Standardisation (ISO) data quality model that identifies several data quality characteristics in the context of Software Engineering.

The work [6] surveys data quality in the context of IoT. The authors enumerate IoT environment-related factors affecting the data quality and analyse the impact of these factors on various data quality dimensions. Also, the authors identify in what form do data quality problems manifest and associate each type of manifestation with its symptoms with respect to the affected data quality dimensions. The survey presents a study regarding data outliers as a major type of data quality problems. The authors [6] define the underlying concept about outliers, enumerate the different types of outliers and analyse their impact on a large scale deployment and acceptance of the IoT paradigm. Also, the authors investigate the impact of outliers by IoT applications. The author also presents and study techniques for enhancing data quality and overcoming data quality problems.

Differently, from [6], which is a data quality survey, the work [7] presents an introduction to the challenges related to dynamic data quality. According to the authors [7], since the data in the context of the Internet of Things (IoT) dynamically changes in content and structure, data quality becomes dynamic since it relates to characteristics of data itself. The work [14] present data quality in the context of mobile sensing. The authors identified an opportunity to benefit from sharing of domain knowledge and methods for data quality assessment in mobile sensing. The authors of [14] argue that we should not focus on mere quantity of sources of data, but instead on the quality of the sensors and the data they produce.

The works [7] [6] [14] are important to introduce the data quality concepts in a deep view in different aspects. Differently from [7] [6] [14], which are conceptual works, our work presents a data quality assessment and enhancement technique. To the best of our knowledge, there is no work on data quality mechanisms to enhance and assess the IoT data quality.

IV. Volcanus

In this section, we present and describe our proposal: Volcanus, which is a data quality assessment and enhancement technique (DQAET) for the internet of things [7], [6] [14] whose objective is to enable the IoT data to meet the confidence requirements of the IoT applications. Volcanus is composed by two modules: a **data quality assessment module** (DQAM) and a **data quality enhancement module** [10] (DQEM). Both DQAM and DQEM can run independently in different nodes or in a single node.

A. Volcanus Operation

Volcanus operates through successive cycles of receiving data and application requirements, assessing the data quality and enhancing data quality, called execution cycles. Prior to starting the execution cycles, Volcanus executes a Set-Up procedure where it populates its data structures with default values and cleans the memory space. Volcanus operation is illustrated in Figure 1.

An execution cycle starts by receiving data and application requirements. Then, the DQAM executes a heuristics that evaluates if a given data meets the quality requirements of the IoT applications. In this work, the DQAM is proposed to evaluate the data quality confidence dimension. The data quality confidence requirement is defined as a statistical error ε such that an interval $[X-\varepsilon, X+\varepsilon]$ contains the real value X with a confidence probability of p [6]. In general words, DQAM receives from each α IoT application a tuple $\tau_\alpha = (\varepsilon, p)$ for the data quality requirement of the α IoT application. Then DQAM applies a heuristics on the data to evaluate if the data meets the τ_α requirements. If the data meets the data quality requirements, it is provided to the applications flagged as data with the desired quality. Else, Volcanus executes the DQEM.

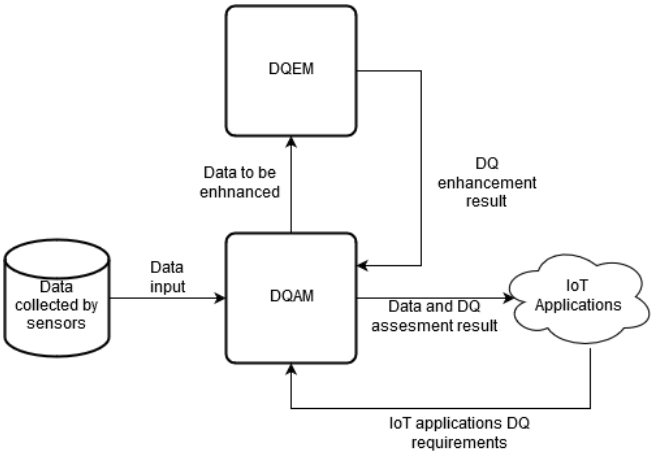

Fig. 1. Volcanus Operation

The DQEM receives the data that does not meet the data quality requirements and executes an information fusion procedure to enhance the data quality. The DQEM integrates the collected data and infers data features. The inferred features are used to separate data generated by different phenomena. By separating the different data, DQEM aims at decreasing the range of the confidence interval of the estimated data parameter. After executing the DQEM, the generated data is given as input to the DQAM. The DQAM evaluates if the generated data meets the quality requirements of the IoT applications. In the case of the data has been already processed by DQEM and it still does not meet the requirements of the IoT applications, it is flagged as data with undesired quality. Volcanus delivers the data to the applications and the IoT applications decide to use such data given the flagged data quality.

B. Algorithm

In this section, we present the algorithm of Volcanus. Since Volcanus is composed of two modules, we describe two algorithms: the Data Quality Assessment Module (DQAM) of Volcanus and Data Quality Enhancement Module (DQEM) of Volcanus.

1) DQAM: Volcanus starts by receiving the IoT applications requirements and the IoT data. The DQAM receives as input the IoT applications data quality requirements(statistical error and the confidence probability) and the environment

collected data. The received data is stored with a Boolean value as a flag (values 1 or 0). This flag is set as default to 0, which indicates that the data has not yet passed through DQEM.

After receiving the IoT applications data quality requirement and the collected data, the DQAM calculates the confidence interval of the mean of the received data using the confidence probability in *p* required by the IoT application and stores it in memory. Then DQAM verifies if the range of the confidence interval is less or equal than the double of the statistical error demanded by the IoT application data quality requirement.

If the range is less or equal than the double of the statistical error, then the data meets the data quality requirements of the IoT application. In this case, DQAM returns the data to the IoT application informing that this data meets the data quality requirements of the IoT application. This is done by returning the data along with a *label informing the data meets the data quality requirements of the IoT application*.

Else, if the range is greater than the double of the statistical error, then the data does not meet the data quality requirements of the IoT application. In this case, DQAM should send the data to the DQEM. However, to prevent that the data was not previously sent to the DQEM, DQAM verifies if the Boolean flag of the received data is set to 0. DQAM verifies if the Boolean flag is set to zero since it is its default value of when the data is received by Volcanus after the data collection from the environment.

If the Boolean flag of the received data is set to 0, DQAM sends the data to the DQEM. Else, If the Boolean flag of the received data is not set to 0, the data has already been sent to the DQEM and there was no effective enhancement on the data. In this case, the data does not meet the data quality requirements and DQEM was not capable of enhancing the data quality to meet the data quality requirements of the IoT application. In this case, DQAM returns the data to the IoT application informing that this data does not meet the data quality requirements of the IoT application. This is done by returning the data along with a *label informing the data does not meet the data quality requirements of the IoT application*.

2) DQEM: The DQAM sends the received data to the DQEM if the data does not meet the data quality requirements of the IoT application. Hence, the input of DQEM is the collected data. The first step of DQEM is to calculate the mean of the collected data. Then, DQEM calculates the skewness of the collected data. When skewness is less or equal than 1 there is a single phenomenon or multiple phenomena generating data with similar data ranges in the monitored area environment [5]. Else, if skewness is greater than 1, then there are multiple phenomena occurring in the environment [5]. By separating the multiple phenomena, DEQM improves the confidence data quality semantic aspect.

If the skewness is less or equal than 1, DQEM was not capable of enhancing the data quality for the data. In such case, DQEM sends the collected data to the DQAM with the boolean flag value of 1 informing to the DQAM that this data has already been sent to DQEM.

Else, if the skewness is greater than 1, DQEM divides the collected data into two pieces of data. DQEM divides the data into less or equal than the average of the collected data and the data greater than the average of the collected data. After dividing the collected data, DQEM sends the data less or equal than the average of the collected data and the data greater than the average of the collected data to the DQAM with a boolean flag value set to 0. DQEM sends the data with the with a boolean flag set to 0 because when it divides the collected data into two, it creates two new pieces of data that should have its data quality assessed.

After executing DQAM and DQEM, Volcanus sends the data to the IoT application along with the quality assessment result. Then the applications receive the data and decide about the use of the data.

V. Discussion: data quality management framework

The objective of Volcanus is to enable the IoT data to meet the confidence requirements of the IoT applications. The realisation of Volcanus is demanding as a part of a data quality management framework whose objective is to improve the overall quality of the IoT data in face of the potential IoT challenge. According to [6] there are several challenges in IoT context such as: (i) high heterogeneity in IoT data sources; (ii) the resource-constrained nature of IoT devices in terms of power and processing capabilities; (iii) the eventual lack precision and loss of calibration; (iv) The maintenance of sensor devices in harsh environments; and (v) vandalism; potentially introduce uncertainty about the data, which lowers the confidence about the IoT data.

In this work, Volcanus is proposed aiming at a specific challenge: the set of different phenomena that can take place in a monitored area (e.g. failures, a fire occurrence or the unwanted people access) is unknown and the set of devices deployed in the monitored area is heterogeneous. However, for the full realisation of a data quality management framework data has to be improved through the assessment and enhancement of data nonconformities that arise from any IoT infrastructural challenges.

To make this scenario feasible, the data quality assessment and enhancement mechanisms have to be set after the data collection for providing information to the IoT application and enabling the control of the quality of the IoT data. Such quality control can be done by mechanisms for promoting the investigation of the source of data and the actuation to improve the quality of the data. This concept is illustrated in Figure 2. The realisation of this concept is a network capable of managing and controlling the quality of its data sources. In Figure 2 the actuation in the IoT network for controlling the data quality is performed by four mechanisms: (i) a data collection mechanism; (ii) a data quality assessment mechanism; (iii) a source investigation mechanism and (iv) a network actuation mechanism.

The (i) data collection mechanism is performed by the network sensors in a monitored area (e.g a temperature sensor

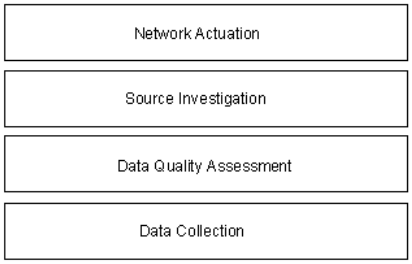

Fig. 2. Data quality assessment and enhancement for IoT

in a room collecting temperature samples with a given rate). The (ii) data quality assessment mechanism is performed by data processing mechanisms responsible for receiving the data collected by the IoT sensors and processing it for performing the assessment of the data quality aspects. In Figure 2, Hygieia is a (ii) data quality assessment mechanism. The (iii) source investigation mechanism is responsible for detecting the source of the data that generated data with a given quality (e.g discovering a source of outlier data) and investigating the reason for the generation of data with the given quality (e.g. this source is generating outlier data, because this room is on fire). And the (iv) network actuation mechanism is responsible for actuating in the network given the reason investigated previously (e.g. since a given area is on fire, the network should change the data collection rate and send all raw data directly to the gateway node).

VI. Experiments

To evaluate the applicability of Volcanus in a real IoT environment, in this section we present the experiments conducted with Volcanus with the purpose of evaluating Volcanus requirements in terms of memory (RAM and ROM) usage and the impact of Volcanus in the delay imposed on the data delivery to the applications due to its data processing. All these properties are evaluated in real sensor nodes. In this section we first we describe the environment configuration, then application scenario, and the metrics used in our experiments. Finally, we discuss the obtained results.

A. Environment configuration

The experiments considered a heterogeneous sensor network composed of smart sensor nodes. We considered a heterogeneous sensor network composed of 10 collector nodes (CN), 1 data processing nodes (DPN) and one sink node (SN). The CN is responsible for collecting data from the environment using its embedded sensors. The DPN is responsible for receiving the data collected by the CN and executing Volcanus on such data. The sink node is the gateway of the network. After executing Volcanus, the DPN sends the result to the SN. We have used a Raspberry Pi 3 Model B as DPN. A Raspberry Pi 3 Model B presents a 1.2 GHz quad-core ARM Cortex A53 CPU with ARMv8 Instruction Set and 1 GB LPDDR2-900 SDRAM. The operating system installed in Raspberry Pi was Raspbian. We implemented Volcanus using Python programming language. We used Python version 3. IEEE 802.15.4 is used as the technical standard for the operation of low-rate wireless personal area networks (LR-WPANs) to transfer data between CN and DPN. The Raspberry Pi node uses an XBee Explorer Dongle for USB interfaces. All experiments lasted 1 hour and each of them was repeated 30 times with a 99% confidence interval. The fusion window was set into 10 samples.

B. Network scenario and metrics

For the network scenario, we have chosen a typical Smart Grid [15] scenario. We have chosen the Smart-Grid scenario to evaluate the applicability of Volcanus in a real IoT environment since a smart grid is an integrated array of grid technologies, devices, and control systems in the IoT context. Also, Smart-Grids provide and utilise digital information, communications, and controls to optimise the efficiency, reliability, and security of electric power delivery. A transmission tower is a tower used to support an overhead power line. An overhead power line is an electric power transmission line suspended by towers. The overhead power-lines are the most common means to transmit energy. In the Smart Grid context, the transmission tower is also responsible for storing energy using local batteries.

The metric used in the performed experiments for evaluating the impact of Volcanus in the delay imposed on the data delivery to the applications is the total delay imposed by Volcanus. The total delay is measured as the time elapsed for Volcanus to execute its procedures.

C. Evaluating Volcanus in terms of resource consumption and delay

This section describes the performed experiments for comparing Volcanus and MAF in terms of resource consumption and delay imposed.

Regarding the values for the delay imposed, we run Volcanus and MAF in DPN. The delay imposed by MAF was of 36.14 ± 7.61 microseconds while the delay imposed by Volcanus was of 1201.14 ± 45.87 microseconds. This delay increased imposed by Volcanus is due to its procedures.

Regarding the values of memory usage, we run Volcanus and MAF in a DPN. In the Volcanus context, DPN uses $99.77MB \pm 0.18$ of RAM memory while in the MAF context, DPN uses $99.29MB \pm 0.28$ of RAM memory. It is important to notice that these values represent the total of ram memory consumed by the raspberry pi with the operating system. Nevertheless, we can conclude that Volcanus is feasible to be employed in IoT scenario.

VII. Conclusions and Future Works

In this work, we proposed Volcanus: a data quality assessment and enhancement technique whose objective is to enable the IoT data to meet the confidence requirements of the IoT applications. We proposed Volcanus as a data quality assessment and enhancement technique based on information

fusion concepts. Volcanus aims to increase data confidence by separating the data from multiple different phenomena. Volcanus has two modules: (i) a data quality assessment module that assess the sensor collected data quality to verify if it meets the data quality requirements of the IoT applications and (ii) a data quality enhancement module that process the data to meet the confidence requirements of the IoT applications. In our experiments we evaluated the applicability of Volcanus in a real IoT environment, in this section we present the experiments conducted with Volcanus with the purpose of evaluating Volcanus requirements in terms of memory (RAM and ROM) usage and the impact of Volcanus in the delay imposed on the data delivery to the applications due to its data processing.

In this work, we did not evaluate the Volcanus capability of assessing and enhancing data quality in order to increase the accuracy of an IoT application. As future works, we aim at evaluating Volcanus in real application scenarios. Also, we aim at expanding this work by evaluating the behaviour of Volcanus with different applications of data quality requirements in the experimentation scenario of this work. Also, we aim at proposing different data quality assessment and enhancement techniques that deal with other data quality aspects, not just the confidence semantic aspect. Another future work is the realisation of the data quality management framework as discussed in section V.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] A. Whitmore, A. Agarwal, and L. Da Xu, "The internet of things—a survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[4] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of manet and wsn in iot urban scenarios," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3558–3567, 2013.

[5] G. Aquino, L. Pirmez, C. M. de Farias, F. C. Delicato, and P. F. Pires, "Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks," in *2016 19th International Conference on Information Fusion (FUSION)*, pp. 59–66, July 2016.

[6] A. Karkouch, H. Mousannif, H. Al Moatassime, and T. Noel, "Data quality in internet of things: A state-of-the-art survey," *Journal of Network and Computer Applications*, vol. 73, pp. 57–81, 2016.

[7] A. G. Labouseur and C. C. Matheus, "An introduction to dynamic data quality challenges," *Journal of Data and Information Quality (JDIQ)*, vol. 8, no. 2, p. 6, 2017.

[8] I. O. for Standardization/International Electrotechnical Commission *et al.*, "Software engineering-software product quality requirements and evaluation (square) data quality model," *ISO/IEC*, vol. 25012, pp. 1–13, 2008.

[9] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of management information systems*, vol. 12, no. 4, pp. 5–33, 1996.

[10] G. R. C. de Aquino, C. M. de Farias, and L. Pirmez, "Data quality assessment and enhancement on social and sensor data," 2018.

[11] M. H. DeGroot and M. J. Schervish, *Probability and statistics*. Pearson Education, 2012.

[12] D. Joanes and C. Gill, "Comparing measures of sample skewness and kurtosis," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 47, no. 1, pp. 183–189, 1998.

[13] K. Pearson, "Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material," *Philosophical Transactions of the Royal Society of London*, vol. 186, no. Part I, pp. 343–424, 1895.

[14] E. Ferreira and D. Ferreira, "Towards altruistic data quality assessment for mobile sensing," in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pp. 464–469, ACM, 2017.

[15] A.-A. Edris and B. W. D'Andrade, "Transmission grid smart technologies," in *The Power Grid*, pp. 37–55, Elsevier, 2017.

Hygieia: Data Quality Assessment for Smart Sensor Network

Gabriel R. Caldas de Aquino
Universidade Federal do Rio de Janeiro - PPGI - Rio de Janeiro - Brazil
gabriel@labnet.nce.ufrj.br

Claudio M. de Farias
Universidade Federal do Rio de Janeiro - PPGI - Rio de Janeiro - Brazil
claudiofarias@nce.ufrj.br

Luci Pirmez
Universidade Federal do Rio de Janeiro - PPGI - Rio de Janeiro - Brazil
luci.pirmez@gmail.com

ABSTRACT

Smart Sensor Networks (SSN) are the core data transmission infrastructure of the Internet of Things. However, SSN devices are resource-constrained and the environment in which the SSN is deployed impose data quality challenges. Data quality refers to the fitness of the data for the data consumer. In this case, mechanisms to assess the data quality for IoT data in network are demanded. In this work we propose Hygieia: a data quality assessment mechanism for constrained SSN devices. Hygieia is designed to impose low memory overhead, low network communication overhead and to not impose a significant delay from its data input to its data quality assessment output. We conducted real node experiments.

CCS CONCEPTS

• **Information systems** → **Information integration**; • **Networks** → **Sensor networks**; • **Computer systems organization** → *Sensors and actuators*;

KEYWORDS

Smart Sensor Networks, Data Quality, Internet of Things

ACM Reference Format:
Gabriel R. Caldas de Aquino, Claudio M. de Farias, and Luci Pirmez. 2019. Hygieia: Data Quality Assessment for Smart Sensor Network. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/3297280.3297564

1 INTRODUCTION

One of the core components of the Internet of Things (IoT) paradigm are the smart sensor networks (SSN) [3] [2] [4]. SSN devices are often placed in harsh environments, susceptible to malfunctioning, interferences or malicious behaviors. Dysfunctional devices introduce uncertainty about the data, which lower data confidence. The challenge is to evaluate if the sensed data corresponds to the necessities of IoT data consumers [12] [11]. This concept is known as the data quality (DQ) of IoT [11].

A DQ challenge refer to difficulties affecting any DQ dimension. DQ challenges can cause data to become entirely or partially unusable, since it may not meet the requirements of data consumers. The presence of outliers in IoT data is a common manifestation of DQ challenges in several dimensions [11] [12] [10]. Outliers can be a product of challenges in SSN context (e.g. a dysfunctional sensor) or can be a product of a legit event in the monitored area (e.g. the detection of a fire).

DQ assessment in IoT context is a challenge. In this work we propose Hygieia: a DQ assessment mechanism for constrained devices in SSNs, whose objective is to assess IoT data with information about its quality to provide DQ information to IoT applications. Hygieia is proposed as a simple DQ assessment technique, to impose low energy consumption, low memory overhead and inconsiderable delay while performing the DQ assessment. In an overview, Hygieia evaluates whether a newly received data is an outlier in face of the historical data received. A data source can be a single device or a group of sensors. This work implemented Hygieia using the interquartile range (IQR) statistical approach to detect outliers. IQR is a well-known statistical outlier detection approach, presents low computational overhead and well suits the IoT devices. Hygieia is not aware about the nature of the data, the output of Hygieia can be provided as input for any data consumer.

In our experiments we evaluate Hygieia in a SSN scenario with real nodes. The rest of this paper is divided as the following: Section 2 presents Hygieia, our proposal; Section 3 presents the experiments we performed for evaluating Hygieia; and finally on section 4 presents the conclusions and future works.

2 PROPOSAL

In this section we present and describe Hygieia, a DQ assessment mechanism for SSN [12], [11] and [9]. Hygieia is designed to impose low memory overhead, communication overhead and not impose a significant delay from its data input to its DQ assessment output. SSN devices present restricted memory and energy resources [2], also, delays in data processing degrades data timeliness. In an overview, Hygieia receives a continuous flow of data and evaluates the interquartile range (IQR) of the data, while performing its DQ procedures. IQR is a simple statistics and takes constant time to be performed [6]. The IQR is the difference between the first and third data quartiles. Hygieia data is stored in the Random Access Memory (RAM) of the network nodes, in which the corresponding values for its quartiles can be directly accessed. The IQR can be calculated to any amount of data in RAM memory and Hygieia is designed to present small executable size. Hygieia is meant to impose low communication overhead.

2.1 Hygieia

Hygieia receives a data input, stores it in device memory to be further analyzed. Hygieia evaluates if there is historical data to be analyzed. If there is no historical data, Hygieia waits for new data, else Hygieia sorts the data and calculates the 25 (P25) and 75 (P75) percentiles. Hygieia calculates the interquartile range (IQR)

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04.
https://doi.org/10.1145/3297280.3297564

by simply executing the difference between the data in 75 and 25 percentiles. After calculating the IQR, Hygieia calculates the lower boundary and the upper boundary. Since a Normal Distribution presents its peak of data within its quartiles [6], outliers in this work are defined as observations that fall below $P25 - 1.5 * IQR$ or above $Q3 + 1.5 * IQR$. Hence, Hygieia uses a factor of 1.5 for multiplying IQR.

Hygieia analyzes if the received data is greater than the upper boundary or smaller than the lower boundary. If the data received is lower or greater than the boundaries, it is considered as outlier. Else it is considered as non-outlier. Hygieia deletes the oldest data from sensor memory and continues its operations. This step is important, for updating Hygieia as the environment changes. By always deleting the oldest data, Hygieia is always renewing its understanding about the environment. The output of Hygieia is a consolidated report of DQ, informing the outlier data, as well the amount of outlier data. This report can be provided as input to a data fusion algorithm for making a decisions or to a data analytics method to perform data analytics. In this case, an application has the possibility to react in a distinct manner in relation to non-outlier data. To make this scenario feasible, Hygieia is set after the data collection. Such quality control can be done by mechanisms for promoting the investigation of the source of data and the actuation to improve the quality of the data. The realization of this concept is a network capable of managing and controlling the quality of its data sources. The actuation in the IoT network for controlling the DQ is performed by data collection mechanisms, DQ assessment mechanisms, source investigation mechanisms and network actuation mechanisms.

The data collection mechanism is performed by the network sensors in a monitored area. The DQ assessment mechanism is performed by data processing mechanisms (i.e. Hygieia). The source investigation mechanism is responsible for detecting the source of the data that generated data with a given quality and investigating the reason for the generation of data with the given quality. And the network actuation mechanism is responsible for actuate in the network given the reason investigated previously. In an hypothetical network, Hygieia can be installed in nodes that collect or route data for detecting and discovering the causes of outlier data. By applying mechanisms to actuate in network in response to an occurrence of an event, the network is able to perform the assessment and the actuation in the network for self improving the quality of the generated data.

3 EXPERIMENTS

In this section we present the experiments in which we evaluate Hygieia accuracy in terms of correct DQ assessment process, Hygieia requirements in terms of memory (RAM and ROM) usage and energy consumption and the delay imposed by Hygieia in its DQ assessment process. All these properties are evaluated in real sensor nodes. The experiments considered a an heterogeneous SSN composed by 3 Arduino UNO nodes and one Raspberry Pi node. We used the 3 Arduino nodes as collector nodes (CN) and the Raspberry Pi as a fusion node (FN). The CN are responsible for collecting data and performing Hygieia, while the FN is responsible for receiving data from CN and taking the applications decisions. The CN used in this experiments abstracts the origin of the data since Hygieia is designed to assess DQ from generic data sources.

The Arduino UNO is an open-source board [13], which is a low-power CMOS 8-bit microcontroller, presents 32 KB of Flash Memory, 2 KB of volatile SRAM and 1 KB non-volatile EEPROM. The clock speed of Arduino UNO processor is 16 MHz. The Raspberry Pi presents a 1.2 GHZ quad-core ARM Cortex A53 CPU with ARMv8 Instruction Set and 1 GB LPDDR2-900 SDRAM. The network uses the IEEE 802.15.4 as the standard for the operation of low-rate wireless personal area networks (LR-WPANs). The Arduino UNO board uses an XBee shield while the Raspberry Pi node uses an XBee Explorer Dongle for USB interfaces. Hygieia was developed using C programming language. All real node experiments lasted 1 hour and each of them was repeated 30 times with a 99% confidence interval. The distance between each CN and FN node was 1 meter.

We have chosen a Smart Grid [8] application called Overhead Power Line Monitoring Application (OPLM) as application scenario [5] [1]. The OPLM application has great importance since the knowledge about the power line temperature is necessary for making decisions about its loading, which could damage the lines causing operational failures, leading to blackouts. In our experiments the temperature of the Overhead Power Line is normal, hence the Overhead Power Line is in a safe condition. The OPLM application evaluates if the line is in ideal condition when the temperature is around 40°C - 65°C [1]. However, due to environmental harsh conditions, dysfunctional smart things are prone to failures and produce data with poor quality. The data from the CN installed in the Overhead Power Line may present degraded quality. 3 CN are installed in the Overhead Power Line. All the CNs collected temperature samples periodically in a TDMA scheme with time slot of 2000ms and transmitted them to the FN. Hygieia is installed on each CN and stores 10 sample data as historical data.

3.1 Metrics

We evaluate Hygieia in terms of: accuracy (correct DQ assessment process), memory, energy consumption and delay imposed. We performed our experiments in two scenarios: one considering Hygieia (ECH) and another do not considering Hygieia (EDCH). In ECH the DQ assessment performed by Hygieia *is considered* by the IoT applications to perform its decision making. In EDCH the DQ assessment performed by Hygieia *is not considered* by the IoT applications to perform its decision making.

The accuracy of Hygieia is evaluated according to: *False Positives (FP)*, *False Negatives (FN)*, *True Positives (TP)* and *True Negatives (TN)*. TP counts the number of situations in which the OPLM application correctly identifies OPL in ideal condition and it is actually in ideal condition. TN counts the number of situations in which the OPLM application correctly identifies OPL not in ideal condition and it is actually not in ideal condition. FP counts the number of situations where the OPLM application identifies OPL in ideal condition and it is not. FN counts the cases in which the OPLM application identifies the OPL is not in ideal condition, but it is. True occurrences are defined as the sum of TP and TN.

The *amount of transmitted bits* in the network is used as metric to evaluate the energy consumption. The use of the radio for data transmission is an energy-demanding operation. When a node has

drained out its battery, the communication and sensing coverage of the application is degraded [7]; The *memory consumption of Hygieia when installed in the Arduino* as memory consmumption metric. SSN devices are resource constrained in terms of memory. Solutions that impose much overhead over the memory resources are infeasible solutions. The *elapsed time to perform DQ assessment* is used as metric to measure if Hygieia imposes much delay while executing the DQ assessment. This metric is the difference in microseconds between the time in which Hygieia receives its data to the time in which Hygieia finishes the DQ assessment.

3.2 Results and Evaluation

The accuracy result for EDCH scenario is of 89.21% σ 0.93% and Confidence Interval (CI) (99%) 0.47 while for ECH scenario the accuracy is of 97.62% σ 1.41% and CI (99%) of 0.71. This result shows an improvement of 8.41% in the accuracy result of the ECH scenario in comparison to the results of the EDCH scenario. This improvement is given by the fact that in EDCH scenario all data is provided to the OPLM application without considering the result of Hygieia with the information about its DQ.

In both EDCH and ECH scenarios the presence of outlier data degrades the information DQ. However, since in EDCH the applications does not consider the DQ assessment of Hygieia, in EDCH the application has no information supplies to whether consider or not a degraded information in its analysis. While in ECH scenario the data is provided to the OPLM along with information about the Hygieia DQ analysis, thus the resulting shows an improvement in the accuracy results.

The aforementioned results shows that Hygieia is capable of performing the assessment of the IoT DQ for improving the accuracy of the OPLM application. The accuracy result shows the importance of considering the DQ for IoT applications in a decision making process. In this experimentation scenario, the presence of dysfunctional smart devices degrades the DQ aspects of the IoT data, which can be manifested in form of outliers. Also, IoT data is a valuable asset for IoT applications decision making. The knowledge about the DQ aspects in ECH was capable of improving the decision making in comparison to EDCH, where the DQ aspects of the IoT data was not considered.

The results for measuring the efficiency of Hygieia according to the amount of transmitted bits for EDCH is of 23705.6 bits σ 272.19 and CI (99%) of 136.98, while for ECH scenario is of 25187.2 bits σ 289.21 and CI (99%) of 145.54. We can observe that when using Hygieia, there is an increase of 5.88% of transmitted bits. This is due to meta information about the DQ aspects that Hygieia produces. In our experiments the information provided by Hygieia is sent in a one bit flag in data frame, either it is an outiler or it is not. Therefore, there is a increase of a bit for each data frame transferred in network. In this work we do not focus on implementing an optimal solution for meta information compression. By using techniques for data, this result might be improved. Also, Hygieia consumes 3454 bits of program storage space. The maximum available is 32256 bits of the Arduino memory. Hygieia presents *memory consumption* of just 10% of the memory of Arduino. This result shows that Hygieia does not impose memory overhead in Arduino platform. The experiments to assess the delay imposed by Hygieia showed that Hygieia imposes a delay of 307.07 microseconds (0.00030707 seconds) on average with standard deviation of σ11.96239318 and a Conf. Interval (99%) of 6.020013505. The results for the experiments to assess the delay imposed by Hygieia shows that Hygieia does not impose a significant overhead when it is used in comparison to when it is not used.

The experiments performed in this work indicates that Hygieia is an effective DQ assessment mechanism, capable of improving the accuracy of the applications by performing DQ assessment in comparison to the scenario where Hygieia is not used. Also, our experiments indicates that Hygieia does not impose much overhead in the network resources (energy and memory) nor Hygieia imposes a significant delay when performing its DQ assessment procedures.

4 CONCLUSIONS

In this work we proposed Hygieia, whose objective was the assessment of IoT data quality for IoT data consumers. Hygieia was designed impose low memory overhead, communication overhead and delay. We conducted experiments using real sensor nodes. Hygieia is an effective DQ assessment technique capable of improving the accuracy of applications in comparison to the scenario where Hygieia is not used. Hygieia did not impose much overhead in the network resources (energy and memory) nor Hygieia imposed a significant delay. As a future work we envision the development of actuation mechanisms using the DQ assessment result.

REFERENCES

[1] G. Aquino, L. Pirmez, C. M. de Farias, F. C. Delicato, and P. F. Pires. 2016. Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks. In *2016 19th International Conference on Information Fusion (FUSION)*. 59–66.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (2010), 2787 – 2805. https://doi.org/10.1016/j.comnet.2010.05.010

[3] Paolo Bellavista, Giuseppe Cardone, Antonio Corradi, and Luca Foschini. 2013. Convergence of MANET and WSN in IoT urban scenarios. *IEEE Sensors Journal* 13, 10 (2013), 3558–3567.

[4] Li Da Xu, Wu He, and Shancang Li. 2014. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics* 10, 4 (2014), 2233–2243.

[5] Claudio M de Farias, Luci Pirmez, Flavia Coimbra Delicato, Igor L Dos Santos, and Albert Y Zomaya. 2012. Information fusion techniques applied to Shared Sensor and Actuator Networks. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*. IEEE, 188–191.

[6] Morris H DeGroot and Mark J Schervish. 2012. *Probability and statistics*. Pearson Education.

[7] F. Z. Djiroun and D. Djenouri. 2017. MAC Protocols With Wake-Up Radio for Wireless Sensor Networks: A Review. *IEEE Communications Surveys Tutorials* 19, 1 (Firstquarter 2017), 587–618. https://doi.org/10.1109/COMST.2016.2612644

[8] Abdel-A Edris and Brian W D'Andrade. 2017. Transmission Grid Smart Technologies. In *The Power Grid*. Elsevier, 37–55.

[9] Eija Ferreira and Denzil Ferreira. 2017. Towards altruistic data quality assessment for mobile sensing. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, 464–469.

[10] International Organization for Standardization/International Electrotechnical Commission et al. 2008. Software engineering-Software product Quality Requirements and Evaluation (SQuaRe) Data quality model. *ISO/IEC* 25012 (2008), 1–13.

[11] Aimad Karkouch, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. 2016. Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications* 73 (2016), 57–81.

[12] Alan G Labouseur and Carolyn C Matheus. 2017. An introduction to dynamic data quality challenges. *Journal of Data and Information Quality (JDIQ)* 8, 2 (2017), 6.

[13] Rajeev Piyare and Seong Ro Lee. 2013. Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing. *arXiv preprint arXiv:1310.2095* (2013).

Asclepius: Data quality framework for IoT

Gabriel R Caldas de Aquino*
gabrielcaldas@cos.ufrj.br
Universidade Federal do Rio de Janeiro - PESC
Rio de Janeiro, Rio de Janeiro, Brasil

Claudio Miceli de Farias
Universidade Federal do Rio de Janeiro - PESC
Rio de Janeiro, Rio de Janeiro, Brasil
cmicelifarias@cos.ufrj.br

ABSTRACT

This work addresses challenges in IoT data quality management. IoT applications in domains like Smart Cities, Smart Healthcare, and Industry 4.0 rely on the Internet for automated services. However, resource limitations and harsh environments of IoT devices raise hardware integrity concerns. IoT data encompasses characteristics like data types, sampling rates, intervals, and specific tasks and states. Since IoT data is not always tied to specific application contexts, managing data quality is crucial to meet diverse application requirements. Asclepius, a data quality framework for IoT infrastructure, is introduced. It comprises software components deployed on IoT devices based on their roles, ensuring interoperability and standardized interfaces. Asclepius serves as a data quality backbone, facilitating reliable data transportation and routing while meeting application requirements. Real device implementation of part of our proposal. The practical incorporation of a segment is present as tangible proof, substantiating its efficacy through real-world device deployment and assessment.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Data Quality, Internet of Things

ACM Reference Format:
Gabriel R Caldas de Aquino and Claudio Miceli de Farias. 2023. Asclepius: Data quality framework for IoT. In *Proceedings of the Int'l ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet '23), October 30-November 3 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3616392.3623407

1 INTRODUCTION

IoT applications utilize the Internet for automated and intelligent services, benefiting areas like Smart Cities, Smart Healthcare, and Industry 4.0 [20] [19]. These applications employ everyday objects equipped with hardware to collect and process data, forming sensor network infrastructures. Interconnecting with the cloud and other Internet services, these infrastructures offer ubiquitous IoT services [4]. However, concerns arise over the hardware integrity of IoT devices due to their resource-constrained nature and challenging operating environments. These devices are limited in size, energy, memory, and processing capability, deployed in settings susceptible to hostile conditions and malicious behaviors.

The requirements of IoT applications encompass diverse data characteristics, including types, sampling rates, intervals, and specific tasks [8]. IoT data isn't always tied to specific contexts [3], necessitating data quality (DQ) management at the infrastructure level to suit various applications. DQ management in IoT spans all layers of the IoT model [32], aligning with contextual requirements. Evaluating IoT data's fitness for an application involves expressing, collecting, transporting, and consuming data meeting the application's specific needs, constituting a DQ problem [21] crucial for successful IoT deployment.

This work introduces Asclepius, a DQ framework for the IoT infrastructure. Asclepius features software components deployed on IoT devices based on their roles within the ITUT reference model [32]. In IoT, diverse devices with distinct hardware capabilities exist. Seamless interoperability and a DQ pipeline necessitate a comprehensive framework. Asclepius includes components and interfaces facilitating data exchange and processing across IoT devices, ensuring consistent and reliable DQ.

Asclepius serves as a DQ backbone for IoT applications, enabling expression, collection, transport, and consumption of IoT data meeting each application's needs. It ensures reliable data transport and routing. In this work we implemented a component of Asclepius and evaluated on real devices. This work is divided as: Section 2 present related works; Section 4 present Asclepius; Section 5 present our implementation; Section 6 present conducted experiments and Section 7 present the conclusions.

2 RELATED WORKS

Numerous works in literature address IoT data quality (DQ) challenges [14, 21, 23]. A survey enumerates IoT environment-related factors impacting various DQ dimensions [21], identifying problem manifestations and associated symptoms [21]. Some introduce dynamic DQ challenges [23] and mobile sensing devices [14], related to contextual changes in IoT data and structure. Mobile crowdsensing validates IoT data using personal devices [26].

To mitigate IoT data uncertainty, various works address missing data [18, 25, 27, 37]. New neural network tools enhance data recovery accuracy [18, 37]. Uncertainty, often incomplete information [18], is addressed with two neural networks for accurate IoT data completion [37]. An approach tackles missing data in time-series gaps [25], proposing an ST-correlated imputation model [27]. A

*Both authors contributed equally to this research.

DIVANet '23, October 30-November 3 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0369-0/23/10…$15.00
https://doi.org/10.1145/3616392.3623407

study examines correlation between IoT elements and DQ dimensions [1]. An IoT DQ assurance layer is proposed [2], transforming and verifying DQ [2].

Certain solutions detect DQ problems [21]. DQ challenges in mobile and crowdsensing IoT are also addressed [14, 23]. These solutions enhance IoT data quality by mitigating missing data [18, 25, 27, 37]. This work investigates DQ and IoT infrastructures. Traditional DQ management is proposed for control [30], presenting best practices and critiquing existing IoT DQ approaches. Overlooked questions are identified [30], proposing traditional best practices [30]. Several studies address data quality in IoT, exploring its dimensions [14, 21, 23]. Other studies ensure sensor accuracy [33], application-level DQ [38], and scaling small IoT components without human intervention. Our work focuses on IoT context, applications, data transportation, and interactions, proposing a novel DQ management framework distinct from traditional methods.

3 DATA QUALITY AND THE IOT SCENARIO

3.1 DQ management for traditional applications

This subsection covers data quality concepts and an overview of data quality management processes based on the ISO 8000-61 standard [17]. Prior to delving into this, the fundamental concepts of **data quality**, **data quality dimensions**, and **data quality problem manifestation** [24] need clarification.

The term "**quality**" bears multiple meanings [10]. ISO 9000:2015 [16], which outlines quality management essentials, defines *quality* as the degree to which inherent characteristics of an object fulfill a need or expectation, whether stated, implied, or obligatory. The phrases "fitness for use" and "conformance to requirements" have both described "quality" [10]. Here, **data quality** is defined as data's fitness for a given application's use.

Numerous **data quality dimensions** exist. A data quality dimension pertains to a data characteristic, attribute, or aspect, measurable empirically [10]. These dimensions serve as metrics for gauging data quality [21], encompassing elements like "data volume," "timeliness," and "completeness."

Organizations can adopt a data quality management process to prevent issues and deliver data application services in line with requirements, enhancing user satisfaction. The ISO 8000-61 standard [17] is one such initiative defining this process. The ISO standard incorporates the PDCA cycle ("Plan-Do-Check-Act"), allowing organizations to plan, execute, and review data quality management processes, ensuring resource adequacy, proper management, and continuous improvement.

Before discussing the ISO 8000-61 standard [17] and the PDCA cycle, it's crucial to explore applications that leverage data. Traditionally, organizations deploy applications in controlled data centers with resources supporting projected workloads. Application hardware requirements shape provisioning decisions, spanning physical/virtual servers, network infrastructure, and databases. The database lifecycle begins with designing a logical database's global schema, allocating data across networks, and defining local database management system schemes [36].

Here, data is a business asset with management responsibilities [28]. Applications use acquired data, performing tasks like customer billing or decision-making routines, which contextualize and interpret data. Poor data quality often stems from process, system, policy, or data design problems [6]. Thus, grasping data-generating, using, and storing processes is vital for understanding data quality.

The ISO 8000-61 standard [17] rests on three principles: process approach, continuous improvement, and people's involvement. It aims to boost data quality and organizational maturity in data quality processes. This standard introduces data quality assessment, improvement, and a plan-do-check-act cycle, enhancing processes as needed.

Quality management operates through a defined PDCA cycle [17], featuring four sub-processes that spotlight people's role in data quality management. PDCA happens via meetings with application owners to gather requirements, define strategies, and set policies for data quality management. Management's efficacy is periodically evaluated for adequacy.

Initially, application owners define a data quality control plan during requirement elicitation, crafting strategies, policies, and quality metrics for database-held data. Implementation follows these definitions. Monitoring ensues to ensure the process aligns with specifications. Meetings review management's compliance and address data quality issues. Modifications prompt new guidelines for altered data quality management, followed by re-execution.

3.2 IoT data characteristics

The characteristics of data collected by IoT infrastructures depend on the phenomena in the monitored area [3]. The IoT infrastructure is dynamic and unpredictable in terms of the phenomena it can capture.

An overview of IoT data characteristics [21]: **Uncertainty, Noise, and Errors**: IoT infrastructures use unreliable constrained devices (e.g., low-memory devices), collect data at a low abstraction level (e.g., electric pulses), and encounter uncertainty in monitored area events (i.e., unexpected phenomena); **Distributed Deployment**: IoT devices are placed in various areas (e.g., different sections of a smart city) or spread across extensive regions (e.g., a forest) to monitor events; **High Volume**: IoT devices generate data at a high rate, surpassing human-generated data. Data aggregation is crucial to distill meaningful insights; **Smooth Variability**: IoT data originates from indirect measurements of physical phenomena (e.g., room temperature) following predetermined models based on material properties. Variations tend to be smooth and well-behaved; **Continuity**: Physical phenomena like temperature change continuously. Sampling addresses computational limitations and enhances energy efficiency, albeit with some loss of information; **Correlation**: Data exhibit temporal and/or spatial correlations. For instance, a fire event might correlate with smoke and increased temperature; **Periodicity**: IoT devices monitor intermittent phenomena that occur unpredictably or at intervals; **Markovian Behavior**: Depending on the phenomenon, recent data could suffice to independently assess situations, bypassing the need for extensive historical data (e.g., real-time or near-real-time applications).

3.3 IoT data quality dimensions

Given the various IoT services and characteristics of IoT data, certain data quality dimensions are crucial in the IoT context to convey environmental aspects and factors endangering data quality. These

factors could adversely affect data consumers and include collection errors, sensor failures, processing-induced delays in the network, incomplete contextual understanding, and security concerns.

The primary factors in the IoT context are: **Accuracy**: Measures systematic error where collected data aligns with the true value; **Likelihood**: Gauges the fit of sample data to a model, indicating expected divergence; **Provenance**: Identifies sources in a given context, summarizing spatial-temporal data correlation; **Data Volume**: Quantifies available data pieces (values) for computation; **Timeliness**: Measures the time lag between current and recording timestamps, reflecting data age and punctuality; **Completeness**: Compares collected items to all available pieces of data (non-interpolated and interpolated) in the stream window; **Vocabulary**: Measures representation capacity to support concepts (unstructured, semi-structured, or fully structured), indicating data's explicit semantic representation; **Security**: Measures the capability to uphold information security's three pillars: confidentiality, integrity, and availability.

3.4 The IoT data quality challenge

The Internet of Things (IoT) embodies a paradigm where ordinary objects (e.g., light bulbs or coffee machines) interconnect via the Internet to exchange data [35] [22]. The IoT infrastructure comprises diverse devices with varying computational capabilities. Generally, its data collection setup consists of cost-effective, battery-operated devices with limited processing and radio communication abilities.

The scope of IoT applications is vast, encompassing domains like smart cities, health, homes, and industry 4.0 [5]. These applications typically acquire data from IoT sensors, employ data fusion methods for processing, and act based on predefined rules [29]. IoT devices are strategically positioned to gather data for various applications [3], often in challenging or harsh environments. These devices are susceptible to malfunctions, interference, and deliberate sabotage [21].

These challenges within the IoT infrastructure can compromise data quality, potentially impacting IoT users. A data quality issue's manifestation indicates a corresponding challenge [21]. Data anomalies degrade data quality, with outliers being one such manifestation [21]. Moreover, problems with data quality may present as bias, drifts, or missing values.

Similar to how organizations adopt data quality management processes, IoT applications can implement analogous procedures. This helps detect symptoms of data quality problems and prevent the use of inadequate data that doesn't meet IoT application standards, ensuring accurate and valuable services for users. However, the contextual aspects of IoT application environments should be considered when implementing these processes.

Contrary to traditional applications that adhere to specific hardware requirements for provisioning servers, network infrastructure, and databases, IoT applications rely on an unreliable infrastructure. Given that IoT devices possess limited resources, the IoT infrastructure must effectively manage their power consumption and handle potential link failures. Thus, the data quality management strategy needs customization to function optimally within this uncertain environment.

For IoT applications, data quality functions as a non-functional requirement, dictating the necessary data quality dimensions for its intended purpose. To facilitate the execution of data quality management within an unstable infrastructure, resources must be allocated within the network. One viable approach is designing a data quality system that employs relevant techniques and protocols to mitigate the potential impact of low-quality IoT data on application performance.

4 ASCLEPIUS

Asclepius oversees IoT data quality (DQ) in shared infrastructure, accommodating multiple applications and varying workloads. It's designed to be lightweight, suitable for resource-constrained IoT devices. The proposed network architecture by Asclepius manages DQ across the entire data lifecycle, from collection to consumption, ensuring that the IoT infrastructure fulfills DQ requirements of its applications. This framework is based on the four-layer reference model for IoT by ITUT [34], encompassing the application layer, service and application support layer, network layer, and device layer. Furthermore, Asclepius integrates management and security functionalities for each of these layers.

4.1 ITUT IoT reference model layers

The ITUT IoT reference model [34] is depicted in Figure 1. The *application layer* contains IoT applications. The *service support and application support layer* comprises generic support capabilities, usable by various IoT applications and may also be invoked by specific ones. Specific support capabilities address diverse application requirements.

The *network layer* includes networking capabilities providing control functions for network connectivity, and transport capabilities facilitating the transport of IoT-specific data and management messages.

The *device layer* is logically divided into *device capabilities* and *gateway capabilities*. *Device capabilities* encompass direct and indirect communication with the network: Devices can communicate information directly and indirectly. Devices may also form ad-hoc networks to conserve energy using "sleep" and "wake-up" mechanisms. *Gateway capabilities* offer support for multiple interfaces, connecting devices via various media. This encompasses protocol conversion when diverse device layer protocols are employed. *Management capabilities* encompass various management types, including fault, configuration, accounting, performance, and security management. Generic management capabilities in the IoT realm comprise device, network topology, traffic and congestion management, and resource reservation for critical data flows. *Security capabilities* cover both generic - independent of applications - and specific security capabilities catering to distinct requirements.

4.2 Asclepius architecture

Asclepius defines specialized components for DQ management across the *device*, *network*, and *application* layers, as per the ITUT IoT reference model [34]. In this proposal, the *application layer* and *service/application support layer* are consolidated into a single layer

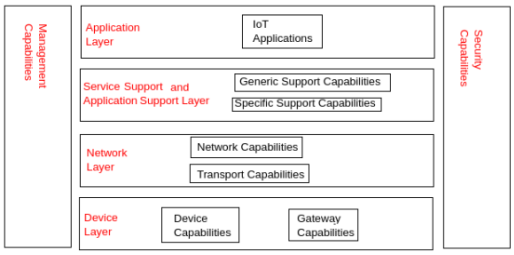

Figure 1: IoT reference model

encompassing IoT applications and their support services. Components for DQ management are formulated to cover *management capabilities* pertinent to these layers, excluding *security capabilities*.

For the mentioned layers, the structure comprises these components and databases (depicted in Figure 2): (i) Device layer: *IoT Sensor* and *data producer manager*; Databases: *environment collected data* and *infrastructure parameters*; (ii) Network layer: *data routing manager*; Database: *routing rules*; (iii) Application layer: *IoT application* and *data consumer manager*; Databases: *service rules* and *application parameters*; (iv) Management capabilities: *data quality manager* and *data quality task scheduler*; Databases: *IoT devices*, *data quality assessment tasks*, *data quality enhancement tasks*, and *scheduling rules*.

4.3 Components, databases and interfaces

The logical architecture of Asclepius (Figure 2). **Device layer**: *IoT Sensor* is the **component** that represents the sensor devices located in the device layer. These devices collect environmental data. The *IoT sensor* component has an interface called *IoTData* that provides the data collected from the environment to the *Data Producer Manager*.

The *IoT Sensor* component uses the *environment collected data* **database**. The *environment collected data* database stores unstructured environment data, including a large volume of phenomena-related information from the monitored area, often not tied to a specific IoT application. *Data producer manager* is the **component** responsible for managing DQ in the device layer. In device layer, the data quality management process applies tasks to analyze and express, and enhance IoT DQ as needed.

The *data producer manager* component consumes the *IoTData* interface provided by the *IoT Sensor* component to get the data collected from the environment. Also, it consumes the *TaskDeploy* interface provided by the data routing manager to receive the data quality tasks used in DQ management. The *data producer manager* component also provides the *IoTData* interface, which is used to forward the collected data, and the *QualityFeedback* interface which is used to forward the data quality metadata. These two interfaces are responsible for transmitting the result of the data quality management executed in the device layer. It also uses the *infrastructure parameters* **database**. The *infrastructure parameters* database stores parameters for data quality tasks on IoT devices, with each task requiring specific parameters for execution.

Network layer: *Data routing manager* component is responsible for routing data and control messages through the network. It resides in the network layer, connecting IoT applications (in the application layer), devices (in the device layer), and the data quality management process. Also, it uses the *routing rules* **database** (the routing table), which is populated statically or dynamically.

To route IoT data and the data quality control messages, the *data routing manager* component provides the following interfaces: *TaskDeploy* interface to the *data consumer manager* and *data producer manager* components; *IoTData* interface to the *data consumer manager* component; *QualityFeedback* interface to the *data quality manager* component. Also, the *data routing manager* component consumes: *TaskDeploy* interface provided by the *data quality manager* component; *QualityFeedback* interface provided by the *data producer manager* and *data consumer manager* components; *IoTData* interface provided by the *data producer manager* component. IoT data is transferred through the *IoTData* interface. As data comes from sensors in the device layer to applications in the application layer, the *data routing manager* component receives IoT data by consuming the *IoTData* interface provided by the *data producer manager* component and sends IoT data by providing the *IoTData* interface to the *data consumer manager* component.

Data quality control messages are transferred by two interfaces: *TaskDeploy* and *QualityFeedback* interfaces. The *data routing manager* consumes the *QualityFeedback* interface provided by the *data consumer manager* and *data provider manager* components to receive the information about the data quality management on *data consumer manager* and *data producer manager*, respectively.

The *data routing manager* component exchange information with the *data quality manager* component. The *data quality manager* component receives the metadata related to the data quality management and schedule data quality tasks to be applied on the data quality management. The *data routing manager* component provides the the *QualityFeedback* interface to the *data quality manager* component to send metadata related to the operation of the data quality management. The *data routing manager* component consumes the *TaskDeploy* interface, which is provided by the *data quality manager* component, to receive the result of the task scheduling process of the data quality manager.

Depending on the location of the task to be deployed (device, network or application layers), the data routing manager component sends the tasks either to *data producer manager*, to itself or to the *data consumer manager* component. So, the data routing manager component provides the *TaskDeploy* interfaces to the data consumer manager and to the data producer manager components.

Application layer: *Data consumer management* component is responsible for managing data quality in the application layer. The *data consumer manager* component analyzes received data based on IoT application's DQ requirements. The *data consumer manager* component basically performs DQ assessment and enhancement mechanisms on the received IoT data. Assessment checks compatibility of IoT data with application requirements, delivering if suitable; otherwise, attempts data enhancement. Data quality enhancement is executed when received data does not meet IoT application requirements. Data quality enhancement transforms IoT data. If the transformed data meets application requirements, it is delivered; else, if it does not meet requirements, it is discarded.

The *data consumer manager* component provides the *DataDelivery* interface to the IoT Application component. This interface is

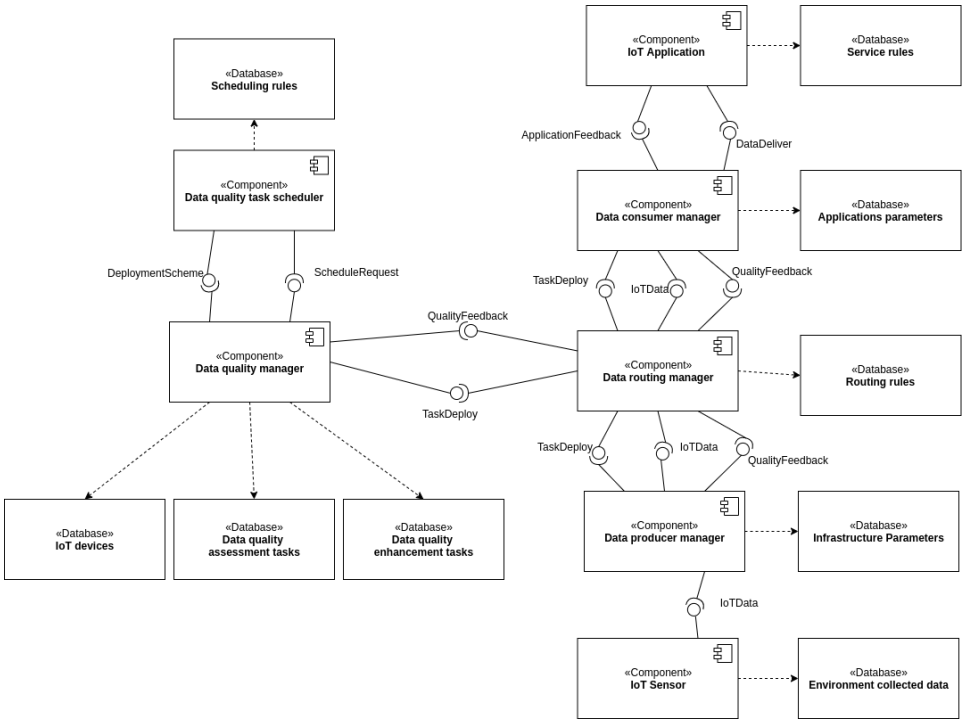

Figure 2: Framework Components

used to deliver iot data to the application. Also, the *data consumer manager* component consumes the *ApplicationFeedback* interface provided by the IoT Application. This interface is used to to receive the feedback from the application about the use of the data. It also provides the *QualityFeedback* interface to the *data routing manager* component to send the feedback of the information quality procedures to be routed by the *data routing manager* component. Also, *data consumer manager* component consumes the *IoTData* and *TaskDeploy* interfaces provided by the *data routing manager* component, which are used to receive IoT data and data quality tasks that are being deployed, respectively.

Also, the *data routing manager* component uses the *applications parameters* **database** to store the data quality requirements of the IoT applications. The application parameters database stores application identification and quality requirements. Each requirement has a quality parameter indicating desired DQ.

IoT Application is the **component** that represents the IoT applications at the the application layer. IoT application receives data, applies fusion techniques, and makes decisions based on rules. Decision-making mechanism may yield unexpected outcomes if input data of undesired quality is considered. The *IoT Application* component uses the *service rules* **database**. The *service rules* database stores the decision-making rules in a structured way. It also provides to the *data consumer manager* component an interface called *ApplicationFeedback*. This interface is used to to send to the *data consumer manager* component a feedback related to the use of the data. Finally, it consumes the *DataDelivery* interface provided by the *data consumer manager* component. This interface delivers IoT data to application after analyzing its quality based on application requirements.

Management Capabilities: *Data quality manager* is the **component** responsible for coordinating the whole data quality management process. The *data quality manager* component plans data quality management by scheduling tasks on device, network, and application layers. Tasks are executed individually on devices for data quality management post-deployment. Devices provide feedback on operations to the *data quality manager* component, which verify the data quality management process. The *data quality manager* component adapts data quality management in response to operational issues. The *data quality manager* component provides and consumes the following interfaces: The *data quality manager* component **consumes** the *QualityFeedback* interface provided by the *data routing manager* component. This interface is used to deliver the data quality management feedback to the *data quality manager* component. This interface receives, via the *data routing manager* component, information about quality procedures executed by the *data consumer manager*, *data producer manager* and *data routing manager* components; The *data quality manager* component **provides** to the *data routing manager* component the *TaskDeploy* interface, which is used to deploy, via the *data routing manager* component, the data quality tasks executed by the *data consumer manager*, *data producer manager* and *data routing manager* components; The *data quality manager* component **consumes** the *DeploymentScheme* interface provided by the *data quality task scheduler* component, which is used to deliver information about which tasks have to be deployed and where they will be deployed;

The *data quality manager* component **provides** to the *data quality task scheduler* component the *ScheduleRequest* interface, which is used to request a deployment scheme, which contains the data quality tasks to be deployed and where they will be deployed. The *data quality manager* component uses the *IoT devices* **database**, which contains: each IoT device with its unique identifier and location (device, network or application layers). It also uses the *data quality assessment tasks* **database**, which stores data quality assessment tasks with its identifiers. These tasks assess IoT DQ through executable code. Finally, it uses the *data quality enhancement tasks* **database**, which stores DQ enhancement tasks with its identifiers. Tasks enhance IoT DQ through executable code. As an example, a data completeness enhancement task can be a task that implements a data interpolation method or uses historical data to mitigate data the effects of missing data in a fusion window. *Data quality task scheduler* is the **component** that represents a data scheduler mechanism that offloads the schedule responsibilities from *data quality manager* component. This component aims to receive task, device, and objective information and generate a deployment scheme by applying scheduling rules. The *data quality task scheduler* component uses the *scheduling rules* **database**. This database stores the rules with its identifiers.

5 DPM IMPLEMENTATION

This section introduces the Data Producer Manager implementation (DPM-I) as a data cleaning technique designed to ensure Data Quality (DQ) within the Internet of Things (IoT) infrastructure. The primary objective of DPM-I is to manage DQ in an unpredictable IoT environment with diverse applications utilizing IoT data. The other components follow conventional implementation methods. The Data Routing Manager component employs the RPL Routing Protocol [15], specifically adapted for smart grid systems. For the IoT application component, the Overhead Power Line Monitoring Application (OPLM) [13] is implemented for the given scenario.

DPM-I is application-agnostic and offers real-time DQ control for any IoT application. It acquires data from IoT sensors and utilizes the Interquartile Range (IQR) statistical approach to detect outliers while considering computational constraints of IoT devices. The outlier detection process entails sending labeled data to an aggregating node for further analysis, distinguishing between event outliers (legitimate events) and error outliers (indicating data inaccuracies). Thus, DPM-I adeptly manages various outlier types, enhancing DQ in IoT scenarios.

Data cleaning is vital in ensuring accurate, consistent, and usable IoT data. It identifies and rectifies errors near IoT sensors, preventing the propagation of low-quality data and maintaining overall DQ for all applications. DPM-I effectively addresses DQ management challenges in IoT infrastructure through outlier detection, differentiation of event and error outliers, and real-time control across diverse applications [7].

5.1 A data cleaning technique

The DPM-I (Data Producer Manager implementation) is designed with the objectives: it aims to have a lightweight memory overhead and low communication overhead to accommodate the limited resources of sensor devices in IoT infrastructure. Secondly, it strives to minimize data delays, as delays can negatively impact real-time or near real-time decision-making.

The DPM-I executes its data cleaning procedures in two steps: outlier detection and outlier characterization. These steps are performed by sensor nodes in a distributed manner. During the outlier detection step, sensor devices receive data streams and assess DQ using the IQR, a simple statistical measure that can be quickly calculated. The IQR is based on quartile values stored in RAM of network nodes. The results of outlier detection are then sent to an aggregating node, which executes the outlier characterization step.

In outlier characterization step, the aggregating node applies heuristics to determine if multiple nodes have detected outliers within the same timeframe. Outliers detected by multiple nodes are considered to be the result of legitimate events, while outliers detected by only one node are labeled as errors caused by challenges in the IoT infrastructure. Outliers labeled as legitimate events are sent to designated applications, while outliers labeled as errors are discarded.

The DPM-I is designed to have a lightweight memory overhead and a small executable size. The IQR calculations can be performed on any amount of data stored in RAM memory. Additionally, the DPM-I ensures low communication overhead by providing binary information regarding the data's outlier status (event outlier or non-outlier). This binary outcome can be represented using a single bit of information, minimizing the communication required in the network. After the DQ assessment, this information serves as input for data processing procedures to facilitate decision-making and data analysis.

5.2 Outlier detection step

Data Structures: The DPM-I outlier detection step has the data structures: *RecData* and *RW*. *RecData* stores the data to be analyzed. *RW* stores the received data. The other data structures used are the following: *P25* stores the percentile 25 of the data in RW. *P75* stores the percentile 75 of the data in RW. *IRQ* stores the *inter quartile range* of the data in RW. *Uboundary* stores the upper boundary of the data analysis and in *Lboundary* stores the lower boundary of the data analysis. *Qualif* stores the result of the DPM-I.

Outlier detection step of the DPM-I: The DPM-I's outlier detection consists of two phases: Setup and Operation. The Setup Phase populates data structures for outlier detection and sets up the RW data structure size. The Operation Phase executes the DPM-I. The Operation Phase begins by receiving and storing data in the RecData structure. The data is then appended to the RW data structure for further analysis. The DPM-I assesses historical data availability by checking if the RW data structure size is greater than one. If historical data exists, the DPM-I sorts it to compute the 25th and 75th percentiles, stored in P25 and P75. These values are used to determine the interquartile range (IQR), saved in the IQR data structure.

Following IQR calculation, the DPM-I computes lower and upper boundaries, stored in Lboundary and Uboundary. Since a Normal Distribution's peak data lies within its quartiles [11], outliers here are defined as data falling below Q1 - 1.5 IQR or above Q3 + 1.5 IQR. The DPM-I employs a 1.5 factor multiplier for IQR data. With boundaries set, the DPM-I analyzes if the RecData is beyond these

limits, marking such data as outliers and removing them from RW. Otherwise, they're marked as non-outliers. The DPM-I then checks if the RW structure is full. If so, the oldest data is purged to update the DPM-I's understanding of the evolving environment. Whether full or not, the DPM-I continues operations. The DPM-I's output can feed into other methods like data fusion for decision-making or data analytics for analysis.

6 EXPERIMENTS

We considered an heterogeneous sensor network composed by 3 Arduino UNO nodes and one Raspberry Pi node. The 3 Arduino nodes act as collector nodes (CN) and the Raspberry Pi is a fusion node (FN). The CN collect data and perform the DPM-I; the FN receives data from CN and makes decisions. The experiments abstracts the origin of the data as the DPM-I is designed to assess DQ from generic IoT sensors.

The Arduino UNO is a memory constrained-device (32 KB and 2 KB RAM), open-source microcontroller board based on the 8-bit microcontroller ATmega328P and 1 KB EEPROM [31]. The Raspberry Pi presents a quad-core ARM CPU with 1 GB RAM. The network operates the IEEE 802.15.4. The DPM-I was developed using C language. The experiments lasted 1 hour, repeated 30 times with a 99% confidence interval. Distance between CN and FN nodes: 1 meter.

We use a typical Smart Grid application: [13] the Overhead Power Line Monitoring Application (OPLM) [9] [3]. OPLM application provides knowledge about the power line temperature and its loading. Due to the increasing load of the power lines, more electric energy turns into heat which could damage the lines causing operational failures, leading to blackouts. In our experiments the temperature of the Overhead Power Line is normal and in a safe condition.

OPLM application checks if the line is in ideal condition when the temperature is around 40°C - 65°C [3]. However, sensor nodes may become dysfunctional due to harsh conditions[21], prone to failures and poor DQ. 3 CN are installed in the Overhead Power Line to collect temperature samples periodically in a TDMA scheme with time slot of 2000ms and transmit it to the FN. The DPM-I is installed on each CN and stores 10 sample data as historical data.

6.1 Metrics

We evaluate the DPM-I in terms of: (i) accuracy (correct DQ assessment process), (ii) memory, (iii) energy consumption and (iv) delay imposed in two scenarios: (Scenario 1 - S1) experiment considering the DPM-I and (Scenario 2 - S2) experiment not considering the DPM-I. In S1 the DQ assessment performed by the DPM-I *is considered* by the IoT applications to perform its decision making. In S2 the DQ assessment performed by the DPM-I *is not considered* by the IoT applications to perform its decision making.

To measure the accuracy we use: *False Positives (FP)*, *False Negatives (FN)*, *True Positives (TP)* and *True Negatives (TN)*. TP metric counts the number of situations in which OPLM correctly identifies OPL in ideal condition and it is actually in ideal condition. TN metric counts the number of situations in which the OPLM application correctly identifies OPL not in ideal condition and it is actually not in ideal condition. FP counts the number of situations where the OPLM application identifies OPL in ideal condition and it is not. FN metric counts the cases in which the OPLM application identifies the OPL is not in ideal condition, but it is. True occurrences are the sum of TP and TN.

The *amount of transmitted bits* in the network is used as metric to evaluate energy consumption, as radio transmission is an energy-demanding operation. Drained battery reduces node's communication and sensing coverage [12]; Regarding memory consumption we used the *memory consumption of the DPM-I when installed in the Arduino* as metric. IoT devices are constrained in terms of memory, therefore using much memory resources is infeasible. To measure delay we used the *elapsed time to perform DQ assessment* as metric. It evaluates if the DPM-I imposes much delay. We calculate the difference in microseconds between the time in which the DPM-I receives its data to when it finishes its operations.

6.2 Results and Evaluation

In this section we discuss the obtained results. The accuracy result for S2 scenario is of 89.21% with standard deviation of 0.93% while for S1 scenario the accuracy is of 97.62% with standard deviation of 1.41%. This result shows an improvement of 8.41% in the accuracy result of the S1 scenario in comparison to the results of the S2 scenario. This improvement is given by the fact that in S2 scenario all data is provided to the OPLM application without considering the result of the DPM-I with the information about its DQ.

Outlier data degrades DQ in both scenarios. However, in S2 where DPM-I's DQ assessment is not considered, applications lack information to determine if degraded data should be considered in analysis. In S1 scenario, data provided to OPLM includes DPM-I DQ analysis information, resulting in improved accuracy. DPM-I assesses IoT DQ, improving OPLM accuracy. Considering DQ in IoT applications is vital as dysfunctional devices degrade data quality. IoT data is valuable for decision making. S1's DQ knowledge improves decision making compared to S2 without DQ consideration.

The experiments for transmitted bits shows that while using DPM-I there is an increase of 5.88% in the amount of transmitted bits. DPM-I produces meta information on DQ aspects, conveyed as a one-bit flag in data frames to indicate outliers. This results in an additional bit per transferred data frame. Optimal meta information compression implementation is not the focus of this work. Techniques for data compression may improve this aspect. Also, the DPM-I consumes 3454 bits of program storage space. The maximum available is 32256 bits of the Arduino memory. The DPM-I presents *memory consumption* of just 10% of the memory of Arduino. This result shows that the DPM-I does not impose memory overhead in Arduino platform. The experiments to assess the delay imposed by the DPM-I showed that the DPM-I imposes a delay of 307.07 microseconds (0.00030707 seconds) on average with standard deviation of σ11.96239318 and a Conf. Interval (99%) of 6.020013505. Experiment results show that DPM-I does not impose significant overhead compared to its absence.

Experiments show DPM-I is effective for DQ assessment, capable of improving applications accuracy in comparison to the scenario where the DPM-I is not used. Also, our experiments indicates that the DPM-I does not impose much overhead in the network resources (energy and memory) nor the DPM-I imposes a significant delay when performing its DQ assessment.

7 CONCLUSION AND FURUTE WORKS

This work proposed and implemented Asclepius - DQ framework designed specifically for the IoT infrastructure. We also implemented the Data Producer Management component of Asclepius on real IoT devices. The results demonstrate the positive impact of implementing our solution on DQ. We observed a significant increase in DQ compared to scenarios where our solution was not employed. Specifically, the implementation of a component part of Asclepius effectively identifies and eliminates outlier data, thereby improving the overall quality of the dataset. Without the utilization of our implementation, the DQ was noticeably worse, with a higher prevalence of outliers that could potentially impact the reliability and accuracy of subsequent data analysis and decision-making processes. These findings highlight the effectiveness and importance of our solution in enhancing DQ within the given context. **Future work** includes completing the implementation of the Asclepius framework. Currently, our focus is on specific components, but a comprehensive implementation of all components will provide a complete solution for IoT DQ management. Further research can explore the implementation of additional components, expanding the capabilities of the framework. Investigating different data scheduling algorithms can optimize data processing and improve framework efficiency. Additionally, exploring the impact of various data routing protocols on reliability, latency, and scalability will enhance DQ management.

REFERENCES

[1] Rawhi Alrae, Qassim Nasir, and Manar Abu Talib. 2020. Developing House of Information Quality framework for IoT systems. *International Journal of System Assurance Engineering and Management* (2020), 1–20.

[2] Fernandes Nicole Ann and Rupali Wagh. 2019. Quality assurance in big data analytics: An IoT perspective. *Telfor Journal* 11, 2 (2019), 114–118.

[3] G. Aquino, L. Pirmez, C. M. de Farias, F. C. Delicato, and P. F. Pires. 2016. Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks. In *2016 19th International Conference on Information Fusion (FUSION)*. 59–66.

[4] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Computer networks* 54, 15 (2010), 2787–2805.

[5] S Balaji, Karan Nathani, and R Santhakumar. 2019. IoT technology, applications and challenges: a contemporary survey. *Wireless personal communications* 108, 1 (2019), 363–388.

[6] Phillip Cykana, Alta Paul, and Miranda Stern. 1996. DoD Guidelines on Data Quality Management. In *IQ*.

[7] Gabriel R Caldas de Aquino. [n. d.]. GitHub - gabrielrca/hygieia: Codes for Hygieia — github.com. https://github.com/gabrielrca/hygieia. [Accessed 29-May-2023].

[8] Claudio de Farias, Henrique Soares, Luci Pirmez, Flávia Delicato, Igor Santos, Luiz Fernando Carmo, José de Souza, Albert Zomaya, and Mischa Dohler. 2014. A control and decision system for smart buildings using wireless sensor and actuator networks. *Trans. on Emerging Telecom. Technologies* 25, 1 (2014), 120–135.

[9] Claudio M de Farias, Luci Pirmez, Flavia Coimbra Delicato, Igor L Dos Santos, and Albert Y Zomaya. 2012. Information fusion techniques applied to Shared Sensor and Actuator Networks. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*. IEEE, 188–191.

[10] Joseph A De Feo. 2017. *Juran's quality handbook: The complete guide to performance excellence*. McGraw-Hill Education.

[11] M. H. DeGroot and M. J. Schervish. 2012. *Probability and statistics*. Pearson.

[12] F. Z. Djiroun and D. Djenouri. 2017. MAC Protocols With Wake-Up Radio for Wireless Sensor Networks: A Review. *IEEE Communications Surveys Tutorials* 19, 1 (Firstquarter 2017), 587–618. https://doi.org/10.1109/COMST.2016.2612644

[13] Abdel-A Edris and Brian W D'Andrade. 2017. Transmission Grid Smart Technologies. In *The Power Grid*. Elsevier, 37–55.

[14] Eija Ferreira and Denzil Ferreira. 2017. Towards altruistic data quality assessment for mobile sensing. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, 464–469.

[15] Li Gao, Zengwei Zheng, and Meimei Huo. 2018. Improvement of RPL protocol algorithm for smart grid. In *2018 IEEE 18th International Conference on Communication Technology (ICCT)*. IEEE, 927–930.

[16] ISO. 2015. *ISO 9000:2015 Quality management systems - Fundamentals and vocabulary* . Standard. International Organization for Standardization, Geneva, CH.

[17] ISO. 2016. *ISO 8000-61 - Data quality – Part 61: Data quality management: Process reference model*. Standard. International Organization for Standardization, Geneva, CH.

[18] Ivan Izonin, Roman Tkachenko, Volodymyr Verhun, and Khrystyna Zub. 2020. An approach towards missing data management using improved GRNN-SGTM ensemble method. *Engineering Science and Technology, an International Journal* (2020).

[19] Mohd Javaid, Abid Haleem, Raju Vaishya, Shashi Bahl, Rajiv Suman, and Abhishek Vaish. 2020. Industry 4.0 technologies and their applications in fighting COVID-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews* 14, 4 (2020), 419–422.

[20] Mohd Javaid and Ibrahim Haleem Khan. 2021. Internet of Things (IoT) enabled healthcare helps to take the challenges of COVID-19 Pandemic. *Journal of Oral Biology and Craniofacial Research* 11, 2 (2021), 209–214.

[21] Aimad Karkouch, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel. 2016. Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications* 73 (2016), 57–81.

[22] Nasr Kasrin, Aboubakr Benabbas, Golnaz Elmamooz, Daniela Nicklas, Simon Steuer, and Michael Sünkel. 2021. Data-sharing markets for integrating IoT data processing functionalities. *CCF Transactions on Pervasive Computing and Interaction* 3, 1 (2021), 76–93.

[23] Alan G Labouseur and Carolyn C Matheus. 2017. An introduction to dynamic data quality challenges. *Journal of Data and Information Quality (JDIQ)* 8, 2 (2017), 6.

[24] Caihua Liu, Patrick Nitschke, Susan P Williams, and Didar Zowghi. 2020. Data quality and the Internet of Things. *Computing* 102, 2 (2020), 573–599.

[25] Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa. 2020. Missing Value Imputation for Industrial IoT Sensor Data With Large Gaps. *IEEE Internet of Things Journal* 7, 8 (2020), 6855–6867. https://doi.org/10.1109/JIOT.2020.2970467

[26] Tie Luo, Jianwei Huang, Salil S Kanhere, Jie Zhang, and Sajal K Das. 2019. Improving IoT Data Quality in Mobile Crowd Sensing: A Cross Validation Approach. *IEEE Internet of Things Journal* (2019).

[27] I. P. S. Mary and L. Arockiam. 2017. Imputing the missing data in IoT based on the spatial and temporal correlation. In *2017 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*. 1–4. https://doi.org/10.1109/ICCTAC.2017.8249990

[28] Richard G Mathieu and Omar Khalil. 1998. Data quality in the database systems course. *Data Quality Journal* 4, 1 (1998), 1–12.

[29] Eduardo F Nakamura, Antonio AF Loureiro, and Alejandro C Frery. 2007. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys (CSUR)* 39, 3 (2007), 9.

[30] Ricardo Perez-Castillo, Ana G Carretero, Moises Rodriguez, Ismael Caballero, Mario Piattini, Alejandro Mate, Sunho Kim, and Dongwoo Lee. 2018. Data quality best practices in IoT environments. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*. IEEE, 272–275.

[31] Rajeev Piyare and Seong Ro Lee. 2013. Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing. *arXiv preprint arXiv:1310.2095* (2013).

[32] Y Recommendation. 2012. 2060 «overview of internet of things». *ITU-T, Geneva* (2012).

[33] Mustafa Al Samara, Ismail Bennis, Abdelhafid Abouaissa, and Pascal Lorenz. 2022. A survey of outlier detection techniques in IoT: review and classification. *Journal of Sensor and Actuator Networks* 11, 1 (2022), 4.

[34] ITU Telecommunication Standardization Sector. 2012. Recommendation ITU-T Y. 2060: Overview of the Internet of things. *Series Y: Global information infrastructure, internet protocol aspects and next-generation networks-Frameworks and functional architecture models* (2012).

[35] Abhishek Singh, Ashish Payal, and Sourabh Bharti. 2019. A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features, and open issues. *Journal of Network and Computer Applications* 143 (2019), 111–151. https://doi.org/10.1016/j.jnca.2019.06.013

[36] Toby Teorey, Sam Lightstone, Tom Nadeau, and H.V. Jagadish. 2011. 1 - Introduction. In *Database Modeling and Design (Fifth Edition)* (fifth edition ed.), Toby Teorey, Sam Lightstone, Tom Nadeau, and H.V. Jagadish (Eds.). Morgan Kaufmann, Boston, 1–11. https://doi.org/10.1016/B978-0-12-382020-4.00001-X

[37] Roman Tkachenko, Ivan Izonin, Natalia Kryvinska, Ivanna Dronyuk, and Khrystyna Zub. 2020. An approach towards increasing prediction accuracy for the recovery of missing IoT data based on the GRNN-SGTM ensemble. *Sensors* (2020).

[38] Feby A Vinisha and L Sujihelen. 2022. Study on Missing Values and Outlier Detection in Concurrence with Data Quality Enhancement for Efficient Data Processing. In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 1600–1607.