DEVELOPING A PROCESS MODEL FOR INCORPORATING DATA QUALITY VALIDATION INTO DATA PIPELINES

Lappeenranta–Lahti University of Technology LUT

Master's Programme in Software Product Management and Business, Master's thesis

2023

Toivo Mattila

Examiners: Professor Maria Paasivaara

Tom Gustafsson, M.Sc.

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Software Engineering

Toivo Mattila

Developing a process model for incorporating data quality validation into data pipelines

Master's thesis

2023

73 pages, 19 figures, and 3 tables

Examiners: Professor Maria Paasivaara and Tom Gustafsson, M.Sc.

Keywords: data engineering, data quality, data quality validation

Having high-quality data is important for companies that rely on data for their operations but the companies may not have the know-how to ensure the quality of their data. Models for data quality management on the organizational level exist but not on the practical, data engineering level.

This thesis aims to provide a process model for practically incorporating data quality validation into a data pipeline. The thesis used the Design Science Research methodology to iteratively develop the process model.

The thesis proposes a generalizable 6-stage process model that data engineers can utilize for reliably implementing continuous data quality validation into their data pipelines. The stages in the model are Environment, Data, Data Specification, Pre-Deployment, Deployment, and Post-Deployment. This enables companies to better monitor and manage the quality of their data as well as further research on the importance of data quality monitoring in the data management lifecycle and improving data quality.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tietotekniikka

Toivo Mattila

Prosessimallin kehittäminen datan laadun vahventamisen sisällyttämiselle dataputkiin

Tietotekniikan diplomtiyö

2023

73 sivua, 19 kuvaa ja 3 taulukkoa

Tarkastajat: Professori Maria Paasivaara ja Tom Gustafsson, DI

Avainsanat: datakehittäminen, datan laatu, datan laadun vahvistaminen

Korkealaatuinen data on tärkeää yrityksille, jotka hyödyntävät dataa toiminnoissaan mutta yrityksillä ei välttämättä ole osaamista taata datan laatua. Datan laadun hallitsemiseen organisaatiotasolla on olemassa valmiita malleja mutta ei käytännön datakehityksen (data engineering) tasolla.

Tämä opinnäytetyö pyrkii tarjoamaan prosessimallin siihen, miten datan laadun vahvistaminen käytännössä sisällytetään dataputkeen (data pipeline). Työ noudatti Design Science Research-tutkimusmetodologiaa prosessimallin kehittämiseen iteratiivisesti.

Työ esittää yleistettävän, 6-vaiheisen prosessimallin, jota datainsinöörit (data engineer) voivat hyödyntää saadakseen luotettavasti toteutettua dataputkiin datan laadun jatkuvan vahvistamisen. Mallin vaiheet ovat Ympäristö, Data, Dataspesifikaatio, Ennen Käyttöönottoa, Käyttöönotto, ja Käyttöönoton Jälkeen. Tämä mahdollistaa yrityksiä seuraamaan ja hallitsemaan käyttämänsä datan laatua sekä lisätutkimusta datan laadun valvomisen tärkeydestä datan elinkaaren hallinnassa ja datan laadun parantamisessa.

ACKNOWLEDGEMENTS

Thank you to Maria Paasivaara from LUT and Tom Gustafsson, Janne Tomperi, Mikael Jantunen, and all the other people from Virnex.

Table of contents

Tables and figures

1 Introduction

1.1 Motivation for thesis

Companies use more and more data for managing their business, such as social media activities and other information from the web, consumer preferences, data produced by IoT devices, and more [1], [2]. The data should also be high-quality as companies for example rely on this data to make business decisions and decisions made based on bad data will likely result in bad decisions [3]. Additionally, as businesses use more machine learning-based solutions, the quality of the data becomes even more important since high-quality data is crucial for machine learning projects as bad data quality results in bad machine learning models [4]. Problems with training or production data may cause wrong or nonsensical outputs from machine learning models and safeguards are needed both in development as well as in production to avoid that [5]. Even though bad-quality data can have significant negative impacts on a company, many companies don't have the resources or know-how to improve data quality [6]. Existing data quality frameworks such as MAMD [7] provide guidance on how to improve data quality, especially on the organizational level but don't provide advice on how to implement data quality assurance from the data engineering perspective.

The research was done at Virnex Group Oy, a consulting company that provides various services to its clients, including data solutions to help companies better leverage their data. The company is described in more detail in the Case company and data pipelines section. The research aims to provide the company with a repeatable and reliable way for incorporating data quality validation into the developed data solutions, which is expected to improve the quality of the produced solutions and help the consultants work more effectively.

Practically from the data engineering perspective, data pipelines are used to produce and manage data [8]. A part of the pipeline may be faulty which leads the whole data pipeline to produce bad data [8]. Incorporating data quality assurance into a data pipeline is expected to facilitate fixing problems and result in higher-quality data.

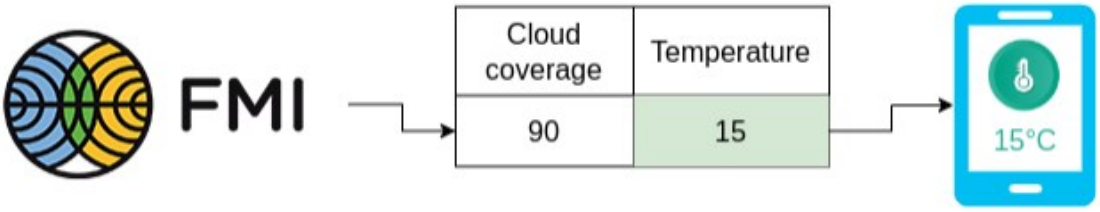

Figure 1 - Simple data pipeline

The context for the thesis can be demonstrated with a simple example which is displayed in Figure 1: a weather application that automatically fetches the current temperature from the Finnish Meteorological Institue (FMI) and shows the temperature to the user. The application would be a data product. However, in this example, FMI provides the current cloud coverage in addition to the temperature which means that the temperature needs to be separated. The program that fetches the weather data from FMI and extracts the temperature is called a data pipeline. However, problems arise if something in the data from FMI changes, and for some reason, the data pipeline extracts the cloud coverage as the current temperature. This results in the application claiming on a particularly cloudy day with a 90% cloud coverage that the average temperature in Finland is 90°C, which is nonsensical. Implementing data validation would automatically notify the owner of the app about the problem with the data and prompt fixing the problem.

1.2 Topic and research questions

The main purpose of the thesis is to create a process model for implementing data quality assurance into an existing data pipeline. The process model aims to provide a way to repeatable and reliably incorporate data quality assurance into data pipelines and therefore improve the quality of the data that the pipelines produce.

The main research goal is to:

"Create a process model for incorporating data quality assurance into an existing data pipeline."

A sub-question related to this goal is:

"Which problems are the people in the case company facing related to low-quality data?"

Other activities such as data management and governance are related to and important for improving data quality, especially at the organizational level [7]. However, the focus of this thesis is on the technical and practical aspects of incorporating continuous data validation into data pipelines.

1.3 Thesis structure

The thesis has the following structure: introduction, literature review, methodology, results, discussion, and conclusion.

The literature review chapter explores the previous research on the topics related to data quality assurance. The methodology chapter describes how the research was conducted and how the process model was created. The findings from the research are presented in the results chapter, and the discussion chapter notes opportunities for future research.

2 Literature review

This section explores the existing literature on how data quality and data pipelines have been defined in the previous research, existing process model related to data quality and the practical aspects of validating data.

2.1 Data Quality

Data quality is defined in the literature to be subjective and context-dependent [9]. This means that a single, universal, and generic definition of what constitutes "high-quality data" can't be defined. Instead, data quality should be situated in the context of the data product and viewed from the perspective of the data consumers and data producers using and developing the data product [9], [10]. Data quality can from this perspective be defined as "fitness for use" or as data that is fit for use by the data consumers [9]. A distinction is also sometimes made in the research literature between data quality (technical) and information quality (non-technical) [11].

Data quality is identified to be multi-dimensional [9]. Different studies have identified various dimensions or provided different definitions for the same dimensions [12]. However, commonly identified data quality dimensions include accuracy, reliability, timeliness, relevance, completeness, understandability, interpretability, accessibility, and ease of use [9], [10], [12], [13]. Figure 2 shows an example of data quality dimensions presented in two layers.

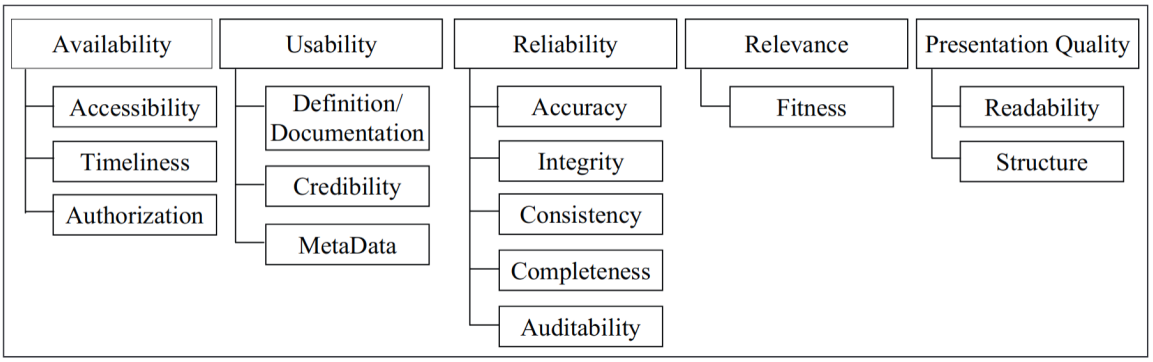

Figure 2 - Two-layer data quality dimensions [10]

2.2 Data Pipelines

Data engineering refers to the process of taking in raw data and transforming, moving, and storing it in a form that is suitable for data consumers [14]. Data pipelines are used for automatically processing data and routing the data to different applications like visualizations or machine learning models [8].

A data pipeline consists of multiple independent data processing steps called data pipeline blocks that take in input data, perform some processing on the data, and produce output data that functions as the input for the next pipeline block [15]. A pipeline block is illustrated in Figure 3.

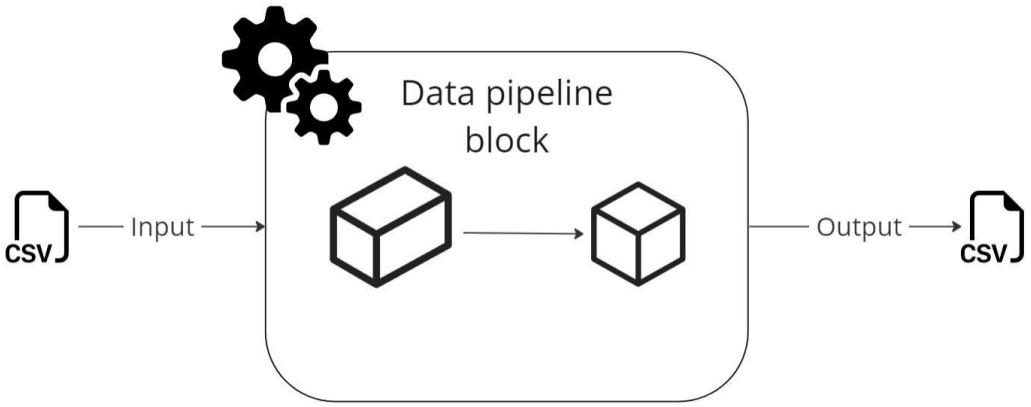

Figure 3 - Data pipeline block

The pipeline blocks are chained together to form a data pipeline that can be used for automatically performing operations on various input data and producing output data that can be utilized by data products [8], [15]. Figure 4 illustrates how these data pipeline blocks are chained together to produce a data pipeline.

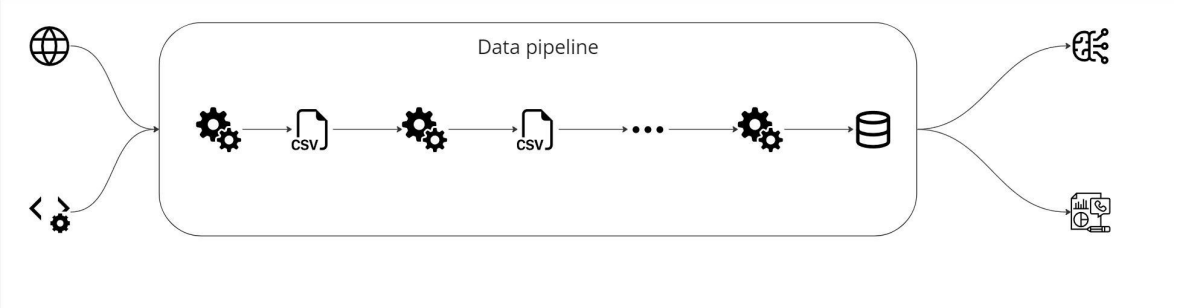

Figure 4 - Data pipeline

The data pipelines can be viewed as data manufacturing systems that take in raw data as input, process it, and as a result produce output data [9]. This idea can be extended to consider the output data as a product of the manufacturing system, which has both cost and value [10].

Triggering the data processing inside the data pipeline needs to be managed by using for example a data pipeline orchestration tool such as Apache Airflow or Argo Workflow [16]. The different techniques that the tools can utilize for orchestrating the processing in a data pipeline can be for example script-based, event-based, adaptive, declarative, or procedural [16], [17].

Benefits of using a data pipeline include improved accessibility to data, automated data processing, reduced operational and human errors, the possibility for automated data pipeline and data quality monitoring, better data workflow traceability, faster recovery from problems with the data, support for heterogeneous as well as streaming data, accelerated data life cycle process, standardized and repeatable data workflow, better monitoring and communication about the data workflow [8].

Challenges with data pipelines have been identified to relate to challenges with infrastructure, organization, as well as data quality, and the problems are magnified with more complex data pipelines [8].

2.3 Data Quality Process Models

The purpose of this section is to examine the process models and frameworks related to data quality that have been presented in the research literature.

In the software development space, multiple good and mature practices, models, and frameworks have been created but similarly mature frameworks don't exist yet for data processing [7], [10].

A systematic approach is required for managing and improving data quality on an organizational level and multiple models have been proposed for managing data quality in organizations, such as Total Data Quality Management (TDQM), Information Integrity Methodology (IIM), AIM Quality (AIMQ), Data Quality Management Maturity Model (DQMMM), and Model Alarcos de Mejora de Datos (MAMD) [7], [13], [18], [19].

Framework	Data management	Data quality management	Data governance
English [14]	X	X	
CALDEA [19]	X	X	
IQM3 [20]	X	X	
IQM [22]	X	X	
Aiken et al. [16]	X		
DMM [5]	X	X	X
IAIDQ [30]	X	X	X
ISO 8000-61[7]	X	X	
DAMA [18]	X	X	X

Table 1 - Comparison of data quality process models [7]

These models aim to help organizations with topics such as data governance, data management, and data quality management [7]. Carretero et al. compared the different models in terms of considering these 3 aspects and the comparison is presented in Table 1. These models and frameworks are useful for providing high-level guidance on how to manage data in organizations. For example, the MAMD model can successfully be used to evaluate the maturity level of an organization's data management practices [20].

Figure 5 - Data Engineering Reference Model (DERM) [21]

Figure 5 presents the Data Engineering Reference Model (DERM) aims to provide a process model for data engineering that covers the full data life cycle [21]. The model identifies 6 data lifecycle phases which are Plan, Create, Access, Use, Transform, and Destroy, and four abstract themes related to handling data which are Metadata, Technology, Data, and Enterprise [21].

No models or frameworks that concentrated on data quality from the data engineering perspective were found.

2.4 Data Quality Assurance In Practice

This section explores the advice presented in the research literature on how to practically perform data quality assurance.

As discussed previously, data pipelines are data processing steps chained together to take in source data and automatically process it into a desired output [8]. Changes in the data, such as data drift may cause problems in the data pipeline resulting in bad data which means that data pipelines should be monitored continuously to recognize data drift and other changes in the data [8], [22]. The data quality can be continuously verified during process execution which means that data quality is verified immediately as it goes through the pipeline and the step in the process where a data defect occurs can be identified immediately [22]. Continuously monitoring the data quality throughout the data pipeline is important as an error or a problem in one component in the data pipeline can cascade throughout the rest of the pipeline [23]. To avoid problems in the data verification from causing overhead in the actual data pipeline, the runtime data verification should be independent of the process execution and just observe a process without intervening in the process execution, simply verifying the data and collecting information about the process [22].

This validation can be performed on structural data using a rule-based approach, where a data quality specification is defined using various rules that the data should fulfill, and the data is checked against these rules [22]. The rules should be stateful, meaning that the acceptable values are stored in the rules, and metrics such as mean and standard deviation are pre-computed, which also enables efficient data drift detection [5]. The rules can be declarative, meaning they are statements about the data that can also be treated as data and stored as metadata for the actual datasets [5], [24].

The data quality dimensions discussed previously such as accessibility or understandability are abstract and no definition was found in the literature for some of the dimensions that could be used for defining rules for data quality validation. However, at least the completeness dimension can be defined in a way that can be turned into a rule for data validation. Data completeness can be represented by breaking it down into 3 separate

aspects: schema completeness, column completeness, and population completeness where schema completeness is defined as the percentage of the required attributes being present in the schema, column completeness is defined as the percentage of missing values in a column, and population completeness as the degree to which the required values are present in the data [25]. For example, only 43 out of the 50 states in the United States being present in the data would correspond to population completeness of 43/50 or 86%.

As discussed previously, data quality is commonly defined as being context-dependent but context-independent methods for approaching data quality have also been proposed. Data smells are defined as context-independent indicators of potential issues in data [26], [27].

Data smells are typically caused by poor practices and from violating recommended best practices in data management and engineering. Typical reasons for data smells to occur include poor quality data sources, data generation, acquisition, and processing. Data smells might not cause errors in the system immediately but make problems later in the data processing more probable. [27]

Data smells can be further categorized into sub-groups that include believability smells, understandability smells, and consistency smells [27]. Figure 1 presents the identified data smells in the different categories.

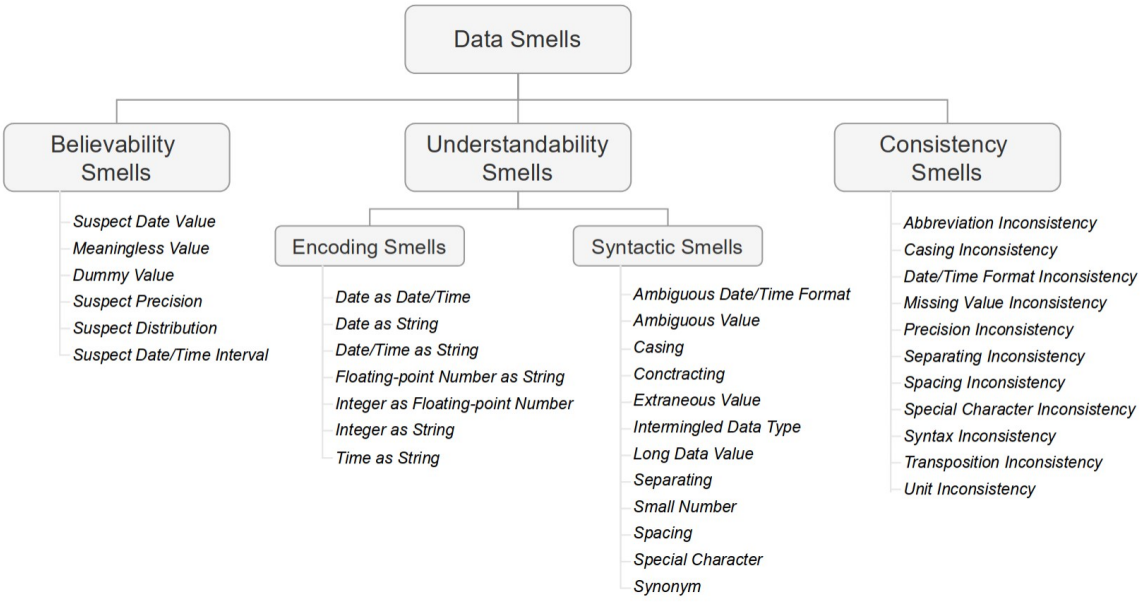

Figure 4 - Data smells and sub-categories [27]

Data smells are typically caused by issues for example with data management, data handling, and data sources [27]. The identified data smells are listed in Figure 4.

Data smells arising from poor data management include for example bad practices with data collection, preparation, documentation, and communication. Bad practices for data collection and preparation include issues such as careless data entry, inconsistent data collection, and faulty transformation processes. Data documentation refers to for example data lineage and data dictionary. Poor communication about the data between different parties involved with the data may also lead to subtle problems and smells in the data. [27]

Pre-built data validation frameworks that have been discussed in the research literature include DaQL Deequ, DuckDQ, Great Expectations, Hooqu, and TFX [4], [5]. These tools are noted to have some advantages and disadvantages: Deequ is designed for very large datasets and brings unnecessary overhead to projects that are handling small to medium-sized datasets, TFX is tightly integrated into Google platforms, and Great Expectations and Hooqu use pandas to calculate the required statistics, which makes them slower than computing similar statistics using an RDBMS [5].

3 Research method

This chapter describes the research methodology. The research was conducted using the Design Science Research methodology and the framework defined by Peffers et al. [28]. This chapter discusses how data was collected and analyzed to define the research problems, and how the research framework was used to iteratively develop a process model for incorporating data quality validation into a data pipeline.

3.1 Research design

3.1.1 Research Approach

The data collected included a mix of qualitative and quantitative data: interviews combined with analyzing logs and data archives.

The purpose of the interviews was to ground the data validation to what was actually important for the company and users of the data products. For example, the research focused on the data validity and ensuring that the values are correct as that was considered highly important by the data consumers. Data developers were interviewed to further understand the current situation.

Log analysis and archive analysis were used for the initial state analysis, to estimate the current problems with data quality, to further motivate adopting data quality validation, as well as to provide something quantitative to measure the effects of the data validation.

The interviews and the analyses were not directly affected by each other. For example, the answers from the interviews did not change how the archived data was validated. No further interviews were conducted based on the initial data collection. Instead, all 3 methods were used in combination when defining the research problems.

These methods provided a good idea about what was important about the data, why data validation was useful, what problems the data consumers and developers were facing, and overall what the situation was before the data validation was implemented.

3.1.2 Research Framework

Design Science Research was used as a research framework for conducting the research. Design Science Research (DSR) is a research paradigm that is concerned with and seeks to produce research through creating and validating artifacts, such as database systems, models for business processes, and more [29]. Given that Design Science Research is well-suited for researching and developing process models, it was chosen as the research paradigm and framework for this thesis as the goal of the thesis is to produce a process model for adopting data quality validation. The process model described below that the thesis follows for conducting design science research was developed and introduced by Peffers et al. [28].

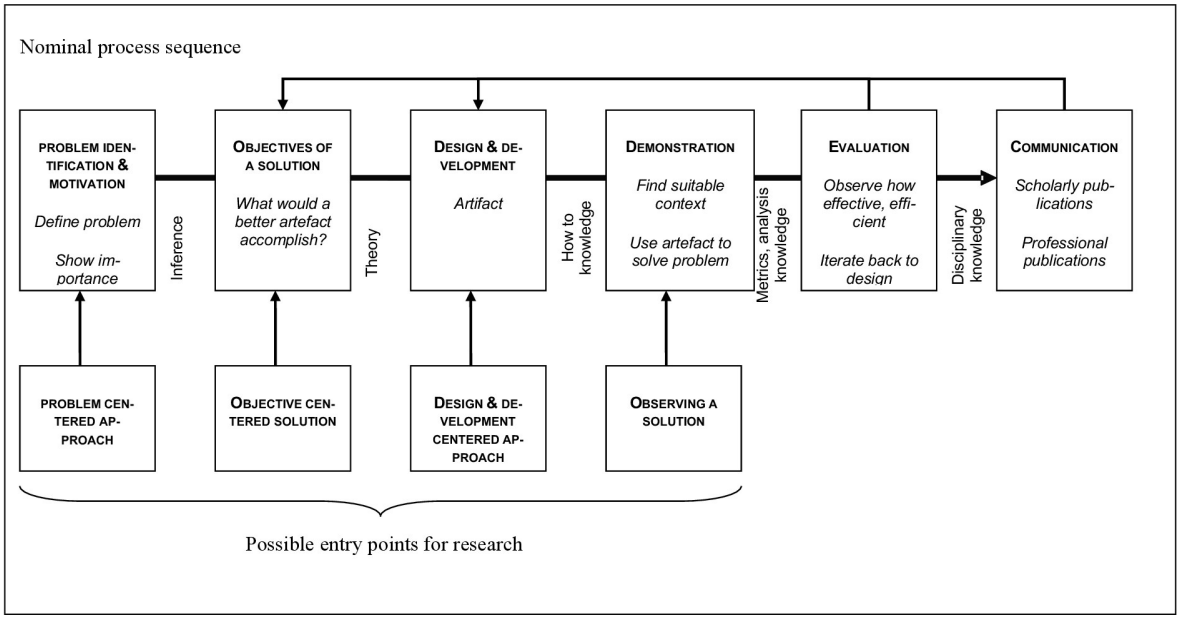

Figure 6 - Design Science Research process model [28]

The overall structure of the process model by Peffers et al. as presented in Figure 6 is divided into 6 different stages [28]:

1. Problem Identification
2. Objectives of a solution

3. Design and development
4. Demonstration
5. Evaluation
6. Communication

The problem identification stage is used for exploring the need for a solution, defining a specific research problem, and providing motivation and justification for developing a solution. In this research, the motivation for the research as well as the specific research problem were explored and defined in the Introduction chapter as well as with interviews combined with log and archive analysis, which are discussed in more detail in the next section.

Based on the defined research problem, the objectives of the desired solution are defined. This definition should define how the expected solution solves the identified research problem and preferably better than existing solutions. This can be measured quantitatively or qualitatively. The objectives for the desired solution are discussed in the Data Quality Validation In Practice section.

Once the objectives for the desired artifact have been identified, the next stage is to create the artifact that solves the research problem. This stage also includes defining and designing how the artifact solves the problem and then creating the artifact to solve the problem. How the artifact was created in this thesis is discussed in the Data Quality Validation In Practice section.

After the artifact has been created, it is concretely utilized to demonstrate its efficacy and effectiveness in solving the problem. In practice, this means utilizing the artifact in some way such as applying it to a case study or involving it in an experiment. The Data Quality Validation In Practice section describes how this process was done in this thesis.

How well the demonstrated artifact solves the defined problem and fulfills the defined objectives for the solution is evaluated in the next stage. In practice, this evaluation might be for example comparing the artifact's functionality to the defined objectives, performance metrics, interviews, or client feedback. After the evaluation, the researchers may choose to go back to the design and development phase and proceed forward to iteratively improve and further develop the artifact to better solve the defined research problem. If they choose to do so, a new version of the artifact is developed, demonstrated,

and evaluated. This process can be repeated as many times as desired. After it is decided to not iterate the artifact, the process proceeds to the next stage. Reasons for stopping the iteration include for example the artifact fulfilling and solving the research problem or some constraint limiting the iteration, such as a pre-defined schedule for the research project. This stage is described in this thesis in the Data Validation in Practice chapter.

The last stage in the process model is communicating the research problem, the research process, and how the resulting artifact solves the problem to a relevant audience. In this thesis, how the research was communicated is described in further detail in the Communication section of the thesis.

The rest of the Methodology chapter in this thesis is structured in a way that multiple stages are described in a single section. Each section clarifies the stages it describes in addition to the sections being mentioned in the process description here.

3.2 Case company and data pipelines

The research was done for the company Virnex Group Oy. Virnex Group is a leading Finnish company specializing in digital solutions, business development, and marketing services.

With years of experience in the industry, Virnex Group has established itself as a trusted partner for businesses looking to enhance their digital presence and achieve sustainable growth.

The solutions provided by Virnex include producing various data solutions for businesses to help them more effectively leverage data in their business and become more data-driven. Virnex has a dedicated team of data engineers, data scientists, and data analysts to provide these data solutions. The size of the team is 20 and growing. The team has also built various internal data products with supporting data pipelines to provide data for the products. Some of these data pipelines didn’t have data quality validation incorporated into them and were available for the research process. As discussed in more detail later in the Research Method chapter, this was crucial for following the Design Science Research

methodology which includes iteratively designing the process model and demonstrating the viability of the designed model.

The criteria for selecting the data pipelines for the thesis were:

- The data pipeline needs to produce data
- The data produced by the data pipeline is consumed by someone
- The data pipeline doesn't have data quality validation
- The data pipeline can be modified during the research
- Brief outages in the data pipeline don't cause significant harm
- The data pipeline processing can be activated on demand

The pipeline needs to produce data so that something that can be validated exists. As discussed in the Literature review chapter, data quality is context-specific and should be observed from the perspective of the data consumer [9]. This means that to be able to adequately define and validate the quality of the data, the data produced by the data pipeline should be consumed by someone, for example in the form of a data product. The data pipeline also shouldn't have data quality validation incorporated into it yet. The data pipeline should also be able to be changed as incorporating the data quality validation into the data pipeline necessarily requires making changes to the pipeline. The assumption is that as a part of the research process, the changes made to the data pipeline may cause unforeseen problems and may cause outages not only in the data validation but also in the data pipeline. While these problems are expected to be fixed relatively quickly, the pipeline should be such that outages don't cause any significant harm or damage. Lastly, to facilitate the development of the process model, it should be possible to trigger the data pipeline on demand. For example, incorporating data quality validation into a data pipeline that is scheduled to run once per month would be impractical for this research process.

5 data pipelines were identified that fulfilled these criteria. The pipelines produced data related to the following topics:

- Electricity spot prices
- Electricity production
- Sun radiation
- Weather measurements such as temperature
- Company's financial data

The pipelines produced data for 2 different data products: a machine learning model that predicted the price of electricity and an internal financial report for better understanding the company financials. The electricity spot prices, electricity production, sun radiation, and weather measurement data were used for the electricity price prediction while the financial data was used in the financial report. The pipelines are described in more detail in the Data Pipelines Analysis section.

3.3 Data collection and analysis

This section describes the different data that were collected and how the data was analyzed to identify the research problems. The data collection methods included interviews, data pipeline log analysis, data archive analysis, data pipeline analysis, and data validation framework comparison.

3.3.1 Interviews

Consumers of data products in the company as well as the team members that develop the data products and the data pipelines for the data products were interviewed.

The purpose of the interviews was to ground the data validation to what was actually important for the company and users of the data products. For example, the research focused on the data validity and ensuring that the values are correct as that was considered highly important by the data consumers. Data developers were interviewed to further understand the current situation.

The development team provided data consumers that represented well the users of the data products that were selected for the interview. All of the 3 developers who were actively developing the data products were interviewed.

The data consumers were asked to freely describe the important aspects of the data in the data product that they used. The exact question that was asked was:

"What are the important aspects of the data in the data product that you use?"

Additionally, as it is generally easier for people to respond to more specific questions and topics, after the initial response, data quality dimensions that were identified in the related literature were provided, one at a time for the data consumer and asked to comment on how important they saw that dimension from their perspective. The provided data quality dimensions were accuracy, reliability, timeliness, relevance, completeness, understandability, interpretability, accessibility, and ease of use.

The data developers were interviewed after the data consumers. The data developers were asked to describe 2 things:

1. *"How well in your opinion the different data quality dimensions that are important for the data consumers currently implemented?"*
2. *"If there are problems in some of these important aspects, how difficult or time-consuming is it to for you to find out the reason for the problem and fix it?"*

The same 7 data quality dimensions were given one at a time for developers as well as briefly described what was important about the dimension for the data consumers.

Figure 7 - Interview analysis on the digital whiteboard tool Miro (blurred)

The interviews were recorded and then analyzed by turning the individual things that the interviewees said into notes on Miro, a digital whiteboarding tool. These notes were then analyzed on the whiteboard by grouping them per interviewee and data quality dimensions, noting any common topics and ideas, agreements and disagreements, anomalies, individual thoughts, and anything else that was noteworthy. For example, whether all of the developers were in agreement that some data quality dimension was not well realized. The resulting whiteboard for the analysis is shown in Figure 7. The figure is intended to only

illustrate the analysis process and is intentionally illegible to preserve the anonymity of the interviewees.

3.3.2 Log analysis

The data pipelines that produced data for the data products log information about their runtime. These logs were analyzed to find out the effects of bad data on the pipeline, such as how often bad data caused errors in the pipeline and what kinds of errors bad data resulted in.

Errors were extracted from the logs produced by the pipeline along with the time information for when the error occurred as well as the error message. The timestamps were used to observe error frequency over time. The error messages were analyzed in 2 ways: firstly, unique errors were extracted along with how many times each unique error occurred. After that, each unique error message was categorized based on whether it was due to bad data or not. For example, an error message that indicated that an external service didn't respond on time was categorized as not being due to bad data while an error that was caused by missing values was categorized as being due to problems in the data.

3.3.3 Archive analysis

The log analysis previously was done to find out and analyze situations where bad-quality data had caused an error in the data pipeline and made it break. The archive analysis was performed to complement the log analysis. The purpose was to find and analyze data that was bad quality but had not caused an error in the pipeline.

As a part of the pipeline architecture, the data from the original source was archived. The archived data was validated against the defined data specification.

Data quality is defined in the research literature as "being fit for purpose". This fitness for purpose can be ensured by defining a specification for the data that the data must conform to. Data can then be validated against this specification to ensure that it is fit for purpose and good quality.

As discussed in the Literature review, problems in data quality can be divided into either context-dependant or context-independent issues. This distinction was also used when defining the data specification.

Context-independent indicators of problems in the data called data smells listed by Foidl et al. [27] were used as a starting point for the context-independent data specification. Context independence enables defining some requirements for the data without knowing the overall pipeline or the data product. A useful data specification can be initalized using context-independent requirements and can then be complemented with context-dependent requirements. These include for example specifying the required data types for columns in the data, specifying the expected format for a column containing datetime objects, and so on. These aspects of the data were identified by examining an example of data that was expected to be of good quality.

Data developers who had existing knowledge about the data were involved in defining the context-dependent aspects of the data quality. An example of a context-dependant data quality issue could be whether datetime values in a column can be in the future or not or whether some specific characters are allowed in a column that is sanitized during the pipeline and finding these characters in that column would indicate an error in that data sanitation.

For validating the archived data, a data specification was defined for each distinct dataset, and the archived data was validated against the defined data specification. The validation produced a list of identified problems in the data. These problems were grouped into unique errors and both the number of different unique errors as well as the number of occurrences of each unique error were noted.

3.3.4 Data Pipeline Analysis

Having an overview and understanding of the data pipeline used for transferring and modifying the data is required for the data validation implementation process. To understand the data pipeline, a team member from the development team was requested to describe the data flow in the data pipeline end-to-end, from the source where the data entered the pipeline up until the point where it was accessed by the data product. This was

combined with the existing documentation about the pipeline and its architecture. The primary focus was on understanding where the interim results of processing the data were stored. Secondary focus points were how the data flowed throughout the pipeline, where it was transformed, how the transformations were orchestrated and triggered, where the data was transferred from one place to another, and lastly where it was accessed by the data products.

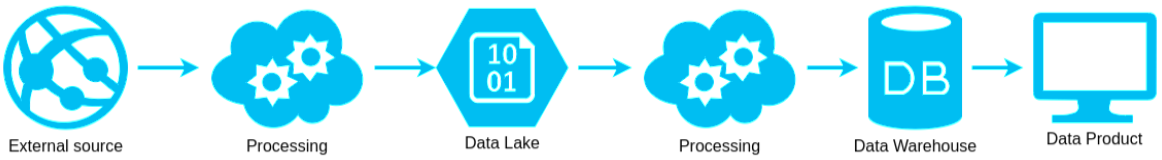

Figure 8 - Simplified data pipeline architecture

All of the pipelines used a similar architecture as presented in Figure 8. The starting point of the pipelines is an external data source, such as an API. The data pipeline extracts data from the source and stores it into a data lake. The processing included for example splitting the results from the external source into smaller chunks that are better suited for archival. From the data lake, the data is processed further and stored in a data warehouse where it is ready for consumption. The data product then pulls the data from the data warehouse. The technologies used for building the data pipeline include various tools such as Azure Blob Storage for the data lake, Azure Functions for the data processing, and Snowflake for data warehousing.

The sources of the data are as follows: weather data from the Finnish Meteorological Institute, electricity production statistics and spot prices from Fingrid, and company financial statistics from accounting software. The company's financial data is sensitive information that must be stored, transferred, and transformed securely.

3.4 Data Quality Validation in Practice

This section describes the iterative process for creating and evaluating the process model for implementing data quality validation in practice. The process of selecting the data validation tool for practically demonstrating the process model is described. After that, the objectives for the solution are defined based on the problems identified in the previous section, and an artifact is iteratively designed, demonstrated, and evaluated to fulfill those objectives. Once the artifact fulfills the research objectives, the research process and the results are communicated to a relevant audience.

3.4.1 Data validation tool comparison

The data validation can be implemented either by building a program for the data validation or by leveraging an existing tool. Building a custom program for the data validation would require a significant amount of work but would also provide a high level of control over the features of the tool, supported data connections, and more while an existing tool that is suitable for the project would require less work, support building up the data validation faster, as well as enable the research to focus on the data validation instead of tool-building [31]. For these reasons, an existing tool was preferred for practically implementing the data validation, if a suitable tool is found.

To find a suitable tool, existing tools were compared. The comparison was done in 3 stages:

1. Find tools for data validation
2. Filter the tools based on project requirements
3. Compare the suitable tools to each other

The filtering and comparison criteria for the tool were 2-fold:

1. Tool can be used to implement the data validation in this thesis
2. Tool is likely going to be a viable tool for data validation in the future

The first criterion is essential for finding a tool that is suitable for the project at all. For example, data must always be stored in an environment that the company owns and controls due to company policy. If a tool doesn't meet these requirements, it cannot be used for data validation in this project. The second criterion is important for assessing and

estimating the future viability of the tool. The data validation is expected to become a part of the data pipeline where it is implemented as a part of this research and needs to be maintained after this research has been completed. The tool receiving updates in the future significantly helps the development team maintain the data validation.

For the initial tool discovery, search engines were used to gather a list of existing data validation tools and frameworks.

The first criterion for the validation tool was evaluated by creating a list of requirements that the current project had for a tool. The list of requirements for the tool was created based on the context of the project, such as company policies and the technical implementation of the existing data pipeline. For example, the tool should support the data storage solution, Azure Blob Storage, that is used by the data pipeline. Each tool had to meet the specified requirements to be considered. If a tool didn't meet the required features, it was excluded from the next comparison. These requirements for the tools were:

- Supports the tools and technologies that the company uses such as data storage platform, data storage type, and deployment platform
- Supports the expected data volume
- Supports specifying the data specification as a declarative ruleset that data can be validated against
- Has an existing library of required rules or allows for developing custom rules
- Data is always processed on company-controlled premises
- Data is always stored on company-controlled premises
- Open source

The tool supporting the tools and technologies used in the data pipeline as well as the expected data volume were baseline requirements as the data validation wouldn’t be possible if these criteria were not fulfilled. As discussed in the Literature review chapter, previous research recommends using a declarative ruleset for validating data [5]. The selected tool should also either have the required rules or provide an option to build the rules. The company has guidelines and policies regarding information security, especially when handling sensitive and personally identifiable information. As one of the objectives for finding the tool includes being able to use the tool in the future and beyond this thesis, it is expected that the tool should comply with these policies regarding information

security. The company policies that are relevant for the tool selection are storing and handling the data on company-controlled premises. Additionally, open source software provide high transparency, publicly available source code, permissive licensing, and security [31] which were desired qualities for this thesis.

Whether each identified tool fulfilled these requirements were evaluated based on each tool's documentation in combination with the discussion by Döhmen et al. [5].

The second criterion, the future viability of the tool, was evaluated with 2 dimensions: current level of support and current popularity. These dimensions are expected to indicate the level of support and maintenance that the tool will have in the future and provide an estimation of the future viability of the tool. The level of support was measured with the number of active developers for a project and the number of commits the project has received during the last year. The popularity of a tool was measured by the amount of downloads and stars the tool had on GitHub. Python Package Index provides the number of downloaded packages publicly for all packages, and it was used for obtaining the download amounts [32]. All of these metrics are public, numerical, and comparable values that may indicate how well-maintained the project is now and in the future.

These results were then compared using a table and the tool that was most popular as well as best-supported was selected.

3.4.2 Objectives of the solution

This section describes how the objectives of the solution were defined. As described in the framework for Design Science Research, the objectives for the designed artifact should be defined before designing, demonstrating, and evaluating the artifact. The objectives help guide the design and development process as well as facilitate evaluating the results from demonstrating the developed artifact.

The objectives were defined based on the results from interviews, log analysis, and archive analysis that were described previously. The objectives were collected into a list of aspects that the solution should fulfill. The objectives were collected and formulated in a dichotomous format with the solution either fulfilling the objective or not instead of the

objectives being for example a scale. This was done to facilitate the design, demonstration, and evaluation of the artifact, and to avoid ambiguity when evaluating the artifact and deciding whether the objectives had been met and the iteration process could be stopped.

3.4.3 Design and development

This section describes the process for designing and developing the process model for incorporating the data validation. The next stage after defining the objectives of the desired solution in the Design Science Research framework as described in the Research Framework chapter earlier is designing and developing an artifact that aims to solve the research problem and fulfill the previously defined objectives.

The Design Science Research framework is iterative in nature and the different stages in the process are expected to be performed multiple times during the research. The assumption in the research framework is that new things are learned and discovered during the research process. This new information can then be utilized when creating the next version of the artifact on each iteration. This assumption was also utilized in this thesis when designing and developing the process model. The approach taken in this thesis was to build the process model iteratively and incrementally. This approach was chosen instead of building a complete initial process model that was expected to include all of the required steps to fulfill the defined objectives. As the assumption was that new information would be learned during the iteration process, this assumption was also extended to the process of designing the model. It was assumed that during the iterations, new information would be learned about the steps required to get to a complete process model that would fulfill the defined objectives. This approach was also assumed to reduce the length of the iterations which would enable faster and more iterations and lead to gaining more information and developing a better process model.

Building the artifact incrementally means that the process model is expanded on each iteration. Each iteration focuses on improving the process model to fulfill only a limited number of the defined objectives. For example, one iteration might concentrate on providing an initial version of the steps required for deploying the data validation. As the model fulfills more and more of the objectives of the solution as it is expanded on each

iteration, after enough iterations the model is expected to fulfill all of the defined objectives. In addition to incrementally expanding the artifact on each iteration, the artifact is also modified based on the new information that is gained from the previous iterations.

At the start of each iteration, the next objective was selected as the focus for this iteration. While the selection didn't follow any specific system as all of the objectives were expected to be met and strict prioritization wasn't required, the dependencies between the different objectives were estimated and that was used as a rough prioritization. Objectives that didn't have any unsolved dependencies were selected first, and the objectives that required other objectives to be solved first were selected after those objectives were met. For example, the process of deploying the data validation was considered only after the process of building data validation that could be deployed was solved first. Then, the existing model was modified in a way that was expected to satisfy the selected objective. Based on the understanding available on that iteration, new steps were added to the model, more details were added to existing steps if information about that was found during the previous iterations, steps or instructions that were found to be redundant were removed, and the steps and instructions were edited and rewritten if necessary as new information was uncovered during the iterations.

3.4.4 Demonstration

This section describes how the process model was followed to demonstrate the effectiveness of the model. After the process model had been designed, the next stage in the Design Science Research process model that was presented in the Research Framework section was to demonstrate the effectiveness and efficacy of the designed artifact. This can be done for example by using the artifact in an experiment or in an example case. In this thesis, as the developed artifact is a process model for data validation, demonstrating the efficacy in practice was done by following and utilizing the process model for practically incorporating data validation into an existing data pipeline. This process was repeated on every iteration using the latest version of the artifact.

The process of demonstrating the efficacy of the process model was to follow the steps and instructions described in the model as they were described in the model. After the process

model included a step for deploying the data validation to a pipeline, a new data pipeline was selected for each iteration. During the demonstration of the artifact and development of the data validation, new information such as missing instructions or invalid assumptions was noted so that the model could be improved on the next iteration.

3.4.5 Evaluation

This section describes how the results from the demonstration were evaluated to consider whether the process model fulfilled the defined requirements and objectives that were set for the solution. After the artifact has been designed and demonstrated, evaluation is the next stage in the Design Science Research framework that was described in the Research Framework section of the thesis. The evaluation stage is intended for assessing the efficacy of the artifact. In this thesis, the evaluation is done by comparing the results from demonstrating and utilizing the defined process model for data validation against the defined research objectives that the model should fulfill. The objectives were defined in a dichotomous format where the solution either did or did not fulfill each objective. As the objectives were defined in this manner, the results from the demonstration could be directly evaluated against the defined objectives for the solution by examining whether the solution fulfilled each objective or not.

The evaluation was performed after each iteration and is crucial for assessing what the next step in the research process is. As discussed in the Research Framework chapter, the iteration would continue based on the results from the evaluation stage. If the model did not fulfill the objectives and no other constraint didn't prevent further iterations and creating new artifacts, the process would proceed to the design and development stage to continue iterating on the artifact. Alternatively, if the model fulfills the research objectives or a constraint prevents from continuing to improve the artifact, the iteration is stopped and the research process advances to the next stage in the framework.

3.4.6 Communication

This section describes how the research and the results from the research were communicated. The final stage in the Design Science Research process framework is communicating and presenting the research to a relevant audience. As discussed in the Research Framework section of the thesis, communicating about the research and the results from the research is an important part of the research process. For this thesis, the results from the research were documented in a documentation tool used in internally, presented in the company internally, and described in this thesis.

The process model that resulted from the research was documented in Confluence, an internal documentation tool used in the company. The results from the research were also presented internally in the company. The team members in the team that is responsible for developing data solutions periodically host demonstrations to the other team members on new tools and technologies they have been using. The demonstrations are performed using Teams and are typically between 30 to 60 minutes. The results from this thesis were demonstrated to the team members in such session. The demonstration included how the data validation works, how the reporting and notifications produced by the data validation can be used and showing the reports and notifications the tool produced, how to develop and maintain the data validation, going through how the solution was integrated into the data pipeline, and more. The feedback from the demonstration was positive and the tool and the proces model were found to be useful.

Lastly, the research, including the research process and the results from the research, are described in this thesis.

3.5 Resources

This section describes the different resources used and required for the thesis. The resources include software and hardware, resources from the cloud, data, and support from the development team.

The software resources used for the data validation included Python, different services on the Azure platform that the data validation was already built on, such as Azure Blob Storage and Azure Functions, and Great Expectations was used for performing the data validation. Documentation software was used for creating documentation about the data

validation and transferring knowledge to the development team. As the data validation was deployed to the cloud, no specific hardware resources beyond a development computer were required. Cloud resources for the deployment were provided by the company and the data validation was able to leverage the existing cloud infrastructure that the existing data pipeline was deployed on.

The data resources that were required for implementing the data validation included the data that was processed inside the data pipelines in addition to the log data discussed earlier.

The support from the development team was necessary for implementing the data validation. The collaboration with the developers provided valuable insights into the data quality as well as technical assistance for the implementation process.

4 Results

This chapter describes the identified problems relating to data quality, the defined research objectives for the process model as well as the process model for incorporating data quality validation into a data pipeline.

4.1 Current problems related to data quality

This section describes the results from the interviews, log analysis, and archive analysis and presents the problems related to data quality that were identified.

4.1.1 Interviews

This section describes the results of analyzing the interviews with the data developers and the data consumers. 5 people were interviewed, including 2 data consumers and 3 data developers. The length of the interviews was between 15 and 20 minutes. The interviews were recorded to facilitate analysis.

Overall, the data consumers considered data quality to be really important. Good data quality was seen as crucial for the data products to be valuable for the data consumers. In order for the data to be useful for them, they needed to have access to the data, the values must be accurate and trustworthy, the data needs to be up-to-date, and the data should be complete. While the data consumers regarded data timeliness to be important, the required window for the data to be timely was smaller for electricity price forecasting.

"It is fine if the - - data is a couple of hours late but after a day the data is useless"
- data consumer

The data developers also regarded data quality as important. From their perspective, good data quality caused fewer errors in the data pipeline. The data developers didn't have an existing process for evaluating or monitoring data quality besides errors produced by the

data pipeline. Notifications about data quality problems that didn't cause an error in the data pipeline were received from the data consumers. Finding out the cause and location of a problem in the data was considered to be relatively hard. Two of the developers considered finding the reason to be hard and only one developer thought that finding the problems wasn't hard, only tedious. All of the developers agreed that finding the source and cause of a problem in the data required a lot of work.

"It might take months for anyone to realize there was a problem with the data. It's typically [data consumer] that sends a message about the data being all over the place."

- data developer

"After [data consumer] sends a message that there's something wrong with the data, it takes a lot of work to find out where the data went wrong and why."

- data developer

4.1.2 Log analysis

This section describes the results of analyzing the data pipeline logs. Figure 9 shows the number of errors logged by the different pipelines over time. The errors logged by the different pipelines are grouped in the figure using color coding. 3 pipelines had produced errors during the observation period and were color-coded in the figure using blue, red, and purple. The length of the observation period is 3 months as the error logs were configured to be stored for that long. The observation period was from late April to late July, though the initial period is not visible in the figure as no errors were logged.

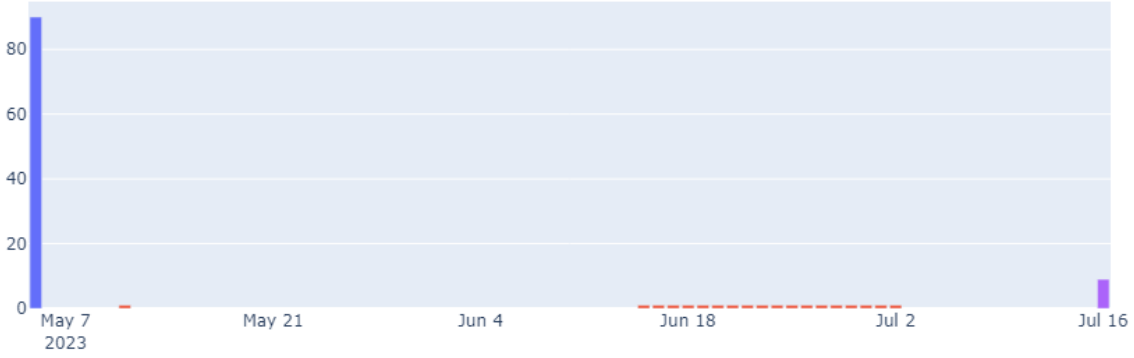

Figure 9 - The number of errors produced by the data pipelines

The amount of errors in the data pipeline that is represented by the blue bar changes the scale of the graph such much that analyzing the amount of errors in other pipelines is not feasible. Figure 10 shows the number of errors when the errors from the blue pipeline are excluded and the scale is adjusted.

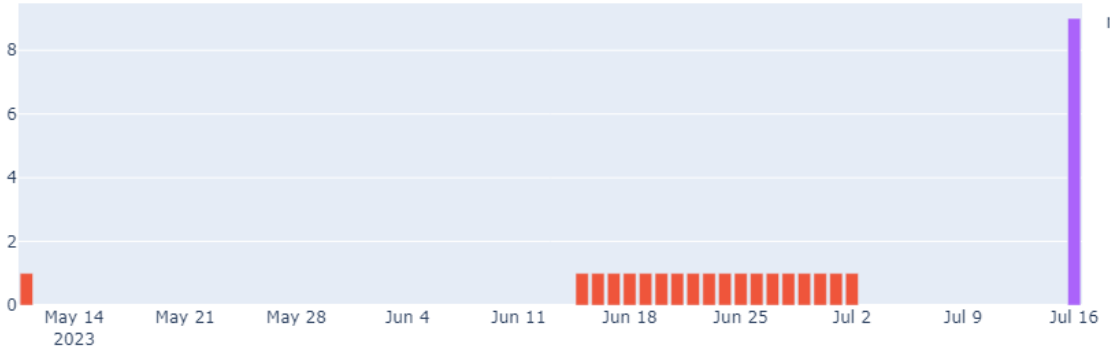

Figure 10 - The number of errors produced by the pipelines (blue excluded)

The error messages were further inspected to better understand the errors in the data pipelines, and analyze whether the errors were caused by bad data or not. The error messages for each color were found to be identical, and only 3 unique errors were found in total. Additionally, based on the error messages, the cause for the errors was evaluated not to be due to bad data but instead due to something that was not related to data quality and therefore outside of the scope of this thesis.

In conclusion, no problems that would have been caused by bad data were identified.

4.1.3 Archive analysis

This section describes the results of analyzing the data archives. The analyzed archives included electricity spot prices, electricity production, sun radiation, and weather data. The accounting data was not validated as archived data didn’t exist.

For the electricity spot prices, 986 files were analyzed. Each file was expected to have a value from each hour and 24 values per day, except for days affected by daylight saving time. 1 file was discovered that contained only 2 rows.

The archived electricity production data contained 11888 files and 285312 rows. Analyzing the electricity production data took roughly 2 hours. No problems in the data were found.

Similarly, no problems were found when validating archived sun radiation and weather data. 37986 files and 911664 files were validated in total.

Overall, the archived data was found to be of good quality and no major problems were found.

4.1.4 Identified problems

Based on the interviews, log analysis, and archive analysis, the most prominent problems were identified to be the delay in discovering problems in the data and finding the source for bad data requiring a significant amount of work. These were identified mainly from the interviews. The log and archive analysis didn't uncover any major problems. The file that was found to have only 2 rows instead of 24 further supports the discovery that there is a delay in discovering bad data.

4.2 Tool comparison

This section presents the results of finding a suitable data validation tool for research as well as a short description of the selected tool.

Based on the initial search, the following tools for data validation were identified in alphabetical order: Deequ, DuckDQ, Great Expectations, Hooqu, Pandera, Soda, and Telmai.

Tool	Supports	Supports	Declarative	Library	On-premise	On-premise	Open

	used technologies	expected data volume	ruleset	of rules	computation	storage	source
Deequ [33]	True	True	True	True	True	True	True
DuckDQ [5]	True	True	True	True	True	True	True
Great Expectations [34], [35]	True	True	True	True	True	True	True
Hooqu [36]	True	True	True	True	True	True	True
Pandera [37]	True	True	True	True	True	True	True
Soda [38]	True	True	True	True	True	True	True
Telmai [39]	True	True	True	True	False	False	False

Table 2 - Validation tools compared against the defined requirements

The second step in evaluating the data validation tools was to observe whether the tools fulfilled the requirements defined for the tools. Table 2 presents the results of this process. Only the data validation tool Telmai didn't fulfill the requirements due to it being a Saas platform and the validation being computed in their premises and since suitable open source options were available, was excluded from the comparison. All of the other tools fulfilled these requirements and any of these tools could be selected as the tool used in the research process. In the next stage, the suitable data validation tools are compared together to estimate the future viability.

Tool	Downloads	Repository contributors	GitHub Stars	Commits
Great Expectations	420 000	394	8670	1999
Deequ	170 000	56	2850	30
Hooqu	0	2	21	0
Soda	13 000	53	1454	288

Pandera	32 000	103	2513	123
DuckDQ	-	1	18	2

Table 3 - Data quality tool comparison

Table 3 presents the results from comparing the data validation tools. The clear winner in all of the dimensions was Great Expectations which was chosen as the tool for performing the data validation in the research process.

Great Expectations is an open-source data validation tool with an Apache-2.0 license with a tagline of “Always know what to expect from your data” [34]. It is built using Python and leverages various tools in the Python ecosystem, such as pandas, PySpark, and SQLAlchemy. This means that Great Expectations can be used in any environment that supports Python. Great Expectations supports a variety of data sources, such as S3, Microsoft Blob Storage, Snowflake, PostgreSQL, and more [35]. Great Expectations also has built-in capability to send notifications and produce reports from the data validation results. Great Expectations is suitable for handling confidential data as it can be self-hosted, the data is processed as close to the storage as possible, only the results of the data validation leave the compute platform, and the data that is included in the reports can be configured to not include any sensitive information. Great Expectations offers an optional cloud service, Great Expectations Cloud, that can facilitate storing the data validation configurations as well as storing, producing, and sharing the validation reports [40].

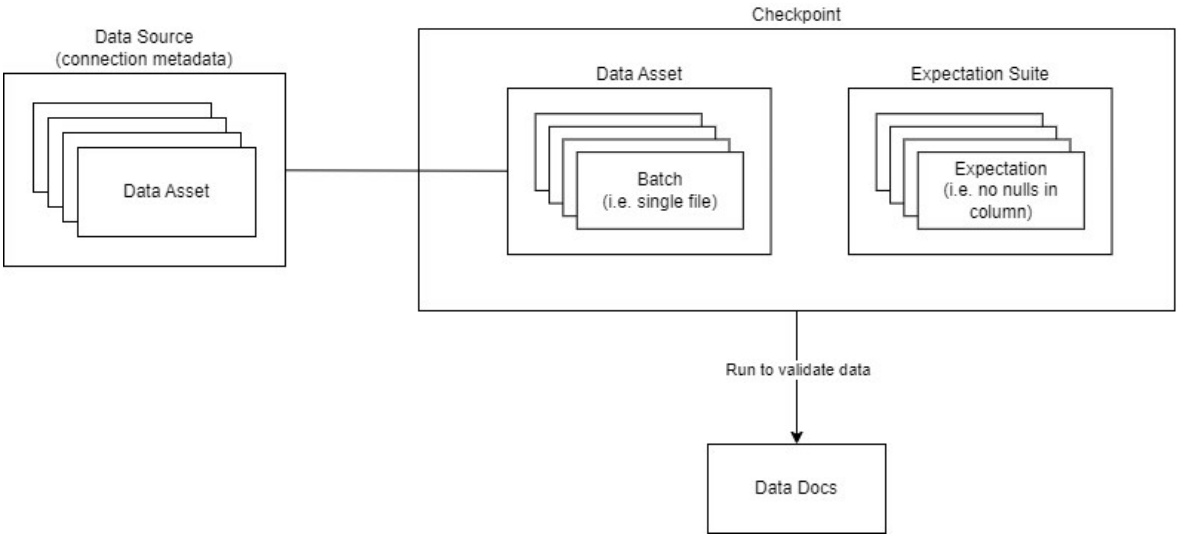

Figure 11 - Great Expectations structure

Figure 11 presents the different concepts used in Great Expectations [35]. A Data Source is used to store the connection metadata, such as a database URL. Data Assets correspond to a collection of records stored in the Data Source, such as a collection of related files in a data lake. Multiple Batches of data can be generated from a Data Asset. If a Data Asset is a collection of related files in a data lake, each Batch can be thought of as corresponding to a single file in the data lake. Great Expectations offers declarative statements called Expectations for defining the requirements for a dataset. These Expectations are collected together to form an Expectation Suite. Data Assets and Expectation Suites are combined to produce a Checkpoint, which encapsulates the instructions for validating data. Checkpoints are practically used for data validation and producing the validation reports called Data Docs.

4.3 Research Objectives

The research objectives are used for designing and evaluating the data validation process model. The research objectives are defined based on the identified problems related to data quality. The identified problems were a delay in finding out bad data and solving problems with the data requiring a lot of work. This section describes the research objectives that were defined for the artifact.

The resulting process model should be relatively simple and straightforward to follow so that implementing the data quality validation is justifiable and doesn't require significant training to implement. Data pipelines can be built using various technologies and ecosystems so the resulting process model should be generalizable and not tied to a specific tool or ecosystem so that it can be used in different contexts. During the initial data pipeline analysis, it was noted that even the pipelines used at Virnex leveraged multiple technologies and ecosystems, such as Azure and Snowflake. The process model should be repeatable and reliably produce the same results.

Following the process model should result in data validation having been incorporated into a data pipeline to continuously monitor the data and data quality flowing through the data pipeline. This is expected to decrease the delay in discovering bad data as well as facilitate locating the source of bad data.

As noted when comparing the data validation tools, information security is important especially when processing sensitive information. The process model should take information security into account.

To facilitate evaluating the artifact, these requirements are transformed into a boolean format that is unambiguous to evaluate:

- The process model is simple and straightforward to follow
- Process model is not tool or ecosystem-specific
- Process model is repeatable
- Data validation is incorporated into a data pipeline
- Data in the pipeline is continuously validated
- No sensitive information stored or processed outside of company controlled premises

4.4 Data validation process model

This section presents the results from iteratively developing the process model for data quality validation. The section presents the generalizable 6-stage process model for incorporating data quality validation that resulted from the iterations, describes each stage

in the model in more detail, and details the steps that were concretely taken when following the process model in the available cases for implementing data validation.

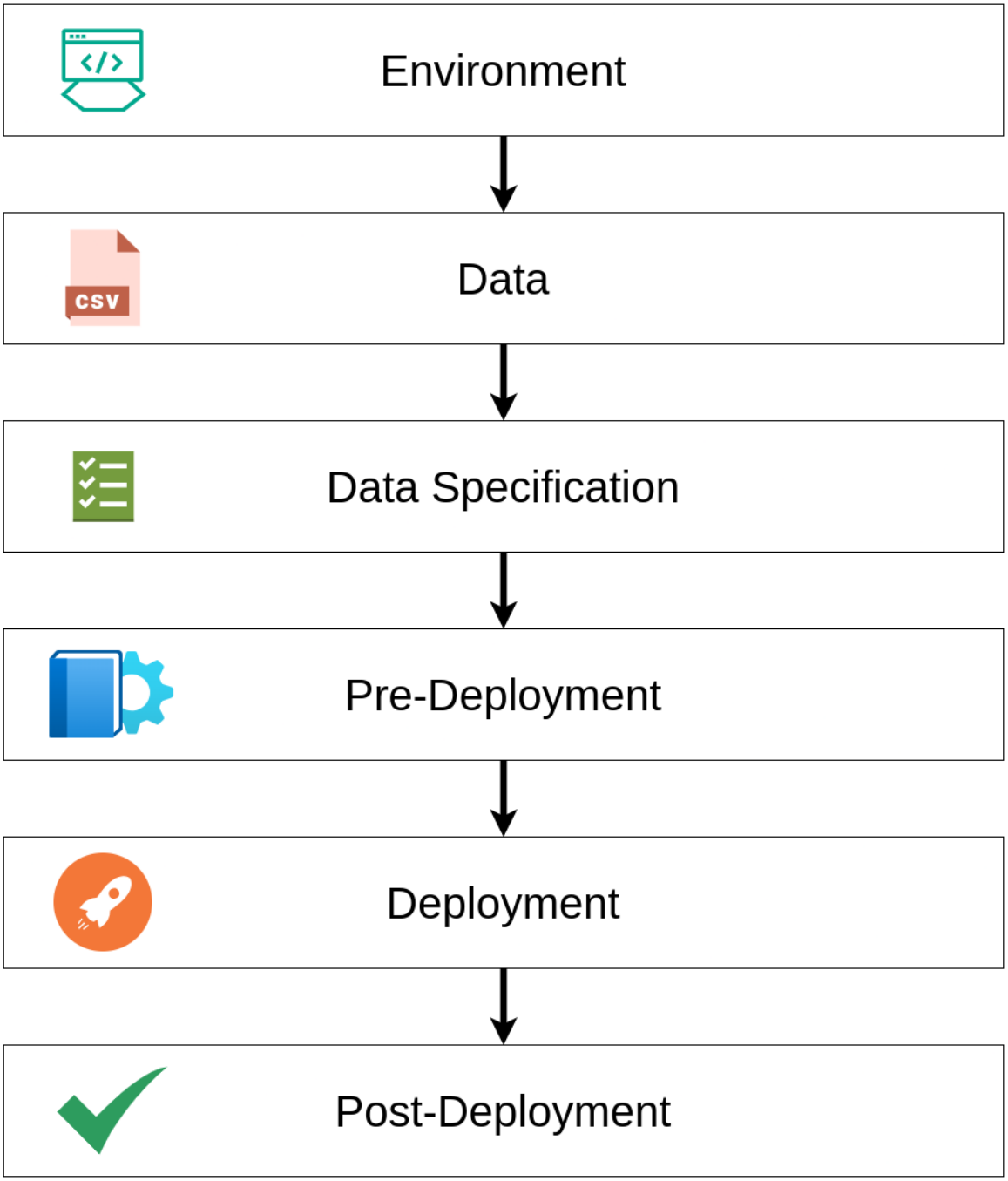

Figure 12 - Process model for data quality validation

Figure 12 presents the process model for the 6-stage process model for incorporating data quality validation into a data pipeline that was developed during the research process. The generic process model contains the following stages:

1. Environment
2. Data
3. Data Specification
4. Pre-deployment
5. Deployment
6. Post-deployment

4.4.1 Environment

The first step in the process model is to initialize a development environment. This includes steps such as environment isolation, external package management, version control, and project initialization. While the specific steps for setting up the development environment depend on the used tools and ecosystems, the overall objective of this step is to have a development environment ready for exploring the data and creating a specification for the data.

The data validation tool Great Expectations and the Python ecosystem were used in this thesis. The steps taken with these tools to set up an isolated development environment and manage external packages are done using virtual environments. Virtual environments in the Python ecosystem are used for containing and isolating a project and the external dependencies required by the project [41]. A virtual environment is created and the external packages are installed and managed inside the virtual environment. The Great Expectations package was installed along with optional dependencies for Azure. No other packages were installed as Great Expectations includes the packages, such as pandas for data manipulation, that were required for developing data validation as dependencies [34]. The initialization command provided by Great Expectations was run to create the files and project structure required for the development [35]. Version control was also initialized as a part of setting up the development environment. The selected version control system was Git, a free, open-source, fast, and lightweight version control system [42].

4.4.2 Data

The second step in the process is to access the data that should be validated. The objective of this step is to have a sample of the data that can be used for development and the tool is successfully connected to the data. The steps for this are finding a dataset that can be used for development, having that data stored in a way that it can be used for development, connecting the tool to that data, and confirming that the tool has access to the data.

The first step is to find a sample dataset that is suitable for the development. Selecting a good sample of the data is important as a specification for the data is built using the selected sample data and this specification is then later used for validating data that goes through the data pipeline. The assumption is that the specification that was built based on the sample data is valid and can be used to validate data in the pipeline. Therefore, the sample should be of good quality without any known issues. Selecting a larger sample or multiple samples may be helpful but having larger or more samples might not be possible in some situations. Finding good sample data may not be trivial, depending on how the data pipeline stores and processes the data. For example, the files that should be validated might not exist if they are only stored as a by-product of the data pipeline normally processing data and are deleted as soon as they are not needed. Consulting other people in the organization is helpful for solving problems with obtaining sample data.

After finding suitable sample data, the data should be stored in a way that the data validation tool can access it when developing the data validation. There are multiple options for storing the data. A copy of the data can be downloaded locally, a specific environment may be available for development purposes, and so on. There are some aspects that should be considered when deciding this. Developing the data validation shouldn’t have an impact on the system that produces the data and the system also shouldn’t impact developing the data validation. The development being done on a local copy of the dataset ensures that the dataset is stable and the data pipeline that produced the data won’t change it while developing the data validation. The dataset may be loaded multiple times when developing the data validation which should be taken into account. The development may be faster when using a local copy of the dataset instead of

downloading the data from the cloud, depending on the size of the dataset, connection speed, and more. Additionally, fees, available resources, and other project-specific aspects should be considered when deciding on where the data that is used for development should be stored.

The next step is to connect the used tool to the data. The process for this varies depending on for example which tool is used and how the data is stored. After the data has been connected, confirming that the correct data is found and can be accessed by the tool is useful.

In the cases observed in this thesis, the data pipeline produced the files as interim results of the processing, and the files were deleted as the processing was completed. However, the cloud file hosting service, Azure Blob Storage, was configured to only soft delete files. This enabled un-deleting the most recent files, downloading a copy of the files locally for development, and then deleting the files again to avoid disturbing the pipeline. A local copy was downloaded instead of connecting directly to the cloud environment to avoid impacting the production system, speed up the development as the file didn't have to be downloaded from the cloud, ensure that the file is stable and doesn't change while developing the data validation, and for being able to modify the dataset if needed. Great Expectations was connected to the dataset by following the instructions in the documentation. The connection was confirmed by listing the available files and inspecting the header and the first few lines of the connected data.

4.4.3 Data specification

After the development environment has been set up and good sample data is available, the next stage is to create a specification for the data. The objective of this stage is to have a specification that can be used for validating other data against it. The steps for creating the specification include exploring the data, creating the specification, defining the data secrecy level, documenting the specification, and storing the specification in a format that can later be used for validating data.

The first step is to explore the data to better understand the data and the aspects of the data that should be considered for the data validation. While exploratory data analysis is a field

in its own right and outside of the scope of this thesis, some useful aspects include summary statistics, observing the data types, small samples, and visualizations. Summary statistics include computed statistics about the data that provide an overview of the data such as minimum and maximum values, median, frequency, and more [43]. Values such as row and column count, number of unique values, and number of missing values are also useful to examine. Observing the data types is important for multiple reasons, such as discovering indicators of problems in the data (data smells) [27], [44], tailoring the data specification based on the data types, and including the data types in the specification. As discussed later in more detail, some of the specifications for the data can only be applied to certain types of data. For example, email address validation can only be meaningfully applied to textual data. Inspecting a small sample of the data was found to be useful for better understanding data and what specifications may be relevant for the data. A sample may be for example the first 10 lines of the data but can vary depending on for example the dataset and project context. Visualizing the data is a powerful tool for exploratory data analysis and can provide insights about the data [45] that can be utilized when defining the data specification.

Due to the iterative nature of the research framework that was used for the research, Design Science Research, the design and demonstration stages for creating the data validation model were repeated multiple times during the research process. As a result of this iteration and as a part of this model, a framework for defining the data validation was developed to facilitate and speed up the process. This framework consisted of defining requirements related to the dimensions of the dataset, completeness, uniqueness, statistical properties, and allowed values. While more dimensions most likely exist, finding them was outside of the scope of this thesis and an opportunity for further research, which is discussed further in the Discussion chapter. These aspects of the data are typically project and context-dependent, and consulting other members of the organization is recommended in regard to any uncertainty on which aspects should be included in the data specification and how. For example, the company policy might be to not store social security numbers, and finding them in the dataset should prompt a discussion with the other members of the organization instead of simply validating the social security numbers.

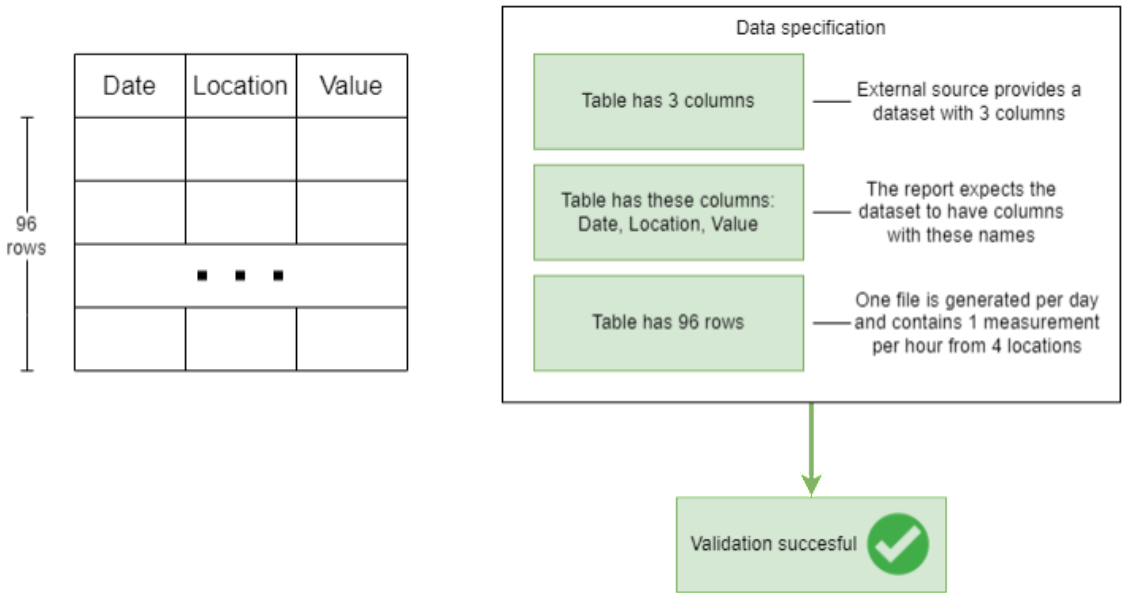

Figure 13 - Example of dataset dimension requirements

The requirements related to the dataset dimensions include the number of rows and columns in the dataset, required column names, and the data types for the columns. Examples of these requirements are presented in Figure 13 where the table both has 3 columns and 96 rows. Since the table is also expected to have 3 columns and 96 rows, the data passes the validation is successfully. The amount of rows and columns is used to essentially define and monitor the size of the dataset. The number of rows and columns that are required to be in the dataset can be a single value or a range with an upper and lower limit. The required column names are important to define especially in cases where other parts of the data pipeline rely on specific columns being present in the dataset. The data type of each column should be included in the data specification as the data type indicates what data is stored in the column and a change in the data type suggests a decrease in the data quality as well as potential problems further in the data pipeline.

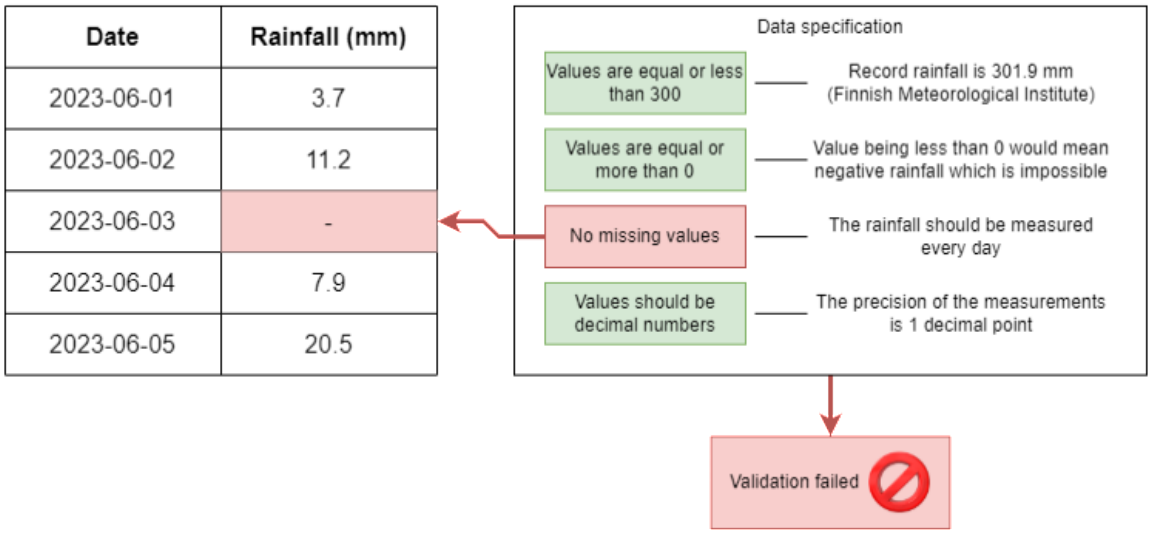

Figure 14 - Example of a failed validation

As discussed in the Literature review chapter, completeness can be viewed as the percentage of missing values in the data [25]. The required level of completeness or the degree to which values can be missing from each column should be included in the data specification. Figure 14 presents an example where the data specification is set to require that there are no missing values in the data but one value from the Rainfall column is missing, which causes the data quality validation to fail. As 4 out of the 5 total values are present in the data, the completeness of the data in this example is 4/5 or 80% while the required level in this case is 100%. The figure also presents examples of other data requirements, such as allowed minimum and maximum values.

The required uniqueness of the data is also included in the data specification. Individual columns such as a column containing identification values may be required to be unique, or a combination of columns may be required to be unique in combination. For example, a duplicate in a column that contains an identification number for different transactions in a store is a strong indicator of a problem somewhere in the data pipeline or the source systems producing the data.

Requirements for the statistical properties can be defined for numerical columns. The statistical properties may be, for example, the minimum and maximum values, the mean or median values in the column, and the standard deviation. It is useful to note that these

values may change between different datasets processed by the data pipeline. Therefore setting an acceptable upper and lower range for these values is recommended. For example, the acceptable average of a column might be defined to be between 10 and 20 instead of an exact value.

The individual allowed values in the columns may be defined using a set of acceptable values or by using rule-based validation. Defining a set of allowed values is a good option when the set of approved values is limited and known in advance. For example, a support ticket system might only accept 3 different types of support requests, and the type of support request is included in each request. In this case, a column is required to have only values from the approved set of options. A rule-based system can also be used for validating data. A rule-based system is useful for validating textual data that follows some rules that can be encapsulated for example using regular expressions. Examples of such data include email addresses, file paths, HEX color codes (i.e. #FFFFFF), and more.

The data secrecy level in the context of the data specification relates to whether the data should be visible in the outputs produced by the data validation tool, such as reports and notifications. Some data handled by the data pipelines may include sensitive information such as social security numbers which should be kept secret and shouldn’t be included in the outputs. Other people in the organization should be consulted regarding the different secrecy levels that should be applied to the data as they are dependent on the project context and organization policies. The tool used for the data validation may impose some limitations on how much the reports and other generated outputs can be modified and controlled. The documentation for the tool should be consulted in regard to which secrecy levels are available.

Documenting the data specification was found to facilitate maintaining the data specification and data validation. Documentation can mean for example explaining why specific requirements were chosen, where the selected values came from, and organizational policies that guided certain decisions. The documentation can be stored in any location, such as an internal documentation tool that might be used already at the organization, or included in the data specification if the data validation tool supports it.

Lastly, the data specification should be stored in a format where it can be used for validating data. This essentially means using the selected data validation tool to store the

specification. The process for storing the specification varies depending on the tool and therefore the documentation for the tool should be used for instructions on how to store the specification.

In this thesis, the data wrangling tool pandas was used for the data exploration. Pandas was the default data processing engine for Great Expectations and was automatically installed in the development environment as a dependency for Great Expectations. The approach taken in the thesis was to use the default data processing engine until some obstacle in implementing the data validation required changing it. However, no such obstacles were faced and pandas was used throughout the research process. In addition to supporting all of the data validation use cases that were encountered during the research, pandas also provides the required functionality out of the box for observing the summary statistics and data types of a dataset as well as examining a sample of the data [46]. The specification was built based on the exploration and by following the framework described previously. Most of the processed data was publicly available statistics and the default secrecy level set by the tool was used. The secrecy level was set to the strictest possible option for the datasets that contained personal information as per company policy which meant that no personal information was included in any of the outputs produced from the data validation. The selected data validation tool, Great Expectations, supported including comments as a part of the data specification which were used for documenting the requirements. In the context of the tool, the individual requirements are called Expectations and are stored in an Expectation Suite. An Expectation Suite was created to store the specification. Great Expectations abstracts storing the Expectation Suite into the project files from which they can be retrieved when validating data against the Expectation Suite. A Checkpoint was also created, which essentially combines the instructions for how to connect to a data source with an Expectation Suite.

4.4.4 Pre-Deployment

The next step in the model is to prepare the deployment environment as well as the data specification for deployment. This includes setting up the connection to the real data source, building the functionality for executing the data validation, identifying how the

data validation should be triggered, and defining the list of actions that should be executed after the data validation has been performed.

The data specification can be developed using a local sample of the data flowing through the pipeline. To validate the actual data in the pipeline, the method for connecting the data validation tool to the data may need to be changed. For example, if the data was stored locally for development but the production system stores the data in a data lake, the data connection to the data lake needs to be set up. The exact process for setting up the connection depends on the selected data validation tool and how the data is stored. The documentation for the tool is expected to have instructions for connecting the tool to the data source. Additionally, the deployment environment may need to be prepared for the data validation, for example by storing secret values required for connecting to the data in environment variables.

The next step is to identify and define how the data validation is triggered and how it fits into the existing data pipeline. As described in further detail in the Literature review chapter, data pipelines use some method, such as an orchestration tool, to organize and trigger the processes in the data pipeline [16]. Triggering the data validation needs to be incorporated into the existing data pipeline orchestration. The results from the data validation may also impact the further processing of the data pipeline, which should also be defined. For example, the data pipeline may be configured to halt processing data any further if the data validation identifies erroneous, problematic, or low-quality data. The data validation needs to be executed somehow, which means that this step may also involve writing code for performing the data validation. The organization may have processes and standards for writing new code, such as peer review or test coverage, which should be followed. The execution should be implemented in a way that it can be triggered, for example by the pipeline orchestration tool. When implementing the execution of the data validation, the broader context where the data validation is situated in the data pipeline should be taken into account to for example avoid race conditions and ensure that the data subject to validation exists and is available. It was discovered during the research process that depending on the data lifetime and how the data pipeline processing is triggered, the data that should be validated may have been deleted before the data validation has been able to access and validate the data, which led to errors and unsuccessful validation. As the data is validated against the data specification defined in

the previous step, the data specification should be included in the implementation and be accessible for the code or process that performs the data validation. Implementing proper error handling and graceful timeouts for the data validation were also found during the research process to be important to not disrupt the normal functioning of the data pipeline if the data validation tool encounters a problem or an error, especially if the error is not related to bad data but for example a bug in the validation tool.

The last step in preparing for the deployment is configuring the different actions that should be taken after the data validation, such as generating reports and sending notifications about the validation results. As these actions may include the data from the datasets, the data secrecy levels defined in the data specification should be adhered to when producing outputs based on the results from the data validation. The data validation tool may have built-in capabilities for these functionalities, such as producing reports from the data validation results which can be leveraged. The reports may, for example, contain the status of the validation (success or failure), examples of the values that didn't satisfy the data specification, information on how to locate the faulty data points, historical results from previous validations, and more. Setting up notifications about failed data validations may be useful for locating and fixing problems with the data faster.

In this thesis, the data source was configured to connect to a data lake, Azure Blob Storage, that was used by the data pipeline for storing the interim output from the data processing. Connecting to the data required setting up authentication between the tool and the Azure Blob Storage. This meant that the data validation code had to have the authentication method and credentials to connect to the data that needed to be validated. The data validation tool Great Expectation supports connecting to Azure Blob Storage using connection strings as authentication credentials [35]. Enabling this support only required installing some optional dependency packages. The authentication credentials were already defined for the production environment as the normal operation of the data pipeline utilized these same credentials for reading and storing the interim datasets that were created as a part of the processing. The data validation tool was able to leverage these existing credentials that were already used in by data pipeline.

The data pipelines that were studied in this thesis were implemented using Python which meant that a Python environment was already set up in the deployment environment. As both the pipeline and the data validation tool used Python, the required code and

environment changes were minimal and the data validation could be integrated into the data pipeline with only a few additional lines of code. The code that performed the data validation was added directly into the data pipeline code base as a function that could be called from the other parts of the data pipeline. Calling the code that performed the data validation was integrated directly into the pipeline code that processed the data. This was done after discovering after one iteration of the development-demonstration-evaluation cycle that the data that should have been validated was deleted due to the data pipeline orchestration and the configured data lifetime causing race conditions. This resulted in the data sometimes being available for the validation and sometimes the validation failing and producing an error because the data wasn't available. This was solved by integrating the validation code more tightly into the data pipeline processing and making the data processing execute the validation which eliminated the race conditions. Additionally, the data validation was implemented as a data pipeline block that could be called using a data orchestration tool to enable flexibility for the data pipeline orchestration in the future. Figure 15 demonstrates the data pipeline block that can be used for data validation. The block takes in some data, validates it against the data specification, and outputs for example the validation and a data quality report.

Figure 15 - Data validation pipeline block

Error handling and graceful timeout were added for the data validation code to ensure that the validation did not disrupt the existing pipeline's operations and prevent the production of the required data. The expected runtime of the data pipeline was derived from the interviews for what was the expected timeliness of the data as well as by consulting the developers if a longer runtime might cause technical problems. The length of the timeout was set to be short enough that it would not disrupt the data pipeline. Another option that was considered was to run the data validation asynchronously and independently of the data pipeline. This would enable for example performing longer, larger, and more extensive validations on the data or validating larger amounts of data. This was not implemented for this case as it would make the pipeline more complex as well as require changes to the pipeline, such as when and how data deletion and life cycle are handled. The first method was selected as it required only minor changes or no changes at all for the existing data pipeline.

The solution built during the thesis utilized Great Expectations Cloud, which meant that the configuration files such as the data specifications are hosted by Great Expectations and are fetched during runtime. The compute instance where the data is validated only needs to have access to the internet and the latest version of the configurations is fetched dynamically right before validating the data. This made maintaining and updating the data specifications simple and straightforward. Authentication to the Cloud service was done by storing authentication credentials to environment variables in the deployment environment. While this option was not used in this case, Great Expectations also provides an option for self-hosting the configuration files, which wouldn’t require the compute instance to have a connection to the Internet.

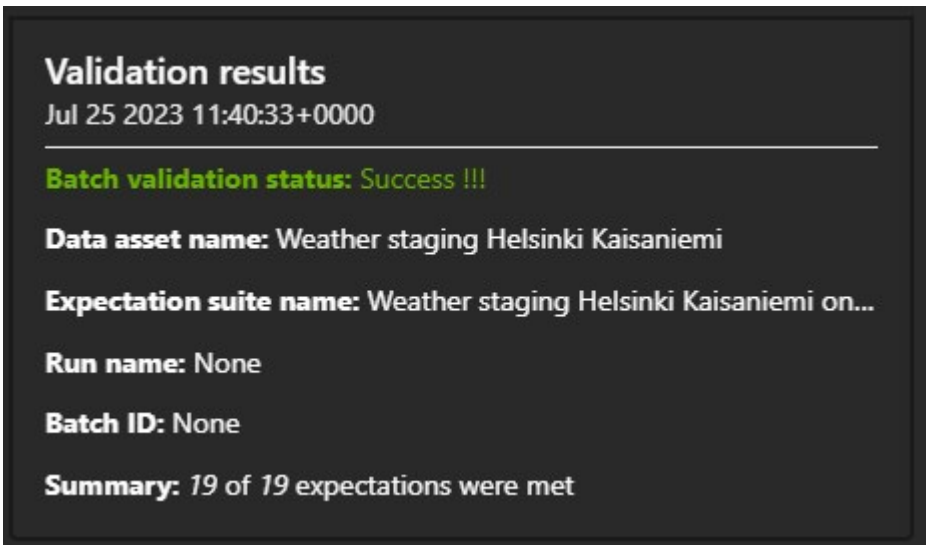

Figure 16 - Example data validation notification

After the data validation, Great Expectations was configured to send a notification if the data validation failed as well as produce a report from the results of each data validation.

Great Expectations has a built-in mechanism to send these notifications to communication software such as Microsoft Teams and Slack. Figure 16 shows an example of a notification sent by Great Expectations after a successful validation. Great Expectations can be configured to send the message either every time the pipeline is run, or only on failed validations. In this thesis, it was configured to send a message to Teams, the tool used by the company for internal communications only when the data validation failed and the validated data didn't meet the defined Expectations. A specific Teams channel was created solely for the data validation notifications and a web hook to that channel was stored in the environment variables of the deployment environment.

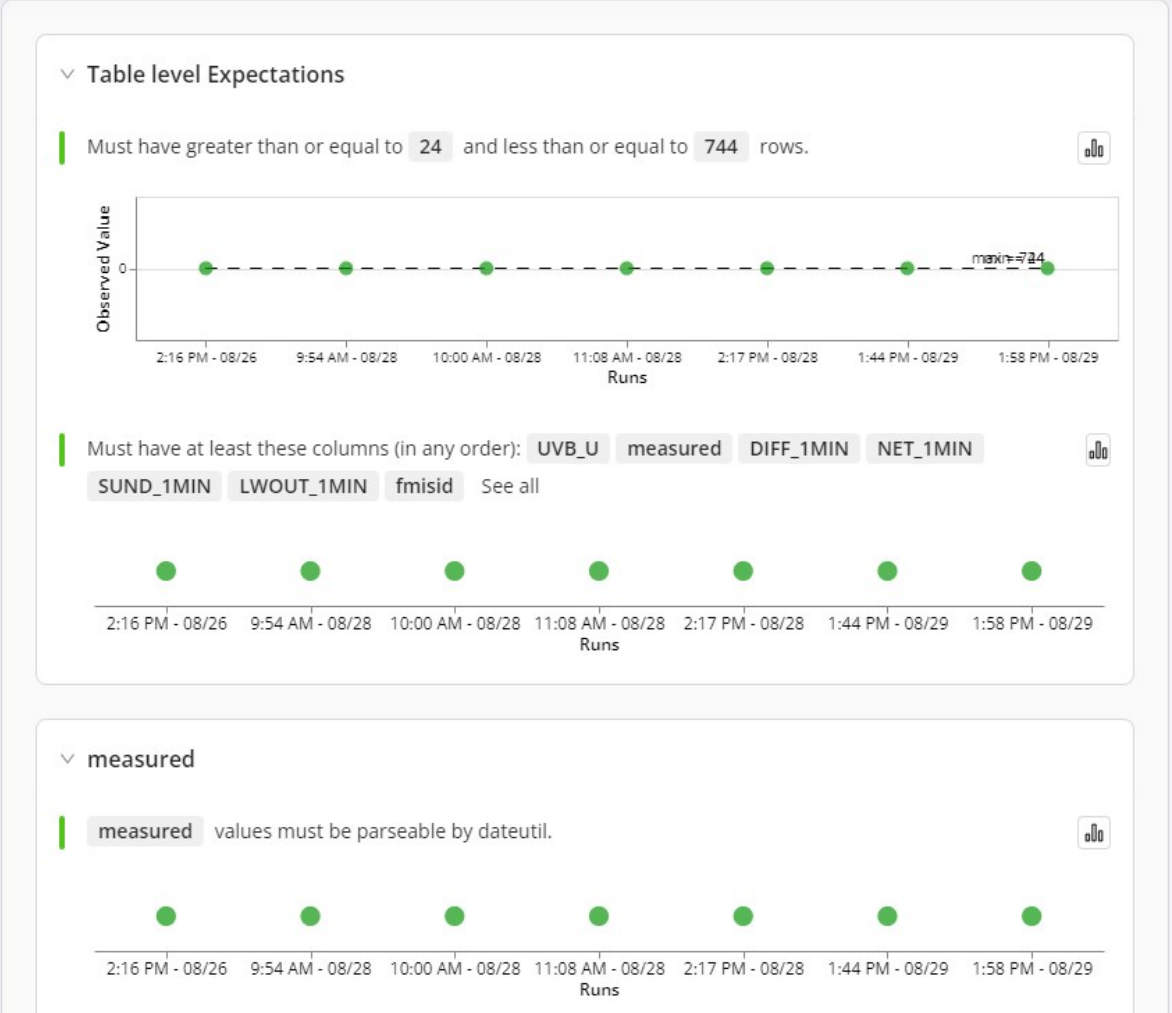

Figure 17 - Example of a validation report in GX Cloud

Great Expectations has also a built-in functionality for generating reports based on the results of the data validation. As the data validation was built using Great Expectations Cloud, the results of the validations were automatically sent to the Cloud service which generated reports from the results. Figure 17 shows an example of a data validation report in Great Expectations Cloud. The report includes information about the latest validation such as the overall result of the validation, the observed values for each metric, and the defined requirements or Expectations for each metric. The reports also contain the validation history which can be used to track and observe the state of the data over time. The reports from previous validations are also available. The information stored and displayed in the reports conforms to the secrecy levels defined in the data specification,

which means that no sensitive information is displayed in the reports or stored in the Great Expectations Cloud.

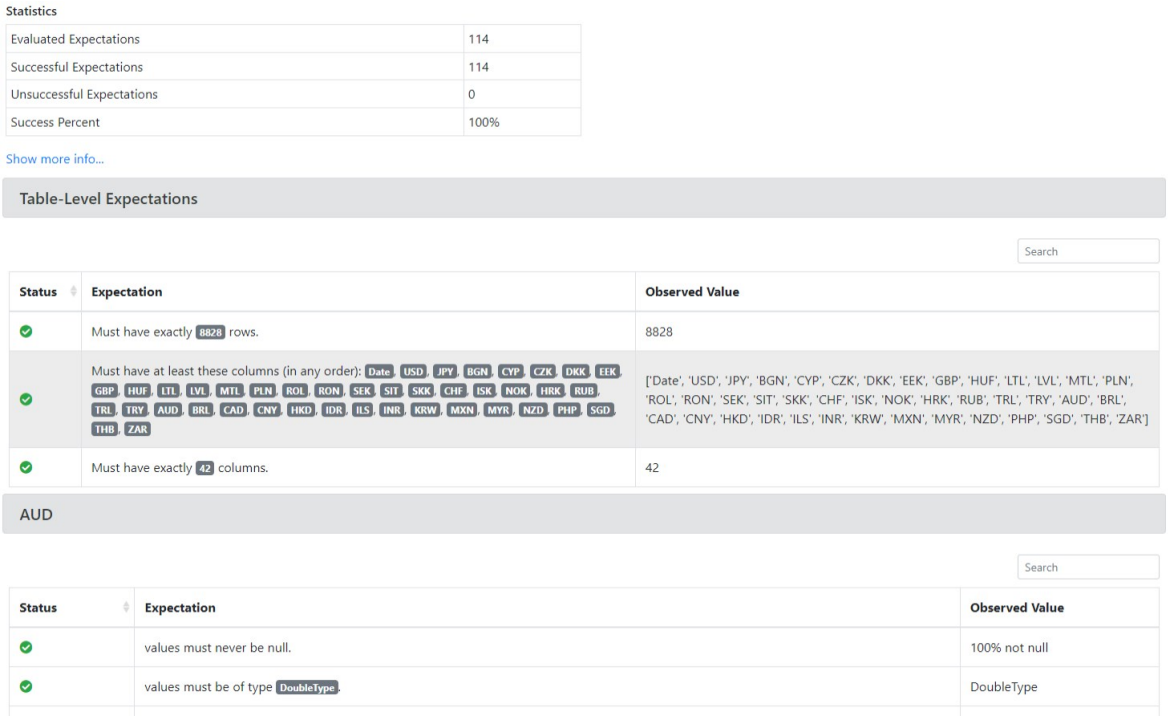

Figure 18 - Example of an HTML validation report

While this feature was not used in this thesis, Great Expectations also has functionality for generating the reports in an HTML format. These reports don't require any additional services and can be stored and hosted in any environment, such as blob storage. Figure 18 shows an example of an HTML report. Just like the report in Great Expectations Cloud, the HTML reports include information about the results of the most recent validation such as the overall results of the validation, the observed value, and more. The reports for previous data validations are also available. Similarly, the reports conform to the defined secrecy levels and don't contain any sensitive information.

Both of these reports are safe in regard to information security and allow control of what data is and is not stored in the reports. Great Expectations Cloud also includes built-in access control to the reports. Users with administrator privileges invite other users in the

organization to join the service. In this thesis, the team members who were involved in the data pipeline development were given access to the reports.

4.4.5 Deployment

After the data validation has been prepared for deployment, the next step in the model is to deploy the validation and integrate it into the data pipeline. The goal of this stage is to have the data validation incorporated into the data pipeline and continuously validate data that goes through the pipeline. The steps to achieve this include pushing the validation to production and configuring the data pipeline orchestration to execute the data validation. Figure 19 demonstrates the desired end result where the data validation has been incorporated as a part of the data pipeline. The data validation tool used in the thesis, Great Expectations, is used in the figure as a stand-in for the data validation.

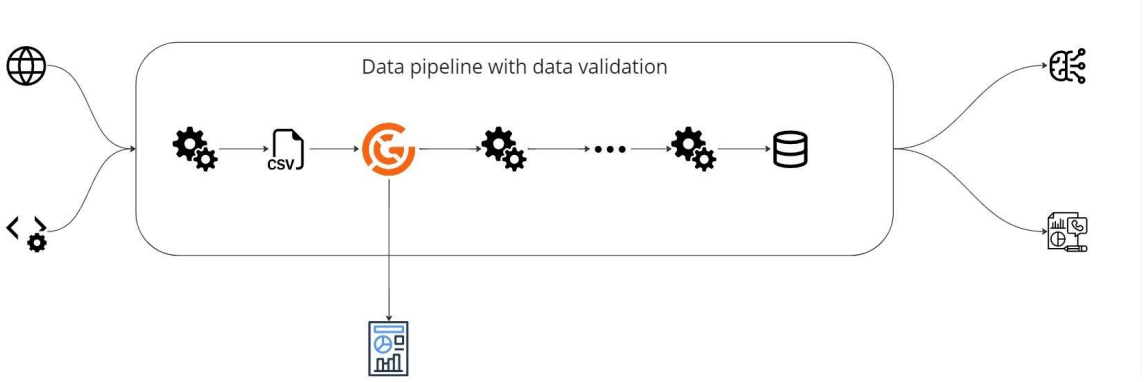

Figure 19 - Data pipeline with data validation

Organizations typically have processes for how to deploy changes to a production system, such as code reviews and scheduled releases. After the data validation has been sufficiently prepared for deployment, pushing the validation to production requires adhering to these company-specific processes.

After the data validation has been deployed to production and is available for the data pipeline orchestration tool, the next step is to connect the orchestration to the data validation. A method for executing the data validation was built in the previous stage and in this stage, the execution only needs to be triggered. As discussed in the previous stage, the orchestration can also be set up to halt processing the data if the data validation fails and encounters bad or problematic data. How this is done practically depends on the orchestration tool and method as well as how the data pipeline overall is orchestrated. The documentation for the orchestration tool as well as other people in the organization should be consulted with problems regarding this step.

In this thesis, the data validation was deployed to the data pipeline with help from developers. The data validation was hosted using the same infrastructure and tooling as the data pipeline. The technologies used for the data pipeline included Azure, Azure Functions, and Python. The data validation tool was also built on Python and supported these platforms which made the deployment relatively straightforward. The deployment process included following the development practices for pushing changes into the production system, such as feature branching.

As discussed in the previous section, triggering the data validation was integrated into the data pipeline code to avoid problems with race conditions. A data pipeline block for data validation was also created to enable flexibility in orchestrating the data pipeline in the future. The data pipeline operations were not configured to halt if the data validation detected problems in the data.

4.4.6 Post-Deployment

After implementing the data validation, integrating it into the data pipeline, and deploying the validation, 3 things should be considered to confirm that the validation functions as intended:

1. Does the data validation continuously validate the data flowing through the data pipeline?
2. Does the data validation automatically catch data that does not conform to the specification?

3. Does the data validation disrupt the normal operation of the data pipeline?

This section discusses the tests that were performed to ensure the validation is working as expected as well as how it was confirmed that the data validation did not disrupt the normal operations of the pipeline.

The functioning of the data validation was confirmed by checking the artifacts that it produced each time the data pipeline was run. These artifacts included the data validation reports automatically produced by the tool in Great Expectations Cloud, the notifications from the failed validations, and the runtime logs produced by the pipeline.

After that, the data validation catching data that did not conform to the data specification was confirmed. This was done by modifying the data specification slightly. Modifying the data specification in a way that valid, high-quality data would not fit the specification was expected to make the validation fail. The failing validation should result in the tool producing a report that highlights the modified specification as well as trigger sending the notification to the internal messaging channel.

To confirm that the data validation didn’t have negative effects on the data timeliness and data pipeline runtime length, the data pipeline was monitored after integrating the data validation. The performance and runtime of the pipeline were monitored by observing the data pipeline orchestration tool Azure Data Factory and the logs produced by the pipeline.

5 Discussion

This chapter discusses the significance of the research on practitioners and researchers as well as the limitations and future research opportunities that were identified during the research process. The identified limitations and research opportunities relate to the process of producing a specification for a dataset, how data validation can be used for improving data quality, and the impact that implementing data validation has.

5.1 Significance for practitioners and researchers

The research produced a process model that practitioners can use to reliably incorporate data quality validation into data pipelines. This is expected to facilitate practitioners with implementing data validation which should result in better observability of the data and data quality in the data pipelines. For example, the data developers in the case company can use the model to more reliably incorporate data validation into the produced data solutions and provide better observability and understanding of the quality of the data in the produced data pipelines. The model is also expected to improve managing data quality and communicating about it more effectively inside organizations, which should lead to higher quality data, faster development, and fewer defects in data products.

Researchers can use the process model for reliably implementing data validation into data pipelines to further study data quality validation, for example how data quality validation can be used for managing and improving data quality, or how the improved visibility to the data quality benefits data engineers and data scientists recover faster from problems in the data pipeline.

5.2 Limitations and Future Opportunities

This section discusses the limitations of the thesis and the future research opportunities that arose during the research process. The research focused on data validation but didn't

explore the impacts that data validation had on the organization or how the data validation could be used for improving data quality. The research also provided some guidelines on how to create a data specification but the guidelines most likely were not comprehensive. All of these limitations provide opportunities for future research.

5.2.1 Producing a data specification

As discussed in the Research Framework section of the thesis, Design Science Research and the used research framework are used to iteratively develop an artifact, which includes repeating the design-demonstrate-evaluate cycle multiple times. When repeatedly performing this cycle, a need for a repeatable framework for defining and producing a data specification for a dataset became apparent. A framework that worked for this thesis was developed but was recognized to be incomplete and likely not generalizable to many other projects. For example, using a set of acceptable values was recommended in the framework and used in the validation when the set of acceptable values was finite and known in advance. But is this a good approach when the set of acceptable values is finite and known but the size of the set is huge?

Data smells, or context-independent indicators of problems in the data [27] that were discussed in the Literature review chapter are useful especially for finding low-quality and problematic data, particularly when the data has not caused errors in the data pipeline but was found insufficient for developing a specification for a dataset. For example, DateTime objects being stored in text format (as strings) was a common data smell in the data that was validated in this thesis. This was useful to note as a change suggestion for the development team but wasn't useful for defining the data specification as the pipeline kept producing this kind of data and the other parts of the data pipeline expected to receive this kind of data.

The data quality dimensions that were also discussed in further detail in the Literature review chapter such as timeliness, usability, and accessibility were found to be useful on a high level for considering data quality and problems in the data but couldn't be used as a framework for practically defining a data specification that data can be validated against. It

should be noted that some dimensions such as completeness can be transformed into metrics that can be used for practically validating data [25].

Defining the different aspects that should be included in a data specification and generating a generalizable framework for defining the requirements for a dataset would be useful for data quality practitioners.

5.2.2 Improving data quality

The goal of this thesis was to create a process model for incorporating data validation into a data pipeline. The data validation only monitors the data in the pipeline, produces reports about the results, and sends notifications as configured. Implementing data validation doesn't automatically increase the data quality in the pipeline. Instead, data quality validation and the outputs such as reports can be used for observing and tracking the quality of the data in the pipeline at a given moment as well as over time. As discussed in the Literature review chapter, data pipelines can be considered as factories that take in an input, process that input, and produce an output [9]. This implies that the quality of the output data depends on the input and the processing. In many cases, the input can't be controlled and the only way to improve the data quality is to modify and improve the data processing. How this is done in practice and how data quality validation can be used for this were outside of the scope of this thesis. This is expected to require changes to the processes that are used for developing the data pipelines and data products.

Some team members discussed during the research process that multiple data specifications could be generated with varying degrees of severity, similar to the severity levels in logging (info, warning, error, and critical) [47]. This was assumed to provide increased levels of control and fidelity over controlling the data pipelines based on the data validation results as well as monitoring the data quality over time. The data specifications could also be updated over time to be stricter as the data quality in the pipeline increased. This approach was suggested after noting that "1000 error messages a day is not useful" instead of setting the specification to be strict initially and aiming to increase the data quality to match the complete data specification.

As discussed in the Introduction chapter, high data quality is important to many businesses and organizations but they lack the know-how to improve the data quality. Researching guidelines and processes for how to improve the data quality as well as maintain the defined data specifications and implemented data validation would be beneficial for organizations struggling to increase the quality of their data.

5.2.3 Impact of data quality validation

As discussed previously, incorporating data quality into a data pipeline doesn't automatically increase the quality of the data in a pipeline but is expected to have some effects and impact. While some effects were observed during the research, identifying the various effects that implementing data quality validation might have was outside of the scope of this thesis. At some point after the data validation had been implemented and deployed in the data pipeline, an external data provider changed the data that they provided. The data validation caught this change and sent a notification about the failed validation. The change in the source data as well as the reason for the change were identified within half an hour after the data had entered the data pipeline. As the change didn't cause errors in the pipeline or affect the processing in any meaningful way, a data developer who worked on the data pipeline estimated that even noticing the change without the data validation might have taken months. While in this case, the change in the source data did not necessitate any changes in the pipeline, the case highlighted the increase in the speed of identifying problems and the sources for the problems.

Mean Time To Recovery (MTTR) is a concept in DevOps that measures the average recovery time after a failure in a production system [48]. The case observed during the research process suggests that implementing data quality validation could provide similar benefits in the context of data pipelines and data products. The data validation and monitoring should not only facilitate identifying problems in the data but also pinpointing the location of the problems faster which should help the data engineers, scientists, analysts, and other development team members work more effectively. The data developers being more effective should also directly or indirectly improve the data product, which should benefit the data consumers. The idea that "what gets measured gets managed" [49]

would suggest that continuously measuring the data quality in the pipelines would be useful for managing the data quality, which should be beneficial for the organization as well.

As discussed in the Introduction chapter, having high-quality data is important for companies. While data quality validation is expected to have advantages for a company, adopting the validation might be more compelling if the specific advantages and disadvantages were known and measured. This could be done by researching and assessing more broadly and comprehensively the benefits and drawbacks of implementing data quality validation.

6 Conclusions

The objective of the thesis was to create a process model for incorporating data quality validation into a data pipeline. The problems that data consumers and developers had regarding data quality were identified and a process model was designed to solve these problems. The process model was designed iteratively by following the Design Science Research framework. The iterative nature of the research framework combined with multiple case pipelines worked well for developing and refining the process model. The process model that resulted from this process was successfully used to develop and incorporate data validation into the case data pipelines.

Data quality has been researched for decades and high-level dimensions such as accessibility and timeliness for measuring data quality have been identified. While the higher-level data quality dimensions are useful, less research was found on measuring data quality in practice. This thesis provides a lower-level, more practical approach for setting up continuous data quality measuring as well as discusses some further gaps in the previous research. Data quality practitioners are expected to be able to use the process model as guidance when developing data quality validation to better understand and facilitate improving the data quality in the data pipelines. This is expected to help data developers work more effectively as well as organizations better measure and manage data quality.

While data validation is beneficial for measuring data quality, a limitation of the process model is that it doesn't offer guidance on improving the data quality. Improving data quality is expected to require broader changes in the data development processes, which was outside the scope of this thesis.

Identified future research opportunities include measuring the impact that implementing data validation has on the different levels of an organization, defining more comprehensive guidelines for the process of producing a specification for a dataset, and determining how data validation can be used for improving data quality.

In conclusion, a generalizable process model for practically incorporating data quality validation into a data pipeline was successfully designed as a result of the research process.

References

[1] E. Raguseo, “Big data technologies: An empirical investigation on their adoption, benefits and risks for companies,” *Int. J. Inf. Manag.*, vol. 38, no. 1, pp. 187–195, 2018.

[2] D. Mourtzis, E. Vlachou, and N. Milas, “Industrial big data as a result of IoT adoption in manufacturing,” *Procedia Cirp*, vol. 55, pp. 290–295, 2016.

[3] T. McCausland, “The Bad Data Problem,” *Res.-Technol. Manag.*, vol. 64, no. 1, pp. 68–71, Jan. 2021, doi: 10.1080/08956308.2021.1844540.

[4] P. Swazinna, S. Udluft, and T. Runkler, “Measuring data quality for dataset selection in offline reinforcement learning,” in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2021, pp. 1–8.

[5] T. Döhmen, M. Raasveldt, H. Mühleisen, and S. Schelter, “DuckDQ: Data Quality Assertions for Machine Learning Pipelines”.

[6] A. Haug, F. Zachariassen, and D. Van Liempd, “The costs of poor data quality,” *J. Ind. Eng. Manag. JIEM*, vol. 4, no. 2, pp. 168–193, 2011.

[7] A. G. Carretero, F. Gualo, I. Caballero, and M. Piattini, “MAMD 2.0: Environment for data quality processes implantation based on ISO 8000-6X and ISO/IEC 33000,” *Comput. Stand. Interfaces*, vol. 54, pp. 139–151, 2017.

[8] A. R. Munappy, J. Bosch, and H. H. Olsson, “Data Pipeline Management in Practice: Challenges and Opportunities,” in *Product-Focused Software Process Improvement*, M. Morisio, M. Torchiano, and A. Jedlitschka, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 168–184. doi: 10.1007/978-3-030-64148-1_11.

[9] R. Y. Wang and D. M. Strong, “Beyond accuracy: What data quality means to data consumers,” *J. Manag. Inf. Syst.*, vol. 12, no. 4, pp. 5–33, 1996.

[10] L. Cai and Y. Zhu, “The challenges of data quality and data quality assessment in the big data era,” *Data Sci. J.*, vol. 14, 2015.

[11] S. E. Madnick, R. Y. Wang, Y. W. Lee, and H. Zhu, “Overview and framework for data and information quality research,” *J. Data Inf. Qual. JDIQ*, vol. 1, no. 1, pp. 1–22, 2009.

[12] V. Jayawardene, S. Sadiq, and M. Indulska, “The Curse of Dimensionality in Data Quality,” 2013.

[13] I. Jaya, F. Sidi, I. Ishak, L. Affendey, and M. A. Jabar, “A review of data quality research in achieving high data quality within organization,” *J. Theor. Appl. Inf. Technol.*, vol. 95, pp. 2647–2657, Jun. 2017, doi: 10.5281/zenodo.5374545.

[14] V. Abeykoon *et al.*, “Data engineering for hpc with python,” in *2020 IEEE/ACM 9th Workshop on Python for High-Performance and Scientific Computing (PyHPC)*, IEEE, 2020, pp. 13–21.

[15] C. K. Dehury, P. Jakovits, S. N. Srirama, G. Giotis, and G. Garg, “TOSCAdata: Modeling data pipeline applications in TOSCA,” *J. Syst. Softw.*, vol. 186, p. 111164, 2022.

[16] M. Matskin *et al.*, “A survey of big data pipeline orchestration tools from the perspective of the datacloud project,” in *Proc. 23rd Int. Conf. Data Analytics Management Data Intensive Domains (DAMDID/RCDL 2021)*, 2021, pp. 63–78. Accessed: Nov. 26, 2023. [Online]. Available: http://ceur-ws.org/Vol-3036/paper05.pdf

[17] M. Barika, S. Garg, A. Y. Zomaya, L. Wang, A. V. Moorsel, and R. Ranjan, "Orchestrating Big Data Analysis Workflows in the Cloud: Research Challenges, Survey, and Future Directions," *ACM Comput. Surv.*, vol. 52, no. 5, p. 95:1-95:41, Sep. 2019, doi: 10.1145/3332301.

[18] A. G. Carretero, I. Caballero, and M. Piattini, "MAMD: Towards a data improvement model based on ISO 8000-6X and ISO/IEC 33000," in *Software Process Improvement and Capability Determination: 16th International Conference, SPICE 2016, Dublin, Ireland, June 9-10, 2016, Proceedings 16*, Springer, 2016, pp. 241–253.

[19] Y. W. Lee, D. M. Strong, B. K. Kahn, and R. Y. Wang, "AIMQ: a methodology for information quality assessment," *Inf. Manage.*, vol. 40, no. 2, pp. 133–146, 2002.

[20] A. G. Carretero, A. Freitas, R. Cruz-Correia, and M. Piattini, "A case study on assessing the organizational maturity of data management, data quality management and data governance by means of MAMD.," in *ICIQ*, 2016, pp. 75–84.

[21] D. Tebernum, M. Altendeitering, and F. Howar, "DERM: A Reference Model for Data Engineering.," in *DATA*, 2021, pp. 165–175. Accessed: Nov. 28, 2023. [Online]. Available: https://www.scitepress.org/PublishedPapers/2021/105173/105173.pdf

[22] J. Bicevskis, Z. Bicevska, A. Nikiforova, and I. Oditis, "Towards data quality runtime verification," in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2019, pp. 639–643.

[23] K. Raman, A. Swaminathan, J. Gehrke, and T. Joachims, "Beyond myopic inference in big data pipelines," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 86–94.

[24] G. Petrakos and G. Farmakis, "A declarative approach to data validation of statistical data sets, based on metadata," *Stat. Appl.*, vol. 12, no. 3, 2000, Accessed: Nov. 29, 2023. [Online]. Available: https://www.researchgate.net/profile/George-Petrakos/publication/238067647_A_declarative_approach_to_data_validation_of_statistical_data_sets_based_on_metadata/links/00463529c733968fdf000000/A-declarative-approach-to-data-validation-of-statistical-data-sets-based-on-metadata.pdf?_sg%5B0%5D=started_experiment_milestone&origin=journalDetail

[25] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," *Commun. ACM*, vol. 45, no. 4, pp. 211–218, 2002.

[26] H. Foidl and M. Felderer, "Risk-based data validation in machine learning-based software systems," in *proceedings of the 3rd ACM SIGSOFT international workshop on machine learning techniques for software quality evaluation*, 2019, pp. 13–18.

[27] H. Foidl, M. Felderer, and R. Ramler, "Data smells: categories, causes and consequences, and detection of suspicious data in AI-based systems," in *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*, 2022, pp. 229–239.

[28] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[29] J. Vom Brocke, A. Hevner, and A. Maedche, "Introduction to Design Science Research," in *Design Science Research. Cases*, J. Vom Brocke, A. Hevner, and A. Maedche, Eds., in Progress in IS. , Cham: Springer International Publishing, 2020, pp. 1–13. doi: 10.1007/978-3-030-46781-4_1.

[30] ggailey777, "Azure Functions documentation." Accessed: Apr. 17, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/

[31] K. Petersen *et al.*, "Choosing component origins for software intensive systems:

In-house, COTS, OSS or outsourcing?—A case survey," *IEEE Trans. Softw. Eng.*, vol. 44, no. 3, pp. 237–261, 2017.
[32] "PyPI Download Stats." Accessed: Nov. 27, 2023. [Online]. Available: https://pypistats.org/
[33] "Deequ - Unit Tests for Data." Amazon Web Services - Labs, Nov. 26, 2023. Accessed: Nov. 27, 2023. [Online]. Available: https://github.com/awslabs/deequ
[34] A. Gong, J. Campbell, and Great Expectations, "Great Expectations." Nov. 24, 2023. Accessed: Nov. 24, 2023. [Online]. Available: https://github.com/great-expectations/great_expectations
[35] "Welcome | Great Expectations." Accessed: Nov. 24, 2023. [Online]. Available: https://docs.greatexpectations.io/docs/
[36] "Hooqu - Unit Tests for Data — hooqu 0.1.0 documentation." Accessed: Nov. 27, 2023. [Online]. Available: https://hooqu.readthedocs.io/en/latest/
[37] N. Bantilan *et al.*, "unionai-oss/pandera: Beta release v0.12.0b0." Zenodo, Aug. 12, 2022. doi: 10.5281/ZENODO.3385265.
[38] "Home," Soda Documentation. Accessed: Nov. 27, 2023. [Online]. Available: https://docs.soda.io/
[39] "Telmai - AI-based Data Observability for Open Architecture," Telmai. Accessed: Nov. 27, 2023. [Online]. Available: https://www.telm.ai/
[40] "GX Cloud: SaaS data quality monitoring & management platform." Accessed: Nov. 27, 2023. [Online]. Available: https://www.greatexpectations.io/gx-cloud
[41] "venv — Creation of virtual environments," Python documentation. Accessed: Nov. 24, 2023. [Online]. Available: https://docs.python.org/3/library/venv.html
[42] "Git." Accessed: Nov. 24, 2023. [Online]. Available: https://git-scm.com/
[43] M. Komorowski, D. C. Marshall, J. D. Salciccioli, and Y. Crutain, "Exploratory Data Analysis," in *Secondary Analysis of Electronic Health Records*, Cham: Springer International Publishing, 2016, pp. 185–203. doi: 10.1007/978-3-319-43742-2_15.
[44] N. Laranjeiro, S. N. Soydemir, and J. Bernardino, "A survey on data quality: classifying poor data," in *2015 IEEE 21st Pacific rim international symposium on dependable computing (PRDC)*, IEEE, 2015, pp. 179–188. Accessed: Nov. 26, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7371861/
[45] A. Perer and B. Shneiderman, "Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florence Italy: ACM, Apr. 2008, pp. 265–274. doi: 10.1145/1357054.1357101.
[46] "pandas - Python Data Analysis Library." Accessed: Nov. 24, 2023. [Online]. Available: https://pandas.pydata.org/
[47] "logging — Logging facility for Python," Python documentation. Accessed: Nov. 27, 2023. [Online]. Available: https://docs.python.org/3/library/logging.html
[48] H. Atwal, *Practical DataOps: Delivering Agile Data Science at Scale*. Berkeley, CA: Apress, 2020. doi: 10.1007/978-1-4842-5104-1.
[49] L. Willcocks and S. Lester, "Beyond the IT productivity paradox," *Eur. Manag. J.*, vol. 14, no. 3, pp. 279–290, Jun. 1996, doi: 10.1016/0263-2373(96)00007-2.