Design Principles for Data Quality Tools

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurswissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Marcel Altendeitering

Dortmund
2023

Tag der mündlichen Prüfung: 23.01.2024
Dekan: Prof. Dr.-Ing. Gernot A. Fink
Gutachter:
Prof. Dr. Falk Howar
Prof. Dr. Christian Janiesch

Abstract

Data Quality represents an essential yet complex aspect of data management. At its core, it aims to support the provisioning of high-quality data to business operations. The availability of high-quality data is vital for organizations and can facilitate accurate decision-making, build competitive advantages, and promote inter-organizational collaborations. The realization of these benefits is complicated by the multi-dimensionality and subjectivity of data quality, resulting in cumbersome and time-consuming procedures. Numerous data quality tools aim to support data quality work by offering automation for different activities, such as data profiling or validation.

Despite the importance of data quality and a long history of tools and research, engineers regularly have to work with erroneous data. Data quality tools face several obstacles that complicate the provisioning of high-quality data. These obstacles rest upon changes in the organizational and technical environment that raise new requirements that data quality tools must address. Organizationally, data quality tools must comprehend changing data architectures and the rising importance of data products and domain-driven design. Technically, they need to incorporate big data and operate within heterogeneous system landscapes. Established data quality tools cannot fully comprehend these changes, resulting in tools unfit for modern systems and data architectures. However, there has been little research on the design of such tools, and comprehensive knowledge guiding their creation is missing, which motivates our research.

This thesis addresses the lack of prescriptive design knowledge for creating successful data quality tools. To inform their design, we adopted a design science research approach and conducted nine individual studies throughout this dissertation. Our findings revealed that *Automation*, *Integrability*, *Standards*, and *Usability* are the four main gaps in current data quality tools, and we formulated adequate meta-requirements. We addressed these gaps with four case studies concerned with implementing data quality tools in real-world scenarios. In each case, we designed and implemented a separate data quality tool and abstracted the essential design elements from the case. A subsequent cross-case analysis helped us accumulate the available design knowledge, resulting in the proposal of 13 generalized design principles. This set of design principles represents the main result of this dissertation. A separate action guideline and reference architecture for data quality tools support the accessibility of our results.

With the proposal of empirically grounded design knowledge, we contribute to the managerial and scientific communities. Managers and practitioners can use our results as a framework for creating successful data quality tools in their individual contexts. Scientifically, we address the lack of prescriptive design knowledge for data quality tools and offer many opportunities to extend our research. Researchers can seize these opportunities and advance our results in multiple directions. The continuous work on data quality tools will help them become more successful in ensuring data fulfills high-quality standards for the benefit of businesses and society.

Keywords: Data Quality, Data Quality Tools, Data Management, Data Engineering, Design Principles, Design Science Research

List of Papers

I **Challenges of Data Management in Industry 4.0: A Single Case Study of the Material Retrieval Process** by Antonello Amadori, Marcel Altendeitering, and Boris Otto. In International Conference on Business Information Systems, (pp. 379-390). 2020. DOI: https://doi.org/10.1007/978-3-030-53337-3_28.

II **Scalable Detection of Concept Drift: A Learning Technique Based on Support Vector Machines** by Marcel Altendeitering and Stephan Dübler. In Procedia Manufacturing, 51, (pp. 400-407). 2020. DOI: https://doi.org/10.1016/j.promfg.2020.10.057.

III **Mining Data Quality Rules for Data Migrations: A Case Study on Material Master Data** by Marcel Altendeitering. In International Symposium on Leveraging Applications of Formal Methods, (pp. 178-191). 2021. DOI: https://doi.org/10.1007/978-3-030-89159-6_12.

IV **Designing Data Quality Tools: Findings from an Action Design Research Project at Boehringer Ingelheim** by Marcel Altendeitering and Tobias Moritz Guggenberger. In ECIS 2021 Research Papers. 2021. URL: https://aisel.aisnet.org/ecis2021_rp/95/.

V **DERM: A Reference Model for Data Engineering** by Daniel Tebernum, Marcel Altendeitering, and Falk Howar. In Proceedings of the 10th International Conference on Data Science, Technology and Applications - DATA, (pp. 165-175). 2021. DOI: https://doi.org/10.5220/0010517301650175.

VI **A Functional Taxonomy of Data Quality Tools: Insights from Science and Practice** by Marcel Altendeitering and Martin Tomczyk. In Wirtschaftsinformatik 2022 Proceedings. 2022. URL: https://aisel.aisnet.org/wi2022/business_analytics/business_analytics/4/.

VII **Data Sovereignty for AI Pipelines: Lessons Learned from an Industrial Project at Mondragon Corporation** by Marcel Altendeitering, Julia Pampus, Felix Larrinaga, Jon Legaristi, and Falk Howar. In 2022 IEEE/ACM 1st International Conference on AI Engineering–Software Engineering for AI (CAIN), (pp. 193-204). 2022. DOI: https://doi.org/10.1145/3522664.3528593

VIII **Data Quality in Data Ecosystems: Towards a Design Theory** by Marcel Altendeitering, Stephan Dübler, and Tobias Moritz Guggenberger. In AMCIS 2022 Proceedings. 2022. URL: https://aisel.aisnet.org/amcis2022/DataEcoSys/DataEcoSys/3/.

IX **Data Quality Tools: Towards a Software Reference Architecture** by Marcel Altendeitering and Tobias Moritz Guggenberger. In Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS). 2024. URL: https://hdl.handle.net/10125/107125.

Comments on My Contribution

I **Challenges of Data Management in Industry 4.0: A Single Case Study of the Material Retrieval Process** In this qualitative study, the authors cooperatively conducted the data analysis. The proposed sets of template codes and identified challenges emerged from discussions among the authors. I am the lead author of sections 1, 2, 3, and 4.2 and co-author of all other sections.

II **Scalable Detection of Concept Drift: A Learning Technique Based on Support Vector Machines** The authors worked out the concept and details for the proposed algorithm collaboratively. I am the lead author for sections 1, 2, and 4 and co-author of all other sections.

III **Mining Data Quality Rules for Data Migrations: A Case Study on Material Master Data** As the only author, I was responsible for the algorithmic approach and implementation and am the lead author of all sections. We conducted the evaluation in cooperation with an industrial partner who provided the use case.

IV **Designing Data Quality Tools: Findings from an Action Design Research Project at Boehringer Ingelheim** The presented concept and design principles for a data quality tool resulted from the interplay between the authors and the experts at Boehringer Ingelheim. I was responsible for implementations, experiments, and evaluation. I am the lead author of sections 1, 2, 4, 5, and 6.

V **DERM: A Reference Model for Data Engineering** The presented data engineering reference model in this paper is the result of discussions and collaborative work among the authors. I am the lead author of sections 1, 2, and 5 and co-author of all other sections.

VI **A Functional Taxonomy of Data Quality Tools: Insights from Science and Practice** The development of the presented functional taxonomy and its application to data quality tools was carried out by me. My co-author provided methodological support. I am the lead author of sections 2, 4, 5, and 6 and am co-author of all other sections.

VII **Data Sovereignty for AI Pipelines: Lessons Learned from an Industrial Project at Mondragon Corporation** The implementation details and results described in this paper resulted from collaborative work and discussions among the authors. I am the lead author of sections 1, 3, and 5 and was co-author of all other sections.

VIII **Data Quality in Data Ecosystems: Towards a Design Theory** The proposed design principles in this paper emerged from the collaboration between the authors and the industrial partner. My co-author Stephan Dübler and I were responsible for the technical architecture and implementation. I am the lead author for sections 1, 3, and 4 and am co-author of all other sections.

IX **Data Quality Tools: Towards a Software Reference Architecture** Both authors worked on the proposed software reference architecture together and conducted the research collaboratively. I am the lead author for sections 1, 4, and 5 and am co-author of all other sections.

Other Peer-Reviewed Publications

Journal papers

- **A Design Theory for Data Quality Tools in Data Ecosystems: Findings from three industry cases** by Marcel Altendeitering, Tobias Guggenberger, and Frederik Möller. In Data & Knowledge Engineering. Major Revision, resubmitted.

Conference papers

- **Data-Flow Programming for Smart Homes and Other Smart Spaces** by Marcel Altendeitering and Sonja Schimmler. In IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). 2020. DOI: `https://doi.org/10.1109/VL/HCC50065.2020.9127268`.
- **Towards a Data Management Capability Model** by Inan Gür, Tobias Moritz Guggenberger, and Marcel Altendeitering. In AMCIS 2021 Proceedings. 2021. URL: `https://aisel.aisnet.org/amcis2021/strategic_is/strategic_is/7/`.
- **Towards a Low-Code Tool for Developing Data Quality Rules** by Timon Sebastian Klann, Marcel Altendeitering, and Falk Howar. In Proceedings of the 12th International Conference on Data Science, Technology and Applications - DATA, (pp. 22-29). 2023. DOI: `https://doi.org/10.5220/0012050400003541`.
- **Design Principles for Quality Scoring - Coping with Information Asymmetry of Data Products** by Tobias Moritz Guggenberger, Marcel Altendeitering, and Christoph Schlueter Langdon. In Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS). 2024. URL: `https://hdl.handle.net/10125/106928`.

Book chapters

- **A Survey-based Evaluation of the Data Engineering Maturity in Practice** by Daniel Tebernum, Marcel Altendeitering, and Falk Howar. In Cuzzocrea, A., Gusikhin, O., Hammoudi, S., Quix, C. (eds). Data Management Technologies and Applications. DATA 2022 2021. Communications in Computer and Information Science, vol 1860. Springer, Cham. 2023. DOI: `https://doi.org/10.1007/978-3-031-37890-4_1`.

Acknowledgements

Time flies. Looking back at the time of my dissertation, I can say it was quite an adventure and full of ups and downs. As with any big journey, this dissertation was a team effort, and I am happy I could spend the time with great colleagues, friends, and family, whom I would like to thank in this acknowledgment.

I would like first to thank Falk Howar for your guidance and for helping me navigate through my dissertation. Falk, thank you for your continuous feedback on my work and your advice in times of uncertainty. Our meetings and discussions offered great insights, and your fruitful suggestions inspired me, and I found myself a better researcher.

I also thank Christian Janiesch and the remaining committee members for taking the time to read my dissertation and providing me with new ideas and valuable feedback.

Of course, I would like to thank my colleagues at Fraunhofer ISST for our countless talks and discussions and our collaborative work on so many projects and papers. I also like to thank the four case study partners who offered me great insights into the industrial practice of data quality and supported me in creating new solutions. A very special thanks goes out to Daniel Tebernum and Tobias Guggenberger. We are brothers-in-arms, and it was a pleasure to share this journey with you. I highly appreciate all the chats, laughs, and discussions we had. I hope for many more projects and publications to come.

Finally, yet most importantly, I thank my family: Julia, thank you for being my favorite human. I could not have done it without your patience and indispensable support during the last four years. I also thank my mom, dad, and my sister for encouraging me to follow my dreams and goals, even if they are far ahead.

Contents

List of Figures

List of Tables

List of Acronyms

ADR	Action Design Research
AI	Artificial Intelligence
API	Application Programming Interface
AR	Action Research
ATAM	Architecture Tradeoff Analysis Method
DBMS	Database Management System
DD	Design Decision
DP	Design Principle
DQ	Data Quality
DQV	Data Quality Vocabulary
DSR	Design Science Research
GTM	Grounded Theory Methodology
IDS	International Data Spaces
IDSA	International Data Spaces Association
IQ	Information Quality
IS	Information Systems
LL	Lesson Learned
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
OA	Organizational Agility
SLR	Systematic Literature Review
SVM	Support Vector Machine
TDQM	Total Data Quality Management
UI	User Interface

1. Introduction

"You can't do anything important in your company without high-quality data" [100, p.1]. As summarized by Redman [100], data quality (DQ) is a critical success factor for any organization and an essential part of data management. Despite its importance, many organizations struggle with DQ and fail to provide business processes with high-quality data [109, 100]. This problem is grounded in the changing environment DQ and data management currently face [48, 56]. Trends like big data, decentralization, and increased awareness of data as a strategic resource raise new requirements [38, 109, 24]. Established DQ tools cannot keep up with these changes and fail to provide DQ practitioners with the support they need [10]. In this chapter, we introduce our research by delving into the following three questions: *Why is DQ an essential topic for organizations?*, *What are the problems of established DQ tools?*, and *How can we address these problems with our research?*.

1.1. The Importance of Data Quality

DQ has long been incorporated into organizational data and information systems (IS) strategies to enable accurate decision-making and seamless operations [83, 81]. It is widely accepted as an essential building block for organizational success [35, 90], and a lack of DQ can incur costs of 8 to 12% of an organization's revenues [99]. Lately, the importance of DQ has shifted from a business-oriented perspective to various influencing factors. Today, four main topics are driving the need for DQ.

First, the prevalence of **automated decision-making** in the forms of artificial intelligence (AI) and machine learning (ML) increases the need for DQ, as multiple studies have shown [27, 13, 56]. Providing high-quality data sets that are unbiased and free of unlabeled or erroneous data for training is vital for the success of any data science project and can increase the business value [36, 35]. At the same time, most data sets face significant quality issues. For instance, Redman [99] analyzed DQ at the field level and observed that 1 to 5% of all entries were erroneous. Consequently, data usually go through a cumbersome and expensive pre-processing before being used in industrial data science projects [62]. Since this is a time-consuming process, data scientists tend to focus on widespread and reoccurring problems and neglect other data errors, although this approach can impair organizational performance [8].

Second, to face increasing uncertainties and remain successful in a changing environment, businesses must be able to adapt to new situations quickly. This capability is often referred to as **organizational agility** (OA) and has developed into an indispensable component of corporate strategies [93, 35, 111, 81]. DQ is a prerequisite for realizing OA, as it forms the basis for all data-related operations and well-informed decision-making

[90, 32]. An organization operating on high-quality data can use business intelligence [93] or big data [35] to utilize internal processes and resources more efficiently and increase its performance [35, 61]. On the contrary, a lack of DQ can negatively affect organizations, leading to inaccurate decision-making and a loss of competitive advantages [90, 32].

Third, the growing use of automated and data-based decision-making has put DQ on the political agenda and led to new **regulatory requirements** [83]. Especially within the European Union, AI systems trained on low-quality data are considered a threat to the ethical use of data and the fundamental rights and Union values [42, 43]. In this context, biased data sets are a major concern, as their use can lead to discrimination against ethnic minorities, older people, and other groups [43]. Several studies demonstrated how low-quality data leads to discrimination in language models [23], facial recognition systems [31], or AI systems trained on sociocultural data in general [71]. To address this issue, the European Union proposed the AI Act, which sees it as the due diligence of organizations to avoid errors and bias in data sets and ensure the creation of trustworthy AI systems [42].

Fourth, organizations increasingly strive to complement their internal data sources with external sources to operate on a more extensive and diverse data basis and promote data-based business models [59, 81]. As a result, **data ecosystems** emerged and promised to simplify inter-organizational data sharing by providing common data services, architectural patterns, and governance mechanisms [88, 56]. A concrete example is the International Data Spaces Association (IDSA), which specifies data space protocols, architectural guidelines, and information models for creating interoperable data ecosystems [67]. Breaking data silos and sharing data for mutual benefit can help foster data-driven innovation, exploit new business opportunities, and overcome typical data challenges such as data governance or data democratization [104, 56]. A critical success factor for the adoption of data ecosystems is DQ. Exchanging low-quality data can mitigate the trust between a data provider and consumer and hinder partners from exploiting the benefits offered by data ecosystems [20, 88]. A joint understanding of DQ, adequate tooling, and suitable metadata are required to ensure that data of sufficient quality is shared with ecosystem partners [6].

1.2. Problem Statement

Although DQ is of paramount importance for organizations, many practitioners, such as data engineers or data scientists, are confronted with DQ issues on a regular basis [100, 62]. Deng et al. [39] quantified the typical DQ work by stating that data scientists spend up to 98% of their time on 'grunt work', including identifying and resolving data errors. In addition, Tebernum et al. [112] empirically investigated data engineering practices and found that DQ is inadequately considered throughout the data life cycle, causing additional manual efforts.

Since humans are usually bad at detecting data errors [75], numerous DQ tools emerged and promised to support DQ management [83]. Today, the market for DQ tools makes up an estimated \$1.77 billion [37]. However, despite decades of previous research and many available tools, DQ remains an important issue and is still considered a problem in industrial practice [56, 109]. The urging question of why existing DQ tools cannot fulfill the need for high-quality data remains and requires new answers.

The obstacles DQ tools face are manifold. Most importantly, the technological and organizational environment in which DQ tools operate is changing, and established DQ solutions cannot fully comprehend these changes [109]. Most of these DQ solutions rely on processes and algorithmic bases unsuited for modern data architectures [10]. For instance, the trend for ubiquitous and big data forces organizations to reorganize their data architecture by shifting from a centralized to a decentralized approach [56, 38]. This development breaks with the concepts of established DQ tools that solve quality issues using a centralized data platform and center of expertise (cf. section 2.2).

A closer investigation of this changing environment revealed that practitioners perceive the centralized approach as cumbersome, time-consuming, and error-prone, thus hindering businesses from fully exploiting their data assets [12, 113, 10]. Centralized DQ teams often lack the required data domain knowledge to resolve data errors efficiently, which leads to lengthy data-cleaning processes [100, 12]. Employees at the data source understand the business context better and can solve DQ issues before feeding them into other business processes [100, 113]. Yet, these employees are typically no data engineers and do not have thorough technical and algorithmic knowledge, raising a need for better accessibility and usability of DQ tools [8]. Moreover, DQ tools must be able to operate in different technical contexts, which requires the implementation of standards and integrative capabilities [10, 48]. In summary, we identified automation, integrability, standards, and usability as the four main problems of DQ tools (cf. section 4.1).

Successful DQ tools must accommodate the changing environment and address the identified gaps to overcome the problems in industrial practice. However, there has been little research on the design and implementation of such tools in concrete socio-technical contexts. Current DQ research focuses on improving single aspects of DQ tools (e.g., automation [106] or validation [110]) and has a rather technical scope. Comprehensive research that consolidates multiple requirements and derives the prescriptive design knowledge necessary for building and adapting adequate DQ tools within an organization is missing. As a result, designing successful DQ solutions remains difficult for practitioners. This lack of knowledge motivates our research, which we specify in the following section.

1.3. Research Goal

In response to the described problem, this thesis aims to overcome the lack of prescriptive design knowledge for DQ tools and offer practitioners actionable knowledge for building solutions that overcome current limitations. Specifically, we aim to provide practical design knowledge for addressing the identified problems of DQ tools in industrial practice and create more successful solutions. The research goal of this thesis reads as follows:

What is the design knowledge required for creating successful data quality tools?

Tackling this research goal calls for investigation into two directions: its problem and solution space. For the problem space, we need to understand and concretize the requirements organizations pose on DQ tools. To gain these insights, we investigate the handling of DQ in real-world settings and understand why many organizations still struggle to cope with DQ (*RQ 1*). To populate the solution space, we want to capture the design principles (DPs) that organizations need to build successful DQ tools from real-world implementations (*RQ 2*). These DPs should address the needs of industrial practice identified in the first step.

Deriving DQ design knowledge is particularly challenging as the 'fitness for use' [118] of DQ usually leads to customized DQ solutions, and their design is subject to influences of their socio-technical contexts. This means we need to abstract the research findings from multiple individual case studies to the higher class of problem, namely the design of DQ tools. By combining the individual results, we can infer generalized and multi-grounded design knowledge and reach theoretical saturation [41, 50, 105]. To achieve the proposed research goal, we answer the following two research questions.

RQ1: What are the objectives for successful data quality tools?

The awareness of DQ in science and practice has proliferated over the past decades, and a large variety of tools and management frameworks emerged supporting organizations in their DQ efforts [83, 40]. Nevertheless, companies face constantly changing technical (e.g., the rise of big data [109, 24]) and organizational (e.g., the value seen in data [90, 38]) environments, creating a need for a new kind of DQ tool [83, 64]. In the first step towards more successful DQ tools, we need to understand what DQ problems organizations face based on theoretical and practical insights. The result motivates our research and serves as the problem-centered entry into the Design Science Research (DSR) process guiding this dissertation [95, 64].

We conducted three studies to inform the proposed research question, which applied different research methods (Paper I [12], V [113], and VI [10]). Paper I describes the results of a qualitative case study in an automotive company. We identified several challenges in the areas of data governance and DQ. In paper V [113], we conducted a Systematic Literature Review (SLR) on data engineering processes described in the scientific literature. Using the SLR result, we developed a reference model that structures data engineering activities and identified the need for improved DQ in different layers of the model. Paper VI [10] presents a taxonomy of functional and non-functional DQ tool capabilities. We applied the taxonomy to commercial DQ tools available in the market and derived gaps and current trends.

The studies revealed *Automation*, *Integrability*, *Standards*, and *Usability* as the four main problems of DQ tools. Given these problems, we derived four meta-requirements for DQ tools that provide a general description of the desired tool and specify functionalities not yet available [95]. It is vital that these meta-requirements are generally applicable to avoid producing narrow theories too close to a case [41]. The meta-requirements guide the creation and presentation of design knowledge in this thesis. We present our results in more detail in section 4.1.

RQ2: How to design successful data quality tools?

To help organizations build successful DQ tools, we need to offer prescriptive design knowledge addressing the previously identified problems [95]. We captured this design knowledge from four case studies concerned with the implementation of four individual DQ tools (Papers III [4], IV [8], VII + II [11, 5], and VIII [6]). Each case features a different empirical setting and addresses one or more of the problems identified in RQ1. We formalized the design decisions and our experiences and abstracted these from the specific case to the class of problem to allow for generalization [95].

In a subsequent cross-case analysis, we aggregated the design knowledge generated in the four cases and identified regularities and irregularities [17]. As a result, we obtained empirically valid and evidence-based design knowledge [64, 103]. We present the accumulated design knowledge in section 4.2 in detail. It consists of concrete DPs that help practitioners build successful DQ tools and enable the provisioning of high-quality data. Furthermore, we provide concrete examples and organizational implications that help the design knowledge become more actionable and applicable.

In addition to the DPs, we created two contributions to improve their accessibility and usage. In paper IX [7], we followed the framework by Angelov et al. [14] and created a software reference architecture for DQ tools based on our findings and related DQ tools. We evaluated the reference architecture qualitatively using the adapted Architecture Tradeoff Analysis Method (ATAM) [15]. Finally, in section 4.3, we offer an action guideline that the target group of this thesis (i.e., DQ system designers) can use to inform their individual DQ tool design by following a systematic guideline.

1.4. Organization

The organization of this thesis follows the guidelines for presenting DSR results by Gregor and Hevner [54] and the recommendations for presenting empirical research by Kitchenham et al. [73].

The following chapter 2 introduces the theoretical foundations of DQ and its tooling. The chapter describes the concept of DQ against its data management background and presents the historical evolution of DQ tools, leading up to the need for a new type of DQ tool addressed in this thesis. We also present related studies and summarize their contributions to DQ design knowledge.

Chapter 3 outlines the research methodology we followed in this thesis. We describe the adopted DSR process and specify how the different papers of this dissertation fit into the overarching DSR method. Moreover, we summarize the four case studies concerned with developing DQ tools and outline our approach for accumulating design knowledge from multiple cases.

In chapter 4, we present the accumulated design knowledge deduced from our case studies. Representing our core result, we communicate the design knowledge in three sections. First, we specify the problems DQ tools face and derive the meta-requirements a successful DQ tool should fulfill. Second, we formulate concrete DPs and structure these along the identified meta-requirements. Third, we present an action guideline for creating DQ tools to make our findings more actionable.

Chapter 5 describes the process and results of our two-tiered approach for evaluating the presented design knowledge, consisting of a scenario-based demonstration and a group discussion on its reusability.

Lastly, chapter 6 concludes this thesis by answering our research questions, critically reflecting on our results, specifying the scientific and managerial contributions, and outlining paths for future work.

2. Related Work

Before continuing with the research methodology and our results, it is beneficial to gain a shared understanding of data management and quality and review the historical development of these fields. For this purpose, we briefly summarize the data management and DQ research and highlight established frameworks and current trends. Afterward, we look closer at the evolutionary developments of DQ tools, leading up to the need for a new, more successful kind of tool. Finally, we present studies related to the design of DQ tools and describe how they can extend our findings.

2.1. Data Management & Data Quality

To grasp the concept of DQ, we start by explaining what data is and what forms it can take on. From an information management perspective, data is often referred to as the unprocessed building block of information, which puts it at the center of any business activity [66, 89]. We can separate data into two classes: relational and non-relational data [90]. Relational data is held in relational databases consisting of sets of tuples [33]. Each tuple contains logically related attributes describing particular objects. A company usually represents its core business objects, such as products or customers, in this form [90]. Non-relational data comprises data that exists in an unstructured or semi-structured format and cannot be organized in tabular form (e.g., images, documents, or audio) [49]. Compared to relational data, non-relational data schemata offer more flexibility, and databases are less rigid regarding consistency [28].

The increased variety of data is part of the 'Big Data' paradigm, which is often defined using the four V's of data [49]. Besides a larger variety, big data also encompasses an increased volume, velocity, and veracity (i.e., believability) of data. Quality concerns both traditional and big data but is much more difficult for the latter. The four V's make it harder to analyze, identify, and correct quality issues efficiently [24, 109].

Data is nowadays widely accepted as vital for organizational success and, consequently, treated as a strategic resource [81, 90]. The ability to efficiently work on data and leverage adequate management capabilities can lead to competitive advantages and increase a firm's performance [60, 61]. For example, data is the source of many digital innovations [58], can be used to optimize business processes [99], or support organizational decision-making [36, 60].

However, to effectively realize organizational benefits, data needs to be "fit for use by data consumers" [118, p.6]. Wang & Strong [118] introduced this definition for DQ in a groundbreaking paper that changed the prevailing view on DQ by taking on a user perspective. Up to that point, DQ was often defined as the accurate representation of its properties and identified via intuition [29, 46, 83]. In the scientific literature, *data* *quality* (DQ) and *information quality* (IQ) are sometimes difficult to differentiate and used interchangeably. Following Madnick et al. [83], DQ comprises technical problems with data, while IQ summarizes organizational and non-technical data issues. In this thesis, we use the term DQ, as the case studies we conducted focused on the technical aspects of DQ, resulting in design knowledge that is more relevant for DQ than IQ tools. Moreover, the term DQ is more common than IQ and thus better suited for communicating our research results to the relevant audience [54].

The 'fitness for use' principle of DQ implies a context sensitivity, meaning its interpretation can vary between stakeholders [118]. For example, the completeness metric of a data set might be sufficient and considered reasonable from a data scientist's perspective. At the same time, it might be insufficient from a sales representative's perspective. This subjectivity of DQ makes it challenging to establish shared definitions and measurements [96, 25]. Moreover, DQ has no single measure but is a multi-dimensional concept, and several scholars have come up with different conceptual frameworks (e.g., [46, 118, 47]). The framework of Wang & Strong [118] is the most popular one. Their work describes 15 DQ dimensions organized into the following four categories. *Intrinsic DQ* encompasses the data's correctness, reputation, and believability. *Contextual DQ* specifies dimensions such as timeliness that aim to measure DQ within the context of the task at hand. *Representational DQ* comprises aspects that measure the format and meaning of data. Finally, *Accessibility DQ* specifies the simplicity and security of accessing data.

Extending the established DQ concepts, a new research stream recently emerged around the term 'data smells'. Following the definition of Foidl et al. [45], data smells are not obvious data errors but problems that arise from bad design decisions in the underlying data architecture. Examples include using synonyms (e.g., New York and NY) and intermingled data types (e.g., differently formatted dates). Although we are not explicitly addressing data smells, our results can inform the design of corresponding tools as they are subject to the same meta-requirements, which emerge from changes in the overall data management.

Research suggests that humans are not good at detecting data errors, and ever-growing amounts of data led DQ to become a daunting task [75, 109]. The complexity of error detection and data validation increased significantly, and assuring high-quality data has become cumbersome and expensive [83, 39]. Consequently, numerous DQ tools emerged from science and practice to support and automate the different activities of DQ management [40, 10].

2.2. Data Quality Tools

Historically, the need for a high level of DQ and adequate tooling are not new. Researchers and practitioners have long known that DQ was a prerequisite for organizational success and used management frameworks and tools to address this need [82, 81]. We can distinguish three general types of DQ tools specializing in different tasks: *data preparation tools* [62], *measuring and monitoring tools* [40], and *general-purpose tools*

[10]. Data preparation tools focus on the pre-processing steps of data analysis and offer data cleaning, validation, and error correction functionalities [62]. Measuring and monitoring tools use various data profiling capabilities (e.g., cardinalities or value distributions) and continuously apply these to data sets to identify potential inconsistencies [40]. General-purpose tools combine features of both former types and offer the most comprehensive set of functionalities [10]. Commercial DQ tools usually fall into this last category.

A clear separation of DQ tools is not always easy and gets even more complicated when comparing them to related solutions, such as data catalogs or data management suites. Since these tools are closely related, many vendors have several solutions in their product portfolio that offer functionalities regarding DQ and combine them into tool suites [10, 40]. For example, a vendor might offer a data catalog, which conducts data profiling tasks, and a separate DQ tool that uses the profiling results for data validation. The design knowledge presented in this thesis addresses meta-requirements relevant to all types of DQ tools, as they are subject to the same changes in the data management landscape. This thesis is, therefore, not limited to specific DQ tasks or a kind of tool. It can instead inform design decisions for DQ tools in general.

All types of DQ tools have been subject to several evolutionary developments in data management over the last few decades. Two developments had a significant impact on their functional and non-functional designs: (1) the value organizations see in data and how they manage it (Evolution of Data Management Maturity) [81], and (2) the way data is stored and organized technically [38, 56] (Evolution of Data Architecture Concepts). Surviving in a rapidly changing environment shaped by data-based business models and big data requires organizations to develop in these directions [81, 48]. The developments in both areas directly influence the organizational scope, functional capabilities, and architectural positioning of DQ tools [9]. We differentiate three generations of DQ tools shaped by the abovementioned developments (see Figure 2.1).

First Generation: Data Quality Checks

In the early stages of data management, lasting until the 1990s, organizations leveraged data to automate business operations (e.g., inventory management) and managed it primarily from a functional perspective [81]. Consequently, data management focused on individual and disconnected databases and the quality and accuracy of the respective data models [81]. In this phase, DQ was concerned with the 'content' of data and focused on the correctness of database entries [109]. To ensure data correctness, DQ was realized as simple accuracy checks in database management systems (DBMS) [29]. These checks were part of the corresponding data models and implemented as integrity constraints in the DBMS. An exemplary constraint might be that data fields containing zip codes must comprise five numbers.

In this sense, the first generation of DQ tools aimed at enabling internal business processes by providing correct data and allowing for data reuse [81]. However, both DQ and data architectures lacked coordinated approaches, resulting in the emergence of data silos [38].

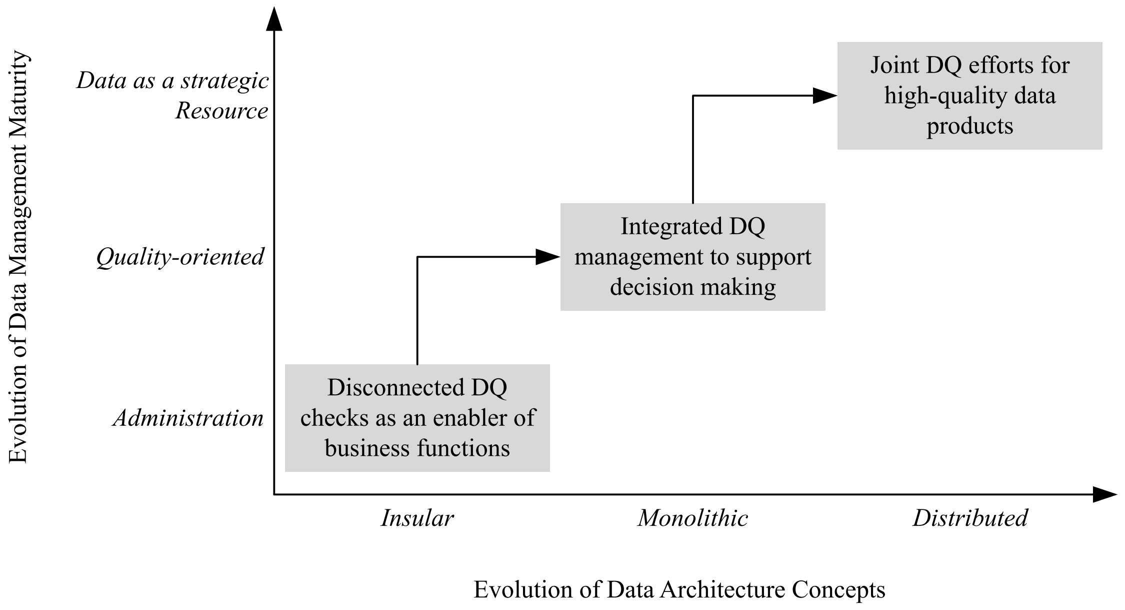

Evolution of Data Management Maturity
Data as a strategic Resource
Quality-oriented
Administration
Joint DQ efforts for high-quality data products
Integrated DQ management to support decision making
Disconnected DQ checks as an enabler of business functions
Insular
Monolithic
Distributed
Evolution of Data Architecture Concepts

Figure 2.1.: Evolution of DQ tools in relation to data management [81] and data architecture [38] advancements (adapted from [9]).

Second Generation: Quality-oriented Data Management
To avoid inaccessible and siloed data, organizations created centralized data platforms with centralized teams owning and managing data starting in the 1990s. Integrated information systems and monolithic business analytics platforms, such as data warehouses, became popular in this data management phase [83]. At the same time, the research around DQ accelerated and established as a new scientific field [83]. Most importantly, Madnick & Wang [82] introduced a research program dedicated to DQ called "The Total Data Quality Management" (TDQM), leading organizations toward quality-oriented management of data resources. These developments helped data become an enabler of enterprise-wide business processes and improved organizational decision-making [81].

In response to these developments, DQ turned into a task involving the entire organization and was no longer solely technical. Organizational capabilities like data governance joined long-established technical tasks such as data integration and integrity constraints [21, 83]. Organizations began to implement integrated DQ management tools, offering assistance with data governance, defining DQ rules, and data validation to support enterprise-wide DQ management. These solutions comprise the second generation of DQ tools [81, 83].

Third Generation: Cooperative Data Quality
A third data management phase emerged in the 2010s, viewing data as a strategic resource. In this new perspective, data was a central object in strategic management and enabling data-driven innovation [81]. Simultaneously, organizations observed that centralized data architectures lacked the necessary scalability and flexibility to handle

big data [109]. Instead, architectures that provide locally owned data products became popular to meet the new demands for flexibility and consumer orientation [38].

These so-called 'Data Mesh' architectures operate on polyglot persistent and distributed data stores, embracing the principles of domain-driven design and encouraging the creation of high-quality 'Data Products' at the source [38, 56, 63]. These architectures offer high scalability and help overcome typical big data problems, such as support for real-time data analytics and high complexity in data management and quality [56, 38]. They, furthermore, shift the responsibility for DQ from the data consumer to the provider side, leading to decentralized and unraveled DQ work [7, 63]. A viable solution for realizing such data mesh architectures is the implementation of an internal data ecosystem comprising a self-service data infrastructure, shared data services, and standards for interoperability [59, 58, 56].

Although the third generation of DQ management emerged, most established DQ tools remain stuck in the second generation and follow centralized approaches to DQ [10, 48]. However, in increasingly decentralized data architectures, a centralized approach to DQ is not feasible, resulting in data not meeting the necessary quality standards and cumbersome data cleaning efforts [8]. These developments raise a need for a new, third-generation DQ tool. This third generation sees DQ tools as suites that assess DQ in several application scenarios or 'contexts' [109]. They facilitate solving quality issues at the 'source' and creating valuable data products in the business domains [100, 63, 90]. At the data source, employees have the necessary domain knowledge and can solve DQ issues more efficiently than centralized teams or users from other domains could [48]. Considering big data and inter-organizational data sharing, these new DQ tools must support joint DQ efforts and be accessible to people from different backgrounds [8, 12]. With this dissertation, we want to understand the requirements for such third-generation DQ tools and inform their design to assist practitioners in realizing successful DQ tools.

2.3. Existing Design Principles for Data Quality Tools

Practitioners who want to realize this new type of DQ tool are confronted with a lack of prescriptive design knowledge. It is difficult for people designing DQ system designers (e.g., DQ managers or solution architects) to define the functional and non-functional capabilities of a DQ tool. Today, the research around DQ tools mainly focuses on the managerial and computer science perspectives. The former studies the impact of DQ tools and how they are managed and positioned in the organizational context. The latter includes research on DQ algorithms and how they can be improved for better results. Consequently, the body of literature concentrates on these two aspects, and there is little research covering both fields [83, 10].

To identify studies related to our work in an unbiased and repeatable fashion, we conducted an SLR following the guidelines of Kuhrmann et al. [79] and Kitchenham [72]. Our goal is to identify studies proposing different forms of design knowledge for creating DQ tools and inform our own study and design recommendations. We formulated the following search term to find relevant articles:

("data quality" OR "information quality") AND "design"

Since design knowledge is often framed differently, such as DPs, theories, or features, we decided to only use the term 'design' alongside DQ and IQ. We limited our search to the study's title to avoid producing numerous results. As Kuhrmann et al. [79] suggested, we selected established sources in the domains of computer science and IS to collect relevant studies. This selection yielded in the databases IEEE Xplore, ACM Digital Library, ScienceDirect, AISeL, and Scopus as a meta-search engine. Using the proposed search term on these databases produced 137 results. Subsequently, we screened these studies for relevance and removed duplicates and our own studies from the result set. The screening process included filtering studies that are too narrow and focus on a specific industry (e.g., medical DQ tools) and removing studies that do not specifically address DQ tools but other kinds of artifacts (e.g., DQ as an aspect of surveys).

As a result of this procedure, we identified three papers that specifically communicate design knowledge for DQ tools and relate to this dissertation (see Table 2.1). A following forward and backward search, as suggested by Webster & Watson [119], revealed no further relevant studies.

Table 2.1.: Overview of related work.

Authors	**Title**	**Source**
Westin & Sein	The Design and Emergence of a Data/Information Quality System	[120]
Walter et al.	Deploying machine learning based data quality controls - Design principles and insights from the field	[116]
Alhamadi et al.	Data Quality, Mismatched Expectations, and Moving Requirements: The Challenges of User-Centred Dashboard Design	[3]

The paper by Westin & Sein [120] describes a five-year Action Design Research (ADR) study concerned with developing and implementing a DQ assessment tool. The authors formulate a set of five DPs, which they derived by analyzing literature and interviews conducted throughout the research project. The described DPs concentrate on improving the assessment and reporting of DQ within a single information system. In contrast to our study, they neglect DQ design requirements that arise in data-sharing scenarios, for example, within data ecosystems. Similar to our findings, they also describe the need for a robust DQ tool that can handle difficulties like inconsistencies or incompleteness.

Walter et al. [116] present the implementation of ML-based DQ checks in an ADR case study. As a result, they propose eight DPs for realizing ML-based quality controls organized into three categories: model development, model deployment, and process integration. Tallied with our results, they present DPs addressing the need for more automation and integration in established processes and tools. Unlike our study, they go into more detail regarding the use of ML models for automating DQ work. Their findings can extend our work in this direction.

The research of Alhamadi et al. [3] proposes design recommendations for dashboards that support the handling of data sets. This study does not explicitly target DQ tools, but we still consider it relevant as it investigates the user experience of data-intensive applications. The findings result from 17 interviews that the authors held with dashboard developers. In line with our findings, the authors describe the need for a high degree of customization and usability to avoid an information overload and ensure perspicuity. Similar to the study by Walter et al. [116], this paper can extend our results as it delves deeper into the usability aspect.

None of the related studies offers the design knowledge necessary for creating successful DQ tools. Instead, they focus on single aspects or trends of DQ tools and offer in-depth design knowledge in this area. Additionally, all presented studies draw their findings from single cases and are of limited generality. This lack of comprehensive and generalizable design knowledge that organizations need to create successful DQ tools motivates our research. The following chapter outlines the research methodology we followed to obtain this design knowledge and accomplish our research goal.

3. Research Methodology

The lack of comprehensive design knowledge for DQ tools is an important problem for businesses [8, 116]. To create such knowledge, we must move from the undesirable problem state described in Section 1.2 to a more desirable solution space [97, 55]. We can reach this goal by incrementally creating new artifacts in concrete socio-technological contexts, which aim for the solution space [55]. These increments can take on different perspectives on the research problem under investigation and embrace the interdisciplinary nature of DQ. Moreover, they help to create a generalized understanding of the complexity that arises from the interactions of various actors involved in socio-technical information systems [107]. In this thesis, we abstract the findings of a multiple-case study and formulate generally applicable design knowledge that benefits the class of artifact, namely DQ tools [54]. Specifically, we theorize on the design of four DQ artifacts created in four case studies and formulate the accumulated design knowledge that supports the creation of DQ tools in general [52]. This way, we can ensure that our results are not tailored to a specific case but are instead applicable to a class of problem and relevant to a wider audience.

This chapter outlines the qualitative approach to obtaining an empirical answer to the proposed research questions. An empirically grounded perspective is essential for software engineering solutions to ensure the results are impactful and evidence-based [72, 50]. We start with an introduction to the overarching DSR process that shaped the research design of this dissertation. We continue by describing the details of the case studies concerned with designing and implementing DQ tools. Finally, we outline the adopted cross-case data analysis process that helped us accumulate and structure the design knowledge we generated in the multiple-case study [64, 95].

3.1. Overarching Design Science Research Approach

DSR is a rigorous approach to creating artifacts that address a "heretofore unsolved and important business problem" [64, p.84]. At its core, it tries to come up with innovative solutions in areas lacking existing theories [64]. It is thus well-suited for guiding research on building new DQ tools and creating contributions based on the design of such artifacts [95, 54].

In this thesis, we applied the DSR methodology as an overarching framework to organize and structure the individual research contributions created throughout this dissertation. Figure 3.1 outlines how the nine papers, which are part of this thesis, fit into the DSR methodology and contribute to our research process. The figure also shows to which DSR stage each paper belongs, thus indicating their primary research focus. Most importantly, we conducted four case studies representing four iterations of the DSR process (cf. Section 3.2). Although we used DSR as a framework, the individual papers can feature different research methodologies more appropriate for their respective settings or publication outlets.

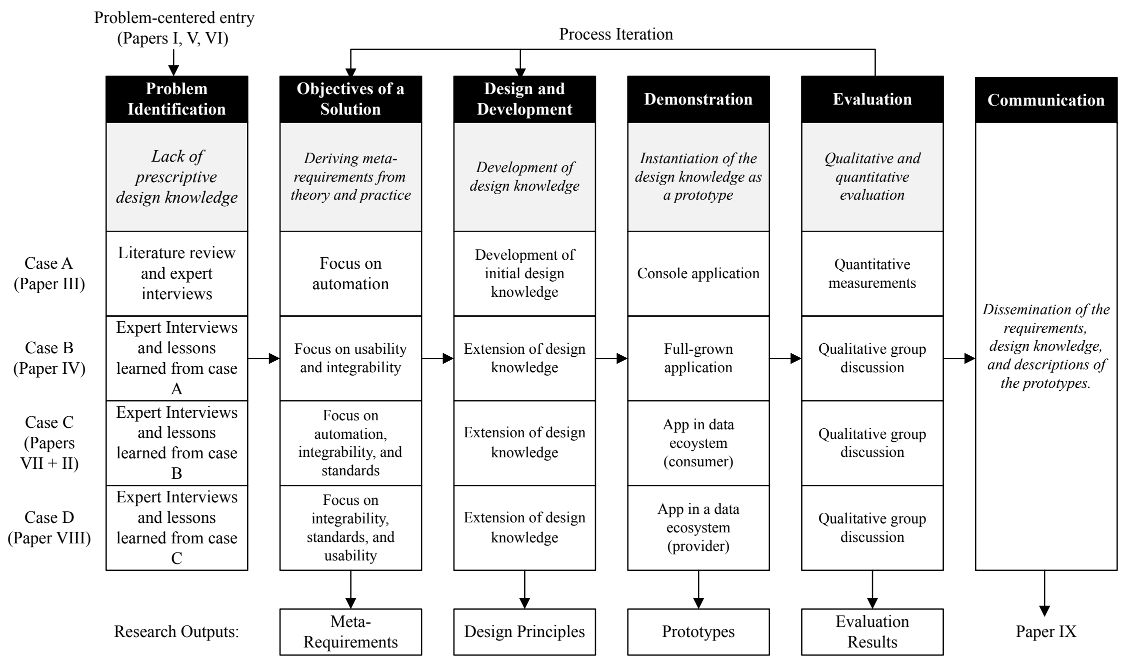

Problem-centered entry (Papers I, V, VI)
Process Iteration
Problem Identification
Objectives of a Solution
Design and Development
Demonstration
Evaluation
Communication
Lack of prescriptive design knowledge
Deriving meta-requirements from theory and practice
Development of design knowledge
Instantiation of the design knowledge as a prototype
Qualitative and quantitative evaluation
Dissemination of the requirements, design knowledge, and descriptions of the prototypes.
Case A (Paper III)
Literature review and expert interviews
Focus on automation
Development of initial design knowledge
Console application
Quantitative measurements
Case B (Paper IV)
Expert Interviews and lessons learned from case A
Focus on usability and integrability
Extension of design knowledge
Full-grown application
Qualitative group discussion
Case C (Papers VII + II)
Expert Interviews and lessons learned from case B
Focus on automation, integrability, and standards
Extension of design knowledge
App in data ecoystem (consumer)
Qualitative group discussion
Case D (Paper VIII)
Expert Interviews and lessons learned from case C
Focus on integrability, standards, and usability
Extension of design knowledge
App in a data ecosystem (provider)
Qualitative group discussion
Research Outputs:
Meta-Requirements
Design Principles
Prototypes
Evaluation Results
Paper IX

Figure 3.1.: Overarching design science research approach (adapted from Peffers et al. [95]).

Following Peffers et al. [95], the DSR process consists of six steps that can be repeated in multiple cycles to refine artifacts incrementally.

1. *Problem Identification:* Specifies the research problem under investigation. In our case, the primary problem is the lack of prescriptive design knowledge for building third-generation DQ tools. Organizations struggle with adapting to a changing data landscape, leading to insufficient DQ tooling (cf. Section 4.1).

2. *Objectives of a Solution:* Based on the identified problems, the solution objectives describe a desirable solution in the form of meta-requirements. We specified the problems we identified throughout our research as four meta-requirements: automation, integrability, standards, and usability (cf. Section 4.1).

3. *Design and Development:* Comprises the search process for designing and building the actual artifact [64]. Throughout this dissertation, we designed four new DQ artifacts addressing one or more of the previously identified meta-requirements and accumulated the design knowledge in the form of DPs (cf. Section 4.2).

4. *Demonstration:* Demonstrates the applicability of the inferred design knowledge. We have realized each DQ artifact as a software prototype and present examples for implementing the derived DPs (cf. Section 4.2).

5. *Evaluation:* Measures if an artifact successfully achieves its purpose and is practically relevant using adequate evaluation strategies [114]. We evaluated each DQ tool individually during the respective case study. Moreover, we evaluated the accumulated design knowledge using a scenario-based demonstration and a group discussion on the DPs' reusability (cf. Chapter 5).

6. *Communication:* Is concerned with publishing the results of the DSR process and contributing to relevant scientific and managerial communities. This thesis aims to summarize individually published research results and communicate accumulated design knowledge to the relevant user groups. We also offer an action guideline that assists practitioners in creating successful DQ tools (cf. Section 4.3) and propose a software reference architecture for DQ tools in paper IX [7].

A common problem of DSR studies is finding the right balance between scientific rigor and addressing the 'street level' of problems practitioners face [78]. To ensure abstracted design knowledge becomes applicable in practice, researchers often rely on prescriptive DPs [77]. Such DPs offer the required know-how to create solutions in practice and help practitioners tackle the complexity of systems [53]. The formulation of DPs has gained prominence within the IS community in recent years. Today, numerous articles communicate codified DPs on all kinds of DSR artifacts.

Searching the meta-search engine Scopus using the search term *"design principles" AND "design science"* on the article's title and abstract yields 362 relevant papers. The results comprise studies reporting DPs for various tools, such as UIs in AI-based decision support systems [84] or energy market applications [101]. However, most of these studies draw their findings from single DSR projects. Deriving design knowledge from multiple cases with a shared interest is less common and lacks adequate methodological support [17]. Searching Scopus again using the search term *("design principles" OR "design theory") AND "multiple case study"* produces only nine relevant papers.

Consequently, there is a need for more studies that generate design knowledge from multiple cases. These would help gain a broader perspective on the phenomenon under investigation, generate more robust and profound results, and formulate DPs that are relevant in practice [121, 53]. Generally, the scientific community should try to bridge the methodological frameworks for conducting multiple case studies (e.g., Eisenhardt [41] and Yin [121]) and formulating design knowledge (e.g., Lee [80] and Gregor and Hevner [54]). Avdiji et al. [17] made a first attempt in this direction by abstracting design knowledge on the results of a comparative cross-case data analysis. However, this area of design science is still in its infancy and requires further attention. With our thesis, we try to address this gap and contribute to the proliferation of DPs grounded in multiple cases and the use of suitable methodologies. The following chapter outlines the details of our multiple-case study.

3.2. Multiple-Case Study

Case studies are well-suited for investigating complex phenomena within a natural setting and provide answers to the 'why' and 'how' questions of research [26]. They can help to unravel convoluted organizational problems and produce testable and empirically valid results [41, 121]. Consequently, case studies are a good fit for achieving our research goal and creating empirically grounded design knowledge in the domain of DQ tools. This thesis draws its findings from four cases that were part of a multiple case study (see Table 3.1). In contrast to single case studies, multiple cases are beneficial for producing more robust research findings, thus allowing better generalizability [121]. Consequently, a multiple case study is well-suited to foster the standardization of modern DQ tools and support practitioners with their design and development.

Table 3.1.: Overview of conducted case studies.

Case	**A**	**B**	**C**	**D**
Paper	III [4]	IV [8]	VII, II [11, 5]	VIII [6]
Use Case	DQ checks in data migrations	Cleaning master data sets	DQ checks in AI pipelines	DQ checks for data sharing
Duration	3 months	3 months	12 months	6 months
DQ Task [10]	Definition, Measurement	Definition, Measurement, Integration	Measurement, Analysis	Measurement, Integration
DQ Dimension [118]	Accuracy	Accuracy, Consistency	Accuracy, Completeness, Validity	Accuracy, Completeness, Validity
Primary Methodology	Case Study [26]	Action Design Research [108]	Action Research [18]	Design Science Research [64, 95]
Design Knowledge	Solution Description	6 Design Principles	11 Lessons Learned	9 Design Principles
Primary Data Source	Literature	Regular feedback	Regular feedback	Expert interviews
Instantiation	Console App	Full-grown App	Component in AI pipeline	Data ecosystem App
Evaluation	Accuracy measurement	Two focus group discussions	Focus group discussion	Focus group discussion

We carried out a different case study in each of the four DSR cycles (see Figure 3.1). All cases were concerned with designing and implementing a DQ tool, representing the respective unit of analysis [121]. The cases are described in papers III [4], IV [8], VII [11], and VIII [6] in detail. To investigate phenomena in their real-life contexts, we collaborated with companies from manufacturing and pharmaceutical business domains between 2020 and 2022 [103]. The partnering companies experienced that their established DQ solutions could not keep up with a rapidly changing environment, resulting in the provisioning of low-quality data. While each case features a different organizational setting, their common denominator is the need for a new DQ tool that addresses the meta-requirements of modern, third-generation DQ tools.

Following Eisenhardt [41, p.537], the goal of theoretical sampling is to select cases that are "likely to replicate or extend the emergent theory". To extend the design theory for DQ tools with new insights, we selected cases focusing on one or more of the DQ tool meta-requirements (cf. section 4.1). After we successfully completed four case studies and addressed all identified meta-requirements in multiple cases, we reached theoretical saturation. At this point, no further case studies were necessary as we were able to inform the problems of established DQ tools with suitable DPs, and additional learning would have been minimal [41, 26].

Figure 3.2 demonstrates the localization of the implemented DQ tools in an exemplary system architecture to differentiate the four case studies better. The DQ tools address different aspects of organizational data architectures, thus incorporating various requirements in their design. The following subsections briefly describe the four cases by highlighting the DQ problems under investigation (marked in *italics*) and specifying our solution design.

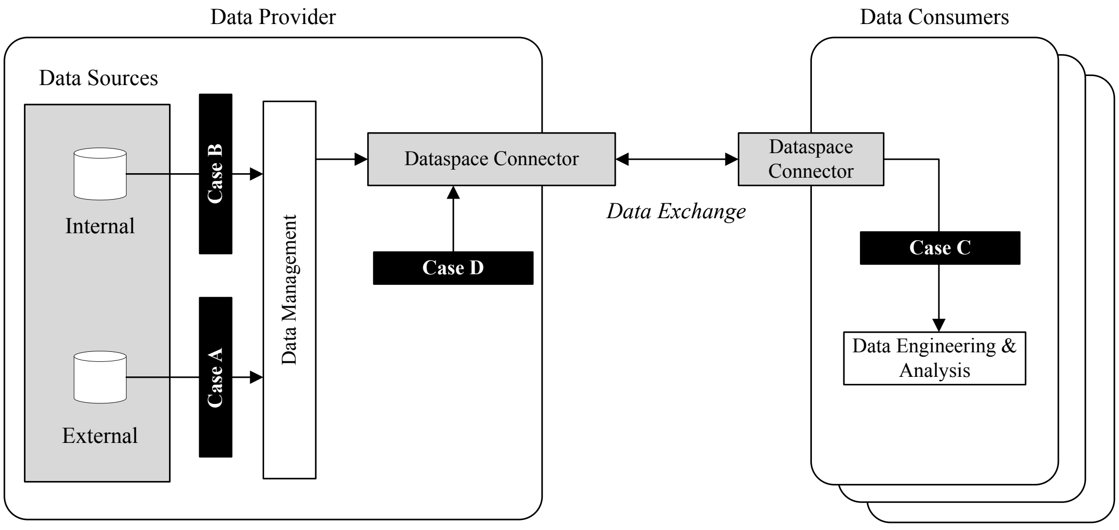


Figure 3.2.: Localization of implemented DQ tools in an exemplary system architecture (adapted from paper VIII [6]).

Case A: Automated Derivation of DQ Rules [4]

The first case was a three-month project aiming to develop a DQ tool for *automating* a manual data validation process. The partnering company experienced DQ issues when migrating external data from affiliated companies into their centralized databases. The DQ issues prolonged the migrations as the errors required manual correction, causing substantial financial costs. There were multiple reasons for quality problems, but they mainly resulted from inconsistencies in database schemata.

To support data migrations and improve DQ, the company's goal was to automatically derive DQ rules from their central database and validate incoming data against these rules. This way, they can avoid introducing new errors and ensure the integrity and correctness of their internal database. The biggest challenges of this approach were the complexity and high dimensionality of the data sets. To overcome these problems, we implemented a support vector machine (SVM) based algorithm to identify DQ rules at the schema level and the Apriori algorithm for generating DQ rules at the instance level. In combination, the algorithms could handle varying data sets subject to real-world characteristics, such as high dimensionality.

We combined quantitative and qualitative methods to evaluate the prototype. The quantitative part included an evaluation of the accuracy of the applied algorithms using test data. For the qualitative measurement, we conducted a workshop on the design and suitability of the application with data migration experts. We contributed the description of our solution design and highlighted potentials for further automating the DQ work.

Case B: User-friendly Data Cleaning [8]
In the second case, we conducted a three-month ADR project in cooperation with Boehringer Ingelheim, a large German pharmaceutical company. To deal with DQ, Boehringer Ingelheim followed a 'consulting approach' [117] in which a specialized DQ team manually reviewed important internal data sets and corrected errors in collaboration with data domain experts. They experienced this approach as time-consuming and expensive. Moreover, a usually large number of data errors forced the DQ team to focus on specific data sets and attributes, resulting in a low DQ coverage and potentially unidentified errors.

Boehringer Ingelheim envisioned a DQ tool that uses ML techniques to *automatically* identify data errors and generate DQ rules. With this tool, they aimed to reduce the amount of manual DQ work and save costs. During the ADR project, we developed a prototypical solution capable of identifying outliers, pattern violations, and rule violations. To create an impactful solution, we worked on the material master data set provided by Boehringer Ingelheim. Specifically, we extended our developments from the first case with interfaces for better *integration* with established data management tools. Moreover, we realized a user interface (UI) offering improved *usability*. The UI allowed users to optimize algorithms for different use cases and improved the presentation of data errors, making it easier to follow up on DQ issues and solve the underlying data problems.

At the end of the ADR project, we conducted a qualitative group discussion with 14 participants, including the ADR team and data domain experts, for evaluation. We received positive feedback for our developments from all participants, and Boehringer Ingelheim planned to continue working on the prototype. By reflecting on our learnings during the case, we derived six DPs that abstract our design decisions.

Case C: Embedding DQ in AI Pipelines [11]
The goal of the third case was to include DQ checks in a collaborative AI pipeline to ensure high-quality data is used for training an AI model. To achieve this goal,

we conducted a twelve-month Action Research (AR) project together with Mondragon Corporation, a manufacturing company from Spain. Mondragon wanted to quality-check sensor data streams to predict potential problems during production processes and reach the goal of zero-defect manufacturing. Since Mondragon lacked the necessary technologies and know-how to conduct quality checks in-house, they aimed to outsource this task to an external DQ service provider.

During the project, we designed and implemented a data-sovereign AI pipeline between Mondragon (i.e., the data provider) and a research institute (i.e., the DQ service provider). The pipeline comprised three main software components: a dataspace connector realizing data sovereignty, a software component at Mondragon used for data collection and preparation, and the AI-based DQ solution offered by the research facility. Using this pipeline, Mondragon sent batches of sensor data, which the DQ service provider analyzed for quality issues and returned a DQ report. We faced several obstacles in designing and implementing the DQ service. The biggest challenge was the *automated* measurement of DQ along several dimensions in a limited time. Another difficulty was establishing a trail between batches of sensor data and the corresponding DQ reports, enabling data lineage. We extended the developments from the second case and implemented multiple algorithms for identifying DQ issues along the dimensions of accuracy, completeness, and validity. To detect potential concept drifts, we measured the accuracy with an algorithm we specifically developed for data stream analysis and presented in paper II [5]. We also developed a *standardized* metadata model and ensured the *integrability* of our solution with the remaining software components.

To evaluate our developments, we conducted a qualitative focus group discussion with nine members of the AR team. Overall, introducing a new DQ component was well-received and enabled Mondragon to exploit its data sets more effectively. In a separate workshop meeting, the core development team reflected on the project's design decisions, success factors, and downsides, resulting in ten lessons learned. Not all lessons learned specifically target the DQ service, but abstract learnings regarding the integration and interaction of the different components of our solution. These learnings are, nevertheless, important for designing DQ tools and supporting their integration with existing tools and processes.

Case D: DQ and Data Ecosystems [6]
We investigated the use of DQ tools in data ecosystems as part of a six-month DSR project in the fourth case. In this project, we cooperated with a large manufacturing company, which we called MCo for anonymization. To promote inter-organizational data sharing, MCo took part in a data ecosystem based on the guidelines of the IDSA [92]. They experienced that sharing faulty data with external organizations entails a labor-intensive data cleaning process as data consumers lack the necessary data domain knowledge for efficient data cleaning. Consequently, they envisioned a solution ensuring they only share high-quality data with business partners. This solution required introducing a technically enforced DQ check prior to data sharing and connecting the currently disjunct processes for DQ analysis and data sharing. Concretely, MCo shared

sensor data streams with an external company offering data analysis services. Operating on high-quality data was vital to ensure accurate analytical results and avoid data-cleaning efforts.

Over three DSR cycles, we incrementally designed and implemented a DQ solution that meets the goals of MCo. The implemented solution follows the guidelines for International Data Spaces (IDS) 'Data Apps' [92] to enable its *integration* into the data ecosystem. We, therefore, built on our previous developments from the third case and extended and *standardized* the DQ tool for operation in the data ecosystem. One of the biggest obstacles with this was the development of generally applicable DQ metrics that can be applied to various use cases and offer high *usability.*

We evaluated our developments qualitatively in a group discussion, including the DSR team members and members of MCo's management. Both the design of our solution and its instantiation received positive feedback and can contribute to leveraging the benefits of data sharing. We formalized the generated design knowledge in the form of nine DPs.

3.3. Cross-Case Data Analysis

To accumulate the design knowledge generated during our four case studies, we theorize on real-world designs and develop a nascent design theory by "reflecting upon what has been done" [52, p.7]. Using the case studies as the primary knowledge base, we aim to elicit meta-requirements and formulate design principles that support the design of future DQ tools [85]. However, as previously noted by Avdiji et al. [17], there is no dedicated method for generating design knowledge from multiple DSR studies. We thus followed their example and conducted a comparative analysis of the design knowledge generated throughout our cases. We argue that the cases are similar and comparable as they focus on the same business problem, namely an insufficient DQ tooling [64]. By systematically comparing the generated design knowledge, we can search for cross-case patterns [41]. Integrating different perspectives on the same business problem leads to a robust and profound solution [121].

For the cross-case analysis, we aim to identify regularities and irregularities across the different case designs. We consider a single design item (i.e., a DP, design decision, or lesson learned) a **regularity** if it appears similarly in at least two cases. The occurrence in multiple cases suggests that the design item is relevant to the phenomenon under investigation and should become part of the accumulated design knowledge [17]. In addition to the regularities, irregularities represent design elements appearing in only one case. Using secondary data sources (e.g., related studies or literature), we can subdivide the irregularities into **considered irregularities** and **unconsidered irregularities**. The former category contains elements that only appear in a single case but are still included in the result, as they are highlighted in other studies or literature as well. We reason their reference in other studies supports their inclusion in the accumulated design knowledge. The latter category comprises irregularities of no relevance, which describe the design of other software artifacts. Figure 3.3 visualizes our conceptual approach for analyzing and accumulating design knowledge from multiple cases.

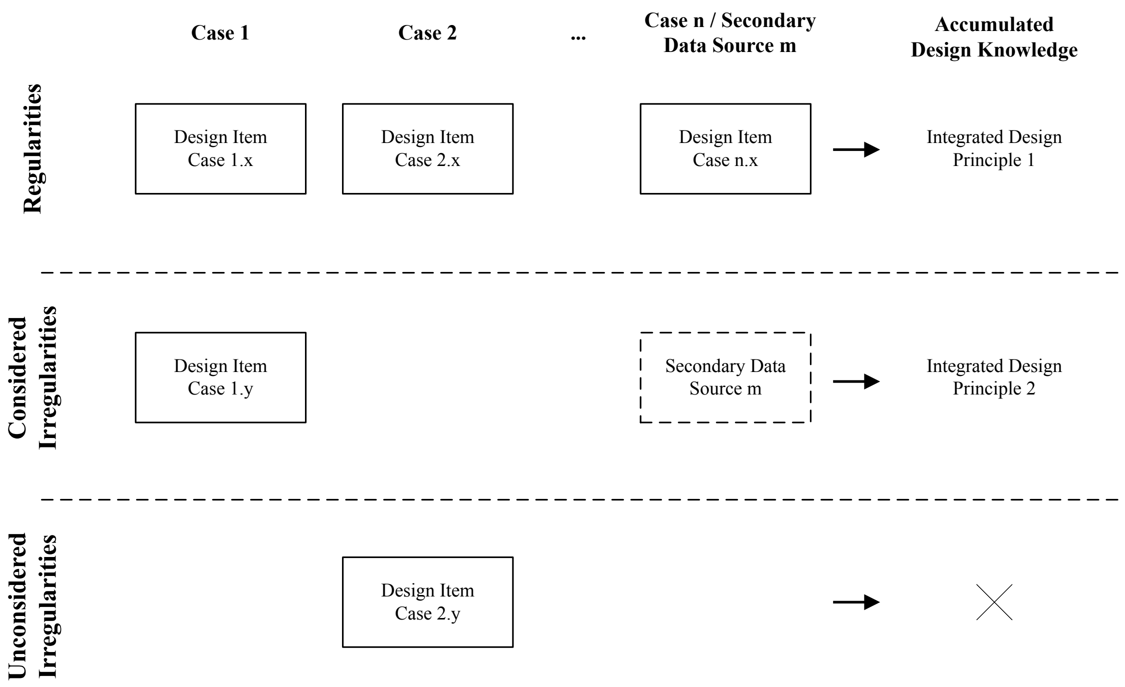

Case 1
Case 2
...
Case n / Secondary Data Source m
Accumulated Design Knowledge
Regularities
Design Item Case 1.x
Design Item Case 2.x
Design Item Case n.x
Integrated Design Principle 1
Considered Irregularities
Design Item Case 1.y
Secondary Data Source m
Integrated Design Principle 2
Unconsidered Irregularities
Design Item Case 2.y

Figure 3.3.: Conceptual approach for the cross-case analysis.

In our case, the design knowledge presented in the case papers forms the primary data source. Depending on the case, it takes on different forms, including DPs (cases B and D), lessons learned (case C), and a design description (case A). In each case, the design knowledge emerged from the interplay between researchers and practitioners, taking the requirements of the socio-technical environment into account [64, 108]. Since case A (paper III [4]) does not offer formalized design knowledge, we reviewed the presented design description and derived four design decisions. These decisions abstract essential design features of the implemented prototype and allow us to better compare the design knowledge with the remaining cases. In addition to the case studies, we used the results from papers II [5] and VI [10] as secondary data sources. These two papers do not describe the realization of a DQ tool per se but describe a DQ algorithm (paper II) and inform about the functional composition of DQ tools in general (paper VI). With this, they contribute additional knowledge to the design of DQ tools.

We initialized the cross-case analysis by aggregating the design knowledge from our primary data sources to identify regularities and irregularities. To systematically compare different design descriptions, we coded the design items using Grounded Theory Methodology (GTM) as an example. GTM is a widely established approach for analyzing different kinds of qualitative data [34]. We initialized the coding procedure with an open coding of the available design items. Subsequently, we grouped similar design items together and repeated this step multiple times until a sound and rigorous structure emerged [121]. This procedure yielded in the identification of ten regularities comparable to 2nd-order concepts in GTM [34]. We moreover identified seven irregularities appearing in only one case. We further subdivided these irregularities into three considered and four unconsidered irregularities by considering secondary data sources.

Throughout the data analysis process, we followed the concept of investigator triangulation to ensure the reliability of our results [94, 17]. Specifically, three independent researchers familiar with the applied research methodologies and DQ tools scrutinized the results of our data analysis. They reviewed our results for their soundness, consistency, and comprehensibility. Moreover, they helped us review the irregularities and decide on their inclusion in our results.

We continued our cross-case analysis by formulating a new DP for the ten regularities and the three considered irregularities. The new DP abstracts the design elements from the individual cases to become universally applicable. To formulate prescriptive DPs, we followed the structured approach by Kruse et al. [77]. As a result, we obtained accumulated design knowledge in the form of 13 DPs. We finally categorized each DP by assigning it to the DQ problem and meta-requirement. This way, our design knowledge is 'multi-grounded' as it is based on real-world instantiations ('empirical grounding') and contributes to fulfilling the identified DQ meta-requirements ('value grounding') [50]. The prescriptive accumulated design knowledge can inform the design of successful DQ tools and constitutes the main result of this dissertation. We describe our results in more detail in the next chapter.

4. Design Knowledge

Formulating the prescriptive design knowledge necessary for building successful DQ tools is the main contribution of this dissertation. The presented design knowledge emerged from accumulating the design knowledge generated in four case studies throughout this dissertation. To gradually build up the design knowledge, we need to answer the following questions: *What are the problems organizations face regarding DQ, and what are the corresponding objectives for successful DQ tools?*, *What are the DPs to address these objectives?*, and *How can these DPs be applied in an organization?*.

The following four subsections will answer these questions and help other researchers follow our conclusions and results [54]. We created two figures summarizing our results. Figure 4.1 provides a holistic view of the accumulated design knowledge derived from the four case studies. In Figure 6.1, we offer our consolidated results, comprising the problem space, made up of DQ problems and meta-requirements, and the solution space in the form of DPs.

4.1. Problem Identification & Meta-Requirements

Eliciting impactful design knowledge requires defining a specific research problem and formulating objectives that future solutions should fulfill [64, 103]. A concretely defined research problem helps to justify the value of our findings and makes it easier for other researchers and practitioners to follow our line of reasoning [95]. Given a specific research problem, we can further substantiate the problem space by formulating meta-requirements that describe the objectives for successful DQ tools [55]. Following Peffers et al. [95, p.55], the meta-requirements offer "... a description of how a new artifact is expected to support solutions to problems not hitherto addressed."

We conducted three studies (papers I [12], V [113], and VI [10]) to investigate the DQ problems organizations face more closely and understand the downsides of established DQ tools. Paper I [12] presents the results of a case study in collaboration with an automotive company. Our goal was to gain an in-depth view of the data management problems at the process level and understand why established solutions fail to address these problems. In response to this question, we conducted seven qualitative interviews and uncovered six significant challenges for DQ and data governance. In paper V [113], we investigated the deficiencies of DQ solutions from a general data engineering perspective. Using an SLR as the methodological basis, we created a data engineering reference model comprising six phases and four layers and identified areas lacking support for DQ. With a taxonomy of DQ functionalities and a systematic review of 18 commercial DQ tools in paper VI [10], we derived emerging capabilities and highlighted current trends.

The results showed that novel functional areas of DQ tools, such as automation, are underdeveloped and require new solutions.

The three studies helped us reveal four main problems of DQ tools, which served as the problem-centered entry to our research [64]. Throughout this dissertation, we confirmed and addressed the identified problems in multiple cases. In the following subsections, we describe the four main problems of DQ tools in detail and define the resulting meta-requirements. We formulated the meta-requirements using 'should' phrases, as suggested by Offermann et al. [86], to indicate that DQ tools should fulfill these requirements to become more successful. Table 4.1 summarizes our results and highlights the studies and cases in which we observed the respective problem.

Table 4.1.: DQ problems and corresponding meta-requirements.

		Identified In	
DQ Problem	**Meta-Requirement**	*Papers*	*Cases*
Automation	*MR1*: DQ tools should automate tasks to ensure that data of different formats and sizes can be quality-checked.	V, VI	A, C
Integrability	*MR2*: DQ tools should enable integration in different contexts to be usable in a distributed data architecture.	I, VI	B, C, D
Standards	*MR3*: DQ tools should apply standards to ensure a common understanding of DQ is in place.	I	C, D
Usability	*MR4*: DQ tools should offer high usability to become accessible for users from various backgrounds.	I	B, D

Automation

Rationale: The findings presented in paper VI [10] revealed that DQ tools face a need for more automation, and the implementation of intelligent algorithms is a major trend. Without automation, DQ tools cannot efficiently evaluate the quality of big and diverse data sets, a capability that is vital for organizational success [36, 100, 109]. Commercial DQ tools started to adapt to this need and increasingly incorporate AI and ML methods. Automated error correction, DQ rule generation, or record linkage are examples of this functional area [10]. However, despite big data becoming increasingly important, most DQ tools focus on simple data errors and specific data sets or have limited options for configuration. Automation and intelligent solutions are only partly supported and often used for marketing purposes without clear documentation [10]. There is a need for additional knowledge on how automation can be achieved in DQ tools and tackle big data [109]. We confirmed the need for improved automation in all four case studies. In particular, cases A and C dealt with handling big data and implementing new DQ algorithms to overcome typical real-world data problems, such as sparsity and high dimensionality [4, 11]. As a result, we realized novel DQ solutions that helped the partnering companies automate previously manual DQ tasks.

MR1: DQ tools should automate tasks to ensure that data of different formats and sizes can be quality-checked.

Integrability

Rationale: The insufficient integration of DQ tools with associated tools and processes is a problem we identified in almost all of our studies. In paper I [12], we observed that DQ tools operating within a heterogeneous system landscape often remain detached and insular. Consequently, users were frustrated as they faced the same errors multiple times and spent much time with recurring DQ work. Generally, we observed that a central DQ tool is no longer sufficient in increasingly decentralized data architectures (e.g., Data Mesh). A centralized solution cannot comprehend the large variety of requirements raised by different business domains and stakeholders, failing to provide users with the support they need [6]. As a result, the DQ work unravels, and each domain takes care of its DQ. Naturally, this affects the design of DQ tools, which need to become easily integrable and adaptable to various technical environments [38, 56, 10]. This way, organizations can avoid operating several different DQ tools and, at the same time, ensure that DQ work follows shared guidelines and rules. As a concrete example, Geisler et al. [48] suggest the creation of knowledge-driven data ecosystems that support quality checks in multiple data domains. The DQ tools in case studies B, C, and D were confronted with different application landscapes and realized various aspects of the integrability requirement.

MR2: DQ tools should enable integration in different contexts to be usable in a distributed data architecture.

Standards

Rationale: The aforementioned decentralization of data architectures and DQ tools raises a need for suitable and commonly agreed standards. Standards can help to establish a shared understanding of quality metrics and how they are measured [48, 56]. Otherwise, it would be difficult for users without data domain knowledge to understand what certain DQ scores mean and how these should be interpreted. Following Geisler et al. [48], standards must be flexible and combine global guidelines with domain-specific DQ definitions to be accepted by all stakeholders. In paper I [12], we observed a lack of standardization regarding DQ definitions and data governance practices, which resulted in high data-cleaning efforts. Some companies tackle this complexity by applying a 'consulting approach' to DQ [117]. In this approach, a team of internal or external DQ experts validates data sets against manually derived DQ rules. However, this method is of limited scalability, time-consuming, and expensive [8]. Throughout case studies C and D, we experienced that standards are vital for the success of inter-organizational collaborations and data exchanges. In these scenarios, all partners must share the same DQ definitions to avoid a significant communication overhead in resolving quality issues and allow for efficient cooperation.

MR3: DQ tools should apply standards to ensure a common understanding of DQ is in place.

Usability
Rationale: The trend for data products that are managed and engineered in a distributed fashion led DQ to become less IT-centric and, instead, a joint effort involving stakeholders from various parts of the organization [38, 10]. This means DQ is no longer solely achieved by a highly skilled team but is a concern for everyone offering data, raising a need for improved usability of DQ tools. In paper I [12], we witnessed how low usability can impair the success of DQ tools. The users were missing transparent and clearly defined DQ workflows in their current tool, which led to a low responsibility for DQ among the workforce [8, 6]. Moreover, in cases B and D, we observed that users without data science expertise had difficulties comprehending DQ scores and taking adequate action [8, 6]. We reasoned that DQ tools must become accessible to a much larger user group and offer a suitable UI, including suggestions for improving DQ, algorithmic explanations, and feedback mechanisms [10]. So far, there is little research on the usability of DQ tools in the scientific literature. However, such features can lower the barrier to DQ and facilitate the tool's long-term success.

MR4: DQ tools should offer high usability to become accessible for users from various backgrounds.

4.2. Design Principles

Building DQ tools that address the described meta-requirements is a complex issue, as the prescriptive know-how to build such tools is currently missing. This section tries to close this research gap by proposing 13 DPs based on the design decisions and experiences we made in four industrial case studies that were concerned with building DQ tools (see Table 3.1). The presented design knowledge accumulates the findings presented in these case studies by following the cross-case analysis approach described in section 3.3. Figure 4.1 depicts the final result of the cross-case analysis and summarizes the accumulated design knowledge. The figure shows similar design elements in the same row alongside the corresponding DP in the accumulated design knowledge. Please note that the figure contains DP3 of Case D twice, as the DP described two different design elements. We highlighted the relevant aspect of the DP in bold.

Using the identified meta-requirements as a basis, we structured the DPs into four subsections. For each DP, we provide a definition and an explanation. To better operationalize the DPs and decouple them of their high level of generalizability, we offer concrete usage examples based on our case studies and a description of organizational implications. The findings can serve as a guide for implementing customized solutions and help practitioners evaluate offerings in the market.

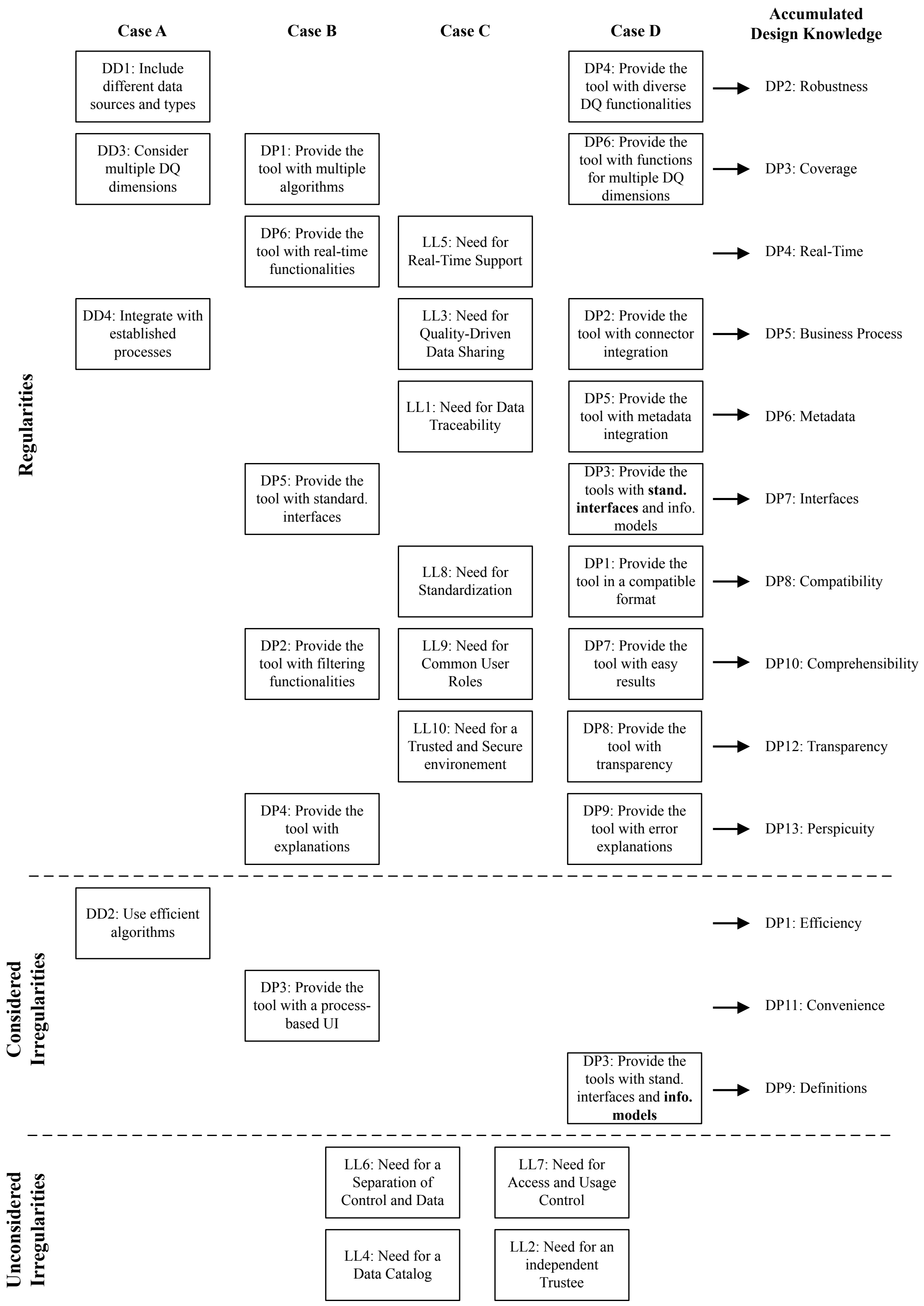

Case A
Case B
Case C
Case D
Accumulated Design Knowledge
Regularities
DD1: Include different data sources and types
DP4: Provide the tool with diverse DQ functionalities
DP2: Robustness
DD3: Consider multiple DQ dimensions
DP1: Provide the tool with multiple algorithms
DP6: Provide the tool with functions for multiple DQ dimensions
DP3: Coverage
DP6: Provide the tool with real-time functionalities
LL5: Need for Real-Time Support
DP4: Real-Time
DD4: Integrate with established processes
LL3: Need for Quality-Driven Data Sharing
DP2: Provide the tool with connector integration
DP5: Business Process
LL1: Need for Data Traceability
DP5: Provide the tool with metadata integration
DP6: Metadata
DP5: Provide the tool with standard. interfaces
DP3: Provide the tools with stand. interfaces and info. models
DP7: Interfaces
LL8: Need for Standardization
DP1: Provide the tool in a compatible format
DP8: Compatibility
DP2: Provide the tool with filtering functionalities
LL9: Need for Common User Roles
DP7: Provide the tool with easy results
DP10: Comprehensibility
LL10: Need for a Trusted and Secure environement
DP8: Provide the tool with transparency
DP12: Transparency
DP4: Provide the tool with explanations
DP9: Provide the tool with error explanations
DP13: Perspicuity
Considered Irregularities
DD2: Use efficient algorithms
DP1: Efficiency
DP3: Provide the tool with a process-based UI
DP11: Convenience
DP3: Provide the tools with stand. interfaces and info. models
DP9: Definitions
Unconsidered Irregularities
LL6: Need for a Separation of Control and Data
LL7: Need for Access and Usage Control
LL4: Need for a Data Catalog
LL2: Need for an independent Trustee

Figure 4.1.: The accumulated design knowledge for data quality tools.

4.2.1. Automation

The first meta-requirement addresses the need for more intelligent and automated approaches to DQ, for example, using AI or ML algorithms. Considering the growing importance of big data, it has become necessary to automate parts of the DQ life cycle to cope with more extensive and diverse data sets [109, 24]. Otherwise, companies would be overwhelmed by the amount of manual DQ work, which often results in insufficient DQ coverage [4, 8]. To address this meta-requirement, we formulate four DPs that DQ tools should realize (see Table 4.2).

Table 4.2.: Design principles for automation in DQ tools.

Aspect	**Design Principle / *Application Example***
Efficiency	DP1: Provide the DQ tool with intelligent algorithms for users to identify DQ issues efficiently, given that the training data and know-how to train such algorithms are available [4, 8, 11, 6]. *Example: We used an SVM-based concept drift analysis to identify inconsistencies that occur over time in semi-structured sensor data (Case C).*
Robustness	DP2: Provide the DQ tool with robust algorithms for users to conduct DQ validations on data sets varying in size and format, given that different data sets are in use [4]. *Example: We implemented the Apriori algorithm, which is capable of handling high-dimensionality and free texts and could maintain a short execution time (Case A).*
Coverage	DP3: Provide the DQ tool with multiple algorithms for users to find different data errors and DQ rules, given that the error types in the data set are unknown [4, 8, 6]. *Example: We used multiple algorithms to identify several kinds of data errors at the schema and instance level of data at once (Case A).*
Real-Time	DP4: Provide the DQ tool with a real-time error detection functionality for users to be automatically informed of DQ issues, given that the data set is volatile [8, 11]. *Example: The case did not offer native real-time support. Instead, we received batches of data over HTTPS in regular intervals, triggering the DQ analysis (Case C).*

DP1 - Efficiency
Provide the DQ tool with intelligent algorithms for users to identify DQ issues efficiently, given that the training data and know-how to train such algorithms are available.

Explanation: There is a clear trend for DQ tools to become more efficient by implementing intelligent AI and ML algorithms [10]. Using such methods can help automate the DQ life cycle, reduce the amount of manual DQ work, and increase the DQ coverage. For instance, DQ tools can learn from past errors and improve the error detection rates in new data. The results can be further enhanced when multiple data sources are combined, as in contextual ML [2]. Although relevant in all cases, only case A mentioned efficiency explicitly. The reference in case A and the trend for efficient algorithms in DQ tools described in the literature (e.g., [10, 116]) convinced us to include the DP in our result.

Example: Generally, intelligent algorithms can be separated into supervised and unsupervised approaches, depending on whether they require training data. In all four cases, we used unsupervised algorithms as no training data was available in the respective industrial context. For instance, in case C, we implemented a concept drift analysis based on an algorithm developed in paper II [5]. Our proposed algorithm extends established concept drift approaches by incorporating an SVM-based classifier, making the concept drift analysis available to high-volume data streams [5].

Organizational Implication: On a general level, several studies derived data-related challenges organizations should address to boost the success of data science initiatives (e.g., [56, 13, 113]). Regarding DQ, organizations need to gain an understanding of their AI/ML capabilities and match them to their DQ requirements. For example, using supervised error detection methods requires an adequate amount of labeled training data. Moreover, the success of intelligent DQ relies on the availability of employees with the required expertise. Besides DQ experts and data scientists, this includes several other roles along the data engineering pipeline [113].

DP2 - Robustness
Provide the DQ tool with robust algorithms for users to conduct DQ validations on data sets varying in size and format, given that different data sets are in use.

Explanation: The data landscape within organizations diversifies and can feature relational databases, data streams, unstructured logs, and many more data sources [38]. Each data type poses new requirements on DQ tools and requires different handling [109]. It is critical for a DQ tool to handle diverse data sets of varying size, complexity, and format [24]. The robustness can be achieved by offering multiple scalable algorithms for different kinds of data [4].

Example: In case A, we faced high-dimensional data sets that combined tabular and free text data. We conducted a systematic review of suitable algorithms for generating DQ rules and decided to implement the Apriori algorithm. It can handle such data and does not require prior training [4].

Organizational Implication: To realize robust DQ tools, organizations should review and categorize their data sets and select suitable DQ algorithms. As a prerequisite, organizations must understand what their data looks like and what DQ tasks they want to achieve. For typical data profiling tasks, Abedjan et al. [1] offer a survey that can act as a starting point for categorization. A human-in-the-loop approach to DQ can be beneficial in more complex situations, such as many different data sets or a lack of suitable algorithms [10]. Hereby, simple issues can be automatically corrected, while complex problems are remediated for manual review.

DP3 - Coverage

Provide the DQ tool with multiple algorithms for users to find different data errors and DQ rules, given that the error types in the data set are unknown.

Explanation: DQ tools must cope with varying data sets and offer broad coverage to address multiple use cases and stakeholders [91]. For this purpose, they must implement several algorithms to automatically assess the quality of a data set along multiple quality dimensions (e.g., completeness, accuracy, timeliness, etc.) [118]. This allows users to obtain a comprehensive view of the DQ and derive suitable actions.

Example: We combined the Apriori algorithm with an SVM-based algorithm in case A to identify quality issues on the schema and instance level of the data simultaneously [4]. This combination was necessary to fulfill the needs of two different data consumers interested in DQ rule compliance and completeness scores.

Organizational Implication: Building up DQ coverage implies that organizations develop an understanding of their DQ problems. For example, Rahm and Do [98] distinguish between single and multi-source and schema and instance-level DQ problems. Ge and Helfert [47] take a different perspective and differentiate between context-dependent and context-independent DQ issues. With a classification of relevant DQ problems, organizations can identify and implement algorithms that address these problems and offer a DQ tool valuable for different users [91]. Moreover, it is essential to carefully review the internal functioning of DQ tools, as the steps in which a DQ analysis and curation take place can influence the results [2]. For example, a preceding deduplication can affect the completeness analysis of a data set [4].

DP4 - Real-Time

Provide the DQ tool with a real-time error detection functionality for users to be automatically informed of DQ issues, given that the data set is volatile.

Explanation: Real-time data streams are on the rise and are leveraged by organizations for real-time data analytics [27]. According to Dehghani [38], a unification of batch and data stream processing is currently taking place that will further increase the importance of data streams for organizations. These developments invoke a need for integrating data streams into organizational data architectures and include them in DQ management efforts [56].

Example: We experienced the growing importance of data streams in cases B and C. Although no real-time data stream was present in either case, both partnering companies mentioned the need for real-time DQ checks in the future. Consequently, we argue that DQ tools should offer real-time error detection capabilities such as real-time data profiling and validation [8, 110].

Organizational Implication: Organizations should evaluate the need for real-time DQ checks based on their use cases. If there is a need, they must ensure that integration with the DQ tool is possible (e.g., using messaging bus technologies) and implement suitable real-time error detection algorithms. Especially when organizations realize AI/ML pipelines and related concepts such as MLOps, they should pay attention to real-time DQ concepts. However, real-time DQ and data management are still in their infancy and viable solutions are part of future work [27].

4.2.2. Integrability

To avoid isolating DQ work from established data management tools and processes and effectively apply DQ tools in decentralized data architectures, they need to integrate with different environments seamlessly [56, 48]. Improving the integrability of DQ tools will support their use in different organizational and technical contexts. Integrable solutions furthermore help democratize DQ work by making DQ tools available to more stakeholders within and across organizational borders and dispersing the responsibility for high-quality data [12]. Specifically, DQ tools should allow an integration on three architectural layers: business processes, metadata, and technical (see Table 4.3).

DP5 - Business Process
Provide the DQ tool with capabilities that enable its integration with established business processes for users to provide high-quality data to various processes, given that multiple DQ-relevant processes are in place.

Explanation: Decentralized data architectures that connect different data domains are becoming increasingly popular within and across organizational boundaries [38, 6]. This trend implies that DQ tools need to integrate with different technical and organizational contexts seamlessly, enabling their connection with the business processes of various data providers [87, 44]. For instance, sharing high-quality data products is vital for successful data ecosystems. Otherwise, data consumers could receive erroneous data, which can deteriorate dependent business processes and mitigate trust [20, 12].

Example: In case D, we integrated our DQ solution into the data-sharing process of the partnering company by offering suitable interfaces and data models [6]. As a result, we embedded DQ in the data-sharing process, which helped avoid disjunct processes and reduced manual DQ efforts as part of the sharing process.

Organizational Implication: To offer high-quality data internally and across organizational boundaries, organizations must include a DQ perspective in their business process management and re-design efforts [87]. Specifically, organizations should develop criteria to assess which business processes need DQ tool support based on factors such as

Table 4.3.: Design principles for integrability in DQ tools.

Aspect	**Design Principle / *Application Example***
Business Process	DP5: Provide the DQ tool with capabilities that enable its integration with established business processes for users to provide high-quality data to various processes, given that multiple DQ-relevant processes are in place [6]. *Example: We used Apache Camel as an integration framework to closely integrate the DQ tool with a dataspace connector used for data sharing (Case D).*
Metadata	DP6: Provide the DQ tool with methods to store the DQ result as metadata for users to establish a trail between the data set and the DQ result, given that such a trail can be established [11, 6]. *Example: Together with the project partner, we negotiated and implemented a new metadata model to enable lineage between data and the corresponding DQ result (Case C).*
Interfaces	DP7: Provide the tool with well-documented and standardized interfaces for users to integrate the tool in different technical environments, given that a heterogeneous system landscape is present [8, 6]. *Example: Integrating with a data management suite already in place was a crucial requirement. We thus reviewed the overall system landscape to identify requirements and create suitable interfaces in our DQ solution (Case B).*

process costs or number of data flows [12]. Subsequently, they can conduct tests to guarantee that integrating DQ tools in different data engineering processes is feasible and contributes to their success [116].

DP6 - Metadata

Provide the DQ tool with methods to store the DQ result as metadata for users to establish a trail between the data set and the DQ result, given that such a trail can be established.

Explanation: Metadata management is among the common data management challenges, and a lack of it can impair the success of AI initiatives [56]. Moreover, the success of data engineering in heterogeneous data landscapes and data ecosystems relies on the availability of adequate metadata [48]. The metadata should include information on the quality of a data set to assist data consumers in finding and selecting high-quality data sets [11]. Especially when data consumers and providers are unfamiliar, quality metadata is vital for data consumers to evaluate the usefulness of data efficiently.

Example: To provide a common understanding of DQ metrics and their interpretation, we negotiated and implemented a shared metadata model in case C. This was important to create lineage between a data set and its quality scores and allow users to follow up on data errors.

Organizational Implication: For organizations, it is crucial to ensure that the results of DQ tools are compatible with their internal data models. Consequently, the DQ metadata will not remain isolated but can help estimate data preparation efforts and build trust in data sharing [6, 20]. Since metadata is often stored in a data catalog, organizations might need to create possibilities for integration between the DQ and data catalog solutions. Providers of commercial DQ tools identified the same requirement and often offer DQ and data catalog solutions combined into a single tool suite [10].

DP7 - Interfaces

Provide the tool with well-documented and standardized interfaces for users to integrate the tool in different technical environments, given that a heterogeneous system landscape is present.

Explanation: The integration of business processes in DP5 necessitates the technical integration of DQ tools with other tools. Nowadays, most organizations operate in a heterogeneous system landscape featuring many different data stores and a variety of software solutions [38]. New tools must integrate with the existing system landscape to avoid being isolated from established processes and remain segregated [8]. A common approach to meet this requirement is implementing a (micro) service-oriented architecture and operating standardized and well-documented APIs. These enhance the agility and flexibility of applications, allowing them to operate in a changing environment [8, 44]. Moreover, standardized interfaces are easier to operate and enable a seamless data transfer [12].

Example: It was an essential requirement in case B to connect our DQ solution to an established data management tool. Without the integration, users would have to manually transfer quality results to the data management tool, resulting in a laborious and cumbersome solution. To address this requirement, we reviewed the system landscape and implemented interfaces that adhere to the data models and communication protocols specified by the data management tool.

Organizational Implication: Using the overview of DQ-relevant business processes described in DP5 as a basis, organizations can review possibilities for integration between the DQ tool and other applications it needs to communicate with. Application integration frameworks are suitable for simplifying the technical integration of different tools. For example, Apache Camel [16] implements established enterprise integration patterns and is suitable for realizing technical process integration.

4.2.3. Standards

While decentralized DQ work is increasingly common, DQ tools must still follow global governance schemes and standards [38, 6]. These help DQ tools become more comprehensible and adaptable to varying contexts. To improve standardization in DQ tools, they need to address two requirements. First, the standardization of the DQ tool, and second, the standardization of DQ definitions. Organizations can meet these requirements by realizing the DPs shown in Table 4.4.

Table 4.4.: Design principles for standards in DQ tools.

Aspect	**Design Principle / *Application Example***
Compatibility	DP8: Provide the DQ tool in a compatible format for users from different contexts to benefit from DQ capabilities, given that the DQ tool has users in various domains [11, 6]. *Example: We prepared our DQ tool as an IDS Data App to ensure its compatibility among participants of the IDS data ecosystem in place (Case C).*
Definitions	DP9: Provide the DQ tool with DQ standards for users to have an agreed definition of DQ metrics, given that existing definitions are ambiguous [6, 8]. *Example: We extended the existing information model with DQ metrics that we based on the DQV (Case D).*

DP8 - Compatibility

Provide the DQ tool in a compatible format for users from different contexts to benefit from DQ capabilities, given that the DQ tool has users in various domains.

Explanation: Trust is an essential building block for the success of any data collaboration [104, 20]. High-quality data is indispensable for creating this level of trust as it ensures the accuracy of data-based decisions and avoids data preparation efforts [11]. Especially for data consumers who do not always have the necessary data domain knowledge, it could otherwise be challenging to resolve data errors efficiently [6]. Technically enforced quality checks are a means for realizing this functionality and can guarantee that data meets the agreed standards in form and content [48, 6]. For this purpose, all data providers need access to adequate DQ tooling, and the DQ tool must be compatible with different organizational and technical contexts [116].

Example: We prepared our DQ tool in case C as an IDS data app, which defines standards for software solutions participating in an IDS-based data ecosystem. These standards include implementing specific interfaces, security measures, and deployment options. Preparing our solution as a data app allowed its distribution over an ecosystem-internal app store, making the tool available to all participants and supporting its proliferation [92].

Organizational Implication: To provide compatible DQ tools, organizations need to establish a global guideline for DQ tools as part of their data governance and management activities. Especially when multiple stakeholders are involved, negotiating such standards can be difficult. In such cases, it can be beneficial to participate in an established data ecosystem initiative (e.g., the IDSA [67]) and leverage the available standards. However, even within a single organization, data ecosystems might be valuable to ensure compatibility across business domains and overcome common data challenges [56].

DP9 - Definitions
Provide the DQ tool with DQ standards for users to have an agreed definition of DQ metrics, given that existing definitions are ambiguous.

Explanation: The lack of harmonized information models and incompatible data sets represent typical data management problems [27, 48]. They complicate data integration, the operation of shared AI pipelines, and hinder data reuse. As a result, manual data preparation work is usually high and causes significant costs [56]. The same applies to DQ definitions, which are often ambiguous and interpreted differently by various stakeholders. This makes it challenging to address DQ issues adequately and can damage trust in the data. To avoid ambiguity and vague DQ definitions, organizations must negotiate common DQ standards acceptable to all DQ stakeholders [48]. We only identified the need for definitions in case D, but its reference in other studies (e.g., [48, 56]) motivated us to add it to our results.

Example: In case D, we extended the metadata model in place with DQ metrics based on the Data Quality Vocabulary (DQV) offered by the W3C. Using an open DQ standard helped all parties involved in the DQ life cycle to interpret quality measurements correctly and create trust in the DQ solution.

Organizational Implication: Organizations must ensure that their DQ tools follow standardized DQ vocabularies and information models. For example, the W3C [115] and the ISO [69] offer frameworks to describe the quality of a data set in a structured and commonly agreed way. Users can advance and customize these standards for the intended purposes and use cases. For some industries (e.g., medical or automotive), it can be valuable to refine these standards further and create an industry-wide DQ standard.

4.2.4. Usability

Providing high-quality data is no longer a task for a specialized data management team but rather a joint effort of data owners across the organization [38, 48]. Naturally, this affects the design of DQ tools, which need to become accessible to a wider audience and usable for people from different backgrounds [10, 8]. As a consequence, it is vital for DQ tools to reduce their inherent complexity and offer an understandable and easy-to-use UI. DQ tools should realize the following four DPs to address the need for better usability (see Table 4.5).

Table 4.5.: Design principles for usability in DQ tools.

Aspect	**Design Principle / *Application Example***
Comprehensibility	DP10: Provide the DQ tool with adequate DQ scores for users to get an easily interpretable result, given that multiple DQ measurements exist [6]. *Example: We combined DQ scores for the DQ dimensions completeness, accuracy, and validity to a single score between zero and one, which is easy to interpret by other users (Case D).*
Convenience	DP11: Provide the DQ tool with a process-based user interface for users to follow a structured DQ approach, given that no process is in place [8]. *Example: The DQ tool featured a single page application that realizes a process-based UI to restrain users from incorrect usage (Case B).*
Transparency	DP12: Provide the DQ tool with transparent descriptions of the algorithms used for users to gain trust and understand their functionality, given that there are users without data science knowledge [6]. *Example: We ensured that the implemented AI/ML algorithms are well-documented and offered use case-specific exemplary outputs and their possible interpretations (Case D).*
Perspicuity	DP13: Provide the DQ tool with explanations of identified DQ issues for users to understand and resolve DQ problems, given that there are users without the necessary data domain knowledge [8, 6]. *Example: In discussions with the project partners, we selected DQ methods that are semantically easy to interpret (e.g., DQ rules) and favored these over black-box methods that are difficult to grasp and can damage the trust in the results (Case B).*

DP10 - Comprehensibility

Provide the DQ tool with adequate DQ scores for users to get an easily interpretable result, given that multiple DQ measurements exist.

Explanation: DQ is a multi-dimensional concept and can be hard to comprehend for users [118]. Data consumers who are unfamiliar with DQ definitions or lack data domain knowledge can have trouble interpreting a low DQ score and deducing adequate actions [48, 8]. In addition to detailed quality scores for different metrics, a DQ tool should offer a simple and aggregated DQ score. This score abstracts the complexity and multi-dimensional nature of DQ and is easy to interpret by data consumers [6]. It helps DQ to become more accessible, and consumers without in-depth data knowledge can quickly evaluate the quality of a data set.

Example: The project partners in case D posed the requirement to combine DQ metrics for completeness, accuracy, and validity to a single score between zero and one. The single score helped users without the necessary expertise obtain an impression of the DQ and inform them about potential data issues. In addition to the single score, we included the three individual metrics in the DQ result so users can identify the origin of a low DQ score.

Organizational Implication: Numerous DQ tools are available on the market, but they often lack simple functionalities for defining DQ properties and rules, validating data, and following up on data errors [110, 10]. For organizations, it is crucial to overcome these downsides by identifying the relevant stakeholders and developing DQ metrics that address the needs of the different users. They should include users from diverse backgrounds in the tool design or review process and retrieve early feedback.

DP11 - Convenience
Provide the DQ tool with a process-based user interface for users to follow a structured DQ approach, given that no process is in place.

Explanation: For any software tool, particularly for data-intensive tools, user acceptance is vital and significantly influenced by its UI [13, 110]. Since DQ used to be a highly specialized task, the UIs of DQ solutions are often inconvenient, with high semantic and syntactic barriers [110, 8]. Some solutions do not provide UIs at all and offer their functionalities as application programming interfaces (APIs) that users can integrate into their source code (e.g., Great Expectations [51]). This combination leads to mixed code bases comprising DQ and application code, which are harder to maintain. Moreover, users must learn new APIs and invoke them to define DQ dimensions and conduct evaluations [110]. These API-based DQ approaches make it difficult for inexperienced users to get involved and contribute high-quality data. Consequently, a DQ tool should offer easy-to-use functionalities for all steps of the DQ life cycle and support established user workflows [8]. Only in case B did we experience the need for a convenient UI. However, we still included the DP in the accumulated design knowledge as the study of Alhamadi et al. [3] supports its importance.

Example: In case B, we implemented the DQ tool as a single-page application, which ensures users follow a specific DQ pipeline. With the structured approach, we guarantee that DQ tasks are executed in a given order. We also included functionalities for filtering DQ issues to avoid users getting overwhelmed by the sheer number of quality problems in their data sets. This feature offered the possibility to focus on specific problems and proved to be a key success factor for the DQ tool.

Organizational Implication: The democratization of DQ work raises the need to integrate DQ tools in established processes (see also DP5). It is crucial to ensure that DQ tools have user-friendly UIs and follow established procedures to promote user adoption and reduce training needs. Organizations could, for instance, implement low-code DQ tools that follow data-flow programming concepts. In a recent paper [74], we made a first attempt in this direction. Generally, organizations should gather user requirements and conduct usability tests with different user groups to realize convenient and suitable DQ tools.

DP12 - Transparency

Provide the DQ tool with transparent descriptions of its algorithms for users to gain trust and understand their functionality, given that there are users without data science knowledge.

Explanation: Commercial DQ tools offered in the market often lack sufficient documentation and explanation of the algorithms used. In particular, AI and ML algorithms are often used for marketing purposes, and vendors offer little information about their internal functioning [10]. This lack of information makes it difficult for users to fine-tune DQ tools for application in different scenarios [6]. Moreover, the results of black-box algorithms are nontransparent and can damage users' trust in the DQ results and impair the tool's success [27].

Example: The DQ tool we developed in case D featured detailed documentation and usage examples based on data used in that case. With this, we helped the users of the DQ tool to understand how DQ metrics were derived and reproduce results.

Organizational Implication: The need for transparency causes organizations to ensure that the algorithmic basis of DQ tools is understandable for its users. Offering a DQ tool with sufficient and legible documentation can help users reproduce DQ results and resolve the underlying data errors. The reproducibility of DQ scores is crucial for resolving complex data errors that require remediation or cooperation of multiple business domains. Moreover, organizations should follow the trend for explainable AI and implement established concepts and approaches [22].

DP13 - Perspicuity

Provide the DQ tool with explanations of identified DQ issues for users to understand and resolve DQ problems, given that there are users without the necessary data domain knowledge.

Explanation: Following the previous DP, it is not only important to understand the internals of the algorithms but also to ensure that users can interpret and understand the results. Otherwise, it can be difficult to deduce suitable actions, and DQ problems might not be adequately addressed [8]. For example, Swami et al. [110] realized this requirement by offering a data validation report describing the results and essential evaluation details. Such explanations are, furthermore, necessary for the remediation of DQ issues to data domain experts as they can quickly grasp the DQ problems and come up with possible solutions.

Example: The DQ solution in case B comprised several algorithms for data profiling and quality rule generation. To derive DQ rules, we implemented the association rule learning algorithms Apriori and TANE. Of all implemented algorithms, these were best received as their results are semantically easy to interpret and do not necessarily require deep statistical or algorithmic knowledge.

Organizational Implication: Organizations should ensure that identified DQ issues are well-explained and interpretable by the relevant stakeholders. Realizing this requirement calls for conducting user acceptance tests and retrieving feedback on the perspicuity of DQ issues by users from various backgrounds early on. Moreover, when applying AI-based algorithms, it can be beneficial to include established guidelines for human-AI interaction (e.g., by Amershi et al. [13]) in the DQ tool design process.

4.3. Action Guideline

DQ is a concern for any organization, and the presented DPs are valid for a class of problem [64]. However, not all organizations are equal, and each organization must become aware of its characteristics and DQ requirements to create impactful solutions. With this knowledge, they can focus their resources on developing a design that fits their operational context and implementing relevant DPs [73]. This is considerably important for DQ tools as their design depends on the surrounding socio-technical context and usually requires a high degree of customization [10]. For example, an automotive company might use high-velocity data streams, while a company from the food industry uses a static set of product data. Real-time error detection capabilities are valid for both, but it is presumably more critical for the automotive company. In designing and building DQ tools, there will be few one-size-fits-all solutions, and organizations must prioritize their efforts to address the business' need for action. The prioritization helps them design and implement the DQ tools the organization needs and realize more successful DQ initiatives.

The DPs we presented above can help build customized tools and enable the provisioning of high-quality data to data consumers. To support practitioners with the design and development of DQ tools, we follow the example of Azkan et al. [19] and aim to translate our DPs into an action guideline. The action guideline comprises a set of rules in a given sequence that practitioners can easily follow [30]. To create the action guideline, we rearranged, subdivided, and related the DPs to each other. This approach led to an action guideline of six steps to help identify the DPs relevant to a specific case (see Figure 4.2). We evaluated the proposed action guideline and our overall results by applying it to an exemplary scenario (cf. Section 5.1).

To create the action guideline, we formulated statements that organizations can use to describe their DQ requirements and assess their individual situation. If a statement applies, the corresponding DPs are likely relevant to the organization. The action guideline also highlights cases where one DP raises the need to implement another. Such dependent DPs are shown as second or third-level DPs in the figure. After running through all steps an organization can make an informed decision about the DQ tool design and implement a customized solution.

The action guideline serves as a simplified model for selecting the DPs most important to an organization. In any case, reviewing all proposed DPs and evaluating their usefulness is valuable. In the following, we describe each step of the action guideline in more detail and elaborate on our rationale.

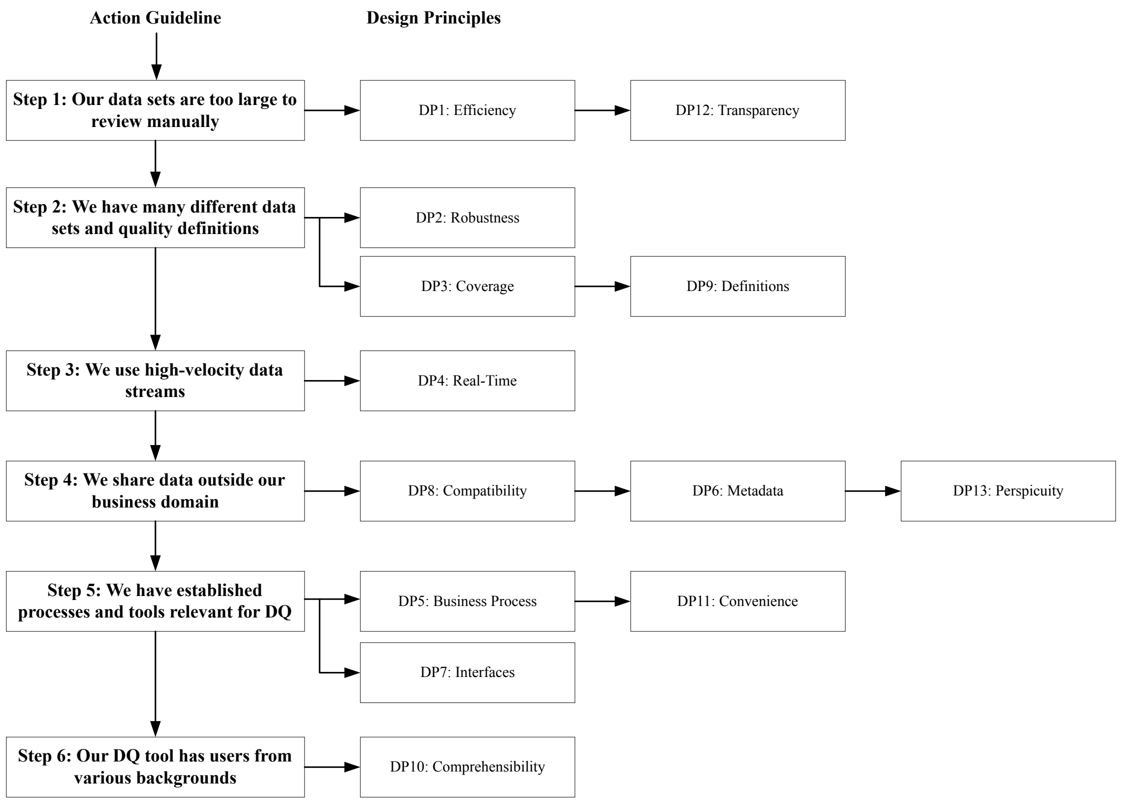

Action Guideline
Design Principles
Step 1: Our data sets are too large to review manually
DP1: Efficiency
DP12: Transparency
Step 2: We have many different data sets and quality definitions
DP2: Robustness
DP3: Coverage
DP9: Definitions
Step 3: We use high-velocity data streams
DP4: Real-Time
Step 4: We share data outside our business domain
DP8: Compatibility
DP6: Metadata
DP13: Perspicuity
Step 5: We have established processes and tools relevant for DQ
DP5: Business Process
DP11: Convenience
DP7: Interfaces
Step 6: Our DQ tool has users from various backgrounds
DP10: Comprehensibility

Figure 4.2.: Action guideline for designing DQ tools.

Step 1. In step one, the organization can differentiate on the volume of the data sets they want to quality check. Once a company experiences that it fails to review data manually and starts focusing on specific data sets, it requires automated and intelligent solutions (we had a similar experience in case B). When facing high-volume data, efficient curation and quality management are practically impossible for humans [109]. For instance, a data scientist cannot review millions of images to assess potential inaccuracies or biases in the data. Efficient algorithms such as AI-supported data profiling or deduplication become necessary DQ capabilities (DP1). However, using intelligent algorithms comes with the need for transparency in the algorithms used (DP12). Following up on DQ problems can be difficult without transparent algorithms, and other users will have trouble replicating the results.

Step 2. Step two covers two DPs, which classify an organization based on the number of different data sets and DQ metrics they want to address. If an organization wants to assess the quality of different data types (e.g., tabular, text, images, etc.), the robustness of a DQ tool becomes relevant (DP2). Often, varying data types require different algorithms or configurations. For example, our concept drift algorithm is suitable for profiling sensor streams but is of no use for profiling images [5]. Moreover, if an organization aims to analyze data along multiple DQ dimensions, a DQ tool should feature a high DQ coverage (DP3), which requires different algorithms. Naturally, a high DQ coverage implies that the different DQ dimensions are concretely defined (DP9). In other words, organizations need to determine their DQ requirements and find the optimal algorithmic basis. In this way, they can fully protect their data sets and contribute significantly to the success of data-intensive applications [113].

Step 3. In step three, an organization can indicate the use of high-velocity data streams. The continuous monitoring of quickly changing data can be vital for businesses for various reasons, including identifying machine failures or spotting changes in the accuracy of ML models [27]. If an organization aims to quality-check such data, it requires real-time DQ capabilities (DP4).

Step 4. Step four includes deciding what data domains take part in data engineering. When multiple data domains (e.g., cooperative data engineering [11] or data repurposing [48]) are involved, they need access to the DQ tool to conduct DQ tasks at the source [38]. To address this requirement, the DQ tool should come in a compatible format to allow its efficient use by data providers (DP8). However, this decentralization of DQ work raises the need to negotiate shared metadata models that feature DQ information and are feasible for different types of stakeholders (e.g., SMEs, enterprises) [48]. Without a DQ-enhanced metadata model, data consumers would have difficulties finding high-quality data sets. Since users from other data domains lack the data domain knowledge of the data provider, it is crucial to offer an explanation of DQ issues for a high level of perspicuity (DP13). A high perspicuity allows other users to understand the composition and rationale of a DQ score.

Step 5. The fifth step delves into an organization's system and process landscape. The success of a DQ tool relies on its capability to integrate with established tools and processes as an integration avoids the isolation of DQ work. To approach these two aspects, step 5 comprises two DPs. First, an organization should evaluate the need to connect a DQ tool with established business processes (DP5). Business process management and re-design initiatives are suitable for solving this task [87, 38]. The integration in different business processes calls for offering a convenient UI that supports and integrates with established user workflows (DP11). Otherwise, the DQ tool creates new barriers in the process execution, and user acceptance might be impaired. Second, the DQ tool must allow technical integration with existing tools and databases by applying standardized technologies and integration frameworks (DP7) [12].

Step 6. In the sixth and final step, the organization should review the homogeneity of the intended DQ tools user base. DQ tools are typically designed for a small team of highly-skilled employees having deep technical knowledge [10, 116]. This knowledge is often absent in other parts of the organization, such as at the data source or in external data domains [38, 48]. A DQ tool must provide DQ results that are comprehensible and easy to interpret by various users to tackle this issue (DP10).

5. Evaluation

The evaluation represents an essential component of rigorous DSR initiatives [114]. Given the goal of design science to address an "unsolved and important business problem" [64, p.84], DSR results must be impactful and relevant for practice. The practical relevance ensures that DSR results can inform the design of real-world solutions and help organizations overcome business problems [95]. To evaluate the accumulated design knowledge presented in this thesis, we followed the example of Janiesch et al. [70] and conducted a two-tiered evaluation by combining artificial and naturalistic evaluation strategies [114]. Our evaluation approach consists of (1) a scenario-based demonstration using a representative setting and (2) a group discussion on the reusability of the DPs. The summative evaluation of our overall results extends the individually evaluated case studies and helps us ensure our accumulated DPs are complete, valid, and impactful. The two-tiered evaluation process furthermore secures that practitioners and DQ system designers can reuse our DPs to create new DQ tools in their contexts [68].

The **scenario-based demonstration** shows that our results can support the creation of new DQ artifacts and helps us identify potential improvements [114]. We selected a descriptive evaluation technique as it is well-suited for demonstrating the applicability of our DPs in a representative case. The **group discussion on reusability** offers insights into the practical relevance and value of our DPs for DQ system designers. For this evaluation step, we relied on the framework by Iivari et al. [68], who suggest the following five criteria for assessing the reusability of DPs: *(1) accessibility, (2) importance, (3) novelty and insightfulness, (4) actability and guidance*, and *(5) effectiveness.*

5.1. Scenario-Based Demonstration

Since the design knowledge we present in this thesis aims to address new and emerging DQ requirements, we decided to base the demonstration on an artificial scenario instead of an implemented prototype [114]. Although the described scenario is artificial, we informed it with insights gathered from recurring real-world cases and throughout our DQ research. As a result, the scenario is typical for organizations experiencing problems with their DQ work and represents their line of thought.

For the evaluation scenario, we investigate the exemplary case of a company operating an electrical network infrastructure called ECo. ECo controls the local power grid system and ensures that the power supply is distributed evenly across the network. To work efficiently, ECo relies on accurate relational master data containing information about customers, suppliers, and devices, among others. Based on the historical build-up of the company, ECo has several distributed and heterogeneous master data sets in place that complicate efficient data management. Table 5.1 shows an exemplary material data set for the case of ECo. Additionally, ECo operates numerous sensors that help maintain the safety and security of the network. The sensors provide numerical values in different time intervals. To benefit from their data sets and promote data-based business models and innovation, ECo takes part in an IDSA-based energy data space and shares master data on the operated energy grid with a few external companies. In the future, ECo wants to promote data sharing and exchange real-time data streams with a larger and more diverse set of partnering companies.

Table 5.1.: Exemplary, artificial data of the ECo material master data set.

ID	**Description**	**Type**	**Size**	**Weight**	**Location**	**Unit**
0001	Container 18x20	1122 Other	20	1kg	W1	Each
0002	Thermostat (NEW)	5633 Safety			W1	Each
0003	Flow sensor (ctrl)	4561 CtrlSys	5	10g	W1	Each
0004	10 CABLE 40MM	1234 Assembly	40	500g	W2	PCS
0005	Heater M12	1122 Other		10kg	W1	Each
0006	Screw 100pcs	1234 Assembly	5	1.5kg	W2	PCS

ECo operates a centralized data management suite that supports general data management and governance tasks but only covers a fraction of ECo's data sets. Whenever data scientists or engineers use data sets, they perform manual quality checks, resulting in a reactive approach to DQ. ECo noticed this approach is time-consuming, expensive, and error-prone. To overcome these downsides, ECo envisions a universal DQ tool that supports different employees (e.g., data owners, scientists, or DQ experts) in ensuring a high level of DQ. The intended DQ tool must address the heterogeneous data landscape at ECo and allow different teams to model their DQ requirements while following global guidelines. Consequently, ECo aims to distribute the tool using an internal application marketplace. Despite being a fictitious company, we argue that ECo is a typical case of an organization afflicted by insufficient DQ tooling and, thus, well-suited for the scenario-based demonstration of our results.

To create a high-level DQ tool for the described scenario, we applied the action guideline described in section 4.3. Following the action guideline helps us consider all aspects relevant to DQ tools and structure the design process. Moreover, it allows us to evaluate the action guideline by reflecting on its usefulness and accessibility. In the following, we describe each step of the action guideline by applying it to ECo's scenario. We highlight necessary architectural components and the corresponding DPs in italics. The architectural components represent an exemplary realization of the DPs.

Design Process

Step 1. *Our data sets are too large to review manually.* In ECo's case, we can differentiate two types of data sets: relational master data and sensor data streams. Although a manual DQ review of master data sets is possible, this process would be cumbersome and lead to significant overhead for the data management team. Given the high velocity and number of sensors, a manual review is impossible for the sensor data streams. We look closer at the implications of sensor data streams for DQ tools in step 3. To quality check the master data sets, ECo should implement algorithms that automatically derive contextual and content-based DQ rules for the data sets in place (*DQ Rule Generation, DP1*) (e.g., Apriori algorithm, Cardinalities). The algorithms should generate rules based on the form and content of the data and allow for configuration to different settings. For instance, a rule could specify that materials of type '1234 Assembly' must feature location 'W2' and unit 'PCS'. Moreover, ECo's employees should be able to define custom rules and import/export DQ rules for reuse and collaborative work (*DQ Rule Definition, DQ Rule Repository, DP1*) [74]. In addition to reusing DQ rules, ECo must ensure the transparency of the algorithms used to help employees understand the results and customize the algorithms. ECo can realize this requirement by offering hints and textual explanations of the algorithms, including exemplary results (*Explanation, DP12*).

Step 2. *We have many different data sets and quality definitions.* As a company with a heterogeneous data landscape, ECo uses various data sets, and many stakeholders are involved in data management activities. As a result, ECo must ensure that the applied DQ algorithms are not limited to specific data sets and can address different requirements. For instance, since ECo aims to quality-check master data sets, the algorithms should work on material and customer data, which vary in terms of dimensionality, volume, and data types. Even within single data sets, there might be obstacles, such as missing data or different value categories (e.g., different weights in Table 5.1). The DQ algorithms must be able to comprehend such typical real-world data problems [8]. Furthermore, the algorithms should be able to identify quality problems in multiple DQ dimensions as ECo aims to obtain a holistic view of their data sets. Consequently, the algorithmic stack must feature a variety of algorithms to cover the DQ dimensions relevant to ECo, while being able to work with heterogeneous data sets (*DQ Measurement, DP2 and DP3*). Possible algorithms include SVMs, Outlier Analysis, or the Apriori algorithm [4]. However, operating a variety of DQ algorithms can lead to unclear and vague definitions, thus requiring the establishment of shared and standardized DQ definition *DQ Definitions, DP9.*

Step 3. *We use high-velocity data streams.* Managing numerous data streams, ECo has to consider two aspects in the design of its DQ tool. First, integrating sensor data streams requires using suitable interfaces to allow for a DQ analysis of the data streams (*Real-Time API, DP4*). Implementing message bus and broker technologies (e.g., Apache Kafka, Message Queuing Telemetry Transport (MQTT)) are suitable for addressing this requirement. Second, ECo must extend the algorithmic stack with algorithms that can handle data streams. These algorithms should be highly scalable to identify quality issues and enable the continuous validation of data streams (*Real-Time Validation, DP4*). For instance, to assess the accuracy of their sensor data streams, ECo could conduct a concept drift analysis [5]. Continuous data monitoring helps detect quality issues early and avoid potential service interruptions, such as power outages.

Step 4. *We share data outside our business domain.* Since ECo shares data in an IDSA-based data space, they must ensure that DQ information is accessible and understandable for all potential data consumers, who might be unknown at the time of designing the tool. Consequently, the intended DQ tool should offer compatibility by adhering to ecosystem-wide standards and guidelines. Most importantly, the DQ tool requires integration with a data space connector, representing the gateway to the data ecosystem. To fulfill this requirement, ECo must review and implement suitable interfaces and metadata models that effectively communicate DQ results (*Connector Interface, DP8; Shared DQ Metadata Model, DP6*). For the IDSA-based energy data space ECo participates in, this implies that ECo must follow the information models, data space protocols, and data app definitions offered by the IDSA [91, 67]. Additionally, ECo should clarify identified DQ issues in a way that is understandable to data consumers who may be unfamiliar with the data domain, so that they can take appropriate actions. To realize this requirement, ECo can extend the algorithmic explanations (cf. Step 1) and provide details on identified DQ issues (*Explanations, DP13*).

Step 5. *We have established processes and tools relevant to DQ.* In addition to the inter-organizational perspective offered by the data space, ECo must consider internal data consumers and DQ-relevant business processes. In ECo's case, this requires integration with the established data management suite and implementing suitable interfaces (*REST API, DP7*). Storing DQ information in multiple systems (i.e., data space connector and data management suite) calls for harmonizing the different metadata models. The harmonization allows making the same DQ information available to internal and external data consumers (*Shared DQ Metadata Model, DP5*). Integrating a DQ tool with multiple business processes necessitates a simple and seamless UI accessible to people of various backgrounds. For instance, correcting data errors in the exemplary material master (see Table 5.1) requires knowledge of the material domain and is easiest for material experts. ECo can approach this requirement by implementing low-code UIs that enable the specification of enterprise-wide DQ workflows, for example, by leveraging data-flow programming frameworks such as Node-RED (*DQ Workflows, DP11*) [74].

Step 6. *Our DQ tool has users from various backgrounds.* At ECo, heterogeneous teams, including members from various organizational units, are responsible for data management and analysis tasks. This diverse user group has varying interpretations of DQ metrics and measurements. Consequently, ECo must develop adequate DQ metrics comprehensible for the users of the DQ tool (*DQ Scoring, DP10*). In its simplest form, this can be a normalized DQ score on a scale of 0 to 1, which abstracts the measurements of multiple DQ dimensions. For this purpose, ECo must develop weightings for the individual DQ metrics and determine how these contribute to an overall score. A detailed DQ report can summarize different DQ results and offer potential solutions and hints for resolving quality problems (*Reporting, DP10*).

Reflection

The high-level software architecture displayed in Figure 5.1 summarizes the identified architectural components necessary for realizing the described DQ tool. We based the architecture on the reference architecture for DQ tools proposed in paper IX [7]. Evaluating our accumulated design knowledge and the proposed action guideline with a scenario-

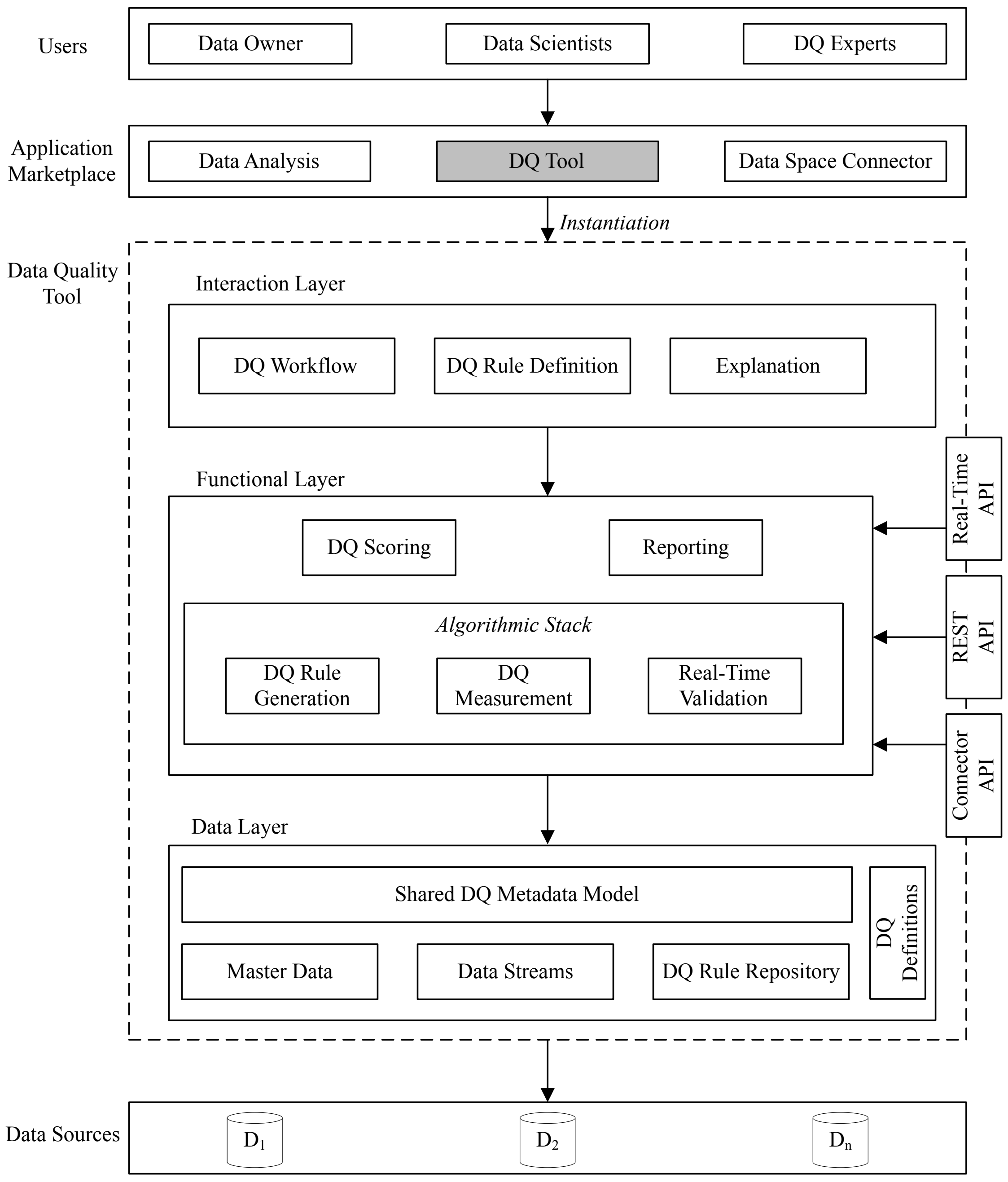


Figure 5.1.: High-level DQ tool architecture based on the requirements by ECo (adapted from paper IX [7]).

based demonstration revealed the benefits and shortcomings of our results, which should be considered in future research.

We identified the following *benefits*. First, the proposed DPs are complete and suitable for designing a full-featured DQ tool, as shown in Figure 5.1. The DPs offer insights into the different architectural layers of software applications and help address the meta-requirements of DQ tools, thus enabling the creation of successful tools. Second, the action guideline offers a structured process for designing DQ tools. The staggered approach helps consider the relevant aspects of DQ tools and can support the tool designer in developing a suitable solution. Third, the DPs are well-balanced as they offer flexibility for application to new use cases (e.g., the scenario of ECo) but provide concrete hints for potential implementation options.

In addition to the described benefits, we revealed the following *shortcomings*. First, we observed that the action guideline has multiple dependencies, and different steps can address the same architectural components. For example, steps 4 and 5 call for establishing a metadata model. In response, it might benefit an organization to consider the DPs individually or in a different order. We extended the description of the action guideline in section 4.3 accordingly. Second, in ECo's case, integration with various systems and data sources posed a crucial requirement. Our design knowledge addresses the technical integration in DP7 (Interfaces). However, the DP focuses on documentation and might not offer enough detail to design a technically viable solution. Future research should investigate the aspect of technical integration more closely. Third, organizations like ECo are confronted with DQ work in several scenarios. The described DQ tool summarizes our response to the requirements raised by ECo. It can be beneficial to design multiple DQ tools specializing in certain DQ aspects (e.g., DQ scoring or validating data streams) and apply these on different architectural levels (e.g., centralized or at the Edge). We aim to conduct further research on distributing DQ tools and varying their design based on specific use cases as part of future work.

5.2. Group Discussion on Reusability

The practical relevance of design knowledge generated in DSR studies is vital for their success [114]. Since practitioners are usually responsible for designing and implementing the intended artifacts, securing the reusability of DPs is crucial [68]. To ensure the reusability of our DPs, we conducted a focus group discussion with five practitioners working in different roles of DQ management. To assess the perceived reusability, we relied on the light reusability framework by Iivari et al. [68]. We selected the framework as it offers a lightweight approach to getting practitioner feedback and can offer first insights before performing more in-depth evaluations, such as prototypical implementations [114]. Moreover, we conducted a focus group discussion as they are generally well-suited for quickly gathering information while mirroring the social interactions and conversations of the participants [76, 65]. As a result, focus groups are high in external validity and can offer in-depth insights into complex phenomena [65].

Table 5.2 provides an overview of the participants who took part in the group discussion. The participants sufficiently represent the target community of our DPs (i.e., DQ system designers) and can offer valuable feedback on the usefulness of our DPs. The group discussion took place as part of an on-site workshop in April 2023 and lasted three hours. During the discussion, we explained each DP described in section 4.2 and provided an exemplary instantiation from one of our cases. Afterward, we put the DP up for discussion and used the template questionnaire offered by Iivari et al. [68] as a basis for structuring the conversation. In the following, we present the summative results for each reusability criterion.

Table 5.2.: Participants of Group Discussion.

ID	**Role**	**Industry**	**Experience**
A	Senior Consultant	IT Consultancy	> 10 years
B	Data Management	Public Sector	5 years
C	System Architect	Public Sector	> 10 years
D	Junior Consultant	IT Consultancy	< 2 years
E	Data Engineer	Manufacturing	8 years

Accessibility. The accessibility criterion relates to whether the target community can understand and comprehend the DPs individually and collectively [68]. Some participants received the proposed DPs as ”vague” or ”too abstract”. One participant highlighted DP2 (*Provide the DQ tool with robust algorithms for users to conduct DQ validations on data sets varying in size and format, given that different data sets are in use*) as a concrete example, asking what ’robust algorithms’ are. A wording that is not specific risks that the DP might remain unnoticed by the target community. We addressed this issue by offering concrete examples and highlighting potential implementation options and organizational implications for each DP. For the broad field of general-purpose DQ tools, it is challenging to formulate more specific DPs while still addressing the general class of problem. Future research could develop more specific DPs for particular industries or application scenarios, using our set of DPs as a basis.

Importance. A DP is considered important when the real-world business problem it addresses is essential and acts as a starting point for a possible solution [68, 102]. Considering that we based our DPs on real-world case studies, we can assume they address business problems important to the partnering companies. In the focus group discussion, we aimed to assess whether the participants perceived the accumulated set of DPs as important. We received mostly positive feedback, and the participants considered all DPs necessary for creating successful DQ tools. Nevertheless, there was some controversy about the relative importance of certain DPs when compared to each other. Some participants (B and E) weighted the DPs related to automation (DPs 1-4) as more significant than the remaining ones. A different participant (D) stated that ”DQ is mostly a standardization problem” and valued the corresponding DPs as more important.

Novelty & Insightfulness. Novelty and insightfulness assess whether the DPs can convey new ideas and insights to the practitioners’ daily practice [68]. Asking the participants if the proposed design knowledge sheds new light on handling DQ issues, we received mixed feedback. Most participants agreed that, individually, the DPs are of limited novelty and instead represent a current state of the art. However, all participants confirmed that, in combination, the DPs offer novel insights into the field of DQ tools. The combination of functionalities and capabilities from various scientific disciplines (e.g., human-computer interaction and data management) is unique and offers an interdisciplinary approach to creating DQ tools. This approach is new to a field that has been predominantly technical or organizational and tackled by multiple teams simultaneously. Future research should deepen the interdisciplinary aspect of DQ tools and develop new and innovative approaches.

Actability & Guidance. The actability and guidance criterion specifies whether a practitioner can effectively and realistically implement a DP in practice [68]. All participants noted that the DPs often offer limited guidance for realizing a DP in real-world scenarios, and further work and creativity are necessary to make a DP "implementation ready". For example, DP1 (Efficiency) specifies the use of 'intelligent algorithms' without defining what these are. This lack of concrete guidance results from the broad application field for DQ tools, and more specific DPs would be too restrictive. We addressed the balance between generalizability to the high class of DQ tools and concrete guidelines by providing implementation examples for each DP. Future work could improve the actability by extending the abstract DPs into more specific design patterns, offering concrete implementation guidelines for certain aspects of DQ tools (e.g., real-time data validation). In contrast to the individual DPs, the participants agreed that the accumulated design knowledge offers guidance for designing DQ tools in general. Moreover, the participants noted that the proposed design knowledge is actable as it helps consider essential functional aspects during the design process. For instance, participant A stated, "the DP overview is well-suited as a checklist for the functionalities a DQ tool should have".

Effectiveness. The effectiveness relates to the effects or consequences of reusing a set of DPs in practice. A rigorous assessment of the efficacy requires naturalistic approaches and measuring the performance of a tool over a longer period [68, 114]. Tallied with the approach of Iivari et al. [68], we rather asked the participants to estimate if our DPs could lead to a more successful DQ tool compared to their current DQ tooling. All participants agreed that the proposed DPs address relevant meta-requirements and their DQ tools in place could not adequately handle trends such as distributed data landscapes or big data. In this regard, all participants believed the DPs could contribute to creating more effective DQ tools. In particular, the participants liked the idea of our action guideline, as it helps organizations come up with the DQ tool most effective in their respective contexts.

6. Conclusion

Operating on high-quality data is vital for organizational success [100]. Among others, it is a prerequisite for automated decision-making, enables seamless business operations, and helps build competitive advantages [100, 35, 36]. DQ tools play an essential role in supporting the DQ work in light of ever-growing amounts of data [109]. However, the organizational (e.g., locally managed data products [38]) and technical (e.g., the rise of big data [109, 24]) environment changes, posing new requirements on DQ tools. In this thesis, we addressed the lack of prescriptive design knowledge for creating successful DQ tools by outlining the problems and meta-requirements DQ tools face (*problem side*) and formulating DPs to address these requirements (*solution side*). To populate the solution side, we accumulated the design knowledge generated in four case studies concerned with implementing DQ tools in real-world contexts.

To conclude this thesis, we cover three aspects. First, we answer the initially proposed research questions and point out limitations in our results. Second, we summarize our contributions to the relevant managerial and scientific communities. Finally, we present an outlook on how our research on DQ tools can continue and lay out potential paths for future work.

6.1. Answers to Research Questions

Using the presented research results, we can now conclude this thesis by answering the research questions raised in section 1.3. For both questions, we will recap the research approach, summarize our results, and highlight their limitations. Figure 6.1 summarizes our results for both research questions. The consolidated data structure comprises the DQ problems and meta-requirements on the problem side (RQ1) and the corresponding DPs on the solution side (RQ2).

RQ1: What are the objectives for successful data quality tools?

We answered this research question with three studies (papers I [12], V [113], and VI [10]). In combination, the three studies offered us an in-depth view of the downsides of current DQ approaches based on practical and scientific insights. We increased the robustness of our findings by applying different research methods to the same phenomenon [41]. Paper I [12] describes the DQ problems a single organization experiences in its daily business operations. We observed that the heterogeneity of system architectures and the lack of standards are the main obstacles to a higher level of DQ. For a higher-level view of DQ, we reviewed the literature on data engineering and derived DQ aspects requiring further research attention in paper V [113]. Following the trend for automated data

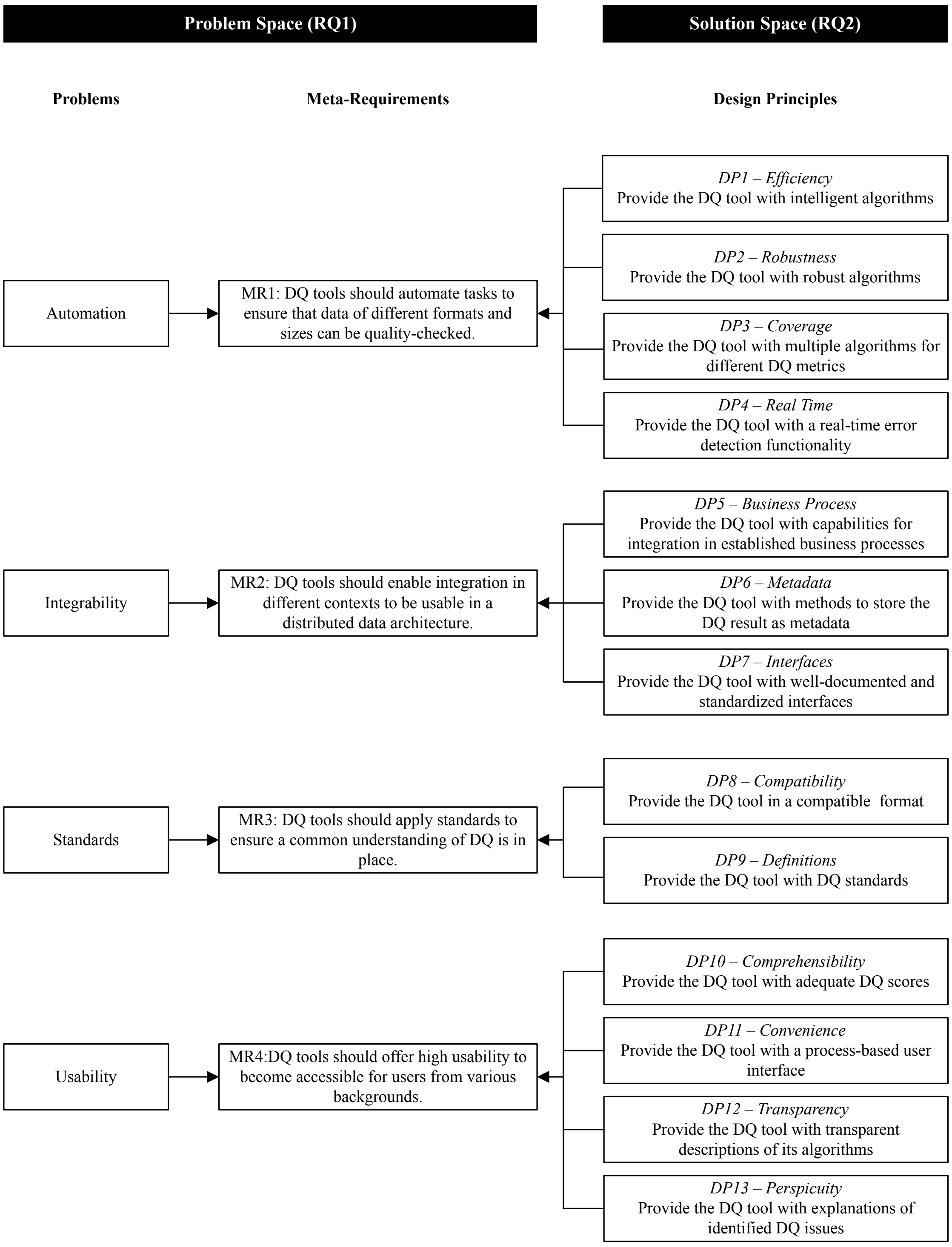

Problem Space (RQ1)
Solution Space (RQ2)
Problems
Meta-Requirements
Design Principles
Automation
MR1: DQ tools should automate tasks to ensure that data of different formats and sizes can be quality-checked.
DP1 – Efficiency Provide the DQ tool with intelligent algorithms
DP2 – Robustness Provide the DQ tool with robust algorithms
DP3 – Coverage Provide the DQ tool with multiple algorithms for different DQ metrics
DP4 – Real Time Provide the DQ tool with a real-time error detection functionality
Integrability
MR2: DQ tools should enable integration in different contexts to be usable in a distributed data architecture.
DP5 – Business Process Provide the DQ tool with capabilities for integration in established business processes
DP6 – Metadata Provide the DQ tool with methods to store the DQ result as metadata
DP7 – Interfaces Provide the DQ tool with well-documented and standardized interfaces
Standards
MR3: DQ tools should apply standards to ensure a common understanding of DQ is in place.
DP8 – Compatibility Provide the DQ tool in a compatible format
DP9 – Definitions Provide the DQ tool with DQ standards
Usability
MR4:DQ tools should offer high usability to become accessible for users from various backgrounds.
DP10 – Comprehensibility Provide the DQ tool with adequate DQ scores
DP11 – Convenience Provide the DQ tool with a process-based user interface
DP12 – Transparency Provide the DQ tool with transparent descriptions of its algorithms
DP13 – Perspicuity Provide the DQ tool with explanations of identified DQ issues

Figure 6.1.: Accumulated design knowledge for DQ tools.

engineering, we identified a need for improved automation in DQ processes. In paper VI [10], we developed a functional taxonomy of DQ tools and applied it to commercial DQ tools. This study revealed that DQ tools increasingly implement integrative functionalities, facilitating collaborative DQ work.

We noted that these developments are symptoms of a technically and organizationally changing environment, resulting in the need for a new kind of DQ tool (cf. section 2.2). Established DQ tools fail to address new requirements, which leads to DQ initiatives that fall short of their promises in industrial practice. In summary, we uncovered that DQ tools are confronted by four main problems *Automation*, *Integrability*, *Standards*, and *Usability*, and formulated corresponding meta-requirements (cf. section 4.1).

Despite conducting different studies, our answer to RQ1 is limited, pertaining to the number of data sources and the potential subjectivity during data analysis. DQ is an inherently complex and subjective matter that users define differently. This makes establishing a shared understanding of DQ difficult and threatens our construct validity. Moreover, the analysis of DQ problems is likely incomplete and does not offer an exhaustive overview. For example, the functional DQ taxonomy presented in paper VI [10] focuses on commercial DQ tools and neglects other tools (e.g., open-source tools). Nevertheless, our answer to RQ1 indicates the main problems of DQ tools, which we also confirmed in our cases. However, further studies will be beneficial for generating additional insights and disclosing more meta-requirements.

RQ2: How to design successful data quality tools?

The development of prescriptive design knowledge for successful DQ tools is the main objective of this dissertation and allows us to achieve the proposed research goal. Our answer to the second research question consists of two elements. First, we presented an accumulated set of 13 DPs in section 4.2. The DPs are 'multi-grounded' [50] as they emerged from four case studies concerned with implementing DQ tools in real-world contexts (cf. section 3.2) and address the previously identified meta-requirements. We aggregated the generated design knowledge by performing an iterative cross-case analysis using the design knowledge from our case studies as the primary data source. By theorizing on the individual case designs, we were able to accumulate and abstract the available design knowledge onto a broader class of problem (i.e., the design of DQ tools) [17]. After we conducted four case studies, we addressed all meta-requirements and reached theoretical saturation where additional learning would have been minimal [41, 26]. Second, to promote the accessibility and application of our findings in industrial practice, we described each DP with a concrete usage example and organizational implications. We also developed an action guideline for practitioners to identify relevant DPs (cf. section 4.3) and proposed a DQ reference architecture in paper IX [7].

However, although we followed a rigorous research approach, our design knowledge has several limitations based on the empirical nature of our case studies. These pose a threat to the external validity of our results [121, 103]. Most importantly, the implemented DQ tools operate on single organizational processes and contexts, which may have introduced a bias in the design knowledge. DQ tools in other application areas (e.g., different processes or data sets) might require different designs. Additionally, we conducted our case studies in large, traditional organizations in the manufacturing and pharmaceutical industries. The solution design might vary in other types of organizations, such as startups. We tried to avoid this organizational bias by triangulating our findings [103]. Specifically, we realized each meta-requirement in multiple cases and only included DPs in our result that have multiple sources. Similar to RQ1, there is a threat to the reliability of our results grounded in the subjectivity during data analysis [103]. Other researchers might deduce different DPs and reach different conclusions. To address this threat, we involved people from various backgrounds in evaluating the DQ tool design in each case study. Moreover, we combined artificial and naturalistic strategies for evaluating the accumulated design knowledge, which helped us secure the practical relevance and reusability of our DPs. Finally, the design knowledge for DQ tools will likely change as new requirements emerge, raising the need to review and update our results continuously.

6.2. Summary of Contributions

As an empirical study that investigates a socio-technical phenomenon, this thesis offers both *managerial* and *scientific* contributions [95]. These can help practitioners design impactful solutions and advance the current body of literature on DQ tools [73]. As depicted in Figure 3.1, our research offers five main contributions to the field of DQ and one methodological contribution:

Data Quality

- Meta-Requirements (cf. Section 4.1 and papers I [12], V [113], VI [10])
- Design Principles (cf. Sections 4.2 and 4.3, and papers III [4], IV [8], VII [11], and VIII [6])
- Prototypes (cf. papers III [4], IV [8], VII [11], II [5], and VIII [6])
- Evaluation Results (cf. papers III [4], IV [8], VII [11], II [5], and VIII [6])
- Software Reference Architecture (cf. Section 5.1 and paper IX [7])

Methodological

- Conceptual approach and visualization of the design knowledge accumulation process (cf. Section 3.3)

In the following, we will discuss these contributions from a managerial and a scientific perspective and highlight potential implications.

Managerial Contributions

DQ system designers can use the proposed design knowledge to inform the functional and non-functional composition of DQ tools and create successful solutions. The design knowledge comprises 13 DPs that contribute to lowering the high uncertainty designers of interdisciplinary information systems usually face [107]. We generalized and codified the DPs, which means that users from other domains can use our findings to inform the DQ tool design and ease the instantiation of artifacts in their contexts [85].

Our design knowledge is grounded in four industrial case studies, which we described in four research papers. Each paper outlines our course of action and the interplay between individuals, organizations, and technology [108]. Practitioners can use the insights into our development process, including the system design choices and considerations, as a guiding example for translating design knowledge into impactful software solutions [73].

Besides their interdisciplinary nature, the individuality and subjectivity of DQ further increase the complexity of building DQ tools [118, 10]. There are no one-size-fits-all solutions, and every organization must create a tool that fits its requirements. To support organizations in this process, we offer three contributions that help practitioners design, create, and find the right DQ solution. First, we propose an action guideline, which acts as a simple entry point to our research results, supports organizations in identifying relevant DPs, and illustrates potential implications and relations among the DPs. Second, we contribute an initial reference architecture for creating state-of-the-art DQ tools in paper IX [7]. DQ system designers can use the reference architecture to inform the architectural design and technical capabilities of their individual DQ tool development projects. Decision-makers can leverage the reference architecture to assess available solutions and support their customization. Third, to inform make-or-buy decisions, we offer a comprehensive overview of commercial DQ tools available in paper VI [10]. In combination with our design knowledge, the overview can help assess the applicability of DQ tools available on the market.

DQ is, without question, an important issue for organizations, and the ones that succeed in DQ can benefit in several ways. We hope that the practice-oriented and hands-on presentation of our research results can lower the barriers to our work for IT managers and help them build successful DQ tools, raise awareness for DQ, and spark new discussions.

Scientific Contributions

Despite the importance of DQ for organizations and a long history of research, there is a lack of prescriptive design knowledge to build DQ tools. Several studies identified this need and formulated research calls for advancing DQ and related solutions (e.g., [109, 62, 24, 48]). Most importantly, this thesis contributes to the scientific knowledge base on two DSR abstraction levels, as defined by Gregor & Hevner [54].

On a semi-abstract level, we contribute codified prescriptive DPs on designing and implementing DQ tools. The DPs address the meta-requirements of DQ tools, thus adding knowledge on designing DQ tools in previously unexplored areas. We derived the DPs from four case studies and generalized our findings by abstracting our design decisions to the class of problem [95]. Hence, the proposed design knowledge is not only relevant for the case studies but serves as a nascent design theory for building DQ tools in general and contributes to answering the aforementioned research calls [54].

On a more specific or 'street level', [78] we contribute the descriptions of situated DSR artifacts [54]. In each case study, we outline the instantiation of a DSR artifact as a software prototype, which is considered a knowledge contribution in itself [54]. Implementing these artifacts in concrete socio-technical contexts helped us derive theoretically grounded and practically relevant design knowledge [95, 107]. Researchers from the DQ and adjacent research communities can use our findings and extend these in various directions. For example, we mentioned the need for more convenient DQ solutions that are easily usable and follow a standardized procedure. Researchers from the human-computer interaction domain could follow up on this aspect and develop novel interfaces or investigate gamification approaches for DQ improvement.

Methodologically, we took inspiration from the process of design knowledge accumulation described by Avdiji et al. [17]. Generally, the scientific community lacks guidance for theorizing on multiple DSR projects that follow similar goals and define a less situated class of problem. Most DSR-related articles instead focus on describing DSR processes (e.g., [95, 64]) or abstracting the findings of single DSR projects (e.g., [80]). To overcome these limitations, Avdiji et al. [17] developed a process for conducting an iterative, retrospective analysis of the design requirements and knowledge generated in multiple case studies. We contribute to the scientific community by offering a conceptual visualization of the process for design knowledge accumulation and describing its application in the cross-case analysis of our cases. Against this background, we hope our work promotes conducting and accumulating the findings of multiple case studies using cross-case data analysis techniques to create more profound and practically relevant results [107].

While the contribution of prescriptive design knowledge is at the heart of this dissertation, we add further knowledge to the DQ community. First, we empirically investigated the problems of industrial data management and identified challenges for data collaborations that other researchers could pursue (paper I [12]). Second, we contributed a new algorithm for detecting concept drift in sensor data that maintains high accuracy and minimizes resource usage (paper II [5]). Third, we propose a data engineering reference model, which points out areas in data engineering that have received limited research attention (paper V [113]). Fourth, we offer a taxonomy for DQ tools other scholars can use to analyze and classify DQ tools systematically (paper VI [10]).

6.3. Outlook

Although the concept of DQ and its research are long established, they are receiving more attention grounded in the rise of automated decision-making and new regulatory requirements. These developments imply a growing need for adequate DQ tooling. The design knowledge offered in this thesis can only be a step towards modern DQ tooling, and further research in multiple directions is required. Most importantly, future work should aim to acquire data from further case studies to obtain more profound design knowledge and overcome limitations based on the organizational settings. Applying our design knowledge in additional industries and different types of organizations (e.g., small enterprises) could help assess the validity of our findings and lead to further learning.

In addition to gathering insights from more domains, integrating DQ tools in different system architectures and observing changes in the DQ tool design is another promising research avenue. Depending on the use case, DQ tools will operate at various architectural levels, ranging from operation at the machine level to DQ tools monitoring data lakes. Each architectural layer poses different functional and non-functional requirements on DQ tools, affecting the corresponding design knowledge. We plan to extend the proposed design knowledge with another conceptual layer representing the varying architectural levels and offer more concrete DP descriptions. The enhanced design knowledge supports organizations in identifying the DQ tool that best suits their use case and provides better accessibility for the different target communities.

We experienced the need for less abstract and more concrete guidance during the evaluation multiple times. Finding the right balance between generalized design knowledge applicable to a class of problem and effective and actable guidelines can be difficult. In this regard, future research should aim to extend the design principles into more concrete design patterns, outlining specific implementation details. For developing these design patterns, it might be necessary to divide the current class of problem (i.e., DQ tools) into more specific sub-classes, like real-time data validation tools. Investigating particular functional aspects of DQ tools opens a way to create specific guidelines and would be a valuable extension to the general design knowledge offered in this thesis.

Future research should also look closely at data ecosystems as a new architectural paradigm. Data ecosystems can offer a fresh start for DQ tools, which often reside as insular solutions in a heterogeneous system landscape [56, 48]. By providing standardized and interoperable solutions, DQ tools can become accessible for many more data providers and consumers and much less of a specialized task [11, 6]. The induced shift of responsibility from a centralized team to the individual data owner will simplify data cleaning and is a promising research field. Another crucial aspect in this regard is the development of universally applicable and standardized DQ scores. These scores must work for all kinds of data sets and be easy to understand for data consumers unfamiliar with the data domain. We made a first attempt in this direction in a recent paper [57].

Apart from the organizational perspective, it would be interesting to investigate what requirements non-profit institutions such as governments or non-governmental organizations pose on DQ tools and advance the DPs in this direction. For instance, to make informed decisions, policymakers rely on high-quality data. Research on the peculiarities of political decision-making can help design and implement tailored DQ solutions. Researchers should also pay attention to regulations' influence on the DQ tool design. The European Union currently works on new laws (e.g., the European AI Act [42]) targeting data-intensive applications. These new regulations will lead to additional DQ requirements for organizations that offer a plethora of research opportunities.

Taking on a societal perspective, DQ is among its grand challenges. Problems of utmost importance, like the spread of misinformation or biased AI algorithms, can be approached with DQ. Researchers have many opportunities to create new interdisciplinary DQ solutions and ensure sustainability, inclusivity, and transparency in data-intensive applications [71].

Bibliography

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. “Profiling relational data: a survey”. In: *The VLDB Journal* 24.4 (2015), pp. 557–581. DOI: 10.1007/s00778-015-0389-y.

[2] Ziawasch Abedjan et al. “Detecting data errors: Where are we and what needs to be done?” In: *Proceedings of the VLDB Endowment* 9.12 (2016), pp. 993–1004. DOI: 10.14778/2994509.2994518.

[3] Mohammed Alhamadi et al. “Data Quality, Mismatched Expectations, and Moving Requirements: The Challenges of User-Centred Dashboard Design”. In: *Nordic Human-Computer Interaction Conference*. NordiCHI ’22. Aarhus, Denmark: Association for Computing Machinery, 2022. DOI: 10.1145/3546155.3546708.

[4] Marcel Altendeitering. “Mining Data Quality Rules for Data Migrations: A Case Study on Material Master Data”. In: *Leveraging Applications of Formal Methods, Verification and Validation*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer International Publishing, 2021, pp. 178–191. DOI: 10.1007/978-3-030-89159-6_12.

[5] Marcel Altendeitering and Stephan Dübler. “Scalable Detection of Concept Drift: A Learning Technique Based on Support Vector Machines”. In: *Procedia Manufacturing* 51 (2020). 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2021), pp. 400–407. DOI: 10.1016/j.promfg.2020.10.057.

[6] Marcel Altendeitering, Stephan Dübler, and Tobias Guggenberger. “Data Quality in Data Ecosystems: Towards a Design Theory”. In: *AMCIS 2022 Research Papers* (2022).

[7] Marcel Altendeitering and Tobias Moritz Guggenberger. “Data Quality Tools: Towards a Software Reference Architecture”. In: *Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS)*. 2024.

[8] Marcel Altendeitering and Tobias Moritz Guggenberger. “Designing Data Quality Tools: Findings from an Action Design Research Project at Boehringer Ingelheim”. In: *ECIS 2021 Research Papers* (2021).

[9] Marcel Altendeitering, Tobias Moritz Guggenberger, and Frederik Möller. “A Design Theory for Data Quality Tools in Data Ecosystems : Findings from three industry cases”. In: *unpublished* ().

[10] Marcel Altendeitering and Martin Tomczyk. “A Functional Taxonomy of Data Quality Tools: Insights from Science and Practice”. In: *Wirtschaftsinformatik 2022 Proceedings* (2022).

[11] Marcel Altendeitering et al. “Data Sovereignty for AI Pipelines: Lessons Learned from an Industrial Project at Mondragon Corporation”. In: *2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN)*. 2022, pp. 193–204. DOI: 10.1145/3522664.3528593.

[12] Antonello Amadori, Marcel Altendeitering, and Boris Otto. “Challenges of Data Management in Industry 4.0: A Single Case Study of the Material Retrieval Process”. In: *Business Information Systems*. Ed. by Witold Abramowicz and Gary Klein. Cham: Springer International Publishing, 2020, pp. 379–390. DOI: 10.1007/978-3-030-53337-3_28.

[13] Saleema Amershi et al. “Guidelines for Human-AI Interaction”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–13. DOI: 10.1145/3290605.3300233.

[14] Samuil Angelov, Paul Grefen, and Danny Greefhorst. “A framework for analysis and design of software reference architectures”. In: *Information and Software Technology* 54.4 (2012), pp. 417–431. DOI: 10.1016/j.infsof.2011.11.009.

[15] Samuil Angelov, Jos JM Trienekens, and Paul Grefen. “Towards a method for the evaluation of reference architectures: Experiences from a case”. In: *Software Architecture: Second European Conference, ECSA*. Springer. 2008, pp. 225–240. DOI: 10.1007/978-3-540-88030-1_17.

[16] Apache Camel. *Apache Camel*. Accessed: 12.01.2023. 2022. URL: https://camel.apache.org/.

[17] Hazbi Avdiji et al. “A design theory for visual inquiry tools”. In: *Journal of the association for Information Systems* 21.3 (2020). DOI: 10.17705/1jais.00617.

[18] David E Avison et al. “Action research”. In: *Communications of the ACM* 42.1 (1999), pp. 94–97.

[19] Can Azkan et al. “Design Principles for Industrial Data-Driven Services”. In: *IEEE Transactions on Engineering Management* (2022), pp. 1–24. DOI: 10.1109/TEM.2022.3167737.

[20] Can Azkan et al. “Service dominant Logic Perspective on Data Ecosystems-a Case Study based Morphology.” In: *Proceedings of the 28th European Conference on Information Systems (ECIS)*. 2020.

[21] Donald P. Ballou et al. “Modeling information manufacturing systems to determine information product quality”. In: *Management Science* 44.4 (1998), pp. 462–484. DOI: 10.1287/mnsc.44.4.462.

[22] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115. DOI: 10.1016/j.inffus.2019.12.012.

[23] Christine Basta, Marta R. Costa-jussà, and Noe Casas. "Evaluating the Underlying Gender Bias in Contextualized Word Embeddings". In: *Proceedings of the First Workshop on Gender Bias in Natural Language Processing.* Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 33–39. DOI: 10.18653/v1/W19-3805.

[24] Carlo Batini et al. "From data quality to big data quality". In: *Journal of Database Management (JDM)* 26.1 (2015), pp. 60–82. DOI: 10.4018/JDM.2015010103.

[25] Carlo Batini et al. "Methodologies for Data Quality Assessment and Improvement". In: *ACM Comput. Surv.* 41.3 (July 2009). DOI: 10.1145/1541880.1541883. URL: 10.1145/1541880.1541883.

[26] Izak Benbasat, David K. Goldstein, and Melissa Mead. "The Case Research Strategy in Studies of Information Systems". In: *MIS Quarterly* 11.3 (1987), pp. 369–386. DOI: 10.2307/248684.

[27] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. "Engineering ai systems: A research agenda". In: *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems.* IGI global, 2021, pp. 1–19. DOI: 10.4018/978-1-7998-5101-1.ch001.

[28] Eric Brewer. "CAP twelve years later: How the "rules" have changed". In: *Computer* 45.2 (2012), pp. 23–29.

[29] Michael L. Brodie. "Data quality in information systems". In: *Information & Management* 3.6 (1980), pp. 245–258. DOI: 10.1016/0378-7206(80)90035-X.

[30] Mario Bunge. *Scientific research II: The search for truth.* Springer Science & Business Media, 2012.

[31] Joy Buolamwini and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency.* Ed. by Sorelle A. Friedler and Christo Wilson. Vol. 81. Proceedings of Machine Learning Research. PMLR, Feb. 2018, pp. 77–91.

[32] Bongsug Kevin Chae et al. "The impact of advanced analytics and data accuracy on operational performance: A contingent resource based theory (RBT) perspective". In: *Decision support systems* 59 (2014), pp. 119–126. DOI: 10.1016/j.dss.2013.10.012.

[33] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". In: *Commun. ACM* 13.6 (June 1970), pp. 377–387. DOI: 10.1145/362384.362685.

[34] Juliet M Corbin and Anselm Strauss. "Grounded theory research: Procedures, canons, and evaluative criteria". In: *Qualitative sociology* 13.1 (1990), pp. 3–21. DOI: 10.1007/BF00988593.

[35] Nadine Côrte-Real, Pedro Ruivo, and Tiago Oliveira. "Leveraging internet of things and big data analytics initiatives in European and American firms: Is data quality a way to extract business value?" In: *Information & Management* 57.1 (2020). Big data and business analytics: A research agenda for realizing business value. DOI: 10.1016/j.im.2019.01.003.

[36] Thomas Davenport and Jeanne Harris. *Competing on analytics: Updated, with a new introduction: The new science of winning*. Harvard Business Press, 2017.

[37] Alan Dayley et al. *Market Share: Data Quality Tools, Worldwide, 2019*. 2020. URL: https://www.gartner.com/en/documents/3984707/market-share-data-quality-tools-worldwide-2019.

[38] Zhamak Dehghani. *How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh*. Accessed: 10.12.2022. 2019. URL: https://martinfowler.com/articles/data-monolith-to-mesh.html.

[39] Dong Deng et al. "The Data Civilizer System". In: 8th Biennial Conference on Innovative Data Systems Research, CIDR 2017. Jan. 2017.

[40] Lisa Ehrlinger and Wolfram Wöß. "A survey of data quality measurement and monitoring tools". In: *Frontiers in Big Data* (2022). DOI: 10.3389/fdata.2022.850611.

[41] Kathleen M Eisenhardt. "Building theories from case study research". In: *Academy of management review* 14.4 (1989), pp. 532–550. DOI: 10.5465/amr.1989.4308385.

[42] European Union. *Proposal for an Artifical Intelligence Act*. Accessed: 19.12.2022. 2021. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%5C%3A52021PC0206.

[43] European Union Agency for Fundamental Rights. *Data quality and artificial intelligence – mitigating bias and error to protect fundamental rights*. Accessed: 19.12.2022. 2019. URL: https://fra.europa.eu/en/publication/2019/data-quality-and-artificial-intelligence-mitigating-bias-and-error-protect.

[44] Eric Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.

[45] Harald Foidl, Michael Felderer, and Rudolf Ramler. "Data Smells: Categories, Causes and Consequences, and Detection of Suspicious Data in AI-based Systems". In: *2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN)*. 2022, pp. 229–239. DOI: 10.1145/3522664.3528590.

[46] Christopher Fox, Anany Levitin, and Thomas Redman. "The notion of data and its quality dimensions". In: *Information Processing & Management* 30.1 (1994), pp. 9–19. DOI: 10.1016/0306-4573(94)90020-5.

[47] Mouzhi Ge and Markus Helfert. "A review of information quality research—develop a research agenda". In: *International Conference on Information Quality.* 2007, pp. 76–91.

[48] Sandra Geisler et al. "Knowledge-Driven Data Ecosystems Toward Data Transparency". In: *ACM Journal of Data and Information Quality (JDIQ)* 14.1 (2021), pp. 1–12. DOI: 10.1145/3467022.

[49] Paulo B. Goes. "Editor's Comments: Big Data and IS Research". In: *MIS Quarterly* 38.3 (2014), pp. iii–viii.

[50] Goran Goldkuhl. "Design theories in information systems-a need for multi-grounding". In: *Journal of Information Technology Theory and Application (JITTA)* 6.2 (2004), p. 7.

[51] Great Expectations. *Great Expectations.* Accessed: 10.01.2023. 2022. URL: https://greatexpectations.io/.

[52] Shirley Gregor. "Building Theory in the Sciences of the Artificial". In: DESRIST '09. Philadelphia, Pennsylvania: Association for Computing Machinery, 2009. DOI: 10.1145/1555619.1555625.

[53] Shirley Gregor, Leona Chandra Kruse, and Stefan Seidel. "Research Perspectives: The Anatomy of a Design Principle". In: *Journal of the Association for Information Systems* 21 (2020), pp. 1622–1652. DOI: 10.17705/1jais.00649.

[54] Shirley Gregor and Alan R. Hevner. "Positioning and Presenting Design Science Research for Maximum Impact". In: *MIS Quarterly* 37.2 (2013), pp. 337–355. (Visited on 10/14/2022).

[55] Shirley Gregor and David Jones. "The Anatomy of a Design Theory". In: *Journal of the Association of Information Systems* 8 (2007), pp. 312–335.

[56] Christoph Gröger. "There is no AI without data". In: *Communications of the ACM* 64.11 (2021), pp. 98–108. DOI: 10.1145/3448247.

[57] Tobias Moritz Guggenberger, Marcel Altendeitering, and Christoph Schlueter Langdon. "Design Principles for Quality Scoring - Coping with Information Asymmetry of Data Products". In: *Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS).* 2024.

[58] Tobias Moritz Guggenberger et al. "Ecosystem Types in Information Systems". In: *Proceedings of the 28th European Conference on Information Systems (ECIS).* 2020.

[59] Tobias Moritz Guggenberger et al. "Towards a Unifying Understanding of Digital Business Models". In: *Proceedings of the Twenty-Third Pacific Asia Conference on Information Systems.* 2020.

[60] Manjul Gupta and Joey F. George. "Toward the development of a big data analytics capability". In: *Information & Management* 53.8 (2016), pp. 1049–1064. DOI: 10.1016/j.im.2016.07.004.

[61] Inan Gür, Tobias Moritz Guggenberger, and Marcel Altendeitering. "Towards a Data Management Capability Model". In: *AMCIS 2021 Proceedings* (2021).

[62] Mazhar Hameed and Felix Naumann. "Data Preparation: A Survey of Commercial Tools". In: *SIGMOD Record* 49.3 (Dec. 2020), pp. 18–29. DOI: 10.1145/3444831.3444835.

[63] M.R Hasan and C. Legner. "Understanding Data Products: Motivations, Definition, and Categories". In: *ECIS 2023 Proceedings*. 2023.

[64] Alan R. Hevner et al. "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1 (2004), pp. 75–105. DOI: 10.2307/25148625.

[65] Jocelyn A Hollander. "The social contexts of focus groups". In: *Journal of contemporary ethnography* 33.5 (2004), pp. 602–637. DOI: 10.1177/0891241604266988.

[66] John van den Hoven. "Information Resource Management: Stewards of Data". In: *Information Systems Management* 16.1 (1999), pp. 88–90. DOI: 10.1201/1078/43187.16.1.19990101/31167.13.

[67] IDSA. *Reference Architecture Model 3.0.* 2019. URL: https://internationaldataspaces.org/wp-content/uploads/IDS-RAM-3.0-2019.pdf.

[68] Juhani Iivari, Magnus Rotvit Perlt Hansen, and Amir Haj-Bolouri. "A proposal for minimum reusability evaluation of design principles". In: *European Journal of Information Systems* 30.3 (2021), pp. 286–303. DOI: 10.1080/0960085X.2020.1793697.

[69] ISO. *ISO/IEC 25012.* Accessed: 10.01.2023. 2022. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25012.

[70] Christian Janiesch et al. "An Architecture Using Payment Channel Networks for Blockchain-Based Wi-Fi Sharing". In: 14.1 (2023). DOI: 10.1145/3529097.

[71] Eun Seo Jo and Timnit Gebru. "Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency.* FAT* '20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 306–316. DOI: 10.1145/3351095.3372829.

[72] B.A. Kitchenham, T. Dyba, and M. Jorgensen. "Evidence-based software engineering". In: *Proceedings. 26th International Conference on Software Engineering.* 2004, pp. 273–281. DOI: 10.1109/ICSE.2004.1317449.

[73] B.A. Kitchenham et al. "Preliminary guidelines for empirical research in software engineering". In: *IEEE Transactions on Software Engineering* 28.8 (2002), pp. 721–734. DOI: 10.1109/TSE.2002.1027796.

[74] Timon Klann., Marcel Altendeitering., and Falk Howar. "Towards a Low-Code Tool for Developing Data Quality Rules". In: *Proceedings of the 12th International Conference on Data Science, Technology and Applications - DATA*. INSTICC. SciTePress, 2023, pp. 22–29. DOI: 10.5220/0012050400003541.

[75] Barbara D Klein, Dale L Goodhue, and Gordon B Davis. "Can humans detect errors in data? Impact of base rates, incentives, and goals". In: *MIS Quarterly* (1997), pp. 169–194. DOI: 10.2307/249418.

[76] Richard A Krueger and Mary Anne Casey. *Focus Groups: A Practical Guide for Applied Research.* SAGE Publications, 2014.

[77] Leona Chandra Kruse, Stefan Seidel, and Shirley Gregor. "Prescriptive Knowledge in IS Research: Conceptualizing Design Principles in Terms of Materiality, Action, and Boundary Conditions". In: *2015 48th Hawaii International Conference on System Sciences.* 2015, pp. 4039–4048. DOI: 10.1109/HICSS.2015.485.

[78] William Kuechler and Vijay Vaishnavi. "Promoting relevance in IS research: An informing system for design science research". In: *Informing Science* 14 (2011), p. 125. DOI: 10.28945/1498.

[79] Marco Kuhrmann, Daniel Méndez Fernández, and Maya Daneva. "On the pragmatic design of literature studies in software engineering: an experience-based guideline". In: *Empirical software engineering* 22.6 (2017), pp. 2852–2891. DOI: 10.1007/s10664-016-9492-y.

[80] Jong Seok Lee, Jan Pries-Heje, and Richard Baskerville. "Theorizing in design science research". In: *Service-Oriented Perspectives in Design Science Research: 6th International Conference, DESRIST 2011, Milwaukee, WI, USA, May 5-6, 2011. Proceedings 6.* Springer. 2011, pp. 1–16.

[81] Christine Legner, Tobias Pentek, and Boris Otto. "Accumulating design knowledge with reference models: insights from 12 years' research into data management". In: *Journal of the Association for Information Systems* 21.3 (2020), p. 2. DOI: 10.17705/1jais.00618.

[82] Stuart E Madnick and RY Wang. "Introduction to total data quality management (TDQM) research program". In: *Total Data Qual. Manag. Program MIT Sloan Sch. Manag* 1 (1992), p. 92.

[83] Stuart E Madnick et al. "Overview and framework for data and information quality research". In: *Journal of data and information quality (JDIQ)* 1.1 (2009), pp. 1–22. DOI: 10.1145/1515693.1516680.

[84] Christian Meske and Enrico Bunde. "Design principles for user interfaces in AI-Based decision support systems: The case of explainable hate speech detection". In: *Information Systems Frontiers* 25.2 (2023), pp. 743–773. DOI: 10.1007/s10796-021-10234-5.

[85] Frederik Möller, Tobias Moritz Guggenberger, and Boris Otto. "Towards a method for design principle development in information systems". In: *International Conference on Design Science Research in Information Systems and Technology.* Springer. 2020, pp. 208–220. DOI: 10.1007/978-3-030-64823-7_20.

[86] Philipp Offermann et al. "Artifact types in information systems design science–a literature review". In: *International Conference on Design Science Research in Information Systems*. Springer. 2010, pp. 77–92. DOI: 10.1007/978-3-642-13335-0_6.

[87] Martin H Ofner, Boris Otto, and Hubert Österle. "Integrating a data quality perspective into business process management". In: *Business Process Management Journal* (2012). DOI: 10.1108/14637151211283401.

[88] Marcelo Iury S. Oliveira and Bernadette Farias Lóscio. "What is a Data Ecosystem?" In: *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*. dg.o '18. Delft, The Netherlands: Association for Computing Machinery, 2018. DOI: 10.1145/3209281.3209335.

[89] Charles Oppenheim, Joan Stenson, and Richard M. S. Wilson. "Studies on Information as an Asset I: Definitions". In: *Journal of Information Science* 29.3 (2003), pp. 159–166. DOI: 10.1177/01655515030293003.

[90] Boris Otto. "Quality and Value of the Data Resource in Large Enterprises". In: *Information Systems Management* 32.3 (2015), pp. 234–251. DOI: 10.1080/10580530.2015.1044344.

[91] Boris Otto, Kai M Hüner, and Hubert Österle. "Identification of Business Oriented Data Quality Metrics." In: *ICIQ*. 2009, pp. 122–134.

[92] Boris Otto and Matthias Jarke. "Designing a multi-sided data platform: findings from the International Data Spaces case". In: *Electronic Markets* 29.4 (2019), pp. 561–580. DOI: 10.1007/s12525-019-00362-x.

[93] YoungKi Park, Omar A El Sawy, and Peer Fiss. "The role of business intelligence and communication technologies in organizational agility: a configurational approach". In: *Journal of the Association for Information Systems* 18.9 (2017). DOI: 10.17705/1jais.00467.

[94] Michael Quinn Patton. "Two Decades of Developments in Qualitative Inquiry: A Personal, Experiential Perspective". In: *Qualitative Social Work* 1.3 (2002), pp. 261–283. DOI: 10.1177/1473325002001003636.

[95] Ken Peffers et al. "A design science research methodology for information systems research". In: *Journal of management information systems* 24.3 (2007), pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

[96] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. "Data Quality Assessment". In: *Communications of the ACM* 45.4 (Apr. 2002), pp. 211–218. DOI: 10.1145/505248.506010.

[97] S. Purao, A. Bush, and M. Rossi. "Problem and design spaces during object-oriented design: an exploratory study". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*. 2001. DOI: 10.1109/HICSS.2001.926343.

[98] Erhard Rahm and Hong Hai Do. “Data cleaning: Problems and current approaches”. In: *IEEE Data Eng. Bull.* 23.4 (2000), pp. 3–13.

[99] Thomas C. Redman. “The Impact of Poor Data Quality on the Typical Enterprise”. In: *Communications of the ACM* 41.2 (Feb. 1998), pp. 79–82. DOI: 10.1145/269012.269025.

[100] Thomas C. Redman. *To Improve Data Quality, Start at the Source.* Accessed: 09.01.2023. 2020. URL: https://hbr.org/2020/02/to-improve-data-quality-start-at-the-source.

[101] Bent Richter, Philipp Staudt, and Christof Weinhardt. “Designing Local Energy Market Applications”. In: *Scandinavian Journal of Information Systems* 34.2 (2022), p. 2.

[102] Michael Rosemann and Iris Vessey. “Toward improving the relevance of information systems research to practice: the role of applicability checks”. In: *Mis Quarterly* (2008), pp. 1–22. DOI: 10.2307/25148826.

[103] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical software engineering* 14.2 (2009), pp. 131–164. DOI: 10.1007/s10664-008-9102-8.

[104] Marcelo Iury S Oliveira, Glória de Fátima Barros Lima, and Bernadette Farias Lóscio. “Investigations into data ecosystems: a systematic mapping study”. In: *Knowledge and Information Systems* 61.2 (2019), pp. 589–630. DOI: 10.1007/s10115-018-1323-6.

[105] Benjamin Saunders et al. “Saturation in qualitative research: exploring its conceptualization and operationalization”. In: *Quality & quantity* 52.4 (2018), pp. 1893–1907. DOI: 10.1007/s11135-017-0574-8.

[106] Sebastian Schelter et al. “Unit testing data with deequ”. In: *Proceedings of the 2019 International Conference on Management of Data.* 2019, pp. 1993–1996. DOI: 10.1145/3299869.3320210.

[107] Reinhard Schütte et al. “Quo Vadis Information Systems Research in Times of Digitalization?” In: *Business & Information Systems Engineering* 64.4 (2022), pp. 529–540. DOI: 10.1007/s12599-022-00759-7.

[108] Maung K Sein et al. “Action design research”. In: *MIS quarterly* (2011), pp. 37–56. DOI: 10.2307/23043488.

[109] Ganesan Shankaranarayanan and Roger Blake. “From content to context: The evolution and growth of data quality research”. In: *Journal of Data and Information Quality (JDIQ)* 8.2 (2017), pp. 1–28. DOI: 10.1145/2996198.

[110] Arun Swami, Sriram Vasudevan, and Joojay Huyn. “Data sentinel: A declarative production-scale data validation platform”. In: *36th International Conference on Data Engineering (ICDE).* IEEE. 2020, pp. 1579–1590. DOI: 10.1109/ICDE48307.2020.00140.

[111] Paul P Tallon and Alain Pinsonneault. “Competing perspectives on the link between strategic information technology alignment and organizational agility: insights from a mediation model”. In: *MIS Quarterly* (2011), pp. 463–486.

[112] Daniel Tebernum, Marcel Altendeitering, and Falk Howar. “A Survey-Based Evaluation of the Data Engineering Maturity in Practice”. In: *Data Management Technologies and Applications*. Ed. by Alfredo Cuzzocrea et al. Cham: Springer Nature Switzerland, 2023, pp. 1–23. DOI: `10.1007/978-3-031-37890-4_1`.

[113] Daniel Tebernum, Marcel Altendeitering, and Falk Howar. “DERM: A Reference Model for Data Engineering”. In: *Proceedings of the 10th International Conference on Data Science, Technology and Applications*. INSTICC. 2021, pp. 165–175. DOI: `10.5220/0010517301650175`.

[114] John Venable, Jan Pries-Heje, and Richard Baskerville. “A Comprehensive Framework for Evaluation in Design Science Research”. In: *Design Science Research in Information Systems. Advances in Theory and Practice*. Ed. by Ken Peffers, Marcus Rothenberger, and Bill Kuechler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 423–438.

[115] W3C. *Data on the Web Best Practices: Data Quality Vocabulary*. Accessed: 10.01.2023. 2016. URL: `https://www.w3.org/TR/vocab-dqv/`.

[116] Valerianne Walter, Andreas Gyoery, and Christine Legner. “Deploying machine learning based data quality controls–Design principles and insights from the field”. In: *Wirtschaftsinformatik 2022 Proceedings* (2022).

[117] Pei Wang and Yeye He. “Uni-Detect: A Unified Approach to Automated Error Detection in Tables”. In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD ’19. Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 811–828. DOI: `10.1145/3299869.3319855`.

[118] Richard Y. Wang and Diane M. Strong. “Beyond Accuracy: What Data Quality Means to Data Consumers”. In: *Journal of Management Information Systems* 12.4 (1996), pp. 5–33. DOI: `10.1080/07421222.1996.11518099`.

[119] Jane Webster and Richard T. Watson. “Analyzing the Past to Prepare for the Future: Writing a Literature Review”. In: *MIS Quarterly* 26.2 (2002), pp. xiii–xxiii. DOI: `10.2307/4132319`.

[120] Soffi Westin and Maung Sein. “The design and emergence of a data/information quality system”. In: *Scandinavian Journal of Information Systems* 27.1 (2015), pp. 3–26.

[121] Robert K Yin. *Case study research: Design and methods*. Vol. 5. Sage, 2014.

A. Papers

Challenges of Data Management in Industry 4.0: A Single Case Study of the Material Retrieval Process by Antonello Amadori, Marcel Altendeitering, and Boris Otto. In International Conference on Business Information Systems, (pp. 379-390). 2020. DOI: https://doi.org/10.1007/978-3-030-53337-3_28.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 00 (2019) 000–000

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

Scalable Detection of Concept Drift: A Learning Technique Based on Support Vector Machines

Marcel Altendeitering[a,*], Stephan Dübler[b]

[a] *Fraunhofer ISST, Emil-Figge-Strasse 91, 44227 Dortmund, Germany*
[b] *Ruhr-Universität Bochum, Universitätsstrasse 150, 44801 Bochum, Germany*

Abstract

The issue of concept drift describes how static machine-learning models build on historical data can become unreliable over time and pose a significant challenge to many applications. Although, there is a growing body of literature investigating concept drift existing solutions are often limited to a small number of samples or features and do not work well in Industry 4.0 scenarios. We are proposing a novel algorithm that extends the existing concept drift algorithm FLORA3 by utilizing support vector machines for the classification process. Through this combination of dynamic and static approaches the algorithm is capable of effectively analyzing data streams of high volume. For evaluation, we tested our algorithm on the publicly available data set 'elec2', which is based on the energy market in Australia. Our results show that the proposed algorithm needs less computational resources compared to other algorithms while maintaining a high level of accuracy.

Keywords: concept drift; SVM; FLORA3; machine learning; energy data

1. Introduction

Commonly referred to as 'Industry 4.0' or 'Smart Manufacturing' the fourth industrial revolution is gaining popularity. The revolution encompasses the shift towards fully-connected manufacturing facilities based on sensor data streams and the Internet of Things (IoT) [1]. These facilities and the exchange of data among connected devices leverages the automation of production and helps the manufacturer to sustain competitive advantages [2].

With the foundation of Industry 4.0 new paths for quality control and Zero-Defect-Manufacturing (ZDM) opened up. ZDM refers to minimizing the number of defects and errors in a production process and aims at reducing the number of defective products to zero. Although ZDM was initially established in the 1960s, many of its concepts like autonomous quality, supply chain optimization or predictive maintenance are even more important nowadays [3]. A vital component of ZDM is the continuous assessment of process and product quality supported by the concept of Autonomous Quality (AQ). AQ is targeted at reducing or even eliminating the human actions in the quality control process [4]. Through gathering product and process data and mining quality relevant features, it becomes possible to automate decision-making and increase the overall process and product quality [5, 6].

In manufacturing environments the accuracy and reliability of predicted values is an important quality feature [7, 8]. For example, in the context of predictive maintenance an inaccurate prediction could hint at an upcoming machine failure. In such cases it is necessary to distinguish between an actual failure and the possible change of an underlying variable (e.g. temperature on the shopfloor), which does not necessarily imply a machine failure. The latter is referred to as 'Concept Drift' [9]. For data streams concept drift is detected by predicting new values using machine-learning and comparing these estimations with sensed values. An increasing prediction error hints at a drifting concept [7]. Since machine-learning classifiers are usually trained on historical data, concept drift is difficult to detect and a challenge for the goals of AQ and ZDM.

Gama et al. [10] define concept drift as the conditional distribution of a target variable while the distribution of the input fea-

* Corresponding author. Tel.: +49-231-97677-461 ; fax: +49-231-97677-0.
E-mail address: marcel.altendeitering@isst.fraunhofer.de (Marcel Altendeitering).

tures remain unchanged. Following Gama et al. [10] and Webb et al. [9] we define a concept at time t as:

$$P_t(X, Y) \tag{1}$$

X is a set of input features, while Y is a set of target variables. Therefore, having two concepts *P(X,Y)* at times *[t,u]* concept drift is defined as:

$$P_t(X, Y) \neq P_u(X, Y) \tag{2}$$

In light of Industry 4.0 and larger amounts of data edge devices are becoming increasingly popular [1]. However, existing concept drift detectors are often not capable of efficiently analyzing fast growing data sets on small computational resources and in a limited amount of time [7, 11]. For this, literature offers adaptive algorithms or windowing methods to find the optimal balance of computational resources and predictive accuracy [10]. For applying our solution in an industrial context as a means towards AQ and ZDM it should furthermore be capable of handling real numbers as these are common in Industry 4.0 scenarios (e.g. machine temperature or deterioration values) [7].

To meet these requirements we present a novel windowing algorithm for concept drift detection in data streams based on Support-Vector-Machines (SVMs). We therefore extended the existing concept drift detector FLORA3 [12, 13] and evaluated our solution on the publicly available and established data set 'elec2' [14, 15]. In comparison to other solutions presented in literature our algorithms maintains a high level of accuracy while requiring less computational resources.

The remainder of this paper is structured as follows. In section 2 we briefly discuss different approaches for concept drift detection. We continue by presenting the algorithm we developed (Section 3) and continue by describing the methodological approach we followed for evaluation and the results we obtained (Section 4). We conclude this study by discussing the results and limitations and highlight paths for future work (Section 5).

2. Detecting Concept Drift in Data Streams

In general there are three different approaches for detecting concept drift [8]. Put simply, these can be described as altering the algorithm (adaptive algorithms), altering the data (windowing techniques) and combining multiple solutions (ensemble technique). In the following we will provide an overview of solutions with regard to detecting concept drift in data streams for each principle.

2.1. *Adaptive algorithms*

A common method in the area of adaptive algorithms is the use of decision trees [16]. One of the most popular algorithms is the 'Very Fast Decision Tree' (VFDT). This algorithm can derive a tree-structure from a subset of the data and predict if the same tree-structure is suitable for the whole data set. With this information it is possible to check if the distribution of the last k events equals the remaining data set [17]. If not a concept drift has occurred and a new tree will be created.

Another well-established algorithm is the k-Nearest Neighbour (kNN) algorithm. Alippi and Roveri [18] developed an adoption of the kNN algorithm for detecting concept drift in data streams. Whenever concept drift occurs their solution decreases the value of k to exclude data that is falsifying the prediction for new data. If, however, the distribution of data is constant k is increased until the optimal value for k has been found.

As adaptive algorithms constantly adjust their processing method to the current environment they are limited in detecting recurring patterns within the data and cannot profit from repetitive concept drifts as efficiently as other solutions.

2.2. *Windowing methods*

The assumption behind windowing methods comes from data management and states that the most recent data is the most informative for a prediction [10]. This means a windowing method only uses the last n values for calculating a predictive model. Hereby, the key challenge is to decide on a suitable value for n. A larger window reacts to concept drift more slowly, but a smaller window can result in a large number of false positives. In order to cope with these challenges the FLORA algorithms were developed [12, 13]. For instance, FLORA3 constantly adjusts the window size to find the optimal value of n [13]. Additionally, each data-point can be weighted to represent its importance for the prediction. By selecting an appropriate window-size and weights the FLORA3 algorithm is capable of detecting concept drift in different scenarios.

Windowing methods are generally fast and easy to build. However, since only a certain amount of data is taken into consideration these techniques disregard some potentially useful information in the data.

2.3. *Ensemble techniques*

Ensemble techniques combine several classifiers to an overall prediction. The advantage of using ensemble techniques is that they are able to combine the advantages of adaptive and windowing methods. For example, different classifiers can be built for older and more recent data and weighted according to their importance. This way no information is lost as compared to windowing methods. This leads to a good performance of ensemble techniques. However, the combination of multiple algorithms can lead to complex solutions, which results in a slow execution [11]. Famous solutions based on this principle are, for example, 'AdaBoost' [19] or 'XGBoost' [20]. Recent reviews of current research around ensemble techniques are provided by Krawczyk et al. and Brzezinski and Stefanowski [8, 11].

3. SVM-based windowing method

In this section we present our 'SVM-based windowing method', which consists of a core algorithm and an optional algorithmic extension. Depending on the use case and available computing power the user can either only utilize the core of the algorithm or implement the full version for more exact results. We decided to use a windowing method because it offers the possibility for the user to determine when concept drift has occurred in the past. The knowledge of the time frames concept drift has occurred in leads to a better understanding of repetitive patterns within the data. Furthermore, no extensive computational resources are needed to execute the algorithm, making it suitable for execution on edge or cloud devices.

3.1. Approach

In order to develop a scalable concept drift detector for Industry 4.0 scenarios we started by analyzing the FLORA3 algorithm as it is the most prominent windowing method [12, 13]. As we implemented the FLORA3 algorithm on the publicly available data set 'Sensor Stream' [21] we noticed that the algorithm suffers from long execution times in its original form. Table 1 exemplary shows the execution times for calculating the concepts for different blocks of training data.

Table 1. execution time of calculating concepts for different settings.

Interval size	Sensors (#)	Block size	execution time
1000	1	100	15s
1000	1	200	73s
1000	1	300	21min
1000	2	100	17s
1000	5	100	39s
1000	3	200	27min
2000	3	200	24min
3000	3	300	22min
1000	54	100	17min

The interval size determines the tolerances for the possible concepts, meaning a large interval size results in faster processing but makes the concepts imprecise. A smaller interval size leads to the opposite effect. We observed that the execution times increase significantly with larger amounts of data, which makes it impractical for high volume data streaming use cases. Another problem is that FLORA3 was designed for boolean attributes but cannot work with real numbers in its original form. However, analyzing data with real numbers as input features is an important requirement we derived. This lead us to extend the algorithm with checks whether an attribute is in an interval of real numbers and based on this information defines the concepts.

We concluded that the FLORA3 algorithm is a useful method to handle streaming data, with the ability to analyze new data as it comes in based on distributions of the past. It is capable of detecting concept drift and to react on it. However, it also comes with a number of limitations that need to be addressed. We therefore built on the general idea of the algorithm and extended it in order to create an intelligent method for industrial streaming data.

When new data comes in the task of predicting the classes for a batch of input features is static for a short period of time. We thus decided to apply an established method from the field of static data analysis and chose to use SVMs as they are well suited for basic categorization tasks. Specifically, we used the established 'caret' package for 'R' [22] for building an SVM that handles the categorization task for a single block. As SVMs are a static tool for data analysis we rely on our own adaptive system, inspired by FLORA3, for working on streaming data.

3.2. Settings

The algorithm for analyzing data streams is suited for data inputs comprised of a finite number of features. Since we are using a SVM for classification, these classes must be separable in two distinct categories (i.e. a two-class problem). For creating static prediction tasks from a larger data stream the algorithm uses small batches of data, called windows. The window size (i.e. time interval) is set by the user and used by the algorithm as data input. A shorter window usually leads to more exact results, and a larger one to faster processing. For selecting the window size it should be considered that the window forms a closed block of analysis. This means that concept drift can only be detected between windows, but not within a window. For very short windows the aspects of constant attributes and underrepresented classes come in to play more often. We will discuss a way of overcoming these problems later on.

Additionally, it is necessary to define at what level a prediction is considered sufficient, which again forms a trade-off between accuracy and execution time. In the following, we will refer to the time interval as T and the threshold to a sufficient prediction as $p \in [0, 1]$.

The task is to predict the class of the data input based on its features. By observing the predictive error our algorithm is capable of detecting concept drift, adapting to it, and report when and in what direction it occurred.

3.3. Core algorithm

After the algorithm has been trained on a static data set with the minimum size of T it can be used to make predictions for streaming data with the same structure. Therefore, the training data set gets partitioned in blocks of length T. The first block, which we call B_1, gets preprocessed according to the type of data and a SVM model is trained on it. After that the SVM model is used to make predictions about the class distributions of the following block B_2 of size T. As the actual class labels become available the accuracy of predictions for B_2 are calculated and saved in $q_2 \in [0, 1]$. If

$$q_2 \geq p \tag{3}$$

holds, a new model is not necessary and the current one is reused for the next block. After the predictions for B_3 are calculated the system reevaluates whether

$$q_3 \geq p \tag{4}$$

is true and decides accordingly. At this point there is still the possibility to train a new model on B_2 even if the previous model has been reused. If the predictions for the third block were not accurate enough a new SVM model is trained on B_2. This process is repeated on every block, which is coming in through the data stream, and forms the core our solution. Algorithm 1 shows the core algorithm in pseudo code.

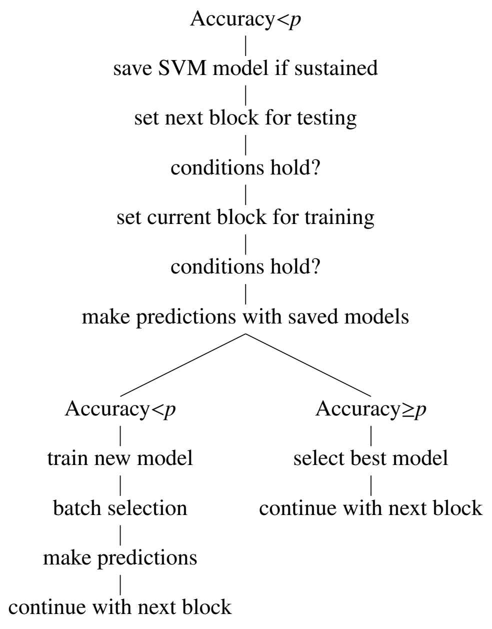
Algorithm 1 Core algorithm

```
partition data set in blocks of size $T$
preprocess $B_1$
train SVM on $B_1$
preprocess $B_2$
predict classes for $B_2$ using trained SVM and calculate $q_2$
if $q_2 \geq p$ then
    preprocess $B_3$; use current SVM on $B_3$
    calculate $q_3$
    if $q_3 < p$ then
        train SVM on $B_2$, continue with $B_3$
    else
        continue with $B_4$
else
    train SVM on $B_2$, continue with $B_3$
proceed with following $B_i$ in the same way
```

In order to make this algorithm smarter, a method to recognize previous patterns is implemented. Since making predictions with an existing model is faster than training a new one, the possibility to reuse existing models makes it more efficient and creates room for a finer analysis. Every time the algorithm attempts to train a new model it first checks if the currently used one makes predictions better than p for at least 3 periods. The number of periods taken into consideration can be changed by the user. If that is the case, the current SVM model is saved for later use. Afterwards, the previously saved models are used to make predictions for the next block of data. In case there is more then one, the one with the best accuracy is chosen and reused for further analysis. If all previous models fail to make accurate predictions for the upcoming block a new one is trained as the last possible option. Our tests have shown that this method leads to a significant difference in efficiency, especially when the algorithm has been running for a while. It is more likely to recognize a pattern if it has already seen a wider range of distributions in the past. We will take a closer look at this idea when discussing the experimental results. Finally, this enables us to identify class distributions that stayed relatively constant for a while because in that case the corresponding SVM was used for multiple periods. Based on this, we are able to detect and quantify concept drift in the incoming data stream.

3.4. Extension

Building on the previously developed pattern we want to improve our algorithm for more accurate predictions. If the predictions for the next block are sufficient the used model is reused for the next block, if not the following procedure applies:



Through enhancing the method with a batch selection mechanism inspired by [23] it is possible to recognize patterns that have been in the data stream before. It also enables us to learn from them and build models considering all the data featuring similar distributions and reduce overfitting. Furthermore, a mechanism to enlarge the used block for training successively such that there is a representative data set to base the predictions on is available.

When aiming for an accurate result it is reasonable to reduce the value of T. However, this creates the problem of constant attributes within a block and of underrepresented classes (i.e. all data in one block belongs to one class). Both of these occurrences make a SVM far less effective. To overcome these issues we can step-by-step enlarge the current block with previous data until none of those problems are present any longer.

Let $\boxed{1}$ and $\boxed{2}$ be the two classes a data-point d_t ($t \in \{1, 2, ..., T\}$), being part of the currently viewed block B_s, can belong to and $a_{t,i}$ be the i-th attribute of data-point d_t then the following condition is part of the algorithm (see Algorithm 2).

Algorithm 2 Extension of the training set

```
while $\forall d_t \in B_s : d_t \in \boxed{1}$ or $\forall d_t \in B_s : d_t \in \boxed{2}$ or $\exists i \forall d_t \in B_s : a_{t,i}$ is constant do
    add $d_0$ to $B_s$ ( $d_0$ is the last data-point before $B_s$)
    T ← T+1
end
```

After this process has been completed the batch selection comes in. The new SVM model gets trained on the (possibly enhanced) data set, only if a new model is needed in the first place. Subsequently, this model is used to make predictions for previous blocks in the data set. Let the number of already analyzed blocks be *s-1* then the SVM model is trained on block *s-1* and after that predictions for block *s-2*, *s-3*, etc. are made. Let B_i be the i-th block with $i \in \{1, 2, ..., s-1\}$ and q_i the predictive accuracy for this block. If for a given i

$$q_i \geq p \tag{5}$$

holds then the concept in B_i is considered to be similar to the current concept in B_s. It is remembered for having this property and to be combined later on with all blocks featuring similar concepts.

This procedure repeats until either all previous blocks have been analyzed or the maximum amount for blocks in a training data set has been reached. This upper limit exists to stop the data set from becoming too big as this would cause long execution times. Also, it prevents very old data from becoming part of the data set because this data has potentially become irrelevant for ongoing predictions. We call this limit m. Once m has been reached the threshold for a similar block gets raised to $\tilde{p}$ with

$$\tilde{p} \geq p \tag{6}$$

as there is enough data available to make a meaningful selection. This allows the batch selection to have higher standards for selecting as soon as there is enough data available such that there is room for a big enough selection of similar blocks.

When this step has finished all those blocks considered to be similar to the current one are combined to one big data set for training. This newly created data set is then used for training a new model. This method is described in pseudo code in Algorithm 3.

This process is repeated whenever there is need for a new model, meaning neither the current nor any of the previous models produces an accuracy that meets p.

Algorithm 3 Training a new SVM model

```
train SVM model on $B_s$
for all $i \in \{1, 2, ..., s-1\}$ do
    make predictions for $B_i$ using trained model
    calculate $q_i$
    if $q_i \geq p$ then
        save i
    if $m$ has been reached then
        $p \leftarrow \tilde{p}$
end
$M := \bigcup_{i\ saved} B_i$
train new SVM model on $M$
```

A full implementation of the proposed solution is available on GitHub [1]

4. Evaluation

For evaluating our solution we conducted an experiment on a real-world data set. Both, the experimental setting and the results are presented below.

4.1. Experimental Setting

Providing an appropriate test scenario for concept drift detection in industrial data streams is a difficult task. There is little data available that is suited for benchmarking and approved for publication [7]. In order to retrieve comparable results we decided to use the publicly available data set 'elec2' [14] even though it is not coming from an industrial context. The 'elec2' data set has been used in many concept drift studies and is well-established in the scientific community [15].

The data set features information about the electricity market in New South Wales, Australia over a period of 2.6 years (135 weeks) and contains a data-point for every half hour in this time period. Each data-point consists of 5 features about the demand and supply of electricity in New South Wales itself and nearby areas. It contains either one of the two classes 'up' or 'down' depending on whether the electricity price has risen or fallen in the respective time period. The data set is subject to concept drift as the patterns in energy consumption change over time due to changing habits or seasonality [14]. Out of those 135 weeks the last 82 weeks are getting used for testing as the earlier measurements do not include all variables. In our experimental setting we have set

$$T = 24h, p = 0.8 \tag{7}$$

because a day is the shortest period of repeated concept drift in this data set. Also, we considered a certainty of 80% about

[1] https://github.com/stephan421/Concept-Drift-Detection

the outcome as sufficient under these circumstances. As the 'elec2' data set is static we simulated streaming data through successively working on the batches of data-points starting from the beginning of the viewed time window. This method is similar to analyzing the data live and therefore making the electricity pricing more predictable.

For testing our algorithm we executed the implementation in R using different settings on a local computer with the following configuration: Intel Core i7-7500U 2.9 GHz, 16GB Memory, 500GB SSD.

4.2. Experimental Results

We conducted several experimental runs on the data set using different subsets of the data. Table 2 features the average accuracies and execution times over a five week period respectively.

Table 2. Accuracies for different data subsets.

Weeks	Accuracy (ø)	execution time
5	88%	24min
10	88%	1h 29min
15	88%	2h 2min
20	88%	2h 37min
25	86%	4h 50min
30	86%	5h 40min

Based on this data we can conclude that the analyzing method is suitable for this use case and produces constant and reliable results with an average predictive accuracy of 87.33%. From our measurements we derived that the execution time for a time period depends on the amount of concept drifts occurring and on average a decreased execution time correlates with a higher amount of data that has been analyzed. As described in the 'Comment on applicability' published by Harris regarding the 'elec2' data set [14], a naive method that always predicts the next class to be the same as the current class achieves an accuracy of 85.3% on this data set. This is based on the fact that the labels are not independently distributed; there are long consecutive periods of 'up' and long consecutive periods of 'down'. As one can see in Table 2 our classifier produces significantly better results then this and therefore is a valid tool to make predictions on this data set. Table 3 compares the accuracy of our algorithm with different algorithms using the 'elec2' data set [14, 15].

Our proposed solution is among the most accurate tested on this data set. In comparison to other solutions, which achieve high accuracy at the cost of high complexity [25, 26], our solution does not require extensive computational resources. Specifically, it delivers accurate results whilst being more efficient in terms of execution time and hardware demands. Most competing algorithms on this data set proceed by analyzing data point by data point in order to achieve a high accuracy (e.g. [29, 31]). In contrast. our method analyzes a full day of 48 data points at once making it faster on high-volume data streams. Furthermore, many of the reported results, as shown in Table 3, were

Table 3. Comparison of our algorithm with other solutions (based on [14, 15]).

Algorithm	Accuracy (%)	Reference
DDM	89.6*	[24]
Learn++.CDS	88.5	[25]
KNN-SPRT	88.0	[26]
GRI	88.0	[27]
our algorithm	**87.3***	
FISH3	86.2	[28]
EDDM-IB1	85.7	[29]
naive classifier	**85.3**	
ASHT	84.8	[30]
bagADWIN	82.8	[30]
DWM-NB	80.8	[31]
Local detection	80.4	[32]
Perceptron	79.1	[33]
ADWIN	76.6	[34]
Prop. method	76.1	[35]
AUE	74.9	[36]
Cont. λ-perc.	74.1	[37]
CALDS	72.5	[38]
TA-SVM	68.9	[39]

** tested on a subset*

achieved using ensemble techniques [31], which provide a high accuracy at the cost of complexity. The proposed algorithm is more suitable for application in Industry 4.0 scenarios, which often requires edge processing and offers only little computational resources.

A shortcoming we encountered while evaluating our algorithm is that for a small number of blocks the predictive accuracy drops significantly as shown in Figure 1. This can most likely be explained with the data itself, as the respective blocks do not fit into the remaining data.

5. Conclusion

In this study we developed a novel tool for dynamic data analysis on streaming data, combining one of the most effective static approaches with a dynamic one. This way, the benefits of SVMs become available for use cases on streaming data and lead to the ability of reusing previously discovered results in the static data analysis for dynamic analysis. Through this combination we were able to build a concept drift detector that meets our requirements and works with limited computational resources, executes in a limited amount of time and handles real numbers. Our solution offers scalability and flexibility as it is adaptable to different settings of computational resources.

In addition to the field of concept drift we are contributing to the research of ZDM and AQ by lowering the threshold for operating a concept drift detector on edge devices. This could cause a more wide-spread adoption of drift detectors and simplify the utilization of concept drifts as a potential quality attribute for industrial data streams. Such a data quality attribute would not only help to easily discover concept drifts but also assist data management in inter or intra-organizational data exchanges and achieve the goal of ZDM.

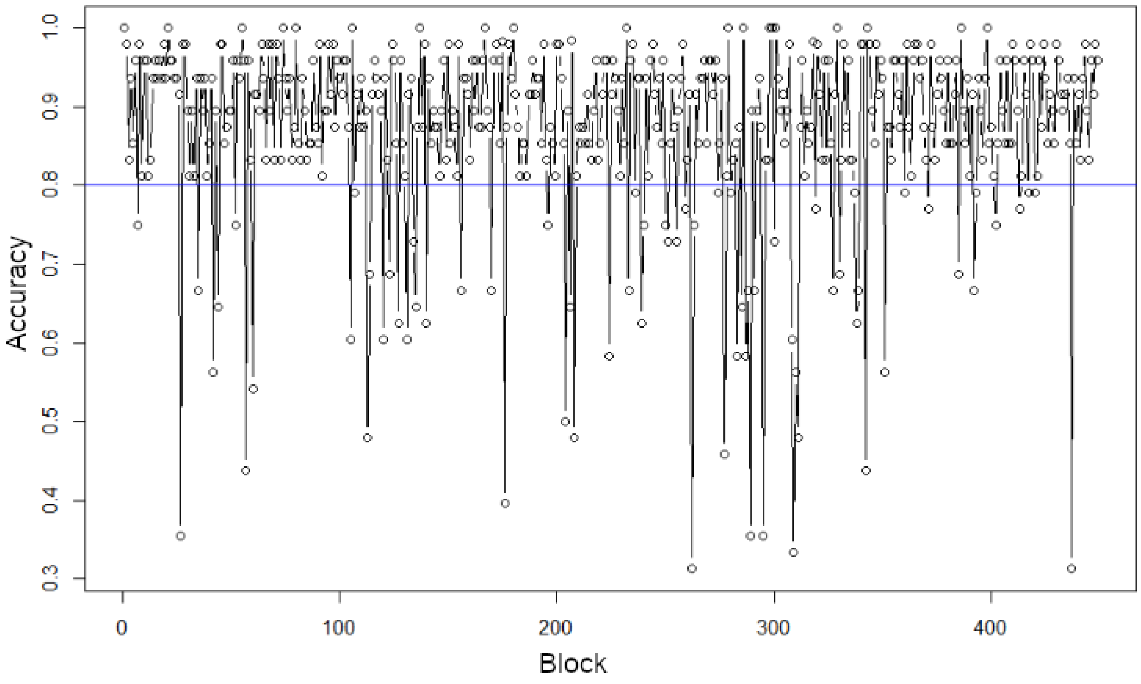


Fig. 1. Accuracy of prediction for different blocks.

However, our study is subject to a few limitations. Most importantly, our solution lacks a deeper evaluation with additional real-world and artificial data sets. We therefore want to apply our algorithm to public and non-public use cases in order to gain a deeper understanding of its accuracy and applicability in different scenarios. Since we are particularly interested in the role of concept drifts in the AQ and ZDM paradigms, we are planning to implement the algorithm in an Industry 4.0 context and assess its usefulness in industrial settings.

Furthermore, we intend to proceed our research by extending and optimizing the algorithm. Towards this end, we want to embed our algorithm in an ensemble of several solutions in order to benefit from other methods such as decision trees. This could further improve the accuracy but may lead to slower execution times. We thus need to find the optimal balance between execution time and accuracy. Another possible extension of our algorithm is the capability of distinguishing different kinds of drifts. So far we are able to discover if any drift occurred. However, it might be useful to investigate if drifts manifest in certain forms (e.g. recurring or gradual). This would be a valuable input for deriving what caused a drift and how it should be handled.

Acknowledgments

This research was partly supported by the EU's Horizon 2020 program and the QU4LITY project (GA no. 825030).

References

[1] R. Y. Zhong, X. Xu, E. Klotz, S. T. Newman, Intelligent manufacturing in the context of industry 4.0: A review, Engineering 3 (5) (2017) 616–630.
[2] E. Brynjolfsson, A. McAfee, The second machine age: Work, progress, and prosperity in a time of brilliant technologies, WW Norton & Company, 2014.
[3] P. B. Crosby, Quality is free: The art of making quality certain, Vol. 94, McGraw-hill New York, 1979.
[4] Y. Monden, "autonomous defects control" assures product quality, in: Y. Monden (Ed.), Toyota Production System: An Integrated Approach to Just-In-Time, Springer US, Boston, MA, 1994, pp. 221–238.
[5] K.-S. Wang, Towards zero-defect manufacturing (zdm)—a data mining approach, Advances in Manufacturing 1 (1) (2013) 62–74.
[6] P. Schlegel, K. Briele, R. H. Schmitt, Autonomous data-driven quality control in self-learning production systems, in: Advances in Production Research, Springer, 2018, pp. 679–689.
[7] J. Zenisek, F. Holzinger, M. Affenzeller, Machine learning based concept drift detection for predictive maintenance, Computers & Industrial Engineering 137 (2019) 106031.
[8] D. Brzeziński, J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, IEEE Transactions on Neural

Networks and Learning Systems 25 (1) (2014) 81–94.

[9] G. I. Webb, L. K. Lee, F. Petitjean, B. Goethals, Understanding concept drift, arXiv preprint arXiv:1704.00362 (2017).

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (2014) 1–37.

[11] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: A survey, Information Fusion 37 (2017) 132 – 156.

[12] G. Widmer, M. Kubat, Effective learning in dynamic environments by explicit context tracking, in: European Conference on Machine Learning, Springer, 1993, pp. 227–243.

[13] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine learning 23 (1) (1996) 69–101.

[14] M. Harries, N. S. Wales, Splice-2 comparative evaluation: Electricity pricing (1999).

[15] I. Žliobaitė, How good is the electricity benchmark for evaluating concept drift adaptation, arXiv preprint arXiv:1301.3524 (2013).

[16] T. R. Hoens, R. Polikar, N. V. Chawla, Learning from streaming data with concept drift and imbalance: an overview, Progress in Artificial Intelligence 1 (1) (2012) 89–101.

[17] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 97–106.

[18] C. Alippi, M. Roveri, Just-in-time adaptive classifiers in non-stationary conditions, in: International Joint Conference on Neural Networks, IEEE, 2007, pp. 1014–1019.

[19] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences 55 (1) (1997) 119–139.

[20] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, ACM, 2016, pp. 785–794.

[21] X. Zhu, Stream data mining repository (2010).
URL `http://www.cse.fau.edu/~xqzhu/stream.html`

[22] M. Kuhn, The caret package, R Foundation for Statistical Computing, Vienna, Austria. URL https://cran. r-project. org/package= caret (2012).

[23] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intelligent Data Analysis 11 (1) (2007) 3–28.

[24] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian symposium on artificial intelligence, Springer, 2004, pp. 286–295.

[25] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE transactions on knowledge and data engineering 25 (10) (2012) 2283–2301.

[26] G. J. Ross, N. M. Adams, D. K. Tasoulis, D. J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern recognition letters 33 (2) (2012) 191–198.

[27] J. M. Tomczak, A. Gonczarek, Decision rules extraction from data stream in the presence of changing context for diabetes treatment, Knowledge and Information Systems 34 (3) (2013) 521–546.

[28] I. Žliobaitė, Combining similarity in time and space for training set formation under concept drift, Intelligent Data Analysis 15 (4) (2011) 589–611.

[29] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth international workshop on knowledge discovery from data streams, Vol. 6, 2006, pp. 77–86.

[30] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 249–260.

[31] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, Journal of Machine Learning Research 8 (Dec) (2007) 2755–2790.

[32] J. Gama, G. Castillo, Learning with local drift detection, in: International Conference on Advanced Data Mining and Applications, Springer, 2006, pp. 42–55.

[33] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: Massive online analysis, Journal of Machine Learning Research 11 (May) (2010) 1601–1604.

[34] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM international conference on data mining, SIAM, 2007, pp. 443–448.

[35] D. Martínez-Rego, B. Pérez-Sánchez, O. Fontenla-Romero, A. Alonso-Betanzos, A robust incremental learning method for non-stationary environments, Neurocomputing 74 (11) (2011) 1800–1808.

[36] D. Brzeziński, J. Stefanowski, Accuracy updated ensemble for data streams with concept drift, in: International conference on hybrid artificial intelligence systems, Springer, 2011, pp. 155–163.

[37] N. G. Pavlidis, D. K. Tasoulis, N. M. Adams, D. J. Hand, λ-perceptron: An adaptive classifier for data streams, Pattern Recognition 44 (1) (2011) 78–96.

[38] J. B. Gomes, E. Menasalvas, P. A. Sousa, Calds: context-aware learning from data streams, in: Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques, ACM, 2010, pp. 16–24.

[39] G. L. Grinblat, L. C. Uzal, H. A. Ceccatto, P. M. Granitto, Solving non-stationary classification problems with coupled support vector machines, IEEE Transactions on Neural Networks 22 (1) (2010) 37–51.

Mining Data Quality Rules for Data Migrations: A Case Study on Material Master Data by Marcel Altendeitering. In International Symposium on Leveraging Applications of Formal Methods, (pp. 178-191). 2021. DOI: https://doi.org/10.1007/978-3-030-89159-6_12.

Designing Data Quality Tools: Findings from an Action Design Research Project at Boehringer Ingelheim by Marcel Altendeitering and Tobias Moritz Guggenberger. In ECIS 2021 Research Papers. 2021. URL: https://aisel.aisnet.org/ecis2021_rp/95/.

DERM: A Reference Model for Data Engineering by Daniel Tebernum, Marcel Altendeitering, and Falk Howar. In Proceedings of the 10th International Conference on Data Science, Technology and Applications - DATA, (pp. 165-175). 2021. DOI: https://doi.org/10.5220/0010517301650175.

A Functional Taxonomy of Data Quality Tools: Insights from Science and Practice by Marcel Altendeitering and Martin Tomczyk. In Wirtschaftsinformatik 2022 Proceedings. 2022. URL: https://aisel.aisnet.org/wi2022/business_analytics/business_analytics/4/.

Data Sovereignty for AI Pipelines: Lessons Learned from an Industrial Project at Mondragon Corporation by Marcel Altendeitering, Julia Pampus, Felix Larrinaga, Jon Legaristi, and Falk Howar. In 2022 IEEE/ACM 1st International Conference on AI Engineering–Software Engineering for AI (CAIN), (pp. 193-204). 2022. DOI: https://doi.org/10.1145/3522664.3528593.

Data Quality in Data Ecosystems: Towards a Design Theory by Marcel Altendeitering, Stephan Dübler, and Tobias Moritz Guggenberger. In AMCIS 2022 Proceedings. 2022. URL: https://aisel.aisnet.org/amcis2022/DataEcoSys/DataEcoSys/3/.

Proceedings of the 57th Hawaii International Conference on System Sciences | 2024

Data Quality Tools: Towards a Software Reference Architecture

Marcel Altendeitering
Fraunhofer ISST
TU Dortmund University
marcel.altendeitering@isst.fraunhofer.de

Tobias Moritz Guggenberger
Fraunhofer ISST
TU Dortmund University
tobias.moritz.guggenberger@isst.fraunhofer.de

Abstract

Organizations crave to succeed in the ongoing digital transformation, and central to this is the quality of data as a major source for business innovation. Data quality tools promise to increase the quality of data by managing and automating the different tasks of data quality management. However, established tools often lack support for the fundamental changes accompanying an ongoing digital transformation, such as data mesh architectures. In this paper, we propose a software reference architecture for data quality tools that guides organizations in creating state-of-the-art solutions. Our reference architecture is based on the knowledge captured from ten data quality tools described in the scientific literature. For evaluation, we conducted two qualitative focus group discussions using the adapted architecture tradeoff analysis method as a basis. Our findings reveal that the proposed reference architecture is well-suited for creating successful data quality tools and can help organizations assess offerings in the market.

Keywords: Data Quality Tools, Software Reference Architecture, Digital Transformation, Systematic Literature Review

1. Introduction

"You can't do anything important in your company without high-quality data" (Redman, 2020, p.1). For several reasons, a high level of data quality (DQ) is vital for organizations. It is required to secure organizational agility and avoid harmful societal effects of automated decision-making (Gröger, 2021; Marjanovic et al., 2021). Moreover, ensuring correct and high-quality data sets creates trust in data ecosystems and is becoming part of legislation (Geisler et al., 2021). For example, the proposed European AI Act sees it as organizations' due diligence to avoid errors in data sets and AI systems based on them (European Union, 2021).

Despite the paramount importance of DQ, many organizations struggle to provide data of adequate quality to business processes, thus impeding the success of digital transformations (Gröger, 2021; Legner et al., 2020). Most significantly, the context in which DQ tools operate is changing. The proliferation of big data uncovered a lack of scalability in centralized data management tools and efforts (Gröger, 2021). Consequently, organizations started to decentralize their data architectures (e.g., data mesh), leading to a distribution of the DQ work and DQ tools being used at the source (Dehghani, 2019; Redman, 2020). Additionally, the emergence of data ecosystems led to varying data consumers and called for new data management standards that DQ tools must incorporate (Geisler et al., 2021; Guggenberger et al., 2020). Finally, DQ is an inherently complex topic grounded in the subjective and multi-dimensional nature of DQ issues (Wang & Strong, 1996). This led to cumbersome and time-consuming processes for resolving DQ problems.

Established DQ tools fail to comprehend these changes fully as the new requirements break with their centralized concept (Altendeitering & Tomczyk, 2022). As a result, there is a need for a new kind of DQ tool that acts as an enterprise-wide framework and supports the creation of high-quality data products. The new tool should support the data owners across an organization in identifying, assessing, and correcting quality problems of heterogeneous data sources (Geisler et al., 2021). Yet, there is a lack of guidance and standardization on the functional composition and architectural design of such state-of-the-art DQ tools. This lack of guidance leads to data analytics and artificial intelligence initiatives that fall short of their promises, and DQ issues prevail in industrial practice (Geisler et al., 2021; Gröger, 2021). Our study addresses this research gap and proposes a software reference architecture that practitioners can use to create successful DQ tools. We formulated the following question that guided our research:

Research Question: *What does a reference architecture for successful data quality tools look like?*

In response to the research question, we propose a software reference architecture outlining essential functional components of DQ tools. Significant to our research was to support organizations in building state-of-the-art DQ tools and incorporating a vision and strategy for the future (Cloutier et al., 2010). For developing the reference architecture, we followed the design guideline by Angelov et al. (2012). Specifically, we informed our reference architecture by conducting a comparative analysis of ten concrete DQ tool architectures, which we identified in a systematic literature review (SLR). We used the adapted Architecture Tradeoff Analysis Method (ATAM) by Angelov et al. (2008) as a basis to evaluate the proposed reference architecture with experts from science and practice.

The remainder of this article is structured as follows. In section 2, we reason why there is a need for a new kind of DQ tool and present how a reference architecture can guide organizations. Section 3 outlines the SLR we conducted to develop the reference architecture. In section 4, we present and describe the proposed reference architecture for DQ tools and continue by describing the evaluation process in section 5. Finally, in section 6, we conclude our study by specifying contributions, limitations, and paths for future work.

2. Background

2.1. The History of Data Quality Tools

For a long time, researchers and practitioners have recognized the importance of high-quality data and developed various approaches to address this need (Madnick et al., 2009). However, the functional and non-functional composition of DQ tools has evolved over the past few decades based on two significant developments. First, the increasing maturity of data management as a response to a rapid digital transformation (Legner et al., 2020). Second, increasingly decentralized data architectures grounded in big data and trends such as 'Data Mesh' (Dehghani, 2019) and 'Data Products' (Hasan & Legner, 2023). Both developments directly influence the organizational scope, functional capabilities, and architectural positioning of DQ tools, which established solutions cannot always fulfill (Altendeitering & Tomczyk, 2022; Gröger, 2021).

Before the 1990s, data was primarily used to facilitate business operations, such as customer or inventory management. Data management involved dealing with single, unconnected databases and ensuring the correctness of data models (Legner et al., 2020). In this stage, DQ focused on the 'content' of data and was realized as conformity checks in database management systems (Shankaranarayanan & Blake, 2017). In this first generation, DQ tools aimed to promote internal business processes by supplying accurate data and allowing for data reuse. Neither DQ nor data architectures followed coordinated approaches, leading to data silos (Dehghani, 2019).

In the 1990s, organizations established centralized data management suites to prevent inaccessible and siloed data. Monolithic business analytics tools such as data warehouses gained popularity and led to the emergence of DQ as a novel research area (Madnick et al., 2009). Total quality management for data (TDQM), conceptualized by Madnick and Wang (1992), resulted in the quality-oriented management of data resources within organizations. Consequently, high-quality data facilitated organizational business processes and decision-making. Organizations utilized a second generation of integrated DQ tools that help coordinate data governance, define DQ rules, and measure DQ, thereby providing a comprehensive solution for these tasks (Legner et al., 2020; Madnick et al., 2009).

2.2. The Need for New Data Quality Tools

As the digital transformation accelerated in the 2010s, a third generation of data management emerged, in which data became central for strategic management (Legner et al., 2020). Organizations soon realized that centralized data architectures lacked scalability and were insufficient in light of continuous and big data. Consequently, locally-owned data products became more popular, providing a flexible solution to the new demands of data management (Hasan & Legner, 2023).

These data products are often organized in data mesh architectures, which rely on polyglot persistence and distributed data stores (Dehghani, 2019; Gröger, 2021). They follow the concepts of domain-driven design and incentivize the creation of high-quality data at the source, shifting the responsibility for DQ from the demand to the supply side (Hasan & Legner, 2023; Redman, 2020). By offering high scalability, data mesh architectures help overcome typical big data problems such as support for real-time data analytics and high complexity in data management (Geisler et al., 2021; Gröger, 2021). To realize such architectures, an internal data ecosystem can be established that relies on a self-service data infrastructure and standards for interoperability and portability (Gröger, 2021; Guggenberger et al., 2020).

However, established DQ tools often face difficulties operating in this new environment (Altendeitering & Tomczyk, 2022). Their reliance on centralized data architectures prevents them from comprehending the requirements of the changing environment, raising the need for a new kind of DQ tool. Additionally, established DQ tools often fail to address the induced shift of responsibility for DQ. In a data mesh architecture, where the data supplier takes care of DQ, a DQ tool must be capable of analyzing different data sets and be accessible to a diverse user group. Several studies (e.g., Altendeitering and Tomczyk (2022), Geisler et al. (2021), and Gröger (2021)) identified the need for new DQ tooling and called for research efforts and new approaches to DQ management. The new kind of DQ tool should support collaborative DQ efforts, thus empowering the ethical use of data-intensive systems (Marjanovic et al., 2021; Shankaranarayanan & Blake, 2017). It also helps perform DQ tasks at the source and offers portability and scalability to cope with big data generated in various sources (Geisler et al., 2021; Gröger, 2021).

3. Research Approach

With our study, we aim to provide a software reference architecture that guides the creation of new DQ tools. The reference architecture can contribute to standardizing DQ tools and reduce uncertainties in their functional design. Following the definitions of Bass et al. (2003) and Angelov et al. (2012), a reference architecture is essentially a generic description of the minimum architectural elements that should be used in designing a concrete software architecture. In this sense, a reference architecture captures the essence of existing architectures and guides the development of software artifacts ”by applying the new product mission, vision, and strategy to the wisdom of the past” (Cloutier et al., 2010, p.19). A viable reference architecture facilitates the development of solutions in different contexts and organizations by offering product-independent descriptions and supports the communication between domain professionals (Kazman et al., 1998).

Using the classification framework of Angelov et al. (2012), we can classify our reference architecture as a *Type 3 Reference Architecture*. This type of reference architecture aims to facilitate software architectures and is proposed by an independent organization to multiple organizations. It should define the main components and interfaces in a 'semi-detailed' way and be comprehensible and accessible to users from various contexts. A reference architecture that is too detailed or formal would impair its facilitation purpose. We argue that this type of reference architecture aligns most with our research setting and forms a suitable base for achieving our research goal.

To develop the intended software reference architecture, we followed the guidelines of design-oriented research in information systems (Hevner et al., 2004). We initialized our study by identifying the architectural building blocks essential to DQ tools and conducted an SLR following the guidelines of Kuhrmann et al. (2017) and Webster and Watson (2002). Our goal was to identify scientific studies reporting on the architecture and the functional and non-functional design of DQ artifacts. Since DQ artifacts are often framed differently, including tools or solutions, we only searched for 'data quality'. Consequently, we devised the following search term:

”data quality” AND (”design” OR ”architecture”)

Following the suggestion of Kuhrmann et al. (2017), we selected established sources in computer science and information systems domains to collect relevant studies. This selection comprised the databases: IEEE Xplore, ACM DL, ScienceDirect, AISeL, and Scopus as a meta-search engine. To avoid producing a very large result set, we limited our search to the study's title. Querying the selected databases using the proposed search term resulted in the identification of 168 studies.

We continued the reviewing process by eliminating any duplicates from the result set. Afterward, we manually screened the remaining papers to fit our research goal. Since we aimed to investigate standalone, general-purpose DQ tools, we filtered studies that were too specialized, such as those solely focused on domain-specific tools. Furthermore, we discarded studies that were too generic or did not directly describe DQ tools but analyzed other types of artifacts, like DQ's role in business processes, machine learning, or surveys. Lastly, we ensured that the selected studies describe state-of-the-art DQ artifacts affected by trends, such as big data, and thus only considered papers published from 2010. This way, we ensured that the reviewed DQ tools were influenced by the third-generation of data management (Legner et al., 2020). The screening procedure yielded in the identification of eight relevant papers. Subsequently, we conducted a forward and backward search, as suggested by Webster and Watson (2002), which revealed three more relevant studies.

To inform the design of our reference architecture, we conducted a comparative analysis of the architectural descriptions available in the papers identified during the SLR. In this step, we compared the concrete architectural designs to each other and derived similarities that form architectural patterns. We considered the appearance of an architectural component in at least three studies a hint towards its inclusion in our reference architecture (Cloutier et al., 2010). We assigned these components IDs to cross-reference them in the reference architecture description and indicate data flows between them. The concept matrix in Table 1 displays the identified architectural components and the corresponding papers. We ordered the studies by their publication date in descending order.

4. A Software Reference Architecture for Data Quality Tools

This section presents the contents of our software reference architecture. We organized this section into four subsections, one for each architectural layer shown in Table 1. Each subsection describes the architectural patterns essential to DQ tools and the relationships and data flows to other architectural components in a semi-detailed way (Angelov et al., 2012).

4.1. Interaction Layer

Seamless user interaction is vital for the success of any software tool. In the past, DQ was a highly technical task conducted by a team of DQ experts, leading to complex user interfaces with high semantic and syntactic barriers (Swami et al., 2020). Moreover, several DQ tools do not provide visual user interfaces and instead offer their functionalities solely as programming interfaces that can be integrated into source code (e.g., Great Expectations (2022)). Considering the trends for data democratization and decentralized DQ tools, the user interface must become accessible to various users.

DQ Workflow. Several studies describe the need for workflow functionalities on the interaction layer of a DQ tool. Often DQ tools realize the workflow using a process-based user interface, in which users can combine and connect different DQ checks or assertions (Swami et al., 2020). Altendeitering and Guggenberger (2021) describe a DQ workflow containing additional steps, such as data import or data pre-processing (*F7*). A significant challenge when designing a DQ workflow is the combination of opposing views on usability between users and developers (Alhamadi et al., 2022). This implies that DQ tools must be flexible to support DQ and data domain experts in their daily work.

Rule Definition. The rule definition component describes the capability to create custom DQ rules, checks, or assertions for validating data sets (*F3*). These custom DQ rules usually extend generally applicable DQ rules and express domain-specific DQ constraints (*D4*) (Swami et al., 2020). They can, for instance, be realized as user-defined functions on top of the underlying algorithmic stack (*F6*).

Explanation. A majority of the DQ artifacts we reviewed offer functionalities for explaining (1) the DQ algorithms in use and (2) the DQ issues identified. The former offers insights into the algorithmic stack (*F7*), which helps adjust tools to specific scenarios and increase trust in DQ metrics (Walter et al., 2022). The latter addresses the comprehensibility of DQ measurements (*F1*). Detailed explanations for low-quality scores can help unravel the multi-dimensional concept of DQ and assist users in taking adequate actions (Altendeitering, Dübler, et al., 2022).

4.2. Functional Layer

The functional layer contains core DQ functionalities and is central to the tool's success. Overall, big data plays an important role in the functional composition of DQ tools and shapes current trends. For instance, automation has become essential to cope with huge data sets, and a broad algorithmic basis is vital to conduct quality measurements on different data sets (Altendeitering & Tomczyk, 2022).

DQ Measurement. Measuring the quality of a data set along several DQ dimensions (e.g., completeness, accuracy, timeliness, etc.) is a key capability of DQ tools (Wang & Strong, 1996). It is featured in almost every tool we reviewed. In addition to different DQ dimensions, the measurement functionality must handle different kinds of data (*D1, D2*) and include the result as metadata (*D3*) (Blechinger et al., 2010).

DQ Rule Generation. In addition to manually defining DQ rules (*I2*), a DQ tool should provide functionality for automatically deriving DQ rules, for example, in the form of integrity constraints (Walter et al., 2022). The automated approach can help tackle ever-growing amounts of data and avoid a low DQ coverage in monitoring data (*F3*) (Altendeitering & Tomczyk, 2022). The generated rules are stored in the DQ rule repository to allow their reuse (*D4*).

Monitoring. The continuous monitoring of data sets is another critical component of DQ tools (Ehrlinger & Wöß, 2022). Its purpose is to validate data against DQ rules (*D4*) in a one-time (*D1*) or ongoing (*D2*) manner. This helps identify data errors early on and take actions (Gerloff & Cleophas, 2017; Swami et al., 2020). In fully automated data management pipelines, automated error correction and data manipulation can follow the monitoring step (Altendeitering & Guggenberger, 2021).

Reporting. The reporting component assembles the

Table 1. Comparative Analysis on the Studies Identified during the SLR. Components with ID appear in at least three studies.

Architectural Layer	*Component*	*ID*	Pradhan et al. (2023)	Alhamadi et al. (2022)	Altendeitering, Dübler, et al. (2022)	Altendeitering, Pampus, et al. (2022)	Walter et al. (2022)	Altendeitering and Guggenberger (2021)	Swami et al. (2020)	Gerloff and Cleophas (2017)	Westin and Sein (2015)	Blechinger et al. (2010)	Olbrich (2010)
Interaction	DQ Workflow	*I1*	x	x				x	x			x	x
	DQ Rule Definition	*I2*						x	x			x	x
	Explanation	*I3*		x	x		x	x	x		x	x	x
	Tutorial		x							x			
	Search						x	x					
	Custom Dashboard		x										
Functional	DQ Measurement	*F1*	x		x	x	x	x	x	x	x		x
	Data Cleaning						x		x				
	DQ Rule Generation	*F2*					x	x			x		
	Monitoring	*F3*	x	x	x	x	x		x	x		x	x
	Reporting	*F4*	x		x		x		x		x	x	x
	Interfaces	*F5*			x	x		x	x	x	x	x	x
	Algorithmic Stack	*F6*	x		x	x	x	x	x	x		x	x
	Data pre-processing	*F7*				x	x	x	x	x			
Data	Master Data	*D1*			x		x	x	x		x	x	x
	Continuous Data	*D2*	x	x	x	x		x	x	x	x		x
	Metadata	*D3*			x	x			x			x	x
	Conceptual Model												x
	DQ Rule Repository	*D4*					x		x		x	x	x
Non-Functional	Interoperability	*G1*		x	x	x	x			x			
	Ecosystem Support	*G2*			x	x		x					
	Process Integration	*G3*					x	x	x		x	x	x
	Availability				x					x			

DQ results from measurement (*F1*) and monitoring (*F3*) data. Afterward, it creates an easy-to-understand report containing details about the identified DQ issues and potential ways to fix them and stores it as metadata (*D3*) (Westin & Sein, 2015). For example, Swami et al. (2020) describe a DQ report that combines a current with a historic view on the monitoring results of a data set, thereby allowing to derive trends.

Interfaces. Seamless integration with the established system landscape within an organization is vital for DQ tools to avoid being isolated from existing processes and remaining segregated (*G3*). For integration, a DQ tool should offer standardized interfaces and connect with message broker or ecosystem technologies (*G2*) in place (Altendeitering & Guggenberger, 2021; Walter et al., 2022). The interfaces component acts as the gateway to access the main DQ functionalities (*F1-F4*) internally and externally, thus allowing for interoperability (*G1*).

Algorithmic Stack. A DQ tool needs a suitable stack of DQ algorithms to conduct its core DQ functions (*F1-F4*). Since organizations are confronted with a heterogeneous data landscape (*D1, D2*) and a proliferation of stream processing, DQ tools must offer a broad algorithmic basis (Gröger, 2021). The algorithms must be flexible and robust to handle data differing in size, complexity, and format and address the multi-dimensionality of DQ at the same time (Olbrich, 2010; Pradhan et al., 2023).

Data Pre-Processing. In several studies, we identified a data pre-processing component that precedes DQ analysis. When the algorithmic stack comprises AI algorithms (*F6*), data pre-processing can help improve the accuracy and execution time (Altendeitering & Guggenberger, 2021). The pre-processing can furthermore support data normalization and prepare heterogeneous data sources for a standardized quality analysis (Walter et al., 2022). For instance, textual data can be converted to numeric data using word embeddings to allow the usage of neural networks.

4.3. Data Layer

To overcome common data challenges, organizations move from centralized data landscapes towards a distributed approach, consisting of individually managed data products (Gröger, 2021; Hasan & Legner, 2023). As a result, DQ tools are confronted with diverse data sets, and to become successful, they must be capable of adapting to different contexts and embracing a variety of data sources. The data layer realizes the connection between the DQ tool and the data sources while ensuring that data is only temporarily held for data security.

Master Data Relational master data sets represent an important organizational asset as they contain information critical to business operations (Legner et al., 2020). Consequently, many DQ initiatives and tools focus on analyzing master data sets and ensuring that the essential data sets remain accurate and of high quality (Altendeitering & Guggenberger, 2021). Given the importance of master data, DQ tools must integrate with master data sets and management suites.

Continuous Data Continuous data streams are on the rise and play an increasingly important role in the organizational data landscape to facilitate real-time analytics and decision-making (Gröger, 2021). The new type of data source raises new requirements for DQ tools, which must offer suitable endpoints for including data streams and enabling DQ management on data streams (Gerloff & Cleophas, 2017).

Metadata DQ tools should integrate with a metadata repository (e.g., data catalogs) and store two pieces of DQ information. First, the history of DQ measurements (*F1*) and validations (*F3*), which data consumers can use to identify high-quality data sets and build trust in data products (Hasan & Legner, 2023). Second, the quality requirements of a data set in the form of DQ rules, manually specified by experts (*I2*) or automatically generated (*F2*) (Blechinger et al., 2010).

DQ Rule Repository The DQ rule repository is a component that stores standardized, generally applicable DQ rules, such as the 'not null' rule (Swami et al., 2020). These rules are usually available for all users of the DQ tool. It can be beneficial to subdivide the rule repository and define DQ rules for specific functional domains (e.g., for material data) (Blechinger et al., 2010).

4.4. Non-Functional Layer

On the non-functional layer, we describe architectural considerations and aspects that are important for the success of DQ tools. These shape the general functioning of the tools and set boundaries and considerations for the functionalities of the remaining architectural layers.

Interoperability In a decentralized data architecture, stakeholders and users from different domains must have access to data management tools. A DQ tool must be interoperable and applicable to different organizational and technical contexts (Altendeitering, Dübler, et al., 2022). To realize this requirement, a DQ tool can be offered in a self-service application marketplace that supports simple deployment and customization (Gröger, 2021).

Ecosystem Support The proliferation of decentralized data management and data ecosystems raised new requirements for DQ tools (Altendeitering, Dübler, et al., 2022; Gröger, 2021). To operate in the new environment, a DQ tool must integrate with ecosystem-specific technologies, such as data space connectors, using suitable interfaces (*F5*) (Altendeitering, Pampus, et al., 2022). Moreover, to provide data consumers with quality information, the DQ tool should adapt existing metadata models and protocols (*D3*).

Process Integration The need for interoperability (*G1*) requires DQ tools to seamlessly integrate with the business processes of various data providers and consumers (Swami et al., 2020). Moreover, integrating

DQ tools with established data management processes is necessary to follow up on low DQ results and fix underlying data errors (Walter et al., 2022). For the efficient use of DQ tools, organizations must include a DQ perspective in their business process management and re-design efforts.

5. Evaluation

We evaluated our reference architecture qualitatively using the adapted ATAM introduced by Angelov et al. (2008) as a basis. The method adapts the well-established practice for evaluating concrete architectures by Kazman et al. (1998) to reference architectures. It consists of three phases and considers the unique requirements for evaluating reference architectures, such as their general applicability.

Phase 1. The first phase identifies the stakeholders relevant to the evaluation process, representing potential users of the software reference architecture. The evaluation procedure combines scientific (phase 2) and organizational (phase 3) feedback, which we gathered in two focus group discussions. A focus group discussion is well-suited for gaining in-depth insights into a phenomenon under investigation and can spark discussions among the participants (Hollander, 2004). The first discussion involved three researchers working on data management and DQ and lasted 90 minutes. The second discussion comprised six professionals working in consultancy and the public domain who experienced the downsides of established DQ tools regarding scalability and usability and shared a vision for a new kind of DQ tool. The professionals worked in solution architecture, data management, or data science roles and have longtime experience in DQ and its tooling. The second meeting lasted three hours.

Phases 2a and b. In the second phase, we elicited the quality attributes of a DQ reference architecture and used these to evaluate our proposed solution. Angelov et al. (2008) differentiate between system and architectural qualities a reference architecture must fulfill. During the SLR, we identified two system qualities relevant to DQ tools: *integrability* and *interoperability*. To determine the architectural qualities, we used Kazman et al. (1998) as a basis and identified *security, flexibility, performance* and *modifiability* as relevant qualities.

We discussed the identified qualities and potential architectural approaches with the first focus group. For this purpose, we defined each quality among the participants and discussed architectural considerations and implementations. In subsequent discussions, we clarified the potential impacts on other architectural components. For example, we positioned the interfaces component outside the *Functional* layer to allow for better integrability and offer the functionalities of the DQ tool to external *Services*. We also highlighted this aspect in the *Users* layer of the architecture. Moreover, one researcher highlighted the need for simple DQ scores to support the interoperability of DQ tools and results. We added this aspect to the explanation component on the *Interaction* layer. To fulfill the need for security, we decoupled the *Data Sources* from the DQ tool, ensuring that data access is limited to quality analyses.

Phase 3. The third phase adds an organizational perspective to the reference architecture by verifying and extending the results from phase two. To gain organizational insights, we conducted another focus group discussion. We initiated the focus group discussion by presenting the current status of our reference architecture and describing its use in two exemplary scenarios. The first scenario encompasses the generation of DQ rules from master data sets, and the second scenario is about DQ measurement on data streams. Subsequently, we asked the experts for feedback on our reference architecture's functional composition and architectural style.

Overall, the proposed reference architecture was well received by the participants, and we received only minor feedback. One participant advised us to decouple the DQ algorithms from the DQ functionalities as this would support the modifiability of the tool. For instance, conducting DQ measurements might require different algorithms based on the data sets used and changing performance requirements. Several other participants highlighted the need to include the delivery mode of a DQ tool. Tallied with current research, we envisioned the tools to be available from an internal or external application marketplace to support its integrability and availability. To highlight this aspect, we included an *Application Marketplace* layer in the architecture.

Since we received only minor feedback in phase three, we concluded the evaluation process. A more detailed evaluation, including additional participants and application scenarios, is part of future work. Figure 1 shows the final version of the reference architecture, offering a consolidated view of the scientific and practical insights of DQ tools.

6. Conclusion

The ongoing digital transformation causes changes in the technological and organizational environment in which DQ tools operate, thus raising the need for new tools. At the same time, little knowledge about the

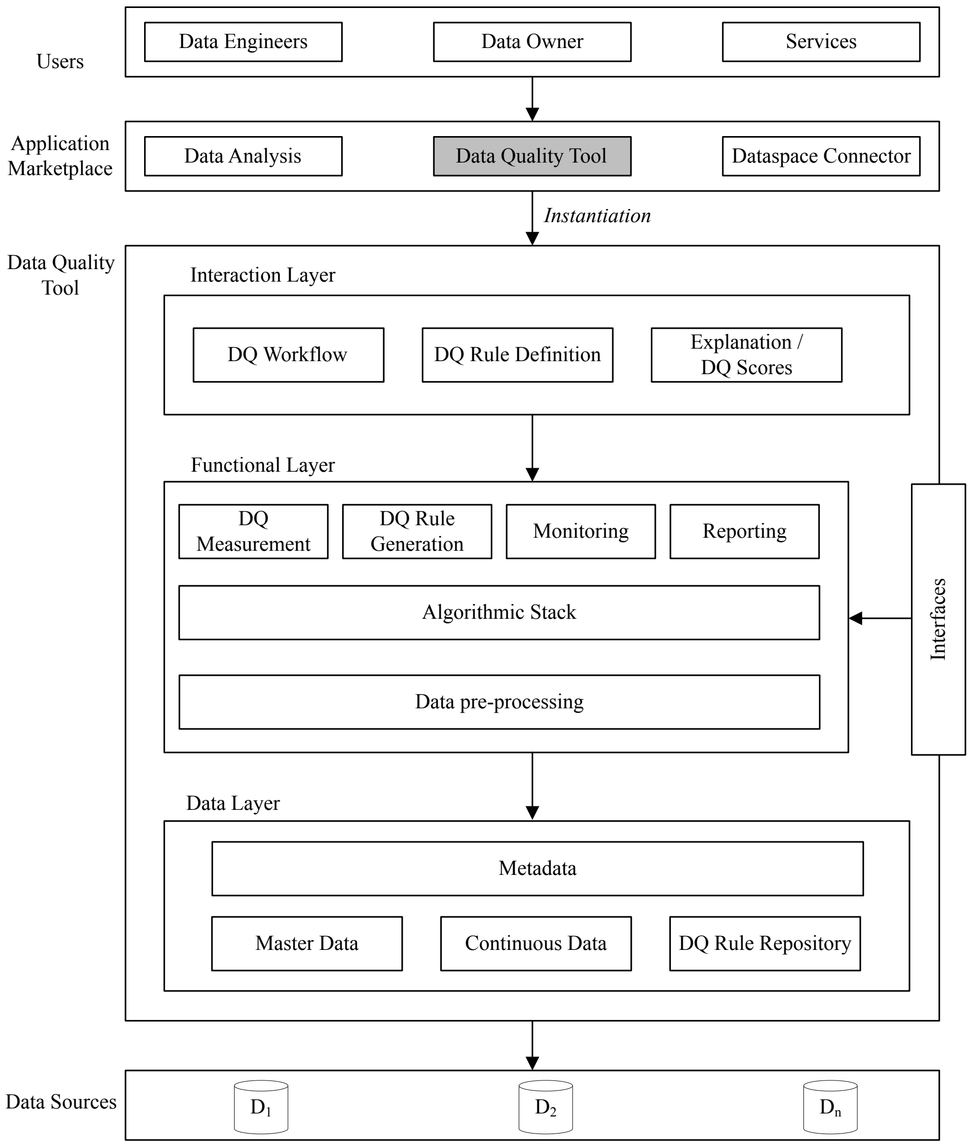


Figure 1. Proposed Software Reference Architecture for Data Quality Tools

functional composition and architectural design of DQ tools is available. This leads to a high degree of uncertainty around their development and deteriorated data initiatives. To address this research gap, we propose a reference architecture for DQ tools consisting of 16 architectural patterns on four architectural layers. We based our findings on a review and comparative analysis of ten concrete DQ tool architectures, which we identified in an SLR. Our proposed reference architecture incorporates a vision for the future and addresses topics that likely become more important in the future, such as the prevalence of data streams.

Our research offers the following *managerial contributions*. Most importantly, our proposed reference architecture guides software architects and DQ experts in designing high-quality solutions and can 'safeguard' DQ implementation projects (Angelov et al., 2012). The reference architecture offers a comprehensive and holistic understanding of DQ tools that lowers the uncertainties involved in DQ initiatives. Moreover, practitioners can use the reference architecture to inform make-or-buy decisions and systematically assess tools available in the market.

Besides managerial contributions, our work has several *scientific implications*. First, by following a rigorous research method, we created a sound reference architecture that extends the existing body of literature on DQ. Second, our work can act as a basis for developing research agendas and guide the future development of the different architectural components of DQ tools. Third, scientists can use the reference architecture as a conceptual framework for classifying DQ tools and systematically analyzing their capabilities.

Despite applying a high level of rigor, our research is subject to multiple *limitations*, which offer paths for *future work*. We cannot rule out subjectivity during the SLR and the qualitative evaluation of the reference architecture. Our selection criteria for including architectural elements in the reference architecture can be challenged, and other researchers might reach other conclusions. Moreover, the evaluation can and should be extended with professionals working in different roles and industries. In future work, we plan to conduct a more exhaustive review of DQ tools and apply the reference architecture to real-world implementation projects to evaluate its usability and validity. Furthermore, DQ tools will likely evolve, and new requirements and functionalities will emerge. The reference architecture should thus be critically scrutinized and updated to new developments regularly.

Acknowledgment

This research was partly supported by the German Federal Ministry of Education and Research (BMBF) and the research project "FAIR Data Spaces" (FAIRDS).

References

Alhamadi, M., Alghamdi, O., Clinch, S., & Vigo, M. (2022). Data quality, mismatched expectations, and moving requirements: The challenges of user-centred dashboard design. *Nordic Human-Computer Interaction Conference*. https://doi.org/10.1145/3546155.3546708

Altendeitering, M., Dübler, S., & Guggenberger, T. (2022). Data quality in data ecosystems: Towards a design theory. *AMCIS 2022 Research Papers*.

Altendeitering, M., & Guggenberger, T. M. (2021). Designing data quality tools: Findings from an action design research project at boehringer ingelheim. *ECIS 2021 Research Papers*.

Altendeitering, M., Pampus, J., Larrinaga, F., Legaristi, J., & Howar, F. (2022). Data sovereignty for ai pipelines: Lessons learned from an industrial project at mondragon corporation. *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*, 193–204. https://doi.org/10.1145/3522664.3528593

Altendeitering, M., & Tomczyk, M. (2022). A functional taxonomy of data quality tools: Insights from science and practice. *Wirtschaftsinformatik 2022 Proceedings*.

Angelov, S., Grefen, P., & Greefhorst, D. (2012). A framework for analysis and design of software reference architectures. *Information and Software Technology*, *54*(4), 417–431. https://doi.org/10.1016/j.infsof.2011.11.009

Angelov, S., Trienekens, J. J., & Grefen, P. (2008). Towards a method for the evaluation of reference architectures: Experiences from a case. *Software Architecture: Second European Conference, ECSA*, 225–240. https://doi.org/10.1007/978-3-540-88030-1_17

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley Professional. https://doi.org/10.1109/ICECCS.1998.706657

Blechinger, J., Lauterwald, F., & Lenz, R. (2010). Supporting the production of high-quality data in concurrent plant engineering using a metadatarepository. *AMCIS 2010 Proceedings*.

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., & Bone, M. (2010). The concept of reference architectures. *Systems Engineering*, *13*(1), 14–27. https://doi.org/10.1002/sys.20129

Dehghani, Z. (2019). How to move beyond a monolithic data lake to a distributed data mesh [Accessed: 06.05.2023]. https://martinfowler.com/articles/data-monolith-to-mesh.html

Ehrlinger, L., & Wöß, W. (2022). A survey of data quality measurement and monitoring tools. *Frontiers in big data*, 28. https://doi.org/10.3389/fdata.2022.850611

European Union. (2021). Proposal for an artifical intelligence act [Accessed: 21.05.2023]. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%5C%3A52021PC0206

Geisler, S., Vidal, M.-E., Cappiello, C., Lóscio, B. F., Gal, A., Jarke, M., Lenzerini, M., Missier, P., Otto, B., Paja, E., et al. (2021).

Knowledge-driven data ecosystems toward data transparency. *ACM Journal of Data and Information Quality (JDIQ)*, *14*(1), 1–12. https://doi.org/10.1145/3467022

Gerloff, C., & Cleophas, C. (2017). Excavating the treasure of iot data: An architecture to empower rapid data analytics for predictive maintenance of connected vehicles. *ICIS 2017 Proceedings*.

Great Expectations. (2022). Great expectations [Accessed: 26.05.2023]. https://greatexpectations.io/

Gröger, C. (2021). There is no ai without data. *Communications of the ACM*, *64*(11), 98–108. https://doi.org/10.1145/3448247

Guggenberger, T. M., Möller, F., Boualouch, K., & Otto, B. (2020). Towards a unifying understanding of digital business models. *PACIS 2020 Proceedings*.

Hasan, M., & Legner, C. (2023). Understanding data products: Motivations, definition, and categories. *ECIS 2023 Proceedings*.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625

Hollander, J. A. (2004). The social contexts of focus groups. *Journal of contemporary ethnography*, *33*(5), 602–637. https://doi.org/10.1177/0891241604266988

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998). The architecture tradeoff analysis method. *Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems*, 68–78.

Kuhrmann, M., Fernández, D. M., & Daneva, M. (2017). On the pragmatic design of literature studies in software engineering: An experience-based guideline. *Empirical software engineering*, *22*(6), 2852–2891. https://doi.org/10.1007/s10664-016-9492-y

Legner, C., Pentek, T., & Otto, B. (2020). Accumulating design knowledge with reference models: Insights from 12 years' research into data management. *Journal of the Association for Information Systems*, *21*(3), 2. https://doi.org/10.17705/1jais.00618

Madnick, S. E., Wang, R. Y., Lee, Y. W., & Zhu, H. (2009). Overview and framework for data and information quality research. *Journal of data and information quality (JDIQ)*, *1*(1), 1–22. https://doi.org/10.1145/1515693.1516680

Madnick, S. E., & Wang, R. (1992). Introduction to total data quality management (tdqm) research program. *Total Data Qual. Manag. Program MIT Sloan Sch. Manag*, *1*, 92.

Marjanovic, O., Cecez-Kecmanovic, D., & Vidgen, R. (2021). Algorithmic pollution: Making the invisible visible. *Journal of Information Technology*, *36*(4), 391–408. https://doi.org/10.1177/02683962211010356

Olbrich, S. (2010). Warehousing and analyzing streaming data quality information. *AMCIS 2010 Proceedings*.

Pradhan, S. K., Heyn, H.-M., & Knauss, E. (2023). Identifying and managing data quality requirements: A design science study in the field of automated driving. *Software Quality Journal*, 1–48. https://doi.org/10.1007/s11219-023-09622-8

Redman, T. C. (2020). To improve data quality, start at the source [Accessed: 06.05.2023]. https://hbr.org/2020/02/to-improve-data-quality-start-at-the-source

Shankaranarayanan, G., & Blake, R. (2017). From content to context: The evolution and growth of data quality research. *Journal of Data and Information Quality (JDIQ)*, *8*(2), 1–28. https://doi.org/10.1145/2996198

Swami, A., Vasudevan, S., & Huyn, J. (2020). Data sentinel: A declarative production-scale data validation platform. *36th International Conference on Data Engineering (ICDE)*, 1579–1590. https://doi.org/10.1109/ICDE48307.2020.00140

Walter, V., Gyoery, A., & Legner, C. (2022). Deploying machine learning based data quality controls–design principles and insights from the field. *Wirtschaftsinformatik 2022 Proceedings*.

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, *12*(4), 5–33. https://doi.org/10.1080/07421222.1996.11518099

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, *26*(2), xiii–xxiii. https://doi.org/10.2307/4132319

Westin, S., & Sein, M. (2015). The design and emergence of a data/information quality system. *Scandinavian Journal of Information Systems*, *27*(1), 3–26.