

Practical Problems in Customer Data - A Use-Case-Driven Classification



Jan-Lucas Deinhard ^{1,2} and Richard Lenz ¹


Abstract: This article presents a comprehensive analysis of data quality issues encountered in customer data at large enterprises. This analysis is based on data collected at a large medical technology manufacturer, and the problems observed there are clustered into distinct classes. Through this classification, nine key prevention requirements can be identified which are essential for improving data fitness. These include changes to data governance and to data architecture, among others. An evaluation of existing tools against these requirements furthermore highlights notable solutions. Despite the availability of numerous tools, gaps remain, especially regarding integration of all functionalities. Our findings suggest that while industry-standard solutions are accessible, integrating them into a cohesive framework posed significant challenges in our use case, necessitating continual adjustments to data architecture and processes to enable and maintain high quality of data.

Keywords: Data Quality, Data Architecture, Industry Case Study

1 Introduction

Research about *data quality* (DQ) often centers on generic descriptions of the different ways in which data can show defects, such as correctness or timeliness. This categorization of problems into larger classes is crucial, but omits an important factor: Many of the problems observed in large data architectures, in particular at large international enterprises, are mere symptoms of an underlying problem. The root-cause may lie deeply hidden in the architecture, or consist of a defective business process, and might not be immediately obvious. Yet, repairing this root-cause may fix a number of symptoms permanently. This relationship between architecture, root-cause analysis and low data quality at the data-consumer-side is often not adequately captured in state-of-the-art DQ analyses. Once the underlying root-causes are identified, it often becomes a simple matter to increase DQ sustainably by selecting the appropriate tool to prevent any further issues. Usually, the market already offers such tools. With this paper, we aim to build on our experiences from an industry case study at *Siemens Healthineers* (SHS), a large German manufacturer of medical technologies, to develop such a architecture-focused and root-cause-centered understanding of DQ. SHS provided access to their operational *Customer Relationship Management* (CRM) system for our research, so we focused mainly on DQ in CRM data. It stands to reason that other

¹ Friedrich-Alexander-University Erlangen-Nuremberg, Chair for Data Management, Martensstraße 3, 91058 Erlangen, Germany, jan-lucas.deinhard@fau.de,  <https://orcid.org/0009-0005-5235-9271>; richard.lenz@fau.de,  <https://orcid.org/0000-0003-1551-4824>

² Siemens Healthineers AG, Commercial Excellence Department, Siemensstraße 3, 91301 Forchheim, Germany, jan-lucas.deinhard@fau.de,  <https://orcid.org/0009-0005-5235-9271>

data infrastructure systems show similar patterns of problems and problem clusters as those which will be outlined herein, especially if they share the core characteristic of human data sources and data sinks with the system available at SHS. With that, this paper is primarily focused on answering three questions:

- **Research Question R1:** *To which extent can data problems in CRM data and their root-causes be clustered into generic problem classes?*
- **Research Question R2:** *Which tools and technologies exist as industry standard for such classes of data problems?*
- **Research Question R3:** *Are there any classes of problems which are difficult to address with existing standard industry solutions?*

A data problem in this context is defined as diminished *fitness for use*, i.e. as an aspect of the data which affects the information consumer and which prevents them from accomplishing their intended goal with the data[25]. A class of data problems clusters multiple data problems of the same sort into a more abstractly defined class and will be sufficiently generic that it applies to a broader context outside of the SHS case as well.

This paper approaches these questions through a case study of the data infrastructure built around CRM data and reporting at SHS. We examine **R1** based on a long-term observation and documentation of the data problems at SHS. These results are captured, integrated and evaluated, yielding different categories of problems. Those can be interpreted as problem classes. For those data problems, the solutions eventually adopted by the data teams at SHS are also evaluated and clustered, which allows us to compose a list of prevention measures. Building on that list of prevention measures, this paper reviews tools commonly available for those measures as an industry standard, thereby answering **R2**. Any identified blind-spots will allow us to answer **R3**.

2 Literature Review

The topic of DQ is commonly understood by researchers in terms of *fitness for use*[25]. In this landmark 1998 publication, the authors outline several concepts which are today taken as cornerstones when approaching the topic, one of those being *fitness for use* as the defining characteristic for the quality of an information asset. Other important concepts are the perspective of information manufacturing as analogous to the manufacturing of products, and the *Total Data Quality Management* (TDQM) approach for improving DQ, which is based on these understandings of data and information quality. This information manufacturing perspective sees information or also the data assets as the result of a production chain which starts with raw data, processes this through the information systems, and arrives at information products at the end. Such a description takes a first step towards emphasizing the root causes leading to low DQ, which are hidden in the data production process. The *fitness for use* idea encourages capturing DQ issues close to the data consumer, which can

then be further clustered into larger domains and classes of problem areas. A first attempt at this was made in [23] and [26], which both precede many of the above publications but already had adopted these core concepts. Both publications aim at identifying what they call *DQ Dimensions* [23]. The latter publication builds on the former and shows that an understanding of DQ broader than just in accuracy terms helps businesses better understand and improve business outcomes. Both publications come to adopt a similar taxonomy of DQ Dimensions. The authors in [26] break DQ into the areas of Intrinsic DQ, Contextual DQ, Representational DQ and Accessibility DQ, which they then refine further into sub-categories such as Completeness, Timeliness, Accuracy, Consistency, etc. In [23], the authors distinguish an internal view of DQ which is related to design and operations, and an external view of DQ related to data use and value. Both classes further break down into data-related problems and system-related problems. These categories then split into similar detail items as in [26], with the most important ones being accuracy, completeness, consistency, currency, relevance, reliability and timeliness. Since then, other DQ Dimensions have been proposed by multiple authors. A more recent survey compiled a comprehensive list of the accepted dimensions and their use throughout literature[24]. The authors identify 21 relevant dimensions which are used, not always under the same name, to cluster DQ. For each dimension, they list the publications where they were used. Accuracy, completeness, relevance and timeliness remain the most frequently used dimensions. These same dimensions can also be confirmed to be applicable for the currently important field of big data, where similar dimensions are encountered and retain their relevance[5]. Building on this, [8] introduced the information-theoretic *Source-Bearer-Receiver* model (S-B-R model) for describing DQ. They outline the difference between problems of low DQ, which they see as inherent to the data, and problems of low *information quality* (IQ), which is inherent to information instead. The former is related to problems within the bearer system, whereas the latter is caused by the quality of the links between the source and bearer, or the bearer and receiver. These publications propose at least a thematic clustering of DQ in terms of effects on the data consumer, which suggests a positive answer to **RI** is a likely outcome. Their approach however neglects to connect clusters of DQ problems to their origins within the data environment where they occur, making it challenging to identify suitable technologies and remedies to combat these problems.

There have been other studies trying to characterize and categorize common problems with DQ empirically. These usually start from individual case studies at specific organizations and track concrete problems occurring there over a period of time. For example, one study investigates DQ at a specific Australian Information Services consulting firm and record the errors encountered there. The authors classify these errors into error types and also analyze the impact on operations. The main error types identified in the article are data entry errors and delays in data processing[21]. Another study examines the causes of insufficient DQ in medical registries and determines that the main source problems lie in poor setup and organization of the registry itself, in problems with the data collection process, and in difficulties with the quality improvement approach[1]. Roots of problematic quality of data frequently are hidden in the *Information Technology* (IT) infrastructure or in business

practices at large organizations. A recent survey suggests five core facets of DQ assessment at large companies: the data facet, the source of the data, the (physical) system, the human and the task facet[17]. The authors identify specific challenges and opportunities associated with each of these facets. Each of the facets consequently also represents a potential source for DQ problems. In spite of the well-established costs of defective data, many organizations still hesitate to invest into improvements. In a 2010 study, [7] examines the reasons why this is the case. They conducted questionnaires at several companies and asked employees about their experience specifically of master data at their workplace. Their literature review and the questionnaire findings imply that there are five major reasons why DQ in master data is not remedied and improved: a lack of the delegation of responsibilities for master data, a lack of master data control routines, missing employee competencies, low usability of the tools used to manage master data and missing rewards for ensuring high quality in master data. These five barriers are organized in descending manner by their relevance as identified through the questionnaires. Even with these studies identifying large problem clusters empirically, we still observe a relative scarcity of literature connecting the problems to the underlying technological and procedural reasons, or to any solutions. With this case study, we aim to examine those aspects in more detail.

3 Case Study at Siemens Healthineers

Research questions **R1**, **R2** and **R3** as described in Section 1 were examined using a case study at the *Commercial Excellence* (CE) department at SHS. SHS is a large manufacturer of medical technologies based in Germany, with business all across the world. Their CE department is responsible for enabling efficient global sales operations. To do this, among other things CE provides a data platform known as the *CRM Data Cloud* (CDC), which provides data products as defined by [4] in support of sales managers and analysts, as well as for corporate management. The CDC is currently in the transition process towards a data mesh approach where their offering will be based around data products. Notwithstanding this transition phase, new DQ problems continue to be reported from the side of the data consumers at a persistent rate.

Sales organizations at SHS are distributed across all countries where SHS is active, and country sales organizations are grouped into larger zones and regions. Production is based around a number of different business lines, and country sales organizations coordinate the customer order from the business lines. The headquarter provides corporate functions such as the finance, CE or IT organizations. All country sales organizations, business lines and corporate functions are afforded a great degree of autonomy. This autonomy also extends to data ownership and governance and thereby affects global information quality at SHS. The CDC focuses on sales data, but also incorporates data elements from finance and the service organizations. The sales data originates in the global CRM system which is used by all countries. The data processing from the source system towards the analytics layer in the CDC follows a standardized process: The source systems replicates the raw data to a global

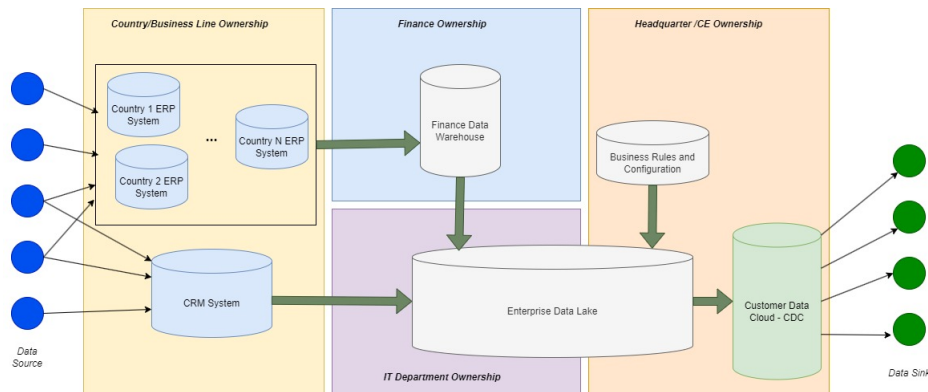


Figure 1: Architecture setup in case study at SHS

data lake in fixed intervals multiple times each day, and the CDC performs a full data load from there, applying preprocessing and business logics before making the data available to data consumers in the form of data products. Finance data follows a slightly different path as the source systems are not unified into one overarching global system, but for historical reasons resides in multiple different systems, each being used by different combinations of country organizations. Ownership lies with the global finance organizations, and the various source systems are first integrated by them before replicating this data to the data lake. From there, the CDC process further integrates this data with other data assets. Once all data assets are integrated and packaged as data products on the CDC, data consumers typically connect reporting tools and dashboards directly. For many problems, especially those related to data semantics, this is the first stage along the data processing pipeline where quality issues are detected. Fig. 1 outlines this setup.

This brief overview of data infrastructure at SHS shows a technical setup as it is present at many large international enterprises. The data infrastructure is distributed, and so is governance and ownership. We will show that the observations at SHS can be generalized into clusters of DQ problems which are independent of this concrete infrastructure and require resolution and prevention methods which consequently also apply to other large companies with similar data architectures. The CE department has been actively tracking metrics to measure DQ and instituting technologies and processes to improve DQ in parallel to the build-up of the CDC. Since middle of 2023, these measures also included the systematic tracking of reports about lackluster quality of the information in the CDC, as experienced by business users. At the time of writing of this analysis, this means we have access to a collection of approximately 14 months worth of case reports which reflect the experience of business users documenting data not fit for use in the CDC. The case reports were captured in the form of two different schemas over time, as the collection is a merge of two different report collections. One schema is focused on describing problems in the form of symptoms and root causes of low DQ and tries to describe the trace of these

types of DQ issues across technical systems and processes. It also captures the resolution which is eventually used to solve the underlying problem. An overview of this schema is shown in Fig. 2. This simple schema can capture a wide array of DQ problems reported by business, and more importantly, relate observations and treatment across the complex system that is the data infrastructure at SHS. In this schema, DQ Symptoms can be observed by anyone working with the data, typically by business users or the data team. Root Causes are identified by the IT department and data teams. They represent the factors which often cause multiple DQ Symptoms, and which can be characterized as the initial problems which start clusters of observed symptoms. Resolutions are steps taken to alleviate Root Causes of problems. All three entities are largely described by a text formulation of the reported observation or action taken. The Process and Technical System entities describe where in the data ecosystem these observations were made, or where an action was taken. The second collection of case reports came from a source which had been purpose-built for reporting the status of DQ problems to the business community in an automated way. This schema captured reports in the form of a simple flat table with the following attributes:

- *Date Reported*: The date and time when the problem had been first reported or observed.
- *Status Type*: An indicator of the problem type which is causing the observation, either *Error* or *Change/Update*.
- *Title of the DQ Problem*: Descriptive title, chosen manually by the member of the data team to first work on the issue.
- *Description*: A text description of the issue, typically around three sentences.
- *Status*: The processing status, either *New*, *In Progress* or *Completed*.
- *Resolution*: A text description of the steps taken to resolve the issue.
- *Assigned to*: Reference to the member of the data team responsible for providing a resolution.
- *Data Product*: A reference to the affected data products.

The table also had additional columns to link the issue to other systems in use at SHS. To answer research question **R1**, a root cause analysis of the problem sources is of particular importance, we decided to map both case collections to the first schema as described in Fig. 2. This description approach is largely compatible with the second schema, and highly useful for understanding the problem source, as root causes are specifically captured as a distinct entity type. After storing all collected cases in the schema shown in Fig. 2, we were able to extract a view of the root causes and accepted resolutions, along with the symptoms observed by business users which prompted them to report low DQ.

The list below shows the collected cases of DQ problems which impacted fitness for use of the information in the CDC. The list entries are structured as follows:

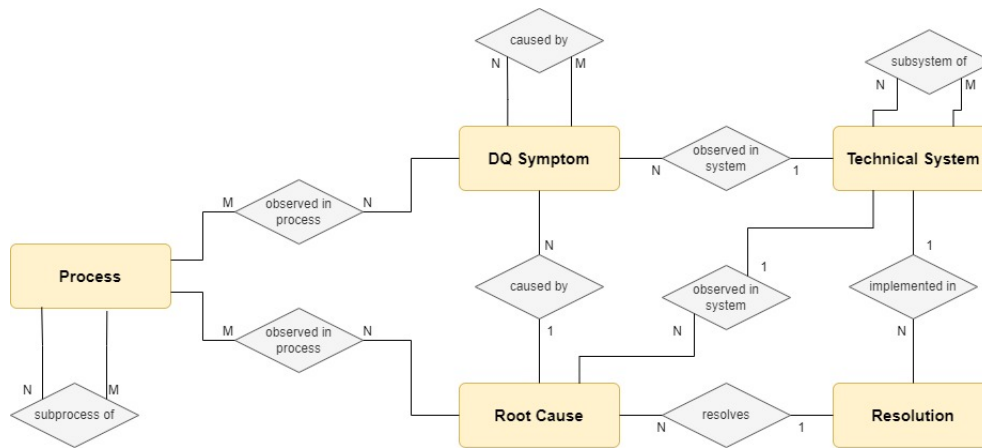


Figure 2: Entities and relations used in the schema at SHS to describe cases of observed low DQ.

- **<Unique Case Identifier>:** <DQ Symptom>. <Root Cause>. <Resolution>.
Affected DQ Dimension: <DQ Dimension Name>

These reports were collected in the time frame between June of 2023 to July 2024. We also included a categorization of the observed symptoms into the accepted DQ Dimensions[23][24]. The unique identifier will make it easier to refer to each case in the subsequent analyses. The subsequent analysis and clustering of problem sources, as well as the recommended tools and technologies to prevent such issues from affecting business users and decision making, are based on this case collection.

- **Ghost Flag Implementation:** This issue was first observed when the value and sold quantity did not match in BI reports. The root cause was that sales staff had entered sales opportunities in the source system which later turned out to be irrelevant but was not flagged as such. The resolution involved implementing multiple SQL rules for different business segments to flag irrelevant cases through business logic.
Affected DQ Dimension: *Consistency*
- **Change-related DQ Issue:** The issue was initially noticed when outdated data appeared in data products due to a reload failure. This problem arose due to the redesign of a central workflow, which removed a crucial row filter. To resolve this, the relevant row filter was added into the redesigned workflow, before merging the two relevant data sources.
Affected DQ Dimension: *Timeliness*
- **Master Workflow Skips Runs 1:** This was identified when outdated last-reload timestamps appeared in data products due to workflow aborts. Insufficient testing of an update to a core software component in the production environment had caused

this issue. The resolution involved rolling back changes in CDC workflows.

Affected DQ Dimension: *Timeliness*

- **Master Workflow Skips Runs 2:** The first symptom was again an outdated last-reload timestamp in data products due to workflow aborts. Sporadic server connectivity loss to the workflow execution server had caused the issue this time. It was resolved by manually re-executing the CDC master workflow.
Affected DQ Dimension: *Timeliness*
- **Revenue Duplications – Country Group CG_1 :** The issue was first observed when countries in CG_1 showed partial differences compared to a reference system. This occurred because the same data load pipeline from the finance data warehouse to the data lake was triggered twice without a delta check. It was resolved by manually purging the data and re-executing the load pipeline.
Affected DQ Dimension: *Consistency*
- **Revenue Duplications – Country Group CG_2 :** The first symptom was that countries in CG_2 showed no revenue, while a reference system reported revenue. The root cause was that finance organizations in CG_2 do not report revenue at the material number level due to policy. The resolution involved communicating the effects of this data policy in CG_2 and the CDC global filtering policy to all stakeholders.
Affected DQ Dimension: *Consistency*
- **No Fresh CRM Data in Data Lake 1:** This issue was first observed as an outdated maximum last-updated timestamp in data products. The root cause was a technical bug in the delta load script, likely due to a coding failure. It was resolved through a bug fix by the data lake team and the CRM platform team.
Affected DQ Dimension: *Timeliness*
- **Data Update Delayed:** The problem was first noticed as an outdated last-reload timestamp in data products due to workflow deadlocks. This issue arose from overburdening the data lake execution server with long-running queries. The resolution involved implementing separate execution contexts by the platform provider.
Affected DQ Dimension: *Timeliness*
- **Duplicates in Data Product P_1 :** The issue was first noticed when duplicates appeared in a data product. Root cause was a misconfigured ingestion pipeline by the CRM platform team. The resolution involved repairing the ingestion pipeline performing the data pre-processing for the CRM platform.
Affected DQ Dimension: *Uniqueness*
- **Duplicates in Data Product P_2 :** The issue was detected when duplicates appeared in a data product. A change in the ETL pipeline introduced a misconfigured joiner. This was resolved by correcting the joiner configuration in the CDC ETL pipeline.
Affected DQ Dimension: *Uniqueness*

- **Data Lake Servers Down:** The first symptom was an error message when connecting to the CDC database. The root cause was a software problem caused by a third-party vendor. It was resolved through an external software bug fix completed by the data lake platform team.
Affected DQ Dimension: *Accessibility*
- **Server Issue:** The issue was first noticed as an outdated last-reload timestamp in data products due to workflow aborts. Faulty time synchronization across regions caused authentication failures between two IT servers for several days. The IT department resolved this by correcting synchronization and manually reloading subsequent systems.
Affected DQ Dimension: *Timeliness*
- **No Fresh CRM Data in Data Lake 2:** The problem first manifested as an outdated last-reload timestamp in data products due to workflow aborts. The CRM-to-data lake ingestion job had been stuck in an infinite loop for several days. This was resolved by the data lake team, who terminated the job and restarted the ingestion process.
Affected DQ Dimension: *Timeliness*

We will be referring to this collection of DQ cases as List 3.

4 Clustering the Problem Classes

Starting from this collection of case reports, this paper aims to identify underlying patterns shared by many of the cases when *fitness for use* of the data is negatively impacted from the perspective of the information consumers at SHS. To accomplish this, we propose a clustering of the problems described in List 3 into larger problem classes. A class is defined abstractly in [9] as *a state of nature that governs the pattern generation process in some cases*. Building on this notion, this paper assigns problem classes which are detached from the concrete context at SHS and also generalize to similar patterns which can be observed in other companies and environments. While the classes should therefore be generic in nature, we at the same time try to make them sufficiently specific to still remain useful in discussing potential solutions. Tab. 1 lists the problem classes identified for the problems discussed above. The assigned problem class identifies the underlying issue in a description which is detached from SHS-specific concerns and can also be applied in other companies. In order to determine it, we proceeded as follows: First, we identified the root cause of the observed problem, i.e. the underlying problem which cannot be explained by defective upstream systems or processes. Then, we restated the root cause in general form by replacing all SHS- or industry-specific details by generic terms. After all problem classes had been assigned in this way, we also reviewed the outcome for closely linked classes another time. The resulting table also includes the measures deemed required for preventing the problem from affecting end users going forward, as identified by the data experts at SHS, and a clustering of those requirements into less specific prevention requirement classes.

Problem Class	Identifier	Prevention Requirement	Requirement Type
A-posteriori insufficiently defined business activity	Ghost Flag Implementation	Data provenance assessment at launch of data product.	Data provenance documentation
Computational resource constraints exceeded	Data Update Delayed	Metadata monitoring and proactive user notification in case of missing updates.	Data observability, automated notifications
Data standards not aligned with stakeholders	Revenue Duplications – Country Group CG_2	Documentation and proactive communication of data source governance practices to data stakeholders.	Data governance accessibility
External hardware outage	Master Workflow skips runs 2	Status monitoring of all architecture components and outage handling	Data observability, automated notifications
	Server Issue	Monitoring of workflow execution, proactive blocking of execution and notification of stewards.	Status-based jobs monitoring, automated notifications
Hardware integration problem	No fresh CRM data in data lake 2	Metadata tracking and monitoring of workflow execution, proactive blocking of execution and notification of stewards.	Data observability, automated notifications, status-based jobs monitoring
Implementation error.	Duplicates in data product P_1	Default duplicate check for keys along the data production chain.	Standardized data testing, metadata tagging
	No fresh CRM data in data lake 1	Metadata monitoring and proactive reload blocking and notification of key users.	Data observability, automated notifications, status-based jobs monitoring
Pipeline redesign errors	Change-related DQ Issue	Test environment, default duplicate check for keys.	Regression testing, standardized data testing, metadata tagging
	Duplicates in product P_2	Default duplicate check for keys along the data production chain.	Standardized data testing, metadata tagging
	Master Workflow skips runs 1	Test cases, Test Environment, components regression testing.	Regression testing
Manual process step error	Revenue Duplications – Country Group CG_1	Pipeline architecture which natively embeds delta check when reading in new data, identify and remove manual process steps.	Data observability, data architecture
Software integration issue	Data lake servers down	Integration testing process on Snowflake side, proactive user notification on SHS side.	Automated notifications

Table 1: Clustering into Problem Classes.

It stands out that the compression factor of this clustering is limited: 13 observed problems lead us to extract 9 different clusters. On the one hand, this suggests a broadly distributed nature of root causes of DQ problems at SHS. On the other hand, it prompted us to try and further generalize our observations. In our effort to approach a true taxonomy rather than just a list of issues, we noticed that the identified problem classes assigned in Tab. 1 seem to further fall into two very different categories: *Technical Problems* and *Business Problems*. This allows us to neatly categorize the DQ problems along the lines shown in Fig. 3. The classification into both classes is largely self-evident. Both classes can be defined as follows:

- *Technical Problem*: Cases where DQ problems were caused by a technical malfunction, outage or error by a technician.
- *Business Problem*: Cases where faulty, incomplete or missing business problems or communication led to low DQ.

This clustering can be considered an answer to **R1**. Based on observations at SHS, a number of recurring problem classes can be identified and further segmented into the large problem classes of *Technical Problems* and *Business Problems*. This taxonomy is based only on the data collected at SHS, and the limited scope of this available data set makes further quantitative assessment of the outcome challenging. Our findings can likely be further refined and extended by carrying out corresponding studies at other sufficiently large companies. As a last note on this topic, it is worth pointing out that when the symptoms of low DQ are clustered into the established DQ Dimensions as shown in List 3, an interesting correspondence between the DQ Dimensions and the categorization of *Technical Problems* and *Business Problems* in Fig. 3 arises: Symptoms caused by the *Business Problems* category seem to reliably lead to violations of the Consistency dimension, whereas symptoms related to *Technical*

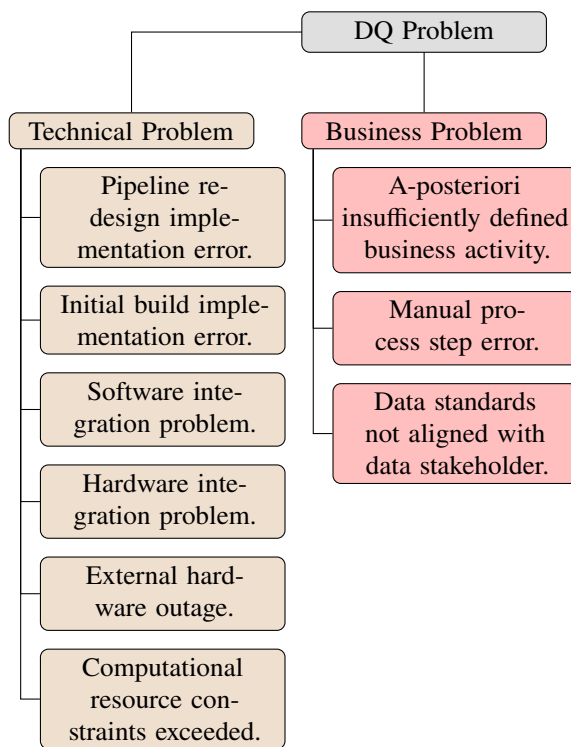


Figure 3: Taxonomy of DQ issues observed at SHS, categorized as Technical Problems and Business Problems.

Problems are associated with a wider range of DQ Dimensions and lead to violations in the domains of Accessibility, Timeliness and Uniqueness. The most-observed problem from a data consumer perspective was a Timeliness problem, although this is likely a consequence of the data architecture specifically at SHS and not indicative of a general trend.

5 From Clusters to Solutions

Many of the problems listed in List 3 already have well-established solutions, usually based around improvements to the data architecture. For the sake of compactness, the prevention measures which were identified at SHS are directly attached to Tab. 1, along with a categorization into classes of prevention measures which follow the understanding of classes which was also used in extracting the problem classes above. The measures listed in this publication were the result of expert review of the problems by the CE data team at SHS. Parts are already deployed, other measures are in the implementation phase. Many of the problems listed require the launch of several different solutions to comprehensively remedy the underlying defect. The overall set of prevention requirement classes however can be distilled to nine items. The following is a brief description and discussion of these items and why they were identified:

Automated Notifications: There is a strong need for a tool or framework to automatically notify specific user groups of technical status anywhere along the data production process. Solutions for this exist and cover surveillance of parts of the data production process, as well as notification of parts of the set of all users, but definition of any additional check is usually a manual process, and no comprehensive tool exists. This is of particular importance in the case of technical problems which cannot be foreseen and therefore need a comprehensive reporting concept.

Data Architecture: Choosing the right data architecture can remedy several problems. There is plenty of literature discussing different data architectures [2][14][12][19] and data warehouse architectures [10][16]. Two key requirements to data architecture in the case at SHS were simple in nature: Remove all manual processing steps from the overall data architecture and select an architecture built around change-data-capture[20]. Both features can prevent unforeseeable errors springing from human error.

Data Governance Accessibility: Data governance efforts do exist at SHS and impact many segments of the data production process. The standards set by governance teams however seem to be inaccessible to many data consumers and especially to potential data consumers who perform exploratory data analysis with new use cases in mind. Documentation is distributed and access to the guidelines often is synonymous with access to the governance experts. A wholistic and easily accessible data governance platform which is targeted at non-expert data consumers across the organization and technical teams alike can prevent the business from mistakenly committing to investment based on faulty expectations about data and data quality.

Data Observability: Among the most critical prevention requirements identified by us, data observability as understood by [18] has the potential to remedy both technical and business problems. Data observability refers to tools which can monitor the data production process and visualize the current status in an easily accessible way. As such, data observability is closely linked to data lineage, metadata tracking and status monitoring of data processing workflows. Full data observability hence relies on an already highly sophisticated data stack. With these aspects available however, data observability tools make the results available to a broad audience at low latencies and thereby contribute to a broad understanding across the organization of what data can do, and what the current status of the data products is. Many unforeseeable errors and outages can be reported directly to the affected users in near-real-time, mitigating the impact.

Data Provenance Documentation: Similar to data governance, our observations also indicate the need for a comprehensive and accessible documentation of the origin of data. In most data architectures, data provenance as defined in [6] is partially documented and often closely tied to data lineage[15]. At SHS, documentation is fractured among multiple teams, and occasionally does not exist in any explicit form but depends on data or domain experts. A system which makes the data provenance fully transparent to use-case owners in business functions can prevent them from relying on data for decision-making which might be available but is not fit for use when contextualized.

Metadata Tagging: This requirement refers to the need for a structured metadata repository of the data products available to the data teams, where the most important data attributes are tagged with respect to important properties, e.g. as primary keys or also as personally identifiable data. This in combination with the standardized data testing requirement allows the default checking of primary key fields for duplicates, leading among other things to an early detection of cases of duplications.

Regression Testing: This analysis technique stores inputs and outputs to production software components as test cases and runs these test cases on the software component in case of any change to the component, triggering an alert if the actual output deviates from the stored expected output[22]. It is a well-established and popular dynamic testing technique in the field of software development, and holds particular appeal for developing data processing software, where snapshots of data input frames along with the expected data output can be captured and stored quite easily. If an automated regression testing framework were in use in the various software development teams, the CI/CD (Continuous Integration/Continuous Development) process could detect damaged software much more easily and ideally prevent the deployment of those components altogether.

Standardized Data Testing: A standardized data testing requirement refers to the need for a framework which allows the low-effort definition and execution of simple tests on data as it is being processed through the IT systems. In the observations at SHS, very simple tests such as row count continuity over time can already detect most issues. Usually the problems uncovered by this technique have technical origins, mostly implementation errors

during software design or refactoring. If data engineers had access to a tool which allows them to set up a number of standardized tests on the data when initially creating the data pipeline, and these tests were evaluated every time when the data is reloaded, many of these implementation error could be caught before they ever could affect data consumers. Such a tool could be combined with regression testing into a testing framework, saving data engineers a lot of time which they now spend manually setting up such tests.

Status-based Jobs Monitoring: The above observations at SHS also suggest the need for a comprehensive monitoring solution which tracks data processing jobs based on their status, and can potentially handle any failures or interrupts. Similar to some of the items mentioned before, this already exists in part for certain segments of the data infrastructure. A comprehensive solution from data source to data sink however is not available. Also, many of the solutions currently in place focus on reporting of job statuses and do not have the capability of handling failed processes safely. Extending the data stack with a reliable tool to do that is one of the few options which can prevent random server outages occurring outside of the organization from impacting data consumers in the organization.

All of the prevention methods listed already exist as industry solutions. A discussion of suitable tools and providers will follow in the next section of this paper. However, none of the available options have so far been implemented at SHS. This is fundamentally due to two reasons: On the one hand, the data architecture at SHS is a typical evolved data architecture and other, more foundational problems took up the spotlight until this point. As a consequence, many ad-hoc partial solutions were deployed and are in operation until now, treating a large part of the DQ problems which typically arise, but not in a comprehensive way. These solutions have already decreased the impact of the various technical and business problems in the data processing pipelines on information consumers to an acceptable level. When considering how to further improve DQ, this is always in reference to a status quo where most of the easy and high-impact challenges have already been confronted and solved. As a direct consequence of the interplay of these different and uncoordinated ad-hoc solutions however, many of the leftover disruptions to the day-to-day work with the data and information cannot be repaired easily, which often impairs especially the discovery of new use cases and the ability to scale up. On the other hand, the data processing infrastructure is not in the scope of a single team at SHS. While this might be different in some other companies, it stands to reason that in most sufficiently large organizations, data architecture is sufficiently complex to demand distributed responsibilities, and the process of scaling up will necessitate the integration of a multitude of historically evolved ownership and management structures. This means different teams with different goals need to grow together and align on a shared strategy before any comprehensive solution can be planned or deployed. SHS is currently approaching a point where both the architecture and ownership structures are sufficiently mature and extensive so that the search for comprehensive solutions makes sense. This is indicated by the emergence of a common data quality strategy as a collaborative project in the data architecture community. The solutions which have been proposed above in particular require the collaboration of the CE data team which works

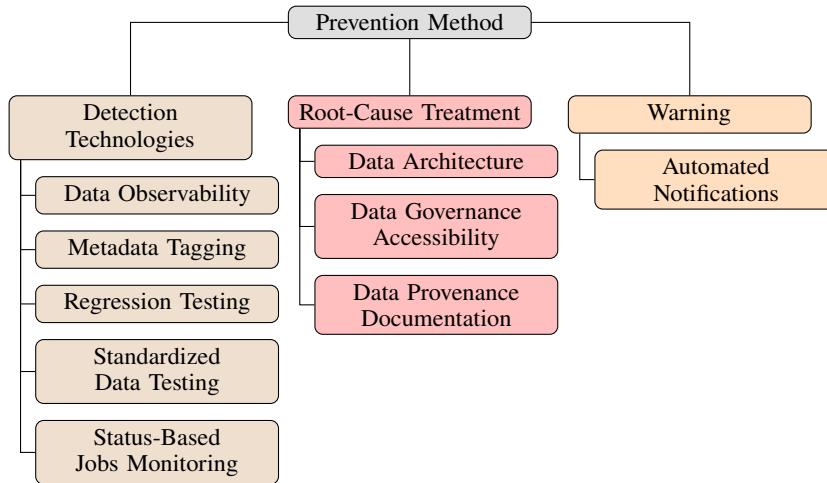


Figure 4: Categorization of Prevention methods.

together with the SHS business community to identify use cases, the IT team responsible for the data lake, and the teams of the various source systems.

Interestingly, the prevention requirement classes group into larger established categories in a way parallel to the clustering of problem classes into the taxonomy shown in Fig. 3. The different classes align along categories which most data engineers and architects would likely naturally understand: Some items are focused on *Detection* of DQ problems, others on *Warning* the data consumer of potentially defective data, and others on *Root-Cause Treatment*. These superordinate classes may be defined in the following way for our purposes:

- *Detection Technologies*: These prevention requirement classes enable the automated finding of defective data as early in the data processing architecture as possible, ideally near to the root cause of the problem. They can be followed by *Warning* solutions to notify key stakeholders.
- *Root-Cause Treatment*: Classes in this category are intended to fix the underlying issue which leads to the defective data in the first place. They will typically target an organizational process or piece of technology.
- *Warning*: These items are aimed at providing an easy method to warn users of broken data somewhere in the data production process, ideally before it impacts their work or information products.

Note that the distinction between the *Detection Technologies* and *Warning* classes consists mainly in the active or passive mode of reporting. The *Detection* methods help data engineers find problems once they have been identified, whereas *Warning* methods proactively alert the data engineering team. Fig. 4 shows how the specified prevention types cluster into

these classes. This clustering also allows one final observation when relating the *Detection Technologies*, *Root-Cause Treatment* and *Warning* classes in Fig. 4 back to the problem classes outlined in Fig. 3. Grouping these problem classes and their associated types (*Business Problem* or *Technical Problem*) by the type of prevention requirement selected by SHS to remedy the problem shows that *Root-Cause Treatments* are only really deemed a requirement for the case of problems identified as *Business Problem*. A majority of these problems requires steps taken to remedy the problem at source, whereas the great majority of problems associated with the *Technical Problem* type can be prevented from affecting users simply through adequate detection and warning prevention. Tab. 2 shows this effect. It charts the resolution used to solve problem types at SHS CE department, along with count of individual problem classes which called for a prevention requirement of that type. Technical problems usually require warning or detection technologies, whereas business problems usually need a root-cause treatment. Solutions targeting the root cause of observed

<i>Resolutions needed to treat different problem types.</i>	Problem Type: Business Problem	Problem Type: Technical Problem
Detection Technologies	1	6
Root-Cause Treatment	3	0
Warning	0	5

Table 2: Prevention Requirement Types.

low *fitness for use* are typically more expensive both in terms of time and resources, and this analysis suggests that organizations can save on their DQ budget by understanding whether problems originate in problematic business processes (*Business Problem* type) or whether the source lies in the technical infrastructure (*Technical Problem* type). In the latter case, a comparatively cheaper detect-and-flag approach, combined with early warning of downstream product owners, could suffice.

6 Existing Solutions and Open Questions

In the following, we will examine which solutions exist for the problem classes which have been outlined in the previous section. Note that we do not claim to provide a comprehensive review of all tools on the market. We only intend to outline standard tools which can be used to address certain classes of problems which have been identified and yielded critical requirements at SHS. These requirements which were described in the previous sections are usually not directly part of the publicly available marketing materials for these tools in the wording chosen for this article, so in order to identify a viable set of tools we decided to

first map them to common functionalities which DQ tools offer, then identify tools which provide these functionalities, and finally determine through a closer inspection whether the selected tool really satisfies our requirement. We also decided to treat the question of appropriate data architecture as an exception, since data architecture insufficiencies are not related to the available tooling. Data architecture requirements will be discussed further below. The requirements were mapped to tool functionalities as follows:

- *Automated Data Testing* for *Automated Notifications*, *Standardized Data Testing* and *Regression Testing*.
- *Data Governance Documentation* for *Data Provenance Documentation* and *Data Governance Accessibility*.
- *Data Observability* for *Data Observability*, no mapping needed.
- *ETL Pipeline Monitoring* for *Status-Based Jobs Monitoring*.
- *Metadata Management* for *Metadata Tagging*.

For each of these functionalities, a large online search engine was fed with the search term functionality + ' tools', and the top-3 pages conformant to our non-exclusion criteria were used to obtain a list of viable tools available on the market for these functionalities. The non-exclusion criteria were a set of conditions identified by us prior to conducting the online search in order to establish relevancy of the compiled results. Relevancy was defined as all tools found on web pages which

- (a) were accessible without a subscription or other form of payment.
- (b) were not sponsored content.
- (c) carried a timestamp from the year 2024 (current calendar year at the time of writing).

Pages which violated any of these criteria were omitted from the analysis. From the pages which did not violate any of these conditions, we selected the three most well-known tools mentioned on these pages. This limited perspective on only the top-3 most well-known entries was imposed to focus on important tools and not have the analysis become too broad to yield actionable insight.

The generated list was then assessed with respect to their capabilities specifically for the requirements outlined in the previous section, which are needed at SHS. The result of this review is charted in Tab. 3. The first column lists all the tools found with out above-described search method, all subsequent columns show the availability of our requirements in the corresponding tools. For the sake of legibility, we used a shortened encoding to denote requirement names in the column headers. The mapping to the requirement references used above can be found in the footnotes of Tab. 3.

The data collected in Tab. 3 clearly shows that solutions exist for all the requirements identified before which could improve DQ as experienced by data consumers at SHS. There

Tool	AN	DGA	DO	DPD	MT	RT	SBJM	SDT
Alation Platform	—	—	—	—	✓	—	—	—
Apache Air Flow	—	✓	✗	✓	—	—	✓	—
Atlan	—	—	—	—	✓	—	—	—
AWS Glue	—	—	—	—	—	—	✓	—
Azure Factory	—	—	—	—	✓	—	✓	—
Chronos	—	—	—	—	—	—	✓	—
Collibra	—	✓	—	✓	✓	—	—	—
Dataedo	—	—	—	—	✓	—	—	—
DataGaps	✓	—	—	—	—	✓	—	✓
DBT	✓	—	—	—	—	✗	✓	✓
Google Cloud DF	—	—	—	—	—	—	✓	—
Great Expectations	—	—	✗	—	—	—	—	—
Hevo Data	—	—	—	—	—	—	✓	—
iceDQ	✓	—	—	—	—	✓	—	✓
Informatica	✓	✓	—	✓	✓	✓	—	✓
Io-Tahoe	—	—	—	—	✗	—	—	—
Jenkins	—	—	—	—	—	—	✓	—
KNIME	—	—	—	—	—	—	✗	—
Metaplane	—	—	✓	—	—	—	—	—
Monte Carlo	—	—	✓	—	—	—	—	—
Open Lineage	—	—	✓	—	—	—	—	—
Oracle MM	—	✗	—	✓	—	—	—	—
Query Surge	✓	—	—	—	—	✓	—	✗
RightData	✓	—	—	—	—	✗	—	✗
Sifflet	—	—	✓	—	—	—	—	—
Snowflake	—	✗	—	✗	✓	—	—	—
Soda	—	—	✓	—	—	—	—	—
Talend	✓	✓	—	✓	—	✗	—	✗
<i>Reviewed Tools</i>	7	6	7	6	8	7	9	7
<i>Feature Covered</i>	7	4	5	5	7	4	8	4

Table 3: Review of tools available on the market which offer functionalities relevant to the prevention methods discussed above. A check-mark signals that the corresponding tool was tested for the functionality needed at SHS and found to provide an adequate solution, whereas a cross-mark signals tools which were reviewed for that functionality and found to be insufficient. A dash is used when the corresponding tool was not reviewed for that functionality. Abbreviations: AN=Automated Notifications, DGA=Data Governance Accessibility, DO=Data Observability, DPD=Data Provenance Documentation, MT=Metadata Tagging, RT=Regression Testing, SBJM=Status-based Jobs Monitoring, SDT=Standardized Data Testing.

are actually tools which cover the set of requirements quite extensively. While not all tools were reviewed with respect to all features due to resource constraints (a horizontal dash in Tab. 3 indicates no analysis performed, and not that the feature is missing, see footnotes to Tab. 3), several tools do stand out. The most comprehensive solution reviewed

by us is the *Informatica* tool suite. They offer a broad variety of data governance and metadata management solutions, and even some regression testing capabilities. *Informatica* is not typically classified as a data observability tool, so it is not surprising that they lack capabilities in that domain. Also, while *Informatica* does offer its own version of jobs scheduling and monitoring, external data production pipelines cannot directly be orchestrated and monitored in the tool. Tab. 3 shows that there are other solutions to that end. For instance, *Monte Carlo* offers a widespread data observability solution and *DBT* implements Status-Based Jobs Monitoring. When looking at the total count of tools in which each feature is covered, it becomes apparent that none of the needs at SHS completely lack a solution. We can, however, identify requirements which are only covered in part of the tools reviewed for these features. Among the most important are Regression Testing and Standardized Data Testing. Regarding data architecture, the problems at SHS arose from manual process steps, and from full data loads rather than change-data-capture. In principle, all modern data architectures surveyed in literature are compatible with these needs, and especially the elimination of remnant manual process steps must remain an important goal in going forward. In a survey of data warehouse architectures, [27] classifies all existing approaches. Neither pipeline automation nor change-data-capture appear as a relevant classification feature, suggesting that these requirements are not tied to specific architectures. However, a detailed capture of data-pipeline-related metadata is proposed in [11] as a first important step in the direction of full pipeline automation as part of their automation concept. This type of metadata can be collected and seamlessly integrates with other data governance projects which aim at compiling and documenting metadata. Change-data-capture methods are also available independent of architecture. One proposal implements this technology using *Apache Spark* in a *Hadoop Distributed File System* and finds a significant decrease in processing time[3]. At SHS, we recommended integrating both changes to data architecture with an on-going effort to convert the existing three-tiered data warehouse into a *Data Vault* architecture[13]. Change-data-capture and full process automation are integral parts of this analytics setup and the associated data warehouse.

Given the case study at SHS, we can therefore answer research question **R2** with the list of tools listed in Tab. 3. These providers offer industry-standard solutions for the required technologies, and they do not leave many gaps in the market. Research question **R3** asks if there are any blind spots which are impossible or difficult to cover, given what the market offers, and this seems to not be the case. However, we can point to several features such as support for Regression Testing and Standardized Data Testing which are still missing in many of the reviewed tools. Furthermore, none of the tools surveyed meet all the requirements, making the integration of these products into a comprehensive DQ framework for any given organization - SHS in our case study - a complex and potentially challenging task.

7 Conclusion

We have outlined in this article an empirical analysis of the DQ problems in large companies, as observed in a case study at a large German medical technologies manufacturer. We have

proposed a classification of the underlying causes leading to low data quality downstream into problem classes which are independent of specific architecture, and we have further mapped these categories to the required prevention strategies. In addition, we have conducted a market analysis to determine whether the technologies required for prevention of the identified problem classes already exist, or whether there are gaps in existing solutions. We find that the market provides tools in response to all identified needs, but that no single solution offers a comprehensive remedy. This analysis allows us to answer the initial research questions **R1** positively, problems with DQ in CRM data do seem to cluster into different classes. Question **R2** is answered by a market analysis, and regarding **R3** we cannot find a relevant market blind-spot. These conclusions are based on one case study, so in order to generalize these results, similar analyses at other large companies are needed. Going forward, this empirical approach to identifying and fixing underlying issues will contribute not only to benefit enterprises with higher-quality information, but can also help researchers enrich the currently accepted DQ Dimensions with findings based in an architecture-centered understanding of data as it prevails in data engineering teams across industry.

References

- [1] Danielle GT Arts, Nicolette F De Keizer, and Gert-Jan Scheffer. “Defining and improving data quality in medical registries: a literature review, case study, and generic framework”. In: *Journal of the American Medical Informatics Association* 9.6 (2002), pp. 600–611.
- [2] Pouya Ataei and Alan Litchfield. “The state of big data reference architectures: A systematic literature review”. In: *IEEE Access* 10 (2022), pp. 113789–113807.
- [3] I Putu Medagia Atmaja, Ari Saptawijaya, Siti Aminah, et al. “Implementation of change data capture in ETL process for data warehouse using HDFS and apache spark”. In: *2017 International Workshop on Big Data and Information Security (IWBIS)*. IEEE. 2017, pp. 49–55.
- [4] Zhamak Dehghani. *Data Mesh*. Marcombo, 2022.
- [5] MI Gabr, YM Helmy, and DS Elzanfaly. “Data quality dimensions, metrics, and improvement techniques”. In: *Future Comput. Inf. J* 6 (2021), pp. 25–44.
- [6] Boris Glavic et al. “Data provenance”. In: *Foundations and Trends® in Databases* 9.3-4 (2021), p. 3.
- [7] Anders Haug and Jan Stentoft Arlbjørn. “Barriers to master data quality”. In: *Journal of Enterprise Information Management* 24.3 (2011), pp. 288–303.
- [8] Wei Hu and Junkang Feng. “Data and information quality: an information-theoretic perspective”. In: *Computing and Information Systems* 9.3 (2005), p. 32.
- [9] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. “Data clustering: a review”. In: *ACM computing surveys (CSUR)* 31.3 (1999), p. 270.

- [10] Vladan Jovanovic and Ivan Bojicic. “Conceptual data vault model”. In: *SAIS 2012 Proceedings* (2012), pp. 130–136.
- [11] Ahmed Kabiri and Dalila Chiadmi. “A method for modelling and organazing ETL processes”. In: *Second International Conference on the Innovative Computing Technology (INTECH 2012)*. IEEE, 2012, pp. 138–143.
- [12] Godson Koffi Kalipe and Rajat Kumar Behera. “Big Data Architectures: A detailed and application oriented review”. In: *Int. Journal Innov. Technol. Explor. Eng* 8 (2019), pp. 2182–2190.
- [13] Daniel Linstedt and Michael Olschimke. *Building a scalable data warehouse with data vault 2.0*. Morgan Kaufmann, 2015.
- [14] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. “Data mesh: concepts and principles of a paradigm shift in data architectures”. In: *Procedia Computer Science* 196 (2022), pp. 263–271.
- [15] Barbara Magagna et al. “Data provenance”. In: *Towards Interoperable Research Infrastructures for Environmental and Earth Sciences: A Reference Model Guided Approach for Common Challenges*. Springer, 2020, p. 214.
- [16] Kamal Matouk, Mieczysław L Owoc, et al. “A survey of data warehouse architectures—Preliminary results”. In: *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2012, pp. 1121–1126.
- [17] Sedir Mohammed et al. “Data Quality Assessment: Challenges and Opportunities”. In: *arXiv preprint arXiv:2403.00526* (2024).
- [18] Barr Moses. “The rise of data observability: Architecting the future of data trust”. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 1657–1657.
- [19] Sonam Ramchand and Tariq Mahmood. “Big data architectures for data lakes: A systematic literature review”. In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2022, pp. 1141–1146.
- [20] Dhamotharan Seenivasan and Muthukumaran Vaithianathan. “Real-Time Adaptation: Change Data Capture in Modern Computer Architecture”. In: *ESP International Journal of Advancements in Computational Technology* 1.2 (2023), pp. 49–61.
- [21] Sing What Tee et al. “Data quality initiatives: striving for continuous improvements”. In: *International Journal of Information Quality* 1.4 (2007), pp. 347–367.
- [22] Mubarak Albarka Umar and Chen Zhanfang. “A comparative study of dynamic software testing techniques”. In: *International Journal of Advanced Networking and Applications* 12.3 (2020), pp. 4575–4584.
- [23] Yair Wand and Richard Y Wang. “Anchoring data quality dimensions in ontological foundations”. In: *Communications of the ACM* 39.11 (1996), pp. 86–95.

- [24] Jingran Wang et al. “Overview of data quality: Examining the dimensions, antecedents, and impacts of data quality”. In: *Journal of the Knowledge Economy* 15.1 (2024), pp. 1159–1178.
- [25] Richard Y Wang. “A product perspective on total data quality management”. In: *Communications of the ACM* 41.2 (1998), pp. 58–65.
- [26] 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.
- [27] Qishan Yang, Mouzhi Ge, and Markus Helfert. “Analysis of data warehouse architectures: modeling and classification”. In: *The 21st International Conference on Enterprise Information Systems* (2019), pp. 3–5.