Towards a Low-Code Tool for Developing Data Quality Rules

Timon Sebastian Klann[1], Marcel Altendeitering[2][a] and Falk Howar[1][b]
[1]*TU Dortmund University, Dortmund, Germany*
[2]*Fraunhofer ISST, Dortmund, Germany*

Keywords: Data Quality Rules, Data Validation, Data Management, Domain Specific Language, Visual Programming.

Abstract: High-quality data sets are vital for organizations as they promote business innovation and the creation of data-driven products and services. Data quality rules are a common approach to assess and enforce compliance with business domain knowledge and ensure the correctness of data sets. These data sets are usually subject to numerous data quality rules to allow for varying requirements and represent the needs of different stakeholders. However, established data quality tools have a rather technical user interface and lack support for inexperienced users, thus hindering them from specifying their data quality requirements. In this study, we present a tool for the user-friendly and collaborative development of data quality rules. Conceptually, our tool realizes a domain-specific language, which enables the graphical creation of rules using common data quality constraints. For implementation, we relied on CINCO, a tool for creating domain-specific visual modeling solutions, and Great Expectations, an open-source data validation framework. The evaluation of our prototype was two-fold, comprising expert interviews and a focus group discussion. Overall, our solution was well-received and can contribute to lowering the accessibility of data quality tools.

1 INTRODUCTION

A high level of Data Quality (DQ) is a critical success factor for organizations but is often impaired by missing, erroneous, or duplicate values (Kandel et al., 2012). These DQ issues are usually difficult to find and resolve and can interrupt business operations (Redman, 2020; Amadori et al., 2020). Consequently, poor DQ is considered one of the major tasks in modern data management (Gröger, 2021) and is responsible for "$15 million per year in losses" (p.1) (Moore, 2018) for an average organization.

The specification of DQ rules is a typical approach to ensure accurate and high-quality data sets. These rules define general or domain-specific data knowledge and can be used for validating the compliance of data (Altendeitering and Tomczyk, 2022). However, defining these rules is complicated by the 'fitness for use' principle of DQ (Wang and Strong, 1996). Fitness for use means that different stakeholders pose different requirements on data and have varying quality definitions based on their tasks. As a result, a data set is usually subject to numerous DQ rules (Wang and He, 2019; Tebernum. et al., 2021). For example, when analyzing a sales data set, a data scientist might want the data to be accurate, while a business analyst is more focused on the timeliness of data.

In the past, defining DQ rules was a technical task conducted by a centralized group of technically skilled employees. However, this approach to DQ does not scale in light of ubiquitous and big data, leading to cumbersome and time-consuming DQ processes (Altendeitering and Guggenberger, 2021). In response to this lack of scalability, DQ is becoming increasingly decentralized and conducted "at the source" (p.1) (Redman, 2020). It evolves from a centralized task to a cooperative effort involving stakeholders across the organization (Gröger, 2021; Dehghani, 2020). DQ tools must embrace these changes and allow various stakeholders to specify their DQ rules and validate data to their needs. Established DQ tools often fail to address this need as they are limited in supporting inexperienced users and lack collaborative functionalities (Altendeitering and Tomczyk, 2022).

We aim to address the need for accessible and user-friendly solutions by developing a low-code application that supports the design and development of DQ rules. The following research question guided our study:

[a] https://orcid.org/0000-0003-1827-5312
[b] https://orcid.org/0000-0002-9524-4459

Klann, T., Altendeitering, M. and Howar, F.
Towards a Low-Code Tool for Developing Data Quality Rules.
DOI: 10.5220/0012050400003541
In *Proceedings of the 12th International Conference on Data Science, Technology and Applications (DATA 2023)*, pages 22-29
ISBN: 978-989-758-664-4; ISSN: 2184-285X

Research Question: *What does a DQ tool for the user-friendly and collaborative specification of DQ rules look like?*

To answer the proposed research question, we developed and prototypically implemented ColDaQ, a DQ tool for the graphical and collaborative specification of DQ rules. ColDaQ offers a data-flow programming environment for constructing DQ rules in a user-friendly way. The low-code approach is well-suited for supporting users from various backgrounds to realize their DQ requirements (Altendeitering and Schimmler, 2022). In this sense, our tool helps data domain experts become self-reliant and empowers them to integrate and operate DQ rules themselves. As a result, we contribute to overcoming the typical divide of DQ tasks between technical (e.g., data engineers) and non-technical (e.g., domain experts) users that complicates DQ management (Altendeitering and Guggenberger, 2021).

For the prototypical implementation of ColDaQ, we used Great Expectations (The Great Expectations Team, 2020) and CINCO (Naujokat et al., 2018) as technological bases. The qualitative evaluation of our application was two-fold. In the first round, we interviewed three experts from the DQ and user experience domains to gain in-depth feedback. Subsequently, we completed a second round of evaluation using a focus group discussion with ten participants working in data-intensive jobs. Overall, we found that our tool can ease the manual specification of DQ rules and inform the design of future DQ tools.

The remainder of this article is structured as follows. First, we describe the theoretical background of our study regarding data management and quality, DQ tools, and domain-specific languages in section 2. In section 3, we outline the conceptual approach of our tool and describe its prototypical implementation. We present and discuss the qualitative evaluation results for our tool in section 4. Finally, in section 5, we describe the contributions of our study, highlight limitations, and outline paths for future work.

2 BACKGROUND

2.1 Data Management & Data Quality

”You can’t do anything important in your company without high-quality data” (p.1) (Redman, 2020). (Redman, 2020) states that DQ forms an important organizational success factor and is widely recognized as an essential building block for organizational agility and a driver of business innovation. For instance, a high level of DQ has a positive influence on the functioning of data-intensive applications (e.g., artificial intelligence) (Tebernum. et al., 2021; Gröger, 2021), allows seamless business processes (Amadori et al., 2020), and builds trust among partners in a data ecosystem (Guggenberger et al., 2020).

To be considered high-quality, data needs to fulfill a ’fitness for use’, which is context-dependent and defined by the data consumer (Wang and Strong, 1996). As a context-dependent and multi-dimensional concept, a sufficient level of DQ can be difficult to achieve. As a result, it has long been incorporated into data management practices but is still considered a significant issue (Wang, 1998; Gröger, 2021). Several frameworks, such as the Total Data Quality Management (TDQM) framework, emerged to support and ease DQ management. The TDQM framework outlines how to define, analyze, measure, and improve DQ. For example, by specifying DQ rules and validating data against these rules (Altendeitering, 2021). However, manual DQ management is complex, cumbersome, and error-prone and is often supported by DQ tools (Altendeitering and Tomczyk, 2022; Altendeitering and Guggenberger, 2021).

2.2 Data Quality Tools

A plurality of tools emerged from science and practice to support DQ management. We can distinguish these tools in data preparation tools for correcting erroneous data, data measuring and monitoring tools for data validation, and general-purpose tools, which offer the most comprehensive set of DQ functionalities (Ehrlinger and Wöß, 2022; Altendeitering and Tomczyk, 2022). Over time, DQ evolved from an algorithmic, IT-centric task to a joint effort involving multiple stakeholders from across the organization (Altendeitering and Tomczyk, 2022). These developments raised new requirements for modern DQ tools, which must offer collaborative functionalities and accessible user interfaces (Altendeitering and Guggenberger, 2021; Swami et al., 2020). However, established tools often cannot fulfill these requirements and lack support for inexperienced users and collaborative approaches to DQ (Altendeitering and Tomczyk, 2022).

In this study, we aim to address this lack of established tools and develop a collaborative, user-friendly solution for defining DQ rules and validating data. For this purpose, we extend the DQ tool Great Expectations (GE) (The Great Expectations Team, 2020). We decided to use GE as it is open-source, offers extensive documentation, and is well-established in the field of DQ. It also has an active community and was used by similar studies (e.g., (Swami et al., 2020)).

GE is a Python-based library for validating and profiling data. By specifying 'Expectations', users can formulate their requirements on data in a declarative language (The Great Expectations Team, 2020). Along with metadata, several Expectations are combined into a list inside an 'Expectation Suite' in JSON format. Besides deriving expectations manually or automatically through profiling, the critical features of GE are data validation and documentation. The latter also offers a graphical and comprehensive presentation of the data and the validation results. GE allows to validate Expectations individually or combined in an Expectation Suite.

2.3 Domain Specific Languages

Domain-Specific Languages (DSLs) are used to describe the context of a specific problem domain efficiently with gains in productivity (Kosar et al., 2008). In contrast to general-purpose languages, a textual or graphical DSL is limited to the bounded context of its domain and does not require general programming knowledge. This means a DSL is more accessible to domain experts and offers expressive power through abstraction without explaining everything in detail (Fall and Fall, 2001; Kosar et al., 2008; Van Deursen and Klint, 2002). Potential advantages of DSLs are high portability, efficiency, and reliability (Deursen and Klint, 1998; Van Deursen and Klint, 2002; Van Deursen et al., 2000).

The CINCO meta tooling suite is a simplicity-oriented DSL tool for developing domain-specific graphical modeling tools (Naujokat et al., 2018). Based on several frameworks, CINCO generates such tools, also called CINCO Products (CP), from simple high-level specifications (MGL: meta graph language, MSL: meta styler language, and CPD: CINCO product definition) automatically (Lybecait et al., 2018). The key idea of CINCO is to hide the complexity of its underlying frameworks and keep the development of the CP as simple as possible (Naujokat, 2016).

Owing to the common meta-domain, every CP has the same structure and basic functionality. Based on structural specifications made in the MGL, users of the CP can place nodes into a graphical editor (canvas) by drag & drop and connect them via edges. The resulting graph will appear as specified in the MSL (Naujokat, 2016). Users can define additional functionality and semantic interpretations, such as code generation or model validation, using meta plugins (Naujokat et al., 2018). We decided to use CINCO to develop our prototype as it supports creating custom domain-specific languages and reduces the development time. It is thus well-suited for quickly creating prototypical applications and retrieving feedback early on. CINCO was also used in several other projects and CINCO Cloud, the web-based successor of CINCO, is currently under development (Naujokat et al., 2018).

3 PROTOTYPE

3.1 Conceptual Approach

With our prototypical DQ tool ColDaQ, we aim to enable the user-friendly and collaborative specification and application of DQ rules. In contrast to established solutions such as GE, users should benefit from a simple UI that is less code-centric. ColDaQ should empower users from technical and non-technical backgrounds to specify their individual DQ needs without modifying codebases (Altendeitering and Guggenberger, 2021; Altendeitering and Tomczyk, 2022). We envision ColDaQ to be operated in a federated, self-serve data infrastructure as a platform where different domains and user groups can customize ColDaQ to their needs (Gröger, 2021). In this architecture, a centralized DQ team could specify DQ information models and offer universal DQ rules, which domain experts can extend and customize (Altendeitering and Tomczyk, 2022; Dehghani, 2020).

ColDaQ relies on CINCO and GE as its technological bases. The tool is specified as an MGL and MSL and generated into a CP by CINCO. To enable the validation of a given data set against user-defined DQ rules, ColDaQ offers a code export that converts the DQ rules into an Expectation Suite suitable for use in GE. For our prototype, we focused on validating CSV files for simplicity. Furthermore, ColDaQ only considers single table data sources and is limited to DQ conditions that can be validated as true or false. In this sense, we assume that DQ rules in ColDaQ can be treated as logical statements about the data.

As a starting point, ColDaQ offers a small set of typical DQ constraints that users can use to describe specific requirements on data (see Table 1). We based the pre-defined set of DQ constraints on two studies reviewing DQ tools and their functionalities (Ehrlinger and Wöß, 2022; Altendeitering and Tomczyk, 2022). Users can create custom DQ rules by combining one or more of these constraints using edges and Or-Container nodes. The key idea of edges and Or-Containers is to cover the basic logical connectives (negation, conjunction, and disjunction) to form a composition of (custom) DQ constraints. The latter is supported by a graphical user interface

Table 1: Set of DQ constraints available in ColDaQ.

Constraint	Description
Not Null	Exclude Null Values
All True	Assert that all values are True
Values In Range	Ensure that values are restricted to given bounds
Matches Regex	Ensures values match a given regular expression
Uniqueness	Exclude duplicate values
Numeric	Assert that all values are numeric
Evenly Distributed	Assert an even distribution with a given parameter for tolerance
Min Between	Check if the minimum of the values is within a given interval
Max Between	Check if the maximum of the values is within a given interval
Sum Between	Check if the sum of the values is within a given interval

that allows users to model DQ rules in a low-code DQ graph called ColDaQ Graph (CG). Following a data-flow programming approach, our concept lowers the semantic and syntactic barriers to creating DQ rules and improves the accessibility to DQ tools for inexperienced users (Altendeitering and Schimmler, 2022).

3.2 Prototypical Implementation

For the prototypical implementation of our concept, we followed the guidelines by (Alavi, 1984) and continuously improved an initial version of our prototype. We initialized the implementation by specifying the MGL and MSL more concretely. The MGL describes all nodes and edges available in the graphical DSL. ColDaQ distinguishes three kinds of edges: Success, No-Success, and Default edge. The latter is a universal edge used in cases where no separation between success and non-success is necessary. Listing 1 displays an example of how edges are specified in CINCO.

Listing 1: Specification of the success edge.

```
@style(success)
edge Success{}
```

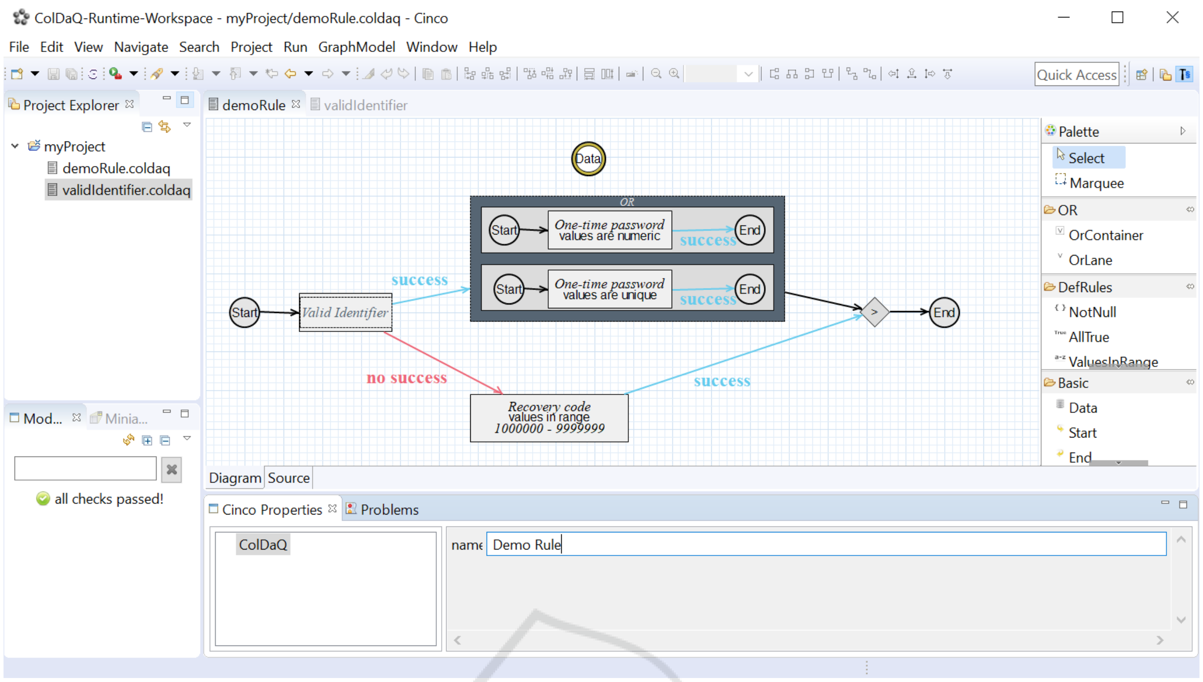
Regarding the provisioning of nodes, ColDaQ implements each of the DQ constraints shown in Table 1 as a node. Listing 2 shows how a node is implemented using the 'Matches Regex' constraint as an example. The abstract Constraint node (l. 1-6) specifies the general handling of edges. Nodes can have outgoing edges depending on whether a constraint is validated successfully. Every concrete node inherits from this abstract node and specifies further details (l. 7-19). For instance, the Matches Regex constraint has an attribute to transfer the display name to the MSL and a column attribute annotated with possibleValuesProvider annotation to choose a column from the data table provided. Furthermore, a DQ node can have additional attributes (l. 18).

Listing 2: Specification of the Matches Regex node.

```
abstract node Constraint{
 incomingEdges(*[1,1])
 outgoingEdges(Success[0,1],
 NoSuccess[0,1],
  {Success,NoSuccess}[1,2])
}
...
@style(ruleRectangle3Arg,
 "${column}","${display}","${regex}")
@palette("DefRules")
node MatchesRegex extends Constraint{
 @possibleValuesProvider("info.scce.
  cinco.product.coldaq.provider.
  PossibleColumnValueProvider")
 attr EString as column
 @propertiesViewHidden
 attr EString as display :=
  "matches regex"
 attr EString as regex
}
```

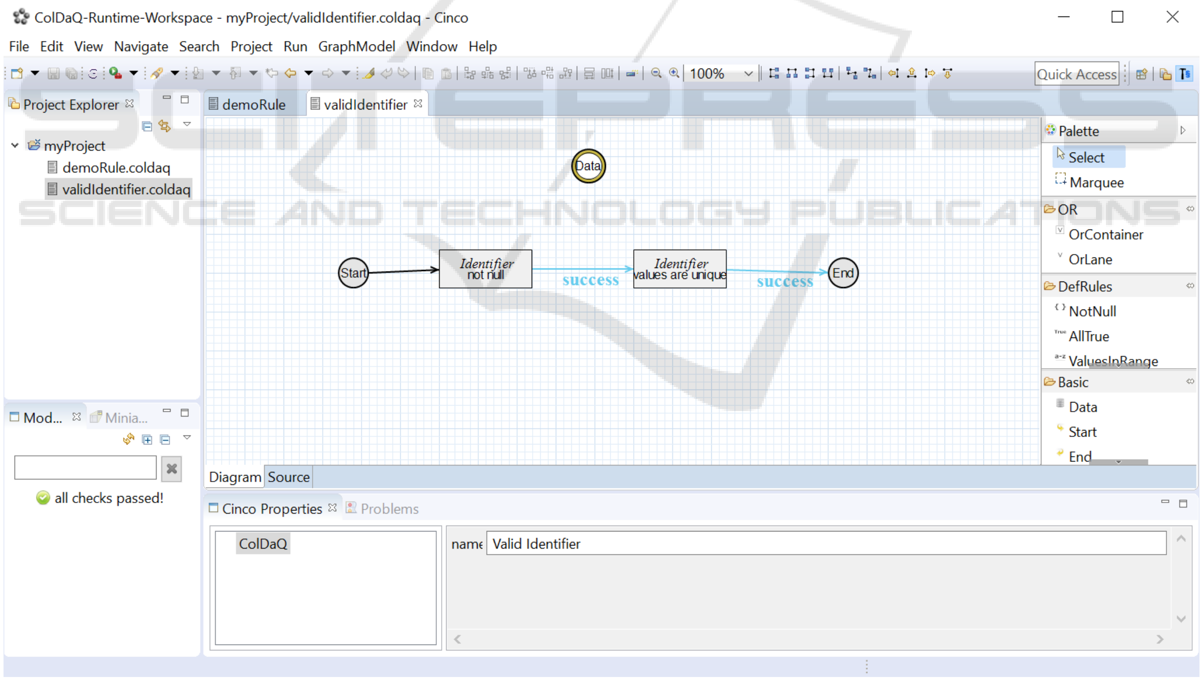
In CINCO, annotations are an easy way to implement additional functionality. For example, the model checking annotation can check whether all nodes except the Data node are connected from the Start to the End node. It can also automatically adjust the size of Or-Container and Or-Lanes whenever the user adds new elements. The MSL only implements the visual concept declaratively illustrated in Figure 1, which shows the user interface of ColDaQ in its final version. As an additional feature, users of ColDaQ can reuse CGs with custom DQ rules as new nodes that behave like a new, single node. For instance, the DQ rule specified in the CG shown in Figure 1b is referred to by a new node named 'Valid Identifier' in the CG in Figure 1a.

For the validation of data against a DQ rule modeled in ColDaQ, we implemented a code generator that creates a Python script and an Expectation Suite. The Expectation Suite summarizes the DQ conditions and uses GE for validation. ColDaQ generates the Python script in three steps. At first, static code for



name Demo Rule

(a)



name Valid Identifier

(b)

Figure 1: Data Quality Rules modeled with the ColDaQ Prototype.

executing the atomic validation with GE is generated. In the second step, the code generator analyzes every path from the Start to the End node and builds a binary parse tree considering all concepts of DQ rules treated as logical statements. At last, ColDaQ uses the tree's structure to generate a logical statement based on the atomic validation results. When executing the Python script, the evaluation of that logical statement equals the overall validation of the modeled CG.

To realize the collaborative functionality of

ColDaQ, we relied on Pyro (Zweihoff, 2022), a meta-plugin to CINCO (Naujokat, 2016). Pyro can generate a full-featured web application that supports simultaneous work on DQ rules. However, technical restrictions and incompatibilities hindered us from fully integrating Pyro. The collaborative aspect is thus currently limited to sharing DQ rules using the import and export function of ColDaQ (see also section 5.2).

4 EVALUATION

Following the guidelines of (Kitchenham et al., 2004), our evaluation aims to critically reflect on the validity, impact, and applicability of our prototype. To achieve this goal, we conducted a two-fold evaluation. In the first step, we interviewed three DQ and user experience experts. With these interviews, we wanted to gain in-depth feedback on our conceptual approach and the functional scope of ColDaQ (Meuser and Nagel, 2009). Subsequently, we conducted a focus group discussion with ten participants working in data-intensive roles. The focus group discussion was well-suited for obtaining additional insights as they are high in external validity and can spark discussions participants have in their daily lives (Hollander, 2004). In the following subsections, we describe each evaluation step more closely.

4.1 Step 1: Expert Interviews

The first expert we interviewed is employed as a frontend software developer. Interviewees 2 and 3 have a background in DQ. We initiated each interview with a demo of our tool, including creating a DQ rule and its application to an exemplary data set. After the demo, we asked the interviewees for feedback regarding intuition and concept of the UI components, relevance, and benefit. We noted their responses, which form the primary source for analysis (Meuser and Nagel, 2009). To systematically analyze the data, we used content analysis (Vaismoradi and Snelgrove, 2019).

Overall, the expert feedback was mainly positive. All interviewees liked the conceptual design of the user interface. Interviewees 2 and 3 emphasized that the concept of composing DQ rules graphically is intuitive. As a potential improvement, interviewee 3 stated that the start and end nodes could feature additional functionality, such as connecting with APIs for direct data imports.

Regarding the conceptual approach for developing DQ rules, interviewees 1 and 2 critically noted that ColDaQ only allows DQ constraints that can be validated as true or false. In addition, they criticized the limitation of the static set of DQ constraints in the node palette and envisioned possibilities to include additional nodes. Moreover, interviewee 3 suggested including a node that uses external code for data validation. Interviewees 2 and 3 proposed more automation, for example, by conducting automated data profiling and offering a set of DQ constraints appropriate for a data set.

4.2 Step 2: Focus Group Discussion

The focus group discussion included ten participants working in data engineering. Similar to the expert interviews, we initiated the discussion with a presentation of the tool and demonstrated its functionality using an exemplary data set. After the presentation, we asked guiding questions on the tool's functional composition and visual style to spark a discussion among the participants (Hollander, 2004). The participants positively noted the conceptual approach and the possibility of graphically designing DQ rules. It can help users inexperienced with DQ to construct rules and apply these to a data set.

On the negative side, the participants noted that some nodes are difficult to understand because they did not recognize the difference between default (atomic) and composed nodes. They suggested simplifying the naming of nodes for better usability. Users might furthermore benefit from additional explanations and usage examples for the individual nodes to improve their comprehensibility. Additionally, the participants stated limitations in the composition of DQ rules. They envisioned further nodes and logical operators for combinations. Finally, the evaluation of DQ rules with GE can be improved. More detailed results about which nodes failed and potential reasons could help users identify underlying data errors and improve the quality of a data set.

5 CONCLUSION

5.1 Contributions

In response to our initial research question, we successfully developed a DQ tool that allows the user-friendly specification of DQ rules using a low-code user interface. With export functionalities, ColDaQ allows the collaborative design of DQ rules and automated validation in GE. The two-step evaluation revealed that ColDaQ was, overall, well-received and can fulfill the goal of lowering the accessibility of DQ tools. Specifically, the participants positively commented on the intuitive user interface and the visual, low-code concept for constructing DQ rules. For further development, the participants noted that the tool should allow the design of more complex DQ rules and improve the comprehensibility of nodes.

Our study offers both scientific and managerial contributions. Scientifically, we conceptually designed and prototypically implemented an artifact that addresses the need for more intuitive, user-friendly, and collaborative DQ tools raised in several studies (Altendeitering and Tomczyk, 2022; Swami et al., 2020; Ehrlinger and Wöß, 2022). Our findings can furthermore contribute to realizing decentralized DQ tools accessible to different stakeholders and inform the design of future tools (Gröger, 2021). For managers, we provide insights on building tools for the manual specification of DQ rules that take current trends into account. This knowledge can support make-or-buy decisions, inform custom implementations, and raise awareness for DQ and adequate tooling.

5.2 Limitations & Future Work

Despite following a rigorous approach, our study is subject to several limitations, which offer paths for future work. Most importantly, we aimed to create an application that supports collaborative work on DQ rules. However, due to technical restrictions, the collaborative aspect is only partly realized and limited to export/import functionalities. We plan to focus the future development of our tool on this aspect and upgrade to 'Cinco Cloud', which was not yet available at the time of writing. Cinco Cloud aims to realize full-featured web applications that enable the collaborative, multi-user modeling of DQ rules.

Moreover, ColDaQ currently features a limited set of pre-defined DQ rules. In the future, ColDaQ should allow more kinds of basic DQ rules from different sources and allow users to specify custom DQ rules for validation. In this regard, it would also be interesting to investigate the difference in DQ rules between different user groups and derive which ones are generally applicable.

To further increase its usability, ColDaQ should offer additional support and automation. For example, we can apply automated data profiling techniques on a given data set and generate DQ rules automatically (Abedjan et al., 2015).

Regarding our methodological approach, our evaluation results are based on qualitative feedback from participants working in computer science and data-related jobs. In future studies, we plan to evaluate ColDaQ with a more extensive and diverse user group to obtain more profound results. These studies should also include participants working in non-technical jobs to assess if our tool can achieve the desired level of accessibility.

ACKNOWLEDGEMENTS

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

REFERENCES

Abedjan, Z., Golab, L., and Naumann, F. (2015). Profiling relational data: a survey. *The VLDB Journal*, 24:557–581.

Alavi, M. (1984). An assessment of the prototyping approach to information systems development. *Communications of the ACM*, 27(6):556–563.

Altendeitering, M. (2021). Mining data quality rules for data migrations: a case study on material master data. In *International Symposium on Leveraging Applications of Formal Methods*, pages 178–191. Springer.

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

Altendeitering, M. and Schimmler, S. (2022). End-user development for smart spaces: A comparison of block and data-flow programming. In *Proceedings of the 11th International Conference on Smart Cities and Green ICT Systems, SMARTGREENS*.

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

Amadori, A., Altendeitering, M., and Otto, B. (2020). Challenges of data management in industry 4.0: a single case study of the material retrieval process. In *International Conference on Business Information Systems*, pages 379–390. Springer.

Dehghani, Z. (2020). Data mesh principles and logical architecture. https://martinfowler.com/articles/data-mesh-principles.html. Accessed: 17.01.2023.

Deursen, A. V. and Klint, P. (1998). Little languages: little maintenance? *Journal of Software Maintenance: Research and Practice*, 10(2):75–92.

Ehrlinger, L. and Wöß, W. (2022). A survey of data quality measurement and monitoring tools. *Frontiers in Big Data*, page 28.

Fall, A. and Fall, J. (2001). A domain-specific language for models of landscape dynamics. *Ecological modelling*, 141(1-3):1–18.

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

Guggenberger, T. M., Möller, F., Boualouch, K., and Otto, B. (2020). Towards a unifying understanding of digital business models. In *Proceedings of the Twenty-Third Pacific Asia Conference on Information Systems*.

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

Kandel, S., Parikh, R., Paepcke, A., Hellerstein, J. M., and Heer, J. (2012). Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 547–554.

Kitchenham, B. A., Dyba, T., and Jorgensen, M. (2004). Evidence-based software engineering. In *Proceedings. 26th International Conference on Software Engineering*, pages 273–281. IEEE.

Kosar, T., Martı, P. E., Barrientos, P. A., Mernik, M., et al. (2008). A preliminary study on various implementation approaches of domain-specific language. *Information and software technology*, 50(5):390–405.

Lybecait, M., Kopetzki, D., Zweihoff, P., Fuhge, A., Naujokat, S., and Steffen, B. (2018). A tutorial introduction to graphical modeling and metamodeling with cinco. In *International Symposium on Leveraging Applications of Formal Methods*, pages 519–538. Springer.

Meuser, M. and Nagel, U. (2009). The expert interview and changes in knowledge production. *Interviewing experts*, pages 17–42.

Moore, S. (2018). How to create a business case for data quality improvement. https://www.gartner.com/smarterwithgartner/how-to-create-a-business-case-for-data-quality\-improvement. Accessed: 24.01.2023.

Naujokat, S. (2016). Cinco user's manual. https://cinco.scce.info/. Accessed: 24.01.2023.

Naujokat, S., Lybecait, M., Kopetzki, D., and Steffen, B. (2018). Cinco: a simplicity-driven approach to full generation of domain-specific graphical modeling tools. *International Journal on Software Tools for Technology Transfer*, 20(3):327–354.

Redman, T. C. (2020). To improve data quality, start at the source. https://hbr.org/2020/02/to-improve-data-quality-start-at-the-source. Accessed: 10.01.2023.

Swami, A., Vasudevan, S., and Huyn, J. (2020). Data sentinel: A declarative production-scale data validation platform. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1579–1590. IEEE.

Tebernum., D., Altendeitering., M., and Howar., F. (2021). Derm: A reference model for data engineering. In *Proceedings of the 10th International Conference on Data Science, Technology and Applications*, pages 165–175. INSTICC.

The Great Expectations Team (2020). Welcome to great expectations! https://greatexpectations.io/. Accessed: 24.01.2023.

Vaismoradi, M. and Snelgrove, S. (2019). Theme in qualitative content analysis and thematic analysis. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, 20(3).

Van Deursen, A. and Klint, P. (2002). Domain-specific language design requires feature descriptions. *Journal of computing and information technology*, 10(1):1–17.

Van Deursen, A., Klint, P., and Visser, J. (2000). Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36.

Wang, P. and He, Y. (2019). Uni-detect: A unified approach to automated error detection in tables. In *Proceedings of the 2019 International Conference on Management of Data*, pages 811–828.

Wang, R. Y. (1998). A product perspective on total data quality management. *Communications of the ACM*, 41(2):58–65.

Wang, R. Y. and Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12(4):5–33.

Zweihoff, P. (2022). Pyro - a collaborative, meta-model-driven, web-based and graphical modeling environment. https://pyro.scce.info. Accessed: 17.02.2023.