Evaluation of Data Quality Assessment

KUANG-YU LI

University of Stuttgart
Master Student in Information Technology
70569 Stuttgart, Germany

Abstract. Accessing a data set is an important step for data-driven research. However, the use and analysis of data must be based on high-quality data, which is a necessary condition for generating value. Therefore, the data quality (DQ) assessment is essential to data-driven research. This paper provides an evaluation of the DQ assessment. This paper organizes the DQ assessment into three research topics and surveys on them, which includes fundamentals of data quality, an overview of DQ assessment methodology, and application of the existing data set. Fundamentals of data quality introduce data types, data quality problems, and data quality dimensions used to represent data quality. Overview of DQ assessment methodology describes phases, strategies, cost, and classification of general assessment methodologies. Application of the existing data set entails the use of TravisTorrent, which puts DQ in a context of continuous integration in the domain of software engineering. The research results contribute to the academic scope of the evaluation of the DQ assessment and lay a foundation for future data-driven research.

1 Introduction

Accessing a data set is an important step for data-driven research, but researchers should also be aware of its quality and potential caveats. Traditionally, data labeling and assessment are done in either a primitive or laborious manner, which becomes almost impossible in the big data era [7]. This seminar paper presents the evaluation of data quality assessment, which is beneficial for data-driven research. The topics of data quality assessment concern with problems in the following:

- What is the overview of methods to assess data quality?
- What are the phases, steps, or techniques included in a data quality assessment method?
- Which data properties can or cannot be analyzed/assessed.
- Are there methods or techniques to improve the quality of a data set?
- What is the application of the assessment methods to an existing data set?

In response to the problems mentioned above, this seminar paper discussed key aspects of data quality systematically and organized them into four research topics in the following:

1. An overview of data quality assessment methods.
2. Data properties that can be analyzed and cannot be analyzed.
3. Possible methods for improving the quality of the data.
4. The current application of the assessment methods to the data set.

The goal of this paper is to perform an evaluation of assessing data quality based on the mentioned research topics. The paper is structured as follows: Data Quality Fundamentals, Assessment Methodology Overview, DQ in Context of Continuous Integration, and Conclusion. To understand the first research topic, data quality assessment methods, the paper first introduced the fundamentals of data quality in Section 2. These fundamentals include Data Type, Data Quality Problem, and Data Quality Dimension [20, 7]. With these fundamentals, the reader can have a better understanding of later topics. The second research topic is addressed in Section 3. This section introduced the overview of the data quality assessment method. The pipelines, phases, steps, strategies, techniques, cost, and classification of data quality assessment methods are presented. The third research topic is also discussed in this section since the improvement method is one of the three phases of DQ assessment method [3, 6]. To address the fourth research topic, an existing open-source data set, TravisTorrent is later introduced followed by its corresponding research in Section 4. TravisTorrent is a freely available data set based on Travis CI and GitHub[5]. Its motivation is to provide quantifiable evidence on the implications of introducing and continuing to use CI. The use of TravisTorrent data set for defect prediction is presented [8, 18]. At the end of the paper, a conclusion is made based on the overall topics for evaluation of data quality assessment. The challenges for DQ assessment are discussed. Lastly, future work regarding the application of continuous integration data sets and possible extensions are presented as guidance for further research.

2 Data Quality Fundamentals

In order to understand the data quality assessment method, the fundamentals of data quality are required. The fundamentals include three main subtopics: Data Types, Data Quality Problem, Data Quality Dimensions. In this section, concepts, definition, and examples are introduced.

2.1 Data Types

The ultimate goal of a DQ methodology is the analysis of data. In the real world, objects need to be created in a format that could further be stored, retrieved, and processed by software programs. In the field of data and computer science, data is either implicitly or explicitly distinguished by three types according to Batini et al. [3]:

- Structured data, is aggregations or generalizations of items described by elementary attributes defined within a domain. Domains represent the range of values that can be assigned to attributes and usually correspond to elementary

data types of programming languages, such as numeric values or text strings. Relational tables and statistical data represent the most common type of structured data.

- Unstructured data, is a generic sequence for symbols, typically coded in natural language. Typical examples of unstructured data are questionnaires containing free text answering open questions or the body of an e-mail.

- Semistructured data, is data that have a structure which has some degree of flexibility. Semistructured data are also referred to as schemaless or self-describing. For example, XML is a markup language commonly used to represent semistructured data.

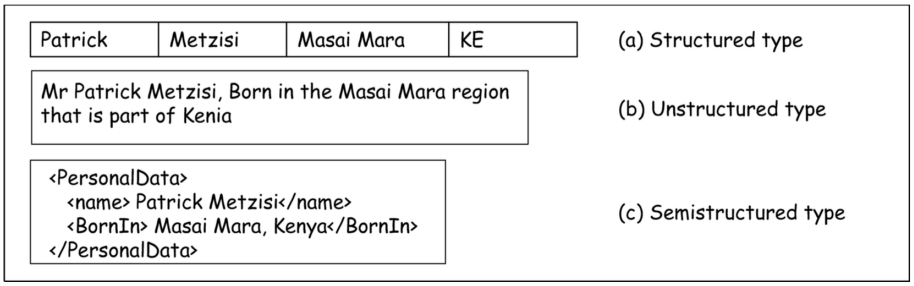

Fig. 1: Data types [3]

According to Batini et al. [3], Data quality techniques become increasingly complex as data loses its structure. For example, let us consider a registry describing personal information such as Name, Surname, Region, and StateOfBirth. Figure 1 shows the representation of Mr. Patrick Metzisi, born in the Masai Mara region in Kenya, by using a structured (Figure 1(a)), unstructured (Figure 1(b)), and semistructured (Figure 1(c)) type of data. The same quality dimension will have different metrics according to the type of data. This would be explained in section 2.3. The large majority of research contributions in the data quality literature focuses on structured and semistructured data. For this reason, this report focuses on structured and semistructured data.

2.2 Data Quality Problems

The data quality problem could be classified into four categories depending on two factors: context-independence and role perspective [6]. Table 1 provides a brief definition for each DQ problem. In the context independent category, spelling errors, missing data, and incorrect values are self-explanatory DQ problems. Duplicate data problems occur when rows are duplicated or when schemas contain redundancies. Data format problems occur when two or more semantically equivalent data values have different representations, including inconsistent and text formatting. Syntax violation problems occur when a

pre-specified format has been assigned to an attribute and a data value for this attribute does not adhere to this format, including incomplete data format. Problems with violations of integrity constraints arise when data values do not adhere to pre-specified database integrity constraints; according to Borek2011AClassficationOfDataQualityAssessmentMethod, they also therefore include unique value violations, rather than have these as a separate problem, because unique value violations are one type of database integrity constraint. Note that, despite its position in Table1 Borek et al. [6] treat outdated data to be a user perspective problem because whether data is out of date depends on the purpose it is used for.

For the context dependent category, the problem of violation of domain constraints is when an attribute value must be in a pre-specified context-dependent domain of values. Violation of organizational business rules is when any set of values do not adhere to a pre-specified rules assigned by the organization. Violation of company and governmental regulations is when any set of values do not adhere to a prespecified rules assigned imposed on the organization by legislating bodies. Similarly, violation of constraints provided by the database administrator is when any set of values do not adhere to a pre-specified rules assigned by the database administrator. In section 2.3, the problems are define in different dimensions with specific indicators.

	Data Perspective	User Perspective
Context-independent	Spelling error	
	Missing data	
	Duplicate data	
	Incorrect value	The information is inaccessible
	Inconsistent data format	The information is insecure
	Outdated data	The information is hardly retrievable
	Incomplete data format	The information is difficult to aggregate
	Syntax violation	Errors in the information transformation
	Unique value violation	
	Violation of integrity constraints	
	Text formatting	
Context-dependent		The information is not based on fact
	Violation of domain constraints	The information is of doubtful credibility
	Violation of organization's business rules	The information presents an impartial view
	Violation of company and government regulations	The information is irrelevant to the work
		The information is incomplete
	Violation of constraints provided by the database administrator	The information is compactly represented
		The information is hard to manipulate
		The information is hard to understand

Table 1: Types of data quality problems [6]

2.3 Data Quality Dimensions

To define data quality dimension, a data quality framework is proposed by Cai and Zhu [7], as shown in Figure 2. With each dimension, there is underlying data

quality elements and indicators. Cai and Zhu [7] choose data quality dimensions commonly accepted and widely used as big data quality standards and redefined their basic concepts based on actual business needs. Each dimension was divided into many typical elements associated with it, and each element has its corresponding quality indicators: Accessibility, Timeliness, Authorization, Credibility, Definition/Documentation, MetaData, Accuracy, Consistency, Integrity, Completeness, Auditability, Fitness, Readability, and Structure. In this way, hierarchical quality standards for big data were used for evaluation. Figure 3 shows a universal two-layer data quality standard. Detailed correspondence dimension, element and indicators are shown in Figure 4.

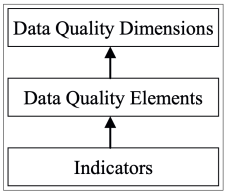

Fig. 2: Data quality framework [7]

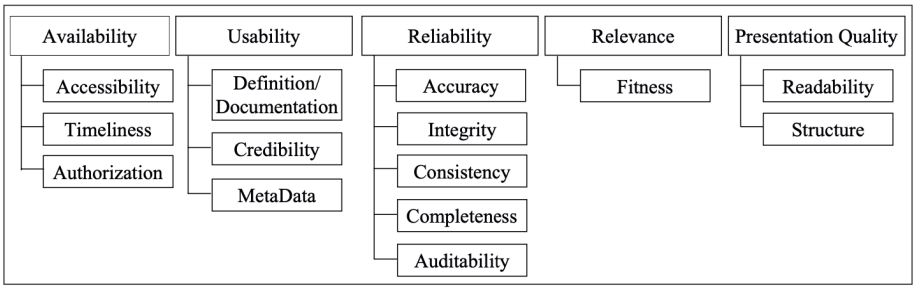

Fig. 3: A universal, two-layer big data quality standard for assessment [7]

Under the same dimension and elements, the indicators varies with type of data structure. For instance, syntactic accuracy is measured as described in Section 2.1 in the case of structured data. With semistructured data, the distance function should consider a global distance related to the shape of the XML tree in addition to the local distance of fields.

Dimensions	Elements	Indicators
1) Availability	1) Accessibility	■ Whether a data access interface is provided ■ Data can be easily made public or easy to purchase
	2) Timeliness	■ Within a given time, whether the data arrive on time ■ Whether data are regularly updated ■ Whether the time interval from data collection and processing to release meets requirements
2) Usability	1) Credibility	■ Data come from specialized organizations of a country, field, or industry ■ Experts or specialists regularly audit and check the correctness of the data content ■ Data exist in the range of known or acceptable values
3) Reliability	1) Accuracy	■ Data provided are accurate ■ Data representation (or value) well reflects the true state of the source information ■ Information (data) representation will not cause ambiguity
	2) Consistency	■ After data have been processed, their concepts, value domains, and formats still match as before processing ■ During a certain time, data remain consistent and verifiable ■ Data and the data from other data sources are consistent or verifiable
	3) Integrity	■ Data format is clear and meets the criteria ■ Data are consistent with structural integrity ■ Data are consistent with content integrity
	4) Completeness	■ Whether the deficiency of a component will impact use of the data for data with multi-components ■ Whether the deficiency of a component will impact data accuracy and integrity
4) Relevance	1) Fitness	■ The data collected do not completely match the theme, but they expound one aspect ■ Most datasets retrieved are within the retrieval theme users need ■ Information theme provides matches with users' retrieval theme
5) Presentation Quality	1) Readability	■ Data (content, format, etc.) are clear and understandable ■ It is easy to judge that the data provided meet needs ■ Data description, classification, and coding content satisfy specification and are easy to understand

Fig. 4: The hierarchical big data quality assessment framework [7]

3 Assessment Methodology Overview

This section give the overview of DQ assessment methodology with the perspectives proposed by Batini et al. [3]. The DQ assessment methodologies can be analyzed and compared through several perspectives in the following:

- Phases and Steps that compose the methodology
- Strategies and Techniques that are adopted in the methodology for assessing and improving data quality levels;
- Types of Costs, which are associated with data quality issues

Methodologies differ in how they consider all of these perspectives. Based on these perspectives, the classification of the DQ method proposed by Batini et al. [3] is presented.

3.1 Phases and Steps

In the most general case, the sequence of activities of a data quality methodology is composed of three phases [3].

1. **State Reconstruction**, which is aimed at collecting contextual information on organizational processes and services, data collections and related management procedures, quality issues and corresponding costs; this phase can be skipped if contextual information is available from previous analyses.
2. **Assessment**, which measures the quality of data collections along relevant quality dimensions; the term measurement is used to address the issue of measuring the value of a set of data quality dimensions.
3. **Improvement**, which concerns the selection of the steps, strategies, and techniques for reaching new data quality targets.

For example, a DQ assessment method called, Comprehensive Data Quality management (CDQ), employed three of the phases in its method, as shown in Figure 5.

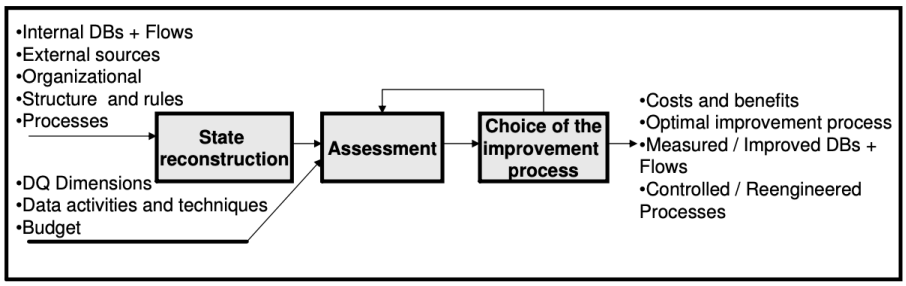

Fig. 5: CDQ Phases [2]

The assessment and improvement phases of the DQ methodologies are organized in the framework of a common set of basic steps [3]. The steps of the assessment phase are:

- Data Analysis, which examines data schemas and performs interviews to reach a complete understanding of data and related architectural and management rules;
- DQ Requirements Analysis, which surveys the opinion of data users and administrators to identify quality issues and set new quality targets;
- Identification of Critical Areas, which selects the most relevant databases and data flows to be assessed quantitatively;
- Process Modeling, which provides a model of the processes producing or updating data;
- Measurement of Quality, which selects the quality dimensions affected by the quality issues identified in the DQ requirements analysis step and defines corresponding metrics.

Metadata plays a relevant role in all stages of the assessment phase. Metadata often provide the information necessary to understand data and/or evaluate them [3].

The steps of the improvement phase includes:

- Evaluation of Costs, which estimates the direct and indirect costs of data quality;
- Assignment of Process responsibilities, which identifies the process owners and defines their responsibilities on data production and management activities;
- Assignment of Data Responsibilities, which identifies the data owners and defines their data management responsibilities;
- Identification of the Causes of Errors, which identifies the causes of quality problems;
- Selection of Strategies and Techniques, which identifies all the data improvement strategies and corresponding techniques, that comply with contextual knowledge, quality objectives, and budget constraints;

Other common steps are Design of Data Improvement Solutions, Process Control, Process Redesign, Improvement Management, Improvement Monitoring.

3.2 Strategies and Techniques

Batini et al. [3] summarize two types of strategies: data-driven and process-driven. Data-driven strategies improve the quality of data by directly modifying the value of data and Process-driven strategies improve quality by redesigning the processes that create or modify data. The common Data-Driven Techniques includes:

- Acquisition of New Data, which improves data by acquiring higher-quality data to replace the values that raise quality problems.

– Standardization, which replaces or complements nonstandard data values with corresponding values that comply with the standard. For example, nicknames are replaced with corresponding names, for example, Bob with Robert, and abbreviations are replaced with corresponding full names, for example, Channel Str. with Channel Street.
– Record Linkage, which identifies that data representations in two (or multiple) tables that might refer to the same real-world object;
– Data and Schema Integration, which define a unified view of the data provided by heterogeneous data sources. Integration has the main purpose of allowing a user to access the data stored by heterogeneous data sources through a unified view of these data.
– Source Trustworthiness, which selects data sources on the basis of the quality of their data;
– Error Localization and Correction, which identify and eliminate data quality errors by detecting the records that do not satisfy a given set of quality rules. These techniques are mainly in the statistical domain.
– Cost optimization, defines quality improvement actions along a set of dimensions by minimizing costs.

Borek et al. [6] also summarized other data-driven techniques, such as Column analysis, Cross-domain analysis, Data validation, Domain analysis,Lexical analysis ,Matching algorithms: identify duplicates ,Primary key and foreign key analysis (PK/FK analysis) : are good candidates for a PK/FK ,Schema matching: two attributes are semantically equivalent, and Semantic profiling.

The common Process-Driven Techniques includes:

– Process control inserts checks and control procedures in the data production process when: (1) new data is created, (2) data sets are updated, or (3) new data sets are accessed by the process. In this way, a reactive strategy is applied to data modification events, thus avoiding data degradation and error propagation.
– Process redesign redesigns processes in order to remove the causes of poor quality and introduces new activities that produce data of higher quality.

3.3 Cost

The cost of a data quality program can be considered a preventive cost that is incurred by organizations to reduce data errors [3]. This cost category includes the cost of all phases and steps that compose a data quality assessment and improvement process. The costs of poor quality can be classified as:

– Process Costs, such as the costs associated with the re-execution of the whole process due to data errors;
– Opportunity Costs, which is due to lost and missed revenues.

The cost of poor data quality is strongly context-dependent as opposed to the cost of a data quality program. This makes its evaluation particularly difficult,

as the same data value and corresponding level of quality has a different impact depending on the recipient. For example, an active trader receiving obsolete information on a stock may incur considerable economic losses as a consequence of wrong investment decisions. In contrast, a newspaper receiving the same obsolete information to publish monthly trading reports may not experience any economic loss.

3.4 Classification of Methodologies

Methodologies can be classified into four categories based on their emphasis on improvement, assessment, cost and quality dimensions [3]. The categories are:

1. Complete methodologies, which provide support to both the assessment and improvement phases, and address both technical and economic issues;
2. Audit methodologies, which focus on the assessment phase and provide limited support to the improvement phase;
3. Operational methodologies, which focus on the technical issues of both the assessment and improvement phases, but do not address economic issues.
4. Economic methodologies, which focus on the evaluation of costs.

Figure 6 shows the methods positioned on a two-dimensional coordinated proposed by Batini et al. [3]. The acronyms, extended names and main references are shown in Table 2.

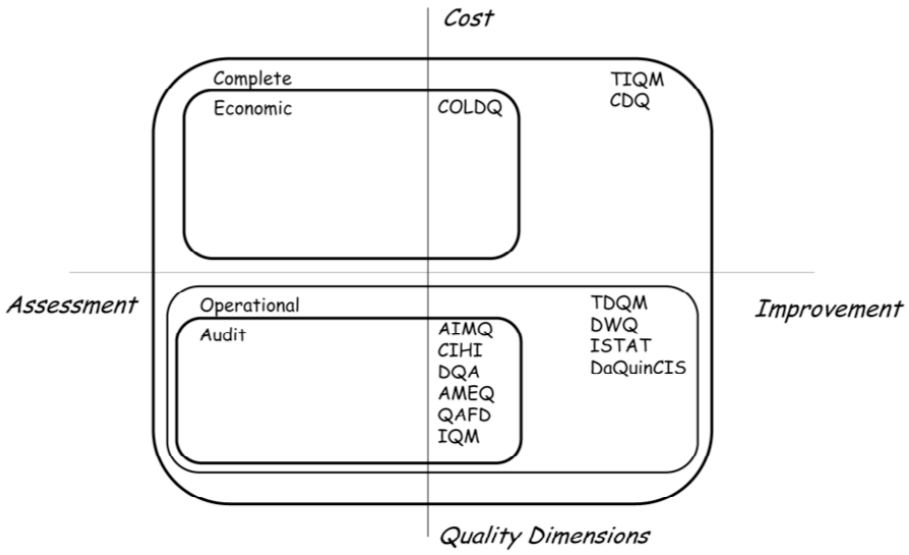

Fig. 6: A classification of methodologies [3]

Acronym	Extended Name	Reference
TDQM	Total Data Quality Management	Wang [24]
DWQ	The Datawarehouse Quality Methodology	Jeusfeld et al. [12]
TIQM	Total Information Quality Management	English [9]
AIMQ	A methodology for information quality assessment	Lee et al. [14]
CIHI	Canadian Institute for Health Information methodology	Long and Seko [15]
DQA	Data Quality Assessment	Pipino et al. [19]
IQM	Information Quality Measurement	Eppler and Muenzenmayer [10]
ISTAT	ISTAT methodology	Falorsi and Scannapieco [11]
AMEQ	Activity-based Measuring and Evaluating of product information Quality (AMEQ) methodology	Su and Jin [22]
COLDQ	Loshin Methodology (Cost-effect Of Low Data Quality)	Loshin [16]
DaQuinCIS	Data Quality in Cooperative Information Systems	Scannapieco and Pierce [21]
QAFD	Methodology for the Quality Assessment of Financial Data	Amicis et al. [1]
CDQ	Comprehensive methodology for Data Quality management	Batini et al. [2]

Table 2: Methodologies Acronyms in Figure 6 [3]

4 DQ in Context of Continuous Integration

In this section, the use of DQ in the context of continuous integration (CI) is introduced. The motivation behind this section is to present an application of DQ on existing data in the Software Engineering domain. An existing open-source data set, TravisTorrent, is later introduced followed by its corresponding research.

CI has become a best practice of modern software development [4]. However, despite its prominent role in Software Engineering in practice, the benefits, costs, and implications of doing CI are all but clear from an academic viewpoint. Little research has been done, and even less was of quantitative nature [4].

From an academic standpoint, Beller et al. [5] think that they still lack quantifiable evidence on the implications of introducing and continuing to use CI. Beller et al. [5] have used TravisTorrent to drive the first investigation into CI testing practices for answering questions:

- Does the use of CI lead to higher-quality products, for example by catching bugs and test failures earlier?
- Does CI lead to fewer test regressions?
- What are CI best practices that successful projects employ (build more often, fail more often, have more tests in the CI)?
- Do CI-enabled projects switch to Continuous Delivery?
- How long may a CI build run still be considered helpful?
- Does a broken build negatively affect other developers' productivity, as is often claimed?

For the application of the data set, three works are introduced [8, 18, 17]. Dimitropoulos et al. [8] analyze this data set and explore the features in order to get the information about the factors that affect build breakage. Madeyski and Jureczko [17] presents an empirical evaluation in which several process metrics were investigated to identify the ones which significantly improve the defect

prediction models based on product metrics. Madeyski and Jureczko [17] extends to TravisTorrent data set with the same process metric with additional parsing and mapping [18].

4.1 TravisTorrent Data Set

TravisTorrent is a freely available data set based on Travis CI and GitHub. Thanks in part to Travis CI's tight integration with GitHub, Travis CI has emerged as arguably the most widely used CI platform for Open-Source Software (OSS) development. TravisTorrent provides easy access to hundreds of thousands of analyzed builds from more than 1,000 projects. Unique to TravisTorrent is that each of its 2,640,825 Travis builds is synthesized with metadata from Travis CI's API, the results of analyzing its textual build log, a link to the GitHub commit which triggered the build, and dynamically aggregated project data from the time of commit extracted through GHTorrent [5].

The TravisTorrent Data Set From the 17,313,330 active OSS repositories on GitHub in August 2015, the data set contains a deep analysis of the project source code, process, and dependency status of 1,300 projects. The data set is restricted to all non-fork, non-toy, somewhat popular ($>$ 10 watchers on GitHub) projects with a history of Travis CI use ($>$ 50 builds) in Ruby (898) or Java (402). Both languages are very popular on GitHub (2nd and 3rd, respectively) [4]. Then, Beller et al. [5] extracted and analyzed build information from Travis CI build logs and the GHTorrent database for each Travis CI build in its history.

TravisTorrent provides convenient access to its archived data sets and free analytic resources: Researchers can directly access an in-browser SQL shell to run their queries on their infrastructure, and download SQL dumps or the compressed data set as a CSV file (1.8 GB unpacked). It also provides documentation and a getting started tutorial. All TravisTorrent tools and data are in the public domain [5].

Data Structure In the TravisTorrent data set, each data point (row) represents a build job executed on Travis. Every such data point synthesizes information from three different sources: The project's git repository (prefixed git_), data extracted from GitHub through GHTorrent (prefixed gh_), and data from Travis's API and an analysis of the build log (prefixed tr_). In total, 55 data fields are provided for each build job [5]. These are described in detail in Table 7.

Data Collection Data Collection is done by TravisPoker and TravisHarvester. TravisPoker is a fast and lightweight application that takes a GitHub project name as input (for example, RAILS/RAILS), and finds out if and how many Travis CI builds were executed for this project. TravisHarvester aggregates detailed information about a project's Travis CI build history. It takes as input a GitHub project name and gathers general statistics on each build in the project's history in a CSV file [5].

Column Name	Description	Unit
row	Unique identifier for a build job in TravisTorrent	Integer
git_commit	SHA1 Hash of the commit which triggered this build (should be unique world-wide)	String
git_merged_with	If this commit sits on a Pull Request (gh_is_pr true), the SHA1 of the commit that merged said pull request	String
git_branch	Branch git_commit was committed on	String
git_commits	All commits included in the push that triggered the build, minus the built commit	List of Strings
git_num_commits	The number of commits in git_commits, to ease efficient splitting	String
git_num_committers	Number of people who committed to this project	Integer
gh_project_name	Project name on GitHub (in format user/repository)	String
gh_is_pr	Whether this build was triggered as part of a pull request on GitHub	Boolean
gh_lang	Dominant repository language, according to GitHub	String
gh_first_commit_created_at	Timestamp of first commit in the push that triggered the build	ISO Date (UTC+1)
gh_team_size	Size of the team contributing to this project within 3 months of last commit	Integer
gh_num_issue_comments	If git_commit is linked to a PR on GitHub, the number of comments on that PR	Integer
gh_num_commit_comments	The number of comments on git_commits on GitHub	Integer
gh_num_pr_comments	If gh_is_pr is true, the number of comments on this pull request on GitHub	Integer
gh_src_churn	How much (lines) production code changed by the new commits in this build	Integer
gh_test_churn	How much (lines) test code changed by the new commits in this build	Integer
gh_files_added	Number of files added by the new commits in this build	Integer
gh_files_deleted	Number of files deleted by the new commits in this build	Integer
gh_files_modified	Number of files modified by the new commits in this build	Integer
gh_tests_added	Lines of testing code added by the new commits in this build	Integer
gh_tests_deleted	Lines of testing code deleted by the new commits in this build	Integer
gh_src_files	Number of production files in the new commits in this build	Integer
gh_doc_files	Number of documentation files in the new commits in this build	Integer
gh_other_files	Number of remaining files which are neither production code nor documentation in the new commits in this build	Integer
gh_commits_on_files_touched	Number of unique commits on the files included in this build within 3 months of last commit	93
gh_sloc	Number of executable production source lines of code, in the entire repository	Integer
gh_test_lines_per_kloc	Test density. Number of lines in test cases per 1,000 gh_sloc	Double
gh_test_cases_per_kloc	Test density. Number of test cases per 1,000 gh_sloc	Double
gh_asserts_cases_per_kloc	Assert density. Number of assertions per 1,000 gh_sloc	Double
gh_by_core_team_member	Whether this commit was authored by a core team member	Boolean
gh_description_complexity	If gh_is_pr is true, the total number of words in the pull request title and description	Integer
gh_pull_req_num	Pull request number on GitHub	Integer
tr_build_id	Unique build ID on Travis	String
tr_status	Build status (pass, fail, errored, canceled)	String

Fig. 7: TravisTorrent data structure (partial view) [5]

4.2 Application of TravisTorrent

The use of TravisTorrent data set can be found in [8] and [18]. Dimitropoulos et al. [8] analyze the data set and explore the features to get the information about the factors that affect build breakage. Dimitropoulos et al. [8] uses k-means++ clustering model and logistic regression for build status analysis. The research goal of Madeyski and Kawalerowicz [18] is to propose Continuous Defect Prediction (CDP) practice supported by a toolset using machine learning (ML)-based prediction models and large datasets to predict defect-prone software changes. At the moment, their target is limited to success/fail continuous integration outcomes. Madeyski and Kawalerowicz [18] first extends this data to a file change level and calculate the software process metrics that may be used, features to predict risky software changes that could break the build if committed to a repository with CI enabled.

Preprocessing To efficiently find the reasons why builds break, Dimitropoulos et al. [8] applied modifications in the given data set as follows:

- All the duplicates are erased because after investigating the data set they identified that many of the rows were identical.
- Select only 28 relevant data fields plus the build status out of 55 data fields.
- Converted all the nominal features to numeric to be able to efficiently apply data mining algorithms.
- Removed all the rows which the builds had no tests executions, had values of specific fields set as NaN (not a number), and had values of specific fields that should be positive (e.g., build duration) set as negative.

Data preprocessing flowchart is shown in Figure 8.

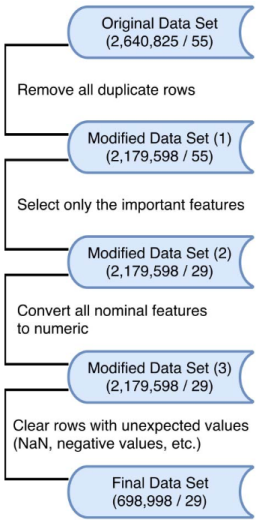

Fig. 8: Data preprocessing flowchart. The values inside the parenthesis represent the volume of the data in rows and columns [8]

Measurement Dimitropoulos et al. [8] use a novel approach that combines two different methods to get a broader understanding and to secure their findings. Firstly, they use the k-means++ clustering method to partition 28 predictors into 4 clusters (passed, failed, errored, canceled) and get a general view of the depending features that affect the build status. Secondly, they use Logistic Regression to model the status of the builds.

Madeyski and Kawalerowicz [18] proposed the use of process metrics as their primary study. Considerable research has been performed on identifying the process metrics which influence the efficiency of defect prediction [17]. Product metrics describe the size and design complexity of software, served as the basis, and the point of departure. The product metrics were used to build simple defect prediction models, while the product metrics, together with the selected process metrics (one at a time), were used to build the advanced models [17]. The most widely used are as following:

- The number of Revisions (NR). The NR metric constitutes the number of revisions (retrieved from a mainline of development in a version control system, e.g., trunk in SVN) of a given Java class during the development of the investigated release of a software system.
- The number of Distinct Committers (NDC). The NDC metric returns the number of distinct authors who committed their changes in a given Java class during the development of the investigated release of a software system.
- The number of Modified Lines (NML). The NML metric calculates the sum of all lines of source code that were added or removed in a given Java class. Each of the committed revisions during the development of the investigated

release of a software system is taken into account. According to the CVS versionâcontrol system, a modification in a given line of source code is equivalent to removing the old version and subsequently adding a new version of the line.
- The Number of Defects in the Previous Version (NDPV). The NDPV metric returns the number of defects repaired in a given class during the development of the previous release of a software system.

Analysis Through k-means++ clustering model, Dimitropoulos et al. [8] found out the following rationales for each cluster:

- Pass: passed builds occur more often when they are committed to the master branch.
- Fail: Failed builds mostly occur when they are not pull requests, they are committed on non-master branches, they have a low amount of issue comments and their test duration is short.
- Error: Errored builds have a massive amount of duration on their tests.
- Cancel: Canceled builds tend to have a small team size, a small number of issue comments, and large changes on the source and test churn. Also, they likely have a small number of tests run and their duration, which is trivial, since their execution is stopped earlier than the predetermined.

Logistic regression models showed that large changes on projects' churn and files surprisingly affect the build status. For example, the bigger the changes on the test churn, the more likely is for a build to pass. Moreover, the larger the number of test runs, the more the possibilities for a build to be successful. It is identified that builds that contain a high number of assert cases and a low number of test cases tend to break more often. Furthermore, the time duration factor seems to be irrelevant to the build status [8].

According to Madeyski and Kawalerowicz [18], they have two findings regarding commit time:

- For the larger projects commits that trigger the CI build are done later in day time, whereas they are done earlier in the average.
- The open source projects integrations are done generally well before 8 PM so on average open source developers are not night owls as they are usually perceived. Actually, the builds are done based on commits done in normal business hours, between 9 AM and 5 PM.

Madeyski and Jureczko [17] later extends to TravisTorrent data set with the same metric with additional parsing and mapping. However, they didn't show further finding regarding process metric with TravisTorrent data set [18].

The findings for process metrics are based on their internal data set, which consists of forty-three releases of 12 open source and 27 releases of 6 industrial software projects [17]. Madeyski and Jureczko [17] found the NDC and NML metrics were valuable about defect prediction. Scatter plots of number of defects against process metrics is shown in Figure 9. Both metrics significantly improved

the efficiency of prediction and, therefore, are worth taking into account while building the defect prediction models. In the case of the NR and NDPV metrics, no statistically significant results were obtained. The NR metric improved the model, but the difference was small. The NDPV metric only slightly affected the model. Defect prediction models that utilize four different process metrics (NR, NDC, NML, and NDPV) were investigated as well. The models usually gave better predictions than the models which did not use the process metrics at all and the difference was statistically [17].

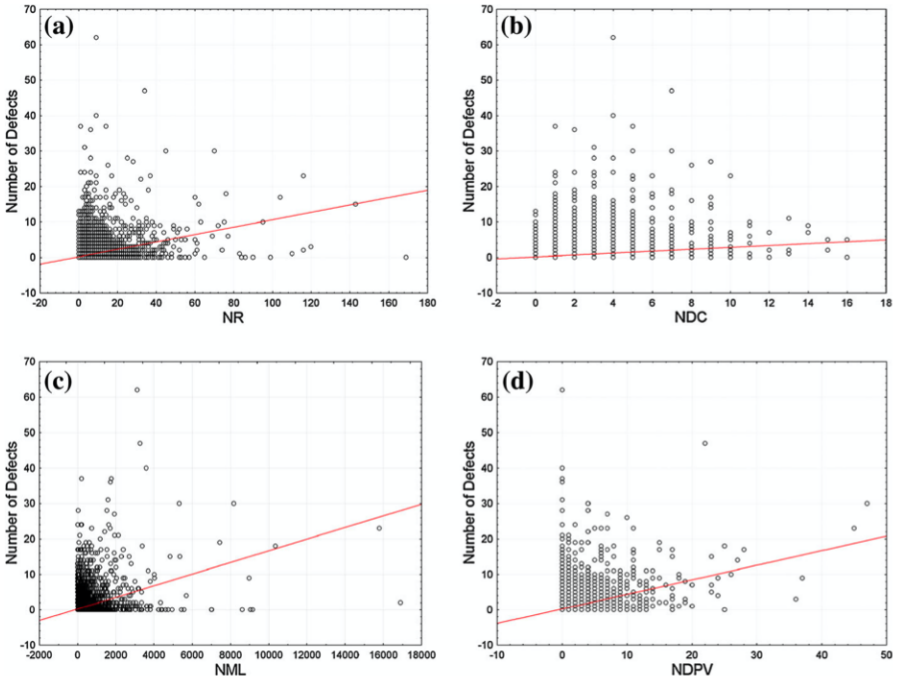

Fig. 9: Scatter-plots of number of defects against process metrics. (a) NR (b) NDC (c) NML (d) NDPV [17]

The summary of their results are presented in Table 3. Especially interesting are the NDC and NML metrics. The NDC metric regards the number of different developers who have changed a given class. When several people are involved in the process of development, it is always probable that they will misunderstand one another's intentions and overwrite the modifications made by their colleagues, which may, in turn, result in defects [17].

The NML metric reflects the size of the change and according to the obtained results, the more lines of code are affected, the greater number of defects to be expected. The early approaches to defect prediction postulated a linear relation

Metric	Prediction efficiency	Statistically significant
Number of Revisions (NR)	Better	No
Number of Distinct Committers (NDC)	Better	Yes
Number of Modified Lines (NML)	Better	Yes
Number of Defects in Previous Version (NDPV)	Not better	No
Combination of process metrics	Better	Yes

Table 3: Metrics evaluations [17]

between the lines of code (LOC) and the number of defects. The result obtained in the case of the NML metric partly moves this rule to the world of process metrics. There is a relation between the size of code and defectiveness, but it turns out that taking into account only the recently changed code gives better results [17].

5 Conclusion

This paper first introduces the fundamentals of data type, DQ problems, and dimensions. This clarifies the first research topic of data properties which can be analyzed and cannot be analyzed. In the second section, the overview of DQ assessment methods is presented. The pipelines, phases, steps, strategy, techniques, cost, and classification of general assessment methods are described. This answers the second research topic. In the same section, two improvement strategies, data-driven and process-driven, are presented along with corresponding techniques. This addresses the research topic of improving DQ. In the third section, an existing open-source data set, TravisTorrent, is introduced followed by its corresponding study. Based on the research of Madeyski and Kawalerowicz [18], the two process metrics, the number of Distinct Committers (NDC) and the number of Modified Lines (NML), are considered valuable for defect prediction. The metrics are extended to TravisTorrent but no result is published. The research of Dimitropoulos et al. [8] showed rationales for build status using the clustering model. They also demonstrate preprocessing of TravisTorrent, which is an example of stage reconstruction in the DQ assessment phase. This response to the research topic of application of the DQ assessment methods to an existing data set. The four research topics covered in this paper present the evaluation of the DQ assessment method.

Several existing challenges for data assessment in the Big Data era presented by Cai and Zhu [7]. According to Katal et al. [13], the characteristics of big data come down to the 4Vs: Volume, Velocity, Variety, and Value. Volume refers to the tremendous volume of the data. Velocity means that data are being formed at an unprecedented speed and must be dealt with promptly. The growth of the data sources in both size and scope consequently has significantly increased the complexity of data quality management. Variety indicates that big data has all kinds of data types, and these multitype data require higher data processing

capabilities. Finally, Value represents low-value density. Value density is inversely proportional to total data size, the greater the big data scale, the less relatively valuable the data. Consequently, the processing cost of the DQ assessment will gain significant importance. How to assess data quality in a fast and efficient way is another topic Taleb et al. [23].

In the end, according to Pipino et al. [19], experience suggests there is no "one size fits all" DQ methodology. From Data Quality Fundamentals to DQ in Context of Continuous Integration, it is shown that DQ assessment varies depending on application and domain. Therefore, assessing data quality is an on-going effort that requires awareness of the fundamental principles underlying the development and context of the application [19].

As for the future work, according to Madeyski and Kawalerowicz [18], they are going to release the whole CDP project together with tools used to gather the data from open source and commercial projects. They hope that by exposing their prediction models over the web in a ready-to-use state the model can aid the development of open-source projects. Moreover, they hope to enrich the collected dataset with more metrics proposed by other researchers from empirical study. The newly-introduced metrics can be used as features to the prediction models. They also hope to extend the research to other areas such as defect prediction on software change level, stability and maturity studies on long-running software projects, developers activity and result examination, or exploration of trends in continuous integration over a period of time.

References

[1] F. Amicis, D. Barone, and C. Batini. An analytical framework to analyze dependencies among data quality dimensions. pages 369–383, 01 2006.

[2] C. Batini, F. Cabitza, C. Cappiello, and C. Francalanci. A comprehensive data quality methodology for web and structured data. 1:205–218, 07 2008.

[3] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. volume 41, New York, NY, USA, July 2009. Association for Computing Machinery.

[4] M. Beller, G. Gousios, and A. Zaidman. Oops, my tests broke the build: An explorative analysis of travis ci with github. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 356–367, 2017.

[5] M. Beller, G. Gousios, and A. Zaidman. Travistorrent: Synthesizing travis ci and github for full-stack research on continuous integration. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 447–450, 2017.

[6] A. Borek, P. Woodall, M. Oberhofer, and A. K. Parlikad. A classification of data quality assessment methods. 01 2011.

[7] L. Cai and Y. Zhu. The challenges of data quality and data quality assessment in the big data era. In *Data Science Journal*, page 2, 2015.

[8] P. Dimitropoulos, Z. Aung, and D. Svetinovic. Continuous integration build breakage rationale: Travis data case study. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pages 639–645, 2017.

[9] L. P. English. *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits*. John Wiley amp; Sons, Inc., USA, 1999. ISBN 0471253839.

[10] M. J. Eppler and P. Muenzenmayer. Measuring information quality in the web context: A survey of state-of-the-art instruments and an application methodology. In *ICIQ*, 2002.

[11] P. S. P. A. A. A. M. E. Falorsi, P. and M. Scannapieco. Improving the quality of toponymic data in the italian public administration. pages 238–250, 01 2003.

[12] M. Jeusfeld, C. Quix, and M. Jarke. Design and analysis of quality information for data warehouses. 12 1999.

[13] A. Katal, M. Wazid, and R. H. Goudar. Big data: Issues, challenges, tools and good practices. In *2013 Sixth International Conference on Contemporary Computing (IC3)*, pages 404–409, 2013.

[14] Y. Lee, D. Strong, B. Kahn, and R. Wang. Aimq: A methodology for information quality assessment. *Information Management*, 40:133–146, 12 2002.

[15] J. Long and C. Seko. A new method for database data quality evaluation at the canadian institute for health information (cihi). pages 238–250, 01 2002.

[16] D. Loshin. *Enterprise Knowledge Management - The Data Quality Approach.* 2004.

[17] L. Madeyski and M. Jureczko. Which process metrics can significantly improve defect prediction models? an empirical study. *Software Quality Journal*, 23(3):393–422, sep 2015. ISSN 0963-9314.

[18] L. Madeyski and M. Kawalerowicz. Continuous defect prediction: The idea and a related dataset. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 515–518, 2017.

[19] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Commun. ACM*, 45(4):211â218, Apr. 2002. ISSN 0001-0782.

[20] Y. W. L. Pipino, Leo L. and R. Y. Wang. Data quality assessment. In *Communications of the ACM 45.4*, pages 211–218, 2002.

[21] M. Scannapieco and E. Pierce. Ip-uml: Towards a methodology for quality improvement based on the ip-map framework. pages 279–291, 01 2002.

[22] Z. Su and Z. Jin. *A Methodology for Information Quality Assessment in the Designing and Manufacturing Processes of Mechanical Products*. 01 2006.

[23] I. Taleb, H. T. E. Kassabi, M. A. Serhani, R. Dssouli, and C. Bouhaddioui. Big data quality: A quality dimensions evaluation. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 759–765, 2016.

[24] R. Y. Wang. A product perspective on total data quality management. *Commun. ACM*, 41(2):58â65, Feb. 1998. ISSN 0001-0782.