Towards a Model and Methodology for Evaluating Data Quality in Software Engineering Experiments

Carolina Valverde

Universidad de la República, Montevideo, Uruguay, mvalverde@fing.edu.uy

Adriana Marotta

Universidad de la República, Montevideo, Uruguay, amarotta@fing.edu.uy

José Ignacio Panach

Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València,Valencia, España, joigpana@uv.es

Diego Vallespir

Universidad de la República, Montevideo, Uruguay, dvallesp@fing.edu.uy

Context. Data collected during software engineering experiments might contain quality problems, leading to wrong experimental conclusions. **Objective.** We present a data quality (DQ) model and a methodology specific to software engineering experiments, which provides a systematic approach in order to analyze and improve data quality in this domain. **Method.** Our proposal considers a multifaceted view of data quality suitable for this context, which enables the discovery of DQ problems that are not generally addressed. We successfully applied the model (DQMoS) and methodology (DQMeS) in four controlled experiments, detecting different quality problems that could impact the experimental results. We present, through a running example, how we applied the DQMoS and DQMeS to one of the four experimental data. **Results.** We found that between 55% and 75% of the DQ metrics applied showed the presence of a DQ problem in all four experiments. In all cases, the experimental results had already been obtained before the DQMeS application. This means that the DQ problems we found, were not discovered by the experimenters during or before making their experiment's analysis. Results yield data quality problems that experimenters did not detect on their own analysis, and that affect the experimental response variables. Our proposal shows a formalized framework that measures and improves the quality of software engineering experimental data. The results of a survey distributed to the experiments' responsibles show that they value the improvements introduced by the model and methodology, and that they intend to apply them again in future experiences. **Conclusions**. DQMoS and DQMeS are useful to increase the confidence in the quality of data used in software engineering experiments, and improve the trust in experimental results.

Additional Keywords and Phrases: data quality, data quality model, data quality methodology, software engineering experiments.

1 INTRODUCTION

During Software Engineering Experiments (SEE) data are collected and used to obtain experimental results. The research community and professionals in SE use these results to improve and adjust their own research projects and professional practices [27, 62]. Data collected in experiments may contain different Data Quality (DQ) problems, which may impact on the experimental results. In particular, in SEE where human subjects are responsible for recording the data, DQ becomes an even more relevant issue and should be carefully considered [5, 31].

DQ research area has focused on defining different DQ aspects and on proposing techniques, methods and methodologies for measuring and dealing with DQ problems [1, 9, 53, 57]. DQ concepts have been applied to different kinds of data belonging to different domains, such as financial, business and organizational [22, 40, 63], web portals [17], bio-medicine [24] and sensors [37], psychology and medical research [64, 65, 66] among others. In these domains, data producers and consumers have recognized DQ problems as an important matter that needs to be considered and attended [38, 46, 47].

Inappropriate levels of quality in scientific data can seriously affect the results obtained. If data contain DQ problems, the decisions made by researchers and professionals (based on this data) might be wrong and lead to inaccurate actions. Some works state the importance of assessing and improving the quality of the data used in Empirical Software Engineering [14, 33, 49, 55]. However, the quality of the data used in SEE is seldom questioned or analyzed, as DQ problems have not received the attention it deserves in this area [15, 34, 36, 52]. In the few cases DQ is considered, the analysis of the quality of the data is normally ad-hoc; i.e. neither a systematic nor a repeatable method is used [31, 49].

Besides, there is a need for defining and using standardized concepts and methods, due to a lack of clear and consistent terminology regarding DQ in SEE domain. Bachmann [3] suggests the development of unified DQ methodologies for the empirical software engineering community, since none of the works found propose or apply one. Rosli *et. al.* [49, 51] suggest, after conducting a mapping study, there is a need to construct a formal definition and to use a standard terminology for DQ issues. They state that DQ terms are used inconsistently, especially regarding the identification of DQ problems. The research community needs to define common terms in order to better understand and communicate DQ problems. Other authors [56] claim that there is no industrial standard practice to evaluate DQ in this context, only guidelines for improving the selection of datasets during the data collection process.

Liebchen and Shepperd [34] after reviewing the findings of their study in 2008 [33] and revisiting the works of eight years, state that there is still a need to develop systematic mechanisms and protocols to assess DQ problems. Even though there seems to be a greater concern in this topic, the community still does not deal with it and would benefit from the usage of unified protocols and approaches to manage DQ.

In this work we present the DQ Methodology and DQ Model that we developed in order to analyze and improve DQ in the SEE domain. Our proposal aims to evaluate the quality of the reported (raw) data that is used to obtain the experimental results. We aim to check that analysed data has enough quality, ensuring that experimental results are properly reported. Other quality features related with the overall quality of the experiment, such as quality related with the experimental design, to mitigate as many threats to validity as possible, or in the process to train subjects; are out of scope of this paper.

The DQ Methodology (DQMeS) uses the DQ Model and provides a systematic approach to enhance DQ. The DQ Model (DQMoS) represents the main DQ concepts that are used and applied to improve data quality in this domain. The aim of the DQMoS and DQMeS application is to analyze and improve the quality of the data resulting from the execution of SEE that involve humans as subjects. It is out of the scope of this work the analysis of computational experiments or any other type of experiments different from SEE with humans, as this will probably result in a different DQMoS than the one proposed in this article. As we did not prove our DQ model and methodology in other SEE that did not involve human as subjects, we cannot assure neither refuse whether our proposal could be applied to other domains.

The DQ concepts used in DQMoS are represented through DQ dimensions, factors and metrics. Dimensions and factors are selected from the DQ research area [9, 57, 60, 61] and are generic, as they are not specific for SEE and could be applied to other domains. Meanwhile, DQ metrics proposed in DQMoS are specific for the SEE domain, since they specify concrete ways for measuring quality to specific types of data objects. The model consists of 6 dimensions, 14 factors and 21 metrics, each metric including: a description, a definition, a measurement result unit and a granularity. As this model (and methodology) includes generic and domain-specific parts, we cannot say a priori whether it is also adequate and can be adapted to other domains or not.

In this work we assume the experiment datasets as conforming to tabular format, i.e., structured in columns and rows, where each row corresponds to an element of interest and each column corresponds to a property of the elements. For example, in a certain dataset, each row may contain information about an execution of the experiment by certain subject, and the columns may be *start time*, *end time*, etc. We use the term *attribute* for a column of the table, following relational databases terminology. During data quality evaluation, the DQ metrics are instantiated to attributes or to sets of attributes of the specific SEE experiment dataset. DQ problems found by the application of these metrics are classified as data errors, questionable values or improvement opportunities.

The DQMeS provides guidelines in order to systematically analyze and improve the quality of experimental data of any SEE. We define three roles: Data Quality Analyst, Experimenter, and DQMoS and DQMeS Responsible Person. The Data Quality Analyst is responsible for conducting the data quality analysis, performing all the steps of the methodology. The Experimenter is the person involved with the SEE and the person who validates the data quality actions. DQMoS and DQMeS Responsible Person is the role accountable for the model and methodology development and maintenance. The DQMeS is composed of four steps: 1) Elicit knowledge about the Experiment, 2) Instantiate DQMeS, 3) Assess Data Quality, and 4) Execute corrective actions. These steps provide help in using the DQMoS correctly.

We built and improved the DQMoS and DQMeS incrementally, through its successive applications to different SEE. We applied the first version of the DQMoS and DQMeS to one SEE data, and as a result, we adjusted our model including new DQ metrics identified. We repeated this procedure with four SEE data until we had our initial DQMoS and DQMeS that are presented here. The applications of our approach to experimental data were done in order to: find support for applying the model and methodology in the SE domain, measure and improve the quality of experimental data, and define metrics that could be generalized to any SEE. In these four cases, our approach contributed in finding and cleaning several DQ problems providing more reliable experimental analysis and results. During the applications we verified the validity of the model and methodology proposed, and improved and extended the metrics applied in each case.

In all four cases, the experimental results had already been obtained before the DQMeS application. This means that the DQ problems we found, were not discovered by the experimenters during or before making their experiment's analysis. The ad hoc approach the researchers followed to assess the experimental data, was less effective than applying the DQMoS. This shows how there is an improvement to the experimental DQ by applying DQMeS, as we discovered data errors that the researchers had not found before.

In a previous work [59], we presented how we applied the first draft version of the DQ model to two of the four experiments, showing the definition and application of one DQ metric as example. In this paper we present the complete DQMoS and DQMeS, including all 21 metrics and its application in the four experiments.

The paper is structured as follows. Section 2 presents our proposal, including the DQ concepts as the conceptual base for the DQMoS, as well as the definition of the DQMoS and the DQMeS. We also use a running example in order to clarify the proposal and the concepts presented. Section 3 shows the results of applying the DQMoS and DQMeS to four SEE. Finally, section 4 compares related work and section 5 presents the conclusions.

2 RELATED WORK

The importance of the quality of data used by empirical studies has been acknowledged in the last years [5, 6, 13, 14, 15, 18, 29, 31, 33, 34, 36, 49, 51, 52, 55] mostly due to the impact that it may have on the decisions made. Some papers explicitly emphasize the importance of DQ in empirical software engineering datasets, as data imperfections can have unwanted impact on the data analysis and might lead to false conclusions [31, 35, 36, 49, 51, 52].

Four literature reviews were carried out in this particular topic, showing interest and concern about how researchers are dealing with DQ problems [3, 12, 31, 49]. They all conclude that empirical software engineering community should pay more attention to this issue, which has been long neglected according to the results. Results of the systematic literature review carried out by Liebchen and Shepperd [31, 33] show that only 1% of analyzed papers explicitly consider noise or DQ as an issue, not necessarily proposing solutions. Even though the majority of the publications in this review recognize its importance, considering DQ to be a threat to the analysis of empirical data (138 out of 161), little work has been done to deal with DQ problems. This results in questionable experiments conclusions, as they are obtained based on poor DQ. Liebchen [33] concludes "*DQ is somewhat neglected in a domain where the most important input is data".*

Bosu and MacDonell [14] carried out a targeted systematic literature review in order to identify evidence of: data collection reporting, data pre-processing and DQ issue identification in ESE studies. The results show that DQ is not a systematic practice in ESE, as just 11% of the 282 papers reviewed indicate some level of consideration in the three perspectives. Only 44% of the studies report consideration in any of the DQ perspectives, confirming the Liebchen and Shepperd [33] findings. The 56% of the reviewed studies (157 of 282) do not report on the quality of the datasets being used. The most frequently identified data quality issues are incompleteness (38%) and outliers (28%), even though causes are not addressed. Bachmann [3] finds as a result of a literature review that only a few publications cover the quality aspects of software engineering process data. Finally, Rosli et. al. [49, 51] performed a systematic mapping study in order to investigate how DQ issues are approached in SE research and to which extent researchers examine DQ issues that might affect their results. As a result, they find 64 papers that address DQ issues in this domain, but only 31 give serious consideration about how DQ impact on their results. They suggest "*researchers should give more attention to the quality of datasets in order to produce trustworthy data for reliable empirical research*". The results of the mapping study also show that many definitions and reporting of DQ issues are unclear because of the inconsistent terminology used, which can lead to interpret data in many different ways [51]. Thus, they suggest there is a need to construct a formal definition of DQ issues, using a standard terminology to specify the quality of data sets.

Eight years after its publication [33], Liebchen and Shepperd [34] review their findings in order to assess how DQ issues are being considered, and which techniques are being used in the ESE community. The original study [33] found only 23 papers explicitly considering DQ. The updated review found 283 articles, even though this still represents only the 1% of the total ESE literature. This shows that the community is starting to become more concerned about DQ issues. However, there is still little systematic work and guidance for researchers in order to assess the quality of datasets used for their works. They conclude "*researchers need to have concern for data quality*". Neither researchers nor practitioners can continue working with suspect data. Bachmann and Bernstein [4, 5] analyze DQ characteristics of closed and opened software projects source, finding that all projects contain DQ issues. These issues may have a major impact on empirical software engineering research results. Bachmann defines a DQ framework and metrics to evaluate and analyze DQ software projects, but he does not state that it could be applied to experimental data. The proposal does not include any method in order to assess DQ in a systematic and structured way. The defined metrics do not consider DQ concepts from the DQ discipline, narrowing the DQ multifaceted vision. Metrics are defined as ratios, and could be associated with the DQ dimensions accuracy and completeness.

Bosu and MacDonell [13] proposes a taxonomy of DQ issues for ESE which considers different dimensions, and are classified in three main classes: Accuracy, Relevance and Provenance. He does not find any study that evaluates all the aspects of the taxonomy, indicating that the treatment of DQ is not a holistic endeavor in ESE. This work succeeds in considering a broaden concept of DQ for ESE that includes all relevant dimensions. However, the proposal does not present DQ metrics or a repeatable method in order to analyze and improve DQ in this domain. In a recent study, Bosu and MacDonell [15] apply the taxonomy proposed in [13] and assess the quality of 13 datasets that are extensively used in research on software effort estimation. They propose a template for dataset collection and submission including 11 DQ challenges, as well as parameters and information for each challenge. The challenges presented can be comparable to the DQ dimensions we propose in our work. The purpose of the template is to ensure that "*datasets are collected, submitted, and used in an informed and consistent manner by ESE researchers and practitioners*". They aim to provide "*a means through which the nature and origin of an ESE dataset will be more transparent to its users*". Both the approach we propose in this article and template presented in Bosu and MacDonell [15], encourage DQ assessment in ESE data following a structured and formal way, and considering a widen view of DQ. However, Bosu does not propose how to improve the DQ issues detected in order to improve the quality of the datasets used by researchers, as well as the results obtained.

Another research conducted by Rosli [51, 52], "*aims to provide a standard way to better understand and interpret data sets*" and to identify DQ issues. She proposes a framework that includes a metamodel in order to help researchers to apply a common interpretation of the data sets, as well as a quality assessment process to evaluate its quality. The framework will allow researchers to understand the quality of data and identify any potential problems that may be present in a data set. Rosli et. al. [52] propose a quality assessment process in four steps: 1) assessing the datasets model; 2) identifying the data quality issues; 3) evaluating the metadata; 4) preparing the assessment report. One of her findings is that after applying the quality assessment process to 92 data repositories, she finds most DQ issues (such as duplicate, incorrect and missing data) are common. In her work, Rosli does not propose a DQ model nor DQ metrics that could be applied in a structured and systematic way to the data she analyzed.

Some works analyze the quality in software engineering historic datasets [4, 7, 18, 32, 48]. Normally, in these types of works, software process and product data are collected, stored, and then used to analyze strategies and methodologies, construct heuristics or prediction models, or apply statistical techniques. These works also agree on the great significance of assessing the quality of the data used, as they have an impact on the obtained results. Other works [2, 10, 11, 30, 54] analyze software engineering data quality such as bugs and changes reports from one particular view.

Most of the works consider noise and missing values as the main DQ problem to be addressed, but they do not apply the holistic perspective of DQ. Systematic reviews of Liebchen and Shepperd [33] and Rosli et. al. [49] considered a narrowed view of DQ, focusing on the accuracy of data (absence of noise or incorrect data), and excluding other important issues such as incompleteness, redundancy and inconsistency. Rosli et. al. [50] proposed a metric for analyzing only consistency conflicts within Software Engineering datasets. Bosu [12], on the other hand, adopts a more general approach including DQ dimensions. While our work proposes a multi-dimensional approach considering six different DQ dimensions, the framework proposed by Rosli [51, 52] aims to identify DQ issues focusing in the interpretation of data as the main dimension, and considering other four common DQ dimensions (accuracy, completeness, consistency, redundancy) as secondary. For this work, they considered our initial DQ Model [59] adapting the definition of the DQ dimensions we proposed. Other difference between our proposal and Rosli [51, 52] is that while they analyze DQ from public data repositories, our focus is on SEE data.

As we presented in this section, we found works that discuss DQ for Empirical Software Engineering, and some proposals about how to analyze DQ in this domain. However, we did not find in the literature any systematic and disciplined approach for analyzing and improving the quality of the data whose source is a controlled experiment in Software Engineering, as we present in this work.

3 DQMOS AND DQMES: DATA QUALITY MODEL AND METHODOLOGY FOR SEE

In this section we present the main DQ concepts defined in the area of Information Systems (IS), as SEE data are always stored in some kind of IS (such as databases or spreadsheets). Then, we describe the DQMoS: our proposal for a Data Quality Model for Software Engineering Experiments. Finally, we present the DQ Methodology (DQMeS) that defines steps and guidelines to analyze and improve data collected through the execution of SEE, and uses the DQMoS.

The proposed DQ model is intended to be applied to the data obtained from the experiments, i.e. generated by the subjects, in order to improve its quality before it is used for analysis. The model is not intended to be used for evaluating quality aspects of the experiment itself, such as those related with the experimental design, those related to mitigate as many threats to validity as possible, quality in the recruitment process, quality in the process to train subjects, etc. Poor experimental design can also lead to DQ problems since the raw measure does not adequately represent what it is claimed. Our DQ model does not consider these aspects. However, we recognize that these kind of quality problems are frequently the causes of quality problems in the generated data. For this reason, we believe that a good strategy is to manage DQ through our proposed DQ model and methodology, but also to identify and eliminate the causes of DQ problems that are related to the quality of the experiment. The application of DQMoS and DQMeS, as a side effect, may shed light on quality problems of the experiment.

3.1 Data Quality Concepts

DQ is generally defined as "*fitness for use*" [20, 26, 33]; that is, if data are suitable for its use or purpose. As the use of the data depends on each context, its quality will be evaluated in function of its specific purpose [57]. DQ can be defined by "*the degree of benefit (or value) perceived by a user using certain data in a certain context*" [57, 35]. Good quality data "*fit for their intended purpose in operations, decision-making, and planning*" [5, 14].

DQ is a multifaceted concept, as it is defined in function of the dimensions it describes. The quality of the data is defined by the set of dimensions or characteristics that are appropriate for a specific context [43]. The subjective view of DQ motivates the selection of DQ dimensions in order to assess the quality of the data in a specific context. Each dimension represents a different aspect (or facet) of DQ [46, 53, 57].

Our approach is based on a DQ meta-model [9] that defines the following concepts. A *DQ dimension* is a concept that captures one facet of DQ; a *DQ factor* represents a particular aspect of a DQ dimension. A dimension can be seen as a set of different factors addressing the same kind of problem.

There exist a huge set of DQ dimensions and factors defined in the literature [9, 46, 60, 61]. In this work, we consider the DQ dimensions presented in [9]. They are based on well-known concepts about which there is general consensus.

We selected the DQ dimensions and factors that are suitable for SEE. There are other DQ dimensions and factors that, after analysis, we understand they are not applicable to this domain. An example of this are the time-related dimensions, such as *currency, volatility* and *timeliness*, which focus on time when data are created or updated. Since data resulting from the execution of SEE will not be modified, they can always be considered fresh and current.

For each considered DQ dimension, we present its definition and its corresponding DQ factors.

Accuracy. Specifies how accurate and valid the data are. It indicates if there exist a correct association between the IS states and the real world objects. Three DQ factors are defined. *Semantic accuracy* refers to how close a data value is to the real world object it represents. *Syntactic accuracy* indicates if a value belongs to a valid domain. *Precision* refers to the level of detail of the data.

Completeness. Specifies if the IS contains all the important data, with the required scope and depth. It indicates the IS capacity to represent all the significant states of the reality through two DQ factors. *Coverage* refers to the portion of real world objects that are represented in the IS, while *density* refers to the amount of missing data items.

Consistency. Specifies if a set of semantic rules are satisfied in the IS. Three DQ factors are defined to represent the different kinds of restrictions. *Domain integrity* refers to rules about the attributes values and their belonging to the corresponding domain. *Intra-relation constraints* refer to rules that must be satisfied by one or more attributes in the same relation, while in the *inter-relation constraints* the attributes entailed are from different relations.

Uniqueness. Specifies the duplication level of the data through two DQ factors. *Duplication* occurs when the same entity is duplicated exactly, while *contradiction* occurs when the entity is duplicated with contradictions.

Representation. Considers the consistent and concise representation of the data in the IS, and the extent in which data are always represented in the same format and structure. We consider the DQ factors *data format* and *data structure*.

Interpretability. Refers to the documentation and metadata available in order to correctly interpret the meaning and properties of the IS. We consider two DQ factors, *ease of understanding* and *metadata*.

In order to obtain concrete values for DQ, these DQ dimensions and factors need to be quantified by *DQ metrics* [8, 9, 16]. A DQ metric is a quantifiable instrument that defines the way a *DQ factor* is measured, and indicates the presence or absence of a DQ problem. In this paper we have identified a set of DQ metrics that can be applied to experiments of SEE. Finally, a *measurement method* is a process that implements a metric. As the same factor can be measured with different metrics, the same metric can be implemented with different methods. The measurement method has to specify the attribute being measured and the procedure to apply [9, 43]. Measurement methods are specific for each experiment. Even though we define some of them in this paper, they should be adapted and instantiated to each experimental data, and could be different to the ones used in the four experiments analyzed in this paper.

Consider for instance a database that records data about clients. In particular, the addresses need to be standardized according to a particular referential of streets and locations. In this case, we can apply the DQ dimension *Accuracy* and DQ factor *Syntactic Accuracy*, and define a DQ metric *Out of Referential Value*. Both DQ dimension and factor are generic and taken from DQ discipline. The DQ metric is defined for a specific domain, so that it could be applied to databases in the same context. The *Out of Referential Value* metric would be applied to *address* data in order to find all values that are not recorded following the defined referential. The measurement method that implements the metric could be a SQL query that returns the values that are out of the referential. Both the metric instantiation to specific (*address*) data as well as the SQL defined are specific to this client database, as it depends on its structure and characteristics.

A DQ model establishes and formalizes the different DQ aspects that will be considered for a particular domain, as well as the metrics that will be applied for DQ assessment. During the definition of a DQ model, quality aspects and data items must be selected following some prioritization criterion, since all aspects cannot be measured for all data items. Therefore, the most relevant data items should be selected and for each one, the most relevant quality aspects should be chosen.

3.2 DQMoS: Data Quality Model for SEE

The DQMoS contains a set of DQ metrics that are suitable for SEE data, and that are based on the DQ dimensions and factors presented in Section 3.1. The model proposed includes the definition of each DQ metric as well as the DQ metadata. This model provides a framework for the analysis and assessment of DQ.

Figure 1 shows how the DQMoS is defined from the general DQ concepts presented in Section 3.1. Considering the DQ dimensions and factors proposed by the DQ discipline, we defined DQ metrics that are specific to the SEE domain. The DQ metrics are then instantiated to particular data (i.e. attributes, tables) when applied to an experimental data repository.

As part of the model that we developed, we defined the following characteristics for each DQ metric of DQMoS.

- *Semantic*: metric's description.
- *Definition*: how the metric returns a value when applied to attributes or set of attributes of the experimental data under consideration. Data collected during the experiments are stored in data repositories. Even though these repositories are not always relational databases, we use relational databases terminology in all the cases, as the concepts for any type of data repository are analogous.
- *Measurement result unit*: how the result is represented (Boolean value, grade, enumeration).
- *Measurement method*: the process that implements a metric (SQL query, programming, manual).

- *Granularity*: the level at which the metric is applied (cell-attribute value for a given tuple, column, table or even database).

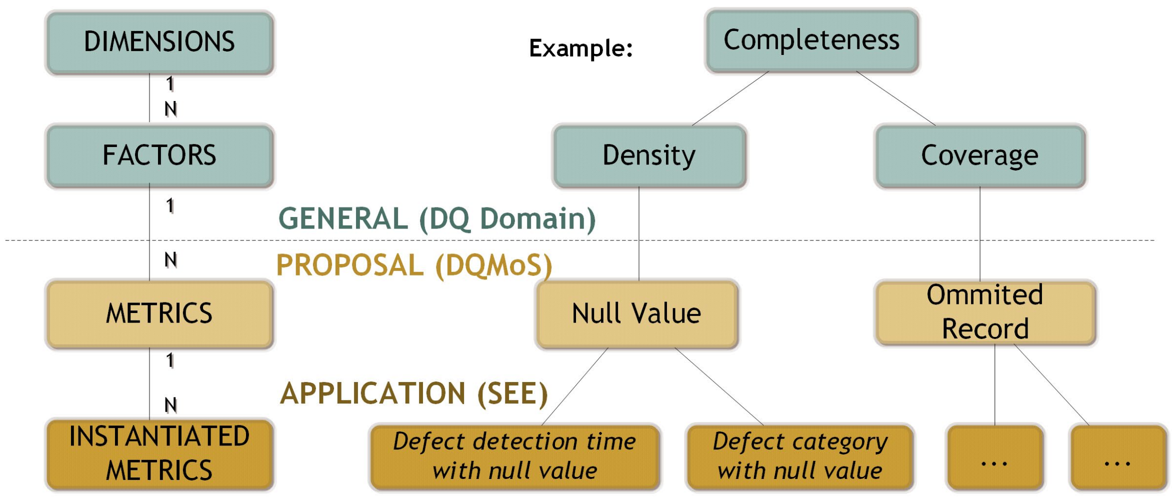

Figure 1: DQMoS and Application example

When the model is applied, that is when a researcher uses the model in order to evaluate and correct data quality issues in a SEE, the DQ metrics defined in the DQMoS are instantiated to particular attributes or set of attributes in the experimental data. DQ problems found as a result of the application of the model are classified as follows.

- *Data Errors (DE)*: correspond to errors in the data that (whenever possible) need correction.
- *Questionable Value (QV)*: in this case it is not possible to assure if the DQ problem corresponds to a real data error (examples of these are outliers). It is necessary to compare data and reality in order to know if a data error exists.
- *Improvement Opportunities (IO)*: correspond to suggestions of aspects that could be improved in order to prevent the occurrence of a DQ problem in the future.

Note that the DQ Model aims to identify DQ problems, no matter its cause.

Consider for instance a database that records data about software defects, and that may contain missing values. The *Completeness* DQ dimension and the *Density* DQ factor (both form DQ discipline) contemplate this DQ aspect. In this case, we apply the DQ metric *Missing Value* in order to identify the values that are missing but need to have a non-empty value. Which values need to be not missing will depend on each application. For example, this metric could be instantiated to *defect detection time* data in order to find time records that contain missing values.

The measurement method that implements the metric could be a *SQL* query that returns the missing values that are present in the data. The metric instantiation and measurement method are specific and depend on the particular database.

As the measurement result unit is Boolean, we assign ‘1’ for each defect detection time containing a missing (null) value, and ‘0’ otherwise. Then, we calculate the DQ Value by counting the proportion of missing data (amount of ‘1’) over the total data measured.

3.2.1 DQ Model Definition and DQ Metadata

In Table 1 we present the DQ dimensions, factors and metrics that define the DQMoS. The DQ metrics are presented in detail in Table 2, where we present the semantics, the definition and the result units of each metric. These units may be Boolean, Grade or Enumeration. When the result unit is Boolean, we assign ‘1’ if the DQ metric is satisfied in the data measured, ‘0’ otherwise. When the result unit is Grade, we assign a value between 0 and 1, representing the closeness between the registered value and the real value. When the result unit is Enumeration, we assign a “Low” value if any of the conditions measured are not satisfied, “Acceptable” if they are partially satisfied and “Good” if they are fully satisfied. For space reasons, metrics’ granularity and measurement method are not included in Table 2. We define DQ metadata in order to record the measurement results. This metadata shows: which DQ metrics were applied, which data were measured, and if each measured data item contains a DQ problem. We can also calculate an aggregated data quality value from these metadata. For each application we define the metadata structure, as it depends on the data model and repository used. For example, if data are stored in a relational database, new tables can be created to store the DQ metadata.

Note that it is possible that the same DQ problem could be found applying different DQ metrics. In this case, the DQ problem is only considered once.

The DQ metrics M18 to M21 are related to the quality of the data representation, interpretability and documentation, which are by definition subjective aspects. This is the reason why human involvement and experience in the specific domain under assessment, is essential in the application of these DQ metrics [46].

Table 2: DQ Metrics defined in DQMoS

Id	DQ Metric Semantics	DQ Metric Definition	Measurement result unit
M1	Define a range of values (minimum and maximum values possible) to which data must belong. The range can be defined statistically, from the researcher experience, or historic data. This includes numeric data (interval, ratio or absolute scales).	Verify if the corresponding data values are out of the range defined as valid.	Boolean or Grade
M2	Define the standard format in which values must be stored. This can include the format itself, units, data types, etc.	Verify if the corresponding data values are not recorded in the specified format.	Boolean
M3	Free texts can contain some embedded values that need to be obtained separately, in order to analyze them individually. If possible, the constraints could be specified as regular expressions.	Verify the existence of embedded data values in free texts.	Boolean
M4	Records represent a real world object. However, there might exist records in the database which corresponding object does not exist in reality. Verify whether the values registered are the true values.	Verify the existence of records in the data that are not associated to any real world object.	Boolean
M5	Records can represent an incorrect real world object. This means that the record associated with the object has incorrect values: although there are syntactically correct, they do not belong to that particular object (this includes measurement errors).	Verify the existence of records in the data whose values are not correct.	Boolean or Grade
M6	Define a set of values (referential) to which some data values must belong. The referential can be defined by the researcher, by a list of pre-defined values, or be specified as part of the experimental design. This includes nominal and ordinal data, by defining the possible set of values, such as categories, rankings, scaling; and its corresponding order if applies.	Verify if the corresponding data values do not belong to the set of values (referential) defined as valid.	Boolean
M7	Define the detailed or precision level required for the values, necessary to make the corresponding calculations. For example, if a number must be registered with two decimals (not rounded) in order to achieve better precision in the results calculations.	Verify if the corresponding data values are not recorded with the required precision.	Boolean
M8	Define which single data should have been recorded with a non-empty value. A missing value could represent data that do not exist in the real world or it was not possible to find out. We are interested in data that exist in reality but its entry was missing.	Verify if data values that exist in the real world are not stored in the database.	Boolean
M9	Define which set of values or information of a particular object was omitted, but should have been recorded with non-empty values. While M8 is applied to a single value, M9 is applied to a set of values. For example, if the start and end session	Verify if the information (set of values) for a particular object exists in the real world, but it is not stored in the database.	Boolean

	times are not null, so the total session time must be calculated and not null.		
M10	Some records may have been omitted. This means there exist real world objects that are not represented in the database.	Verify the existence of records which entry was omitted.	Boolean
M11	Define the domain of values to which some data must belong. In this metric the domain defined is well known, while in "Out of Range Value" the values could be defined arbitrarily.	Verify if the corresponding data values are out of the domain defined as valid.	Boolean
M12	Define rules involving attributes that belong to the same relation that must be satisfied in the database.	Verify if the defined rules are not satisfied in the data.	Boolean
M13	Define which data should have been recorded with a unique value.	Verify if data which values must be unique, contain a repeated value.	Boolean
M14	Define rules involving attributes that belong to different relations, and that must be satisfied in the database.	Verify if the defined rules are not satisfied in the data.	Boolean
M15	There could exist references to records from within the collected data that are not stored in the database, so they result in invalid references.	Verify the existence of references to records that are not in the database.	Boolean
M16	Records could have been stored twice in the database. Duplication could happen: if two or more records have the same value in their identifiers and other data items; or if two or more records have different identifiers but reference to the same real world object. A duplication criterion must be established in order to identify duplicated records.	Verify the existence of duplicate records, according to the duplication criteria defined.	Boolean
M17	Records could have been stored more than once but in a contradictory way. They represent the same real world object, but contain different values in their identifiers and/or other data.	Verify the existence of contradictory records, according to the contradiction criteria defined.	Boolean
M18	Data must be represented in an appropriate way, considering the reality and context under study.	If the data are stored in a DBMS, verify the lack of definition of integrity rules and restrictions. Regardless of the data storage, also verify if the data representation is not concise and consistent, or appropriate to the reality represented. Assign a value "Low" if any of these conditions are not satisfied, "Acceptable" if they are partially satisfied and "Good" if they are satisfied.	Enumeration
M19	Data must be represented in a consistent format, so as the same data must always be represented in the same way.	Verify if different formats are used to represent the same data. Assign a value "Low", "Acceptable" or "Good", according to the quantity of cases where this condition is not satisfied.	Enumeration
M20	Data must be appropriate for each class of stakeholder who might make use of them (apart from the researcher).	Verify if data are not clear and with ambiguities in its meaning and use. Also verify the degree in which the information can be understood and interpreted, the scope and comprehensibility of the data. Assign a value "Low" if any of these conditions are not satisfied, "Acceptable" if they	Enumeration

		are partially satisfied and "Good" if they are satisfied.	
M21	Documentation and metadata must be used in order to describe and define the data characteristics. This enhances the correct interpretation of the meaning and properties of the data and its sources.	Verify the lack of definition of conceptual scheme (if applies), metadata, and historic information and traceability of data. If all these aspects are covered assign a value "Good", if only some of them are given assign "Acceptable", otherwise assign "Low".	Enumeration

3.3 DQMeS: Data Quality Methodology for SEE

We propose a DQ Methodology (DQMeS) that defines steps and guidelines to apply the DQMoS to the data collected through the execution of SEE. During each application to SEE, we found that we followed a systematic and structured approach, and that this approach and guidelines could be repeated in other SEE.

Figure 2 shows the steps to identify DQ problems and improve the quality of SEE data. Next, we describe each step including inputs, tasks and outputs, illustrated by a running example. As running example to illustrate our approach, we select a replication experiment conducted at the Technical University of Valencia (UPV), Spain [44]. The aim of this experiment is to compare the MDD paradigm with traditional software development methods. We consider this case to be representative, complete and enriching. By presenting a running example, we are able to show how we successfully applied the model and methodology to SE experimental data, as well as the results and benefits obtained.

3.3.1 Roles

The roles that participate in the DQMeS are:

- *Data Quality Analyst (DQA)*. She/he participates during all the steps, as she/he is responsible for carrying out the tasks described in each step. This role can be executed by anyone who is familiar with the DQ metrics and the DQMoS; even the Experimenter could assume this role.
- *Experimenter*. She/he participates during the steps in which knowledge about the experiment is required. She/he is also responsible for the validation tasks, including the validation of DQ metrics identified and corrective actions proposed.
- *DQMoS and DQMeS Responsible Person (DQMR)* is the role accountable for the model and methodology development and maintenance. This role is in charge of analyzing all change requests and improvement suggestions after each application. He also adjusts the DQMoS and DQMeS in case the proposed changes are approved. For example, if new metrics are identified, the DQMR analyzes their inclusion as part of the DQMoS.

In the running example, the DQA and DQMR was the first author of this paper (and the proposal), and the Experimenter did not belong to the DQMoS and DQMeS research group.

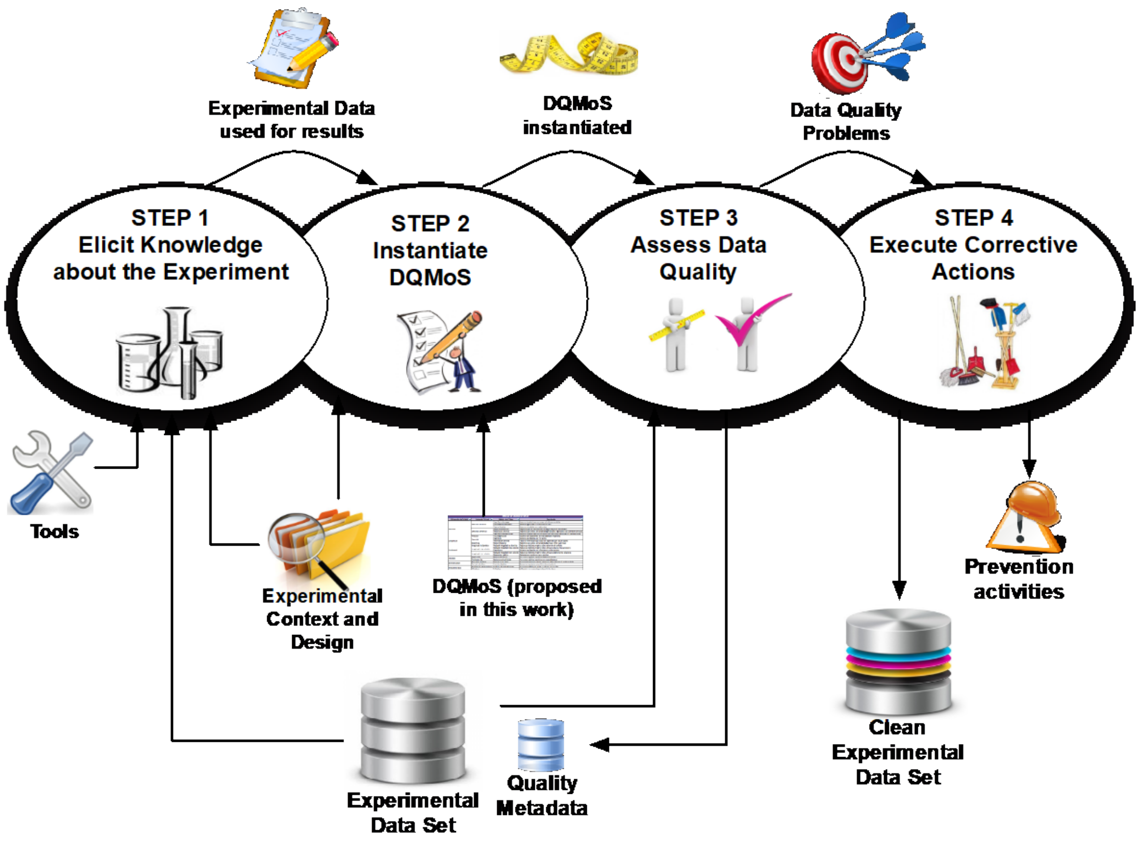
Experimental Data used for results
DQMoS instantiated
Data Quality Problems
STEP 1 Elicit Knowledge about the Experiment
STEP 2 Instantiate DQMoS
STEP 3 Assess Data Quality
STEP 4 Execute Corrective Actions
Tools
Experimental Context and Design
DQMoS (proposed in this work)
Prevention activities
Clean Experimental Data Set
Experimental Data Set
Quality Metadata

Figure 2: DQMeS: Data Quality Methodology for SEE

3.3.2 Steps

In this section we describe in detail the steps proposed by the DQMeS, showed in Figure 2. During Step 1, the DQA understands the context and design of the experiment, focusing on data collected and stored. Next, in Step 2, the DQA identifies which DQ metrics (from the ones defined in the DQMoS) will be applied, and to which specific data. In Step 3 the DQA implements the measurement methods in order to assess the quality of experimental data. Finally in Step 4, the DQA defines and execute the corrective actions in order to correct the data errors that were previously identified. For each step, we present the inputs needed, the tasks executed and the results (outputs) produced.

Step 1: Elicit knowledge about the Experiment

The aim of this step is to understand the experimental context and design. In particular: which data are collected and how, who collects and uses the data, where data are stored, and if any tool is used to collect and store the data. The focus of this step is to identify the “significant data”: this is the data (raw or calculated) used to obtain the experimental results.

For example, consider an experimental design where *effort* is defined as one response variable. Subjects will store data about *“time”* in the experimental database. This is considered as “significant data”.

The Experimenter participates actively during this step communicating in detail the experiment knowledge to the DQA.

Inputs needed for Step 1:

- Experimental material: documentation, conceptual models, and experimental context.
- Data repository.
- Tools, if used.

Tasks executed in Step 1:

- Working meetings between Experimenter and DQ Analyst.
- Elicitation of experimental information and data.
- Analysis of material (papers, models, etc.).
- Analysis of tools and data repository.

Outputs resulting of Step 1:

- Experimental context known and studied: description, design, data, repositories, and tools.
- Significant data identified.

Step 1 – Running example

We gather information about the experiment by having interviews with the experimenters, and studying reports and papers [23, 44, 45].

The experiment aims to compare the Model-Driven Development (MDD) paradigm versus a traditional software development method.

The subjects of the experiment are 20 master students from the UPV, and the experimenters who collect the data are three teachers. They have 2 sessions of 2 hours per development method (MDD and traditional) and 2 problems (P1 and P2). In the replication, the problems are extended to increase the difficulty. This new version of the problems is divided into 3 exercises (parts) such a way the first exercise is the same problem used in the base experiment and exercise 2 and 3 are extensions.

The following variables are defined:

- *Software Quality*: percentage of success after applying test cases.
- *Effort*: time spent to develop the system.
- *Productivity*: quality-effort ratio. It is defined as the amount of software quality work done per effort. By software quality and effort we mean the variables defined previously.
- *Satisfaction*: range of values from 1 to 5 (1: totally unsatisfied, 5: totally satisfied). It is defined as how at ease developers are as they develop a system. The instrument used to measure this variable was a satisfaction questionnaire. The experimenters evaluate satisfaction in terms of perceived usefulness (PU), perceived ease of use (PEOU), and intention to use (ITU) [41, 42].

Figure 3 shows the UPV experimental process, and the data that is registered in each step. The experiment starts with a demographic questionnaire that subjects have to fill in on paper to identify their background. Traditional development treatment is applied first to all the subjects. The beginning and end of each session are automatically registered since subjects must upload the problem to a repository at the end of each session.

Next, the experimenters add the time spent in both sessions to get the time of the whole development. Once the session is finished, each subject must fill in a satisfaction questionnaire.

Finally, the experimenters check the quality of the systems developed by each subject through test cases and write down the results. Each test case is defined as a sequence of items. A test case is considered fulfilled when every item is passed. The test cases as well as the items result have two possible values: success (1) or failure (0). All the data collected during the experiment is recorded in spreadsheets by the experimenters. The same process is repeated to apply the MDD treatment.

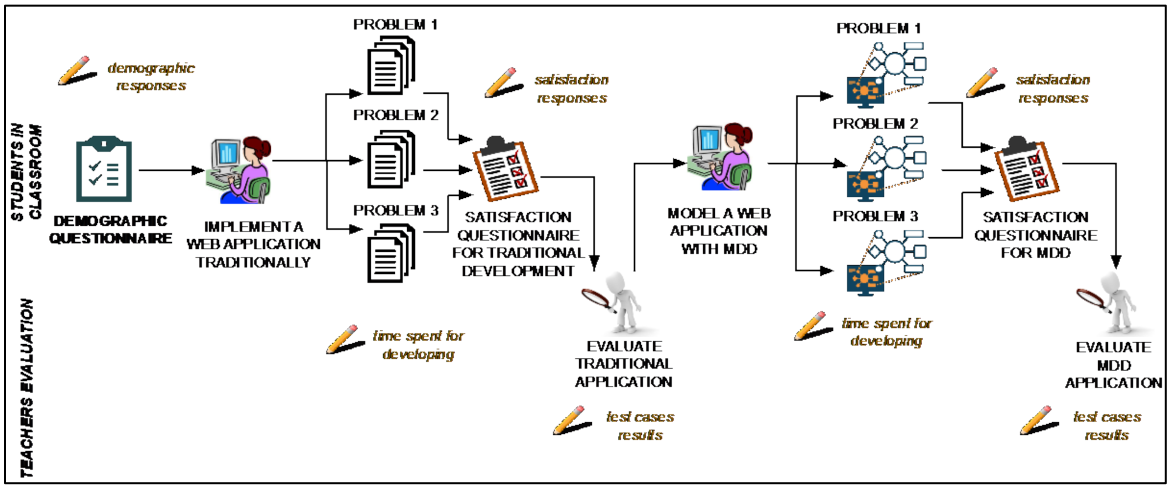
STUDENTS IN CLASSROOM
TEACHERS EVALUATION
demographic responses
DEMOGRAPHIC QUESTIONNAIRE
IMPLEMENT A WEB APPLICATION TRADITIONALLY
PROBLEM 1
PROBLEM 2
PROBLEM 3
time spent for developing
satisfaction responses
SATISFACTION QUESTIONNAIRE FOR TRADITIONAL DEVELOPMENT
EVALUATE TRADITIONAL APPLICATION
test cases results
MODEL A WEB APPLICATION WITH MDD
PROBLEM 1
PROBLEM 2
PROBLEM 3
time spent for developing
satisfaction responses
SATISFACTION QUESTIONNAIRE FOR MDD
EVALUATE MDD APPLICATION
test cases results

Figure 3: UPV Replication Experimental Process

Step 2: Instantiate DQMoS

The DQMoS defines, for each DQ dimension and factor, DQ metrics that could be applied to experimental data in SE.

In this step, the DQA identifies which DQ metrics (from the ones defined in the DQMoS) will be applied to the data. For this task, three possible strategies are defined that could be applied complementarily to the different data items and DQ aspects.

Strategy I: Bottom-up, from collected data

Considering the experimental data used for the experiment results (significant data), the DQA identifies possible DQ problems that could occur during the collection and recording of the data. Then the DQA maps these problems to the corresponding DQ metrics.

For example, if *time* data are collected, then a *time* value less than 0 would show the presence of a DQ problem. This can be measured applying the DQ metric *Domain Integrity Rule* that is defined in the DQMoS.

Strategy II: Top-down, from DQ metrics defined

For each DQ metric in the DQMoS, the DQA identifies possible DQ problems that could occur to the significant data.

For example, considering the DQ metric *Duplicate Record*, a possible DQ problem would be if the same subject registered the same defect more than once.

Strategy III: From the tools used (if any)

The DQA examines the tools used to collect and store the data, and which validations are included. The DQA identifies possible DQ problems that could arise when using the tool.

Then, the DQA joins the results obtained after applying the strategies, verifying if all the metrics identified are possible and worthy to apply. For example, if the tool used validates the insertion of missing values, then the *Missing Value* metric will not be worth to apply.

Finally, the DQA identifies all the data to which each DQ metric will be applied. Remember that each metric can be instantiated more than once, on different data. The Experimenter is responsible for validating the DQ metrics identified and the data to which they will be applied.

Inputs needed for Step 2:

- DQMoS.
- Data schema.
- Rules and restrictions defined in the data.
- Experimental context documentation, including tools to collect data.
- Significant data.

Tasks executed in Step 2:

- Identification of DQ metrics by applying the three strategies.
- Work sessions to validate metrics.
- Instantiation of DQ metrics.

Outputs resulting of Step 2:

- DQMoS instantiated (selection of DQ metrics).
- DQ metrics instantiated and validated (definition of the data to which each DQ metric would be applied).

Step 2 – Running example

Based on the experimental response variables and other contextual information related to these variables, we examine the experimental data and identify which DQ metrics (defined in Table 2) could be instantiated to the corresponding attributes.

Table 3 presents the DQ Dimensions, Factors and Metrics applied to each experimental data (raw and calculated). Due to space restrictions, we show the information for the response variables *Software Quality* and *Effort* as an example. This example shows how each response variable component (raw or calculated data) is analyzed separately in order to select which DQ dimension, factor and metric assess its quality. The multi-faceted view of DQ is clearly shown, as different DQ dimensions and factors are applicable to each response variable.

Consider for instance the response variable *Software Quality*. We identify *test_case_items* as raw data recorded to calculate this variable. This data represents the result of each test case item. Following Strategy I (from data collected), we instantiate the DQ metrics presented next.

- As the experimenters register this data manually, typing errors may occur, generating values that do not correspond to reality. We apply the DQ metric *Incorrect Record* to identify these values.

Table 3: DQ Metrics applied to MDD experimental data (UPV-Replication)

Response Variable	Calculation	Components: Raw (R) and Calculated (C) data	DQ Dimension / DQ Factor	DQ Metric

Software Quality	success_test_cases_porcentage=success_test_cases*100/total_test_cases
	
	success_test_cases=AND (test_case_items)
		success_test_cases (R)	Accuracy / Syntactic Accuracy	Lack of Standardization

			Completeness / Density	Missing Value
			Completeness / Coverage	Omitted Record
			Consistency / Inter-relation Integrity	***Inter-relation Integrity Rule (R1)***
		sucess_test_cases_porcentage (C)	Accuracy / Precision	Lack of Precision
			Completeness / Density	Missing Value
			Consistency / Domain Integrity	Domain Integrity Rule
		test_case_items (R)	Accuracy / Syntactic Accuracy	Lack of Standardization
			Accuracy / Semantic Accuracy	Incorrect Record
			Completeness / Density	Missing Value
			Consistency / Inter-relation Integrity	***Inter-relation Integrity Rule (R1)***
Effort	total_time=session1_time +session2_time	**** session1_time (R) * session2_time (R)***	Accuracy / Syntactic Accuracy	Out of Range Value
			Accuracy / Syntactic Accuracy	Lack of Standardization
			Accuracy / Semantic Accuracy	Incorrect Record
			Completeness / Density	Missing Value
			Completeness / Density	Omitted Information
			Consistency / Intra-relation Integrity	Intra-relation Integrity Rule
			Consistency / Inter-relation Integrity	***Inter-relation Integrity Rule (R2)***
	total_time<=exercise_time	**** total_time (C) * exercise_time***	Accuracy / Syntactic Accuracy	Out of Range Value
			Accuracy / Syntactic Accuracy	Lack of Standardization
			Completeness / Density	Missing Value
			Consistency / Inter-relation Integrity	***Inter-relation Integrity Rule (R2,R3)***

- As *test_case_items* is used to manually calculate *the success_test_cases* value, an inter-relation integrity rule is defined in order to verify this calculation. Thus, we apply the DQ metric *Inter-relation Integrity Rules*.

Following Strategy II, for each DQ metric in the DQMoS, the DQA identifies possible DQ problems that could occur in data used to calculate *Software Quality* response variable. We instantiate the following DQ metrics:

- *Missing Value*: as all test case items have to be executed, we apply this metric to verify if values are not empty.
- *Lack of Standardization*: as test case items are Boolean values, it must always be 0 (failure) or 1 (success). We apply this metric to verify no other value was registered.

In this case, as no tool was used, Strategy III is not considered.

We can observe that some DQ metrics do not apply to all types of data. For instance, for Boolean data we would not measure *Lack of Precision* or *Duplicate Record*. DQ metrics are selected based on the data nature (format, structure), aim of use (why data is registered) and user requirements (what benefits do users expect to obtain from data).

In order to show a complete example of how a DQ metric is instantiated to experimental data, we choose *Inter-relation Integrity rules definition* (M14) as it corresponds to a DQ aspect that is not generally addressed in SE experimental data.

This metric, when instantiated to our example, produces eight instantiated metrics: three related to time, two related to test cases, and three related to demographic questionnaire responses. Here we present three of them as an example. As we can observe, in this case the instantiated metrics correspond to data integrity rules.

- *R1. Test case result will be 1 (success) if and only if each of its items result is 1. Otherwise, the test case result will be 0.*
- *R2. Total execution time is the sum of the time spent in the two working sessions.*
- *R3. Time spent in making the first exercise has to be less or equal than the total execution time (only for the replication).*

Step 3: Assess Data Quality

In this step, the DQA defines and implements the measurement methods and the DQ-metadata database in order to apply each instantiated DQ metric and register the results. The measurement methods used depends on the data repository characteristics, and can be manual or automatic.

For instance, if data are stored in a relational database, then SQL queries can be defined to implement the measurement methods. However, during this step the DQA may identify measures that cannot be executed (i.e., for their cost of implementation).

As a result of executing the measurement methods, DQ problems are identified in the data. The DQA classifies them as: data errors, questionable value and improvement opportunity (as presented in Section 2.2). Finally, the DQ problems are validated with the Experimenter.

The DQA records the measurement results in the DQ-metadata. This allows the identification of data with DQ problems, as well as the calculation of DQ values. The DQ value presents (by a numerical calculation) the presence or absence of a DQ problem.

Table 4: DQ Metrics Results (UPV-Replication)

DQ Metric	DQ Metric instantiation	DQ Problem	DQ measurement result	Corrective Actions proposed
M8: Missing Value	Test case items and exercise times with missing values	Data Error	6 test case items 3 exercise times	No actions were taken as real values cannot be known.
M9: Omitted Information	***Second session times calculations omitted***	***Data Error***	***4 sessions times calculations omitted***	***Calculate second session times (as the difference between end and start time). Update total execution time.***
M10: Omitted Record	Test case results missing	Data Error	4 test case results	No actions were taken. Requirements were not implemented, so they were not tested.
M14: Inter-relation Integrity Rule	***Consistency rules not satisfied in test cases results (R1)***	***Data Error***	***2 test case results where the test case items are missing***	***Replace test cases results with '0'. Update success test cases percentages.***
	Consistency rules not satisfied in total execution time (R3)	***Data Error***	***6 total execution times where exercise time is longer***	***Same corrections applied for M9. Update exercise time with total execution time.***
	Consistency rules not satisfied in demographic questionnaire responses	Data Error	5 demographic questionnaire responses	No actions were taken. Manual depuration is proposed, as automatic manipulation of demographic information is not possible.
M1: Out of Range Value	First and second session times out of range value	Questionable Value	2 session times registers	No actions were taken as the time exceeded is lower than 15 minutes, which is considered acceptable.
M12: Intra-relation Integrity Rule	Consistency rules not satisfied in session times	Questionable Value	3 session times registers	No actions were taken as the time exceeded is lower than 15 minutes.
M18: Data Structure	Adjustment of spreadsheets	Improvement Opportunities	Database structure to be improved	Definition and implementation of database schema, rules and restrictions.
M20: Ease of understanding	Adjustment of spreadsheets	Improvement Opportunities	Database understanding to be improved	Use references to ease the understanding of data.
M21: Metadata	Adjustment of spreadsheets	Improvement Opportunities	Database metadata to be improved	Record the origin, responsible and way of collecting the data.

Inputs needed for Step 3:

- DQMoS and metrics instantiated.
- Data repository.

Tasks executed in Step 3:

- Implementation and execution of measurement methods.
- Recording of DQ-metadata.
- Work sessions and validation.
- Calculation of DQ aggregated values.

Outputs resulting of Step 3:

- DQ problems identified.
- DQ-metadata recorded.
- DQ values calculated.

Step 3 – Running example

The assessment of data quality consists in, whenever possible, defining and executing formulas in the spreadsheets where data were registered. This allows the implementation of the measurement methods.

Continuing with the example of the DQ metric *Inter-relation Integrity rules definition* (M14), we measure the rules (R1 to R3) satisfaction through formulas implemented in the spreadsheets. The result unit is a Boolean value, indicating if the object measured presents a DQ problem (0) or not (1). Its granularity is at cell level.

Table 3 shows (in bold and italics) which are the response variables, components, DQ dimensions, factors and metrics implied in R1 to R3.

As a result, we can observe that:

- R1 was not satisfied. We found 2 test case records whose items results did not exist, but the test case result had a value (1 or 0). Both cases correspond to data errors. The subjects had not implemented the functionality being tested, so the test case result should have been ‘0’. This DQ problem impacts on the experimental results obtained (regarding the variable *software quality*).
- R2 was satisfied in the experimental data.
- R3 was not satisfied. We found 6 records where the time spent in making the first exercise was longer than the total execution time. In 4 cases, the second session time values did not exist. Note that these cases were also found by the metric “*Omitted Information*”. Thus, the total execution time only considered the time of the first session, making the exercise time longer. This data error has to be corrected because it impacts on the results obtained (regarding the variable *effort*). In the remaining cases, the difference was lower than 3 minutes.

The DQ problems found as a result of executing the measurement methods are presented in Table 4. We can observe how the instantiation of M14 in R1 and R3 result in data errors; R2 is not included as it did not show the presence of any DQ problem.

Step 4: Execute corrective actions

In this step, the DQA defines data cleaning actions in order to correct the identified data errors. The questionable values are analyzed with the Experimenter in order to identify if they correspond to real data errors. The cleaning can be automatic, semi-automatic or manual, depending on each error characteristics.

For example, if a missing time value is found, a corrective action could consist in asking to the responsible of recording the data which is the real value, and updating that value in the database (manual correction). If the data responsible does not know the real value, it would not be possible to correct the DQ problem.
If the data model contains errors, a new schema can be defined in order to correct the errors and migrate the data.

Finally, the DQA identifies DQ prevention actions and improvement opportunities in order to prevent data errors from occurring in future replications of the experiment.

Inputs needed for Step 4:

- DQ problems.
- Data cleaning techniques.
- Data repository with DQ-metadata.

Tasks executed in Step 4:

- Selection of data cleaning techniques.
- Implementation of data cleaning tasks.
- Definition of new data schema (if applies).
- Migration of data (if applies).
- Identification of preventive actions

Outputs resulting of Step 4:

- “Clean” data repository (errors corrected)
- Preventive actions identified

Step 4 – Running example

We showed in the previous step that the execution of the measurement processes shows the presence of DQ problems. For *M14 Inter-relation Integrity Rule*, only data errors were found and corrected. We did not identify any questionable value or improvement opportunity.

The following corrective actions were taken.

- For R1, we replace the values in two test case results with ‘0’, and execute the calculations again. We found that in one case the test case result differs before and after the correction.
- For R3, we applied two different corrective actions. The first one is related to *M9 Omitted Information*. Applying the corrective actions for the data errors found for this DQ Metric, we were also correcting the errors related to R3. For this, we made the calculations for second session and total time that were omitted. We found that for 4 cases the results differ before and after the corrections. Besides, in 3 cases we replaced the time for the first exercise with the value of the total time.

Corrections applied for UPV-Replication Experiment

Table 4 shows all DQ problems found in this experience, and its corresponding classification. It also includes the amount of objects found containing DQ problems as a result of the measurement, and the corrective actions proposed for each case.

Note that in this case, three corrective actions were executed successfully involving M14 and M9, as described above (showed in bold and italics in Table 4). In the remaining cases (*M8 Missing Value* and *M10*

Omitted Record) it was not possible to apply corrections as missing values could not be known. After analyzing the DQ problem regarding the demographic questionnaire responses with the experimenters, they understand they will not have an impact in the experimental results.

We found questionable (unusual) values for 2 of the metrics (*M1 Out of Range Value* and *M12 Intra-relation Integrity Rule*). After validation with the experimenter, we did not identify any corrective actions as they did not correspond to real data errors. For instance, we found time values that are not in the defined range. It was not possible to assure if this corresponds to an error in data registration, or if students used that time to complete the exercises. Besides, the time is exceeded by less than 15 minutes, which is considered acceptable as sessions' duration was 240 minutes.

We propose specific improvement opportunities for 4 metrics, referred to: *M18 Data Structure* (i.e., how data is stored), *M19 Data Format* (i.e., how data is represented), *M20 Ease of Understanding* (i.e., if data can be understood for making the analysis) and *M21 Metadata* (i.e., if metadata is stored). We suggest, for example, the definition of a database schema and integrity rules, the standardization in the registration of the same kind of data and the identification of the sources of information, among others.

Impact of corrections in response variables for UPV-Replication Experiment

Once we applied all the corrections proposed to UPV-Replication experiment, we repeated all the statistical tests that were conducted with the raw data to analyze the impact of such changes.
Table 5 shows the p-values obtained through the Mixed Model and the Effect Size before and after applying the corrections. Note that we only show the variables affected by the changes after applying the DQMoS and DQMeS. These results conclude that even though there are small variations in the p-values, the results of the statistical test have not changed. For all variables, the level of significance remains stable: Productivity, Time and Accuracy with 3 exercises are significant; while Accuracy with 1 exercise is not significant. Regarding Productivity and Time, we notice that both variables have a considerable better value for Effect Size, which implies that the differences between treatments were more important than suspected before the corrections. Effect Sizes for both Accuracies does not involve important changes.

Table 5: P-values and Effect Sizes of variables affected by the corrections

Response Variable	**Significance**		**Effect Size**	
	Before	**After**	**Before**	**After**
Productivity	0.008	0.030	1.25	1.68
Time	0.023	0.001	1.35	2.49
Accuracy (1 exercise)	0.946	0.900	1.13	1.09
Accuracy (3 exercises)	0.022	0.027	1.76	1.76

Analyzing more in detail the impact of the corrections, we have also conducted a comparison of descriptive data through Box and Whiskers plots. Figure 4 shows the plots for Productivity before and after corrections respectively. Note that differences between treatments are more evident after corrections since there is no overlapping among first and third quartiles (including medians).

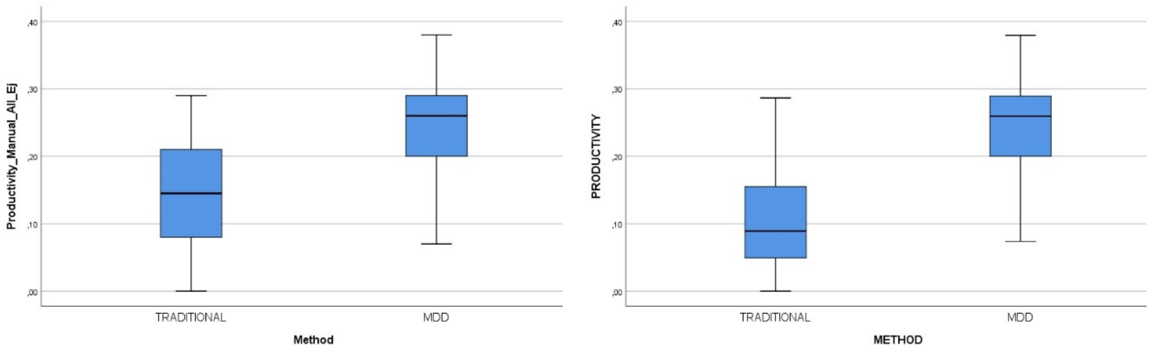

Figure 4: Box and Whiskers plot for Productivity before and after corrections

Figure 5 shows the plots for Time before and after corrections respectively. Medians, first and third quartiles do not present any overlapping between treatments, which means that differences between treatments are more evident after the correction. This difference, for example, clearly shows the importance of a comprehensive DQ analysis before statistical analysis of the experiments.

Figure 6 shows the Box and Whiskers plot for Accuracy of Exercise 1 before and after applying corrections. We see that the overlapping of first quartile, median and third quartile is exactly the same in both treatments. The only difference is the lower extreme, that reaches to 0 after the correction. The Box and Whiskers plot for Accuracy considering all 3 exercises show the same difference as Figure 5 before and after applying corrections.

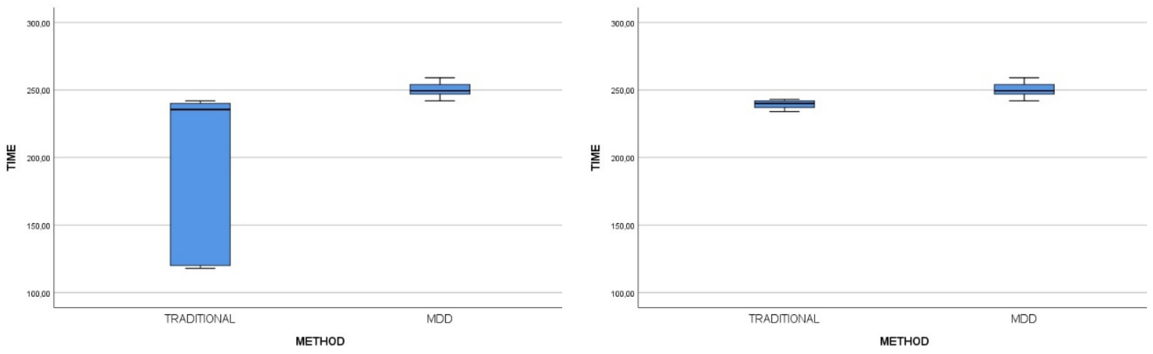

Figure 5: Box and Whiskers plot for Time before and after corrections

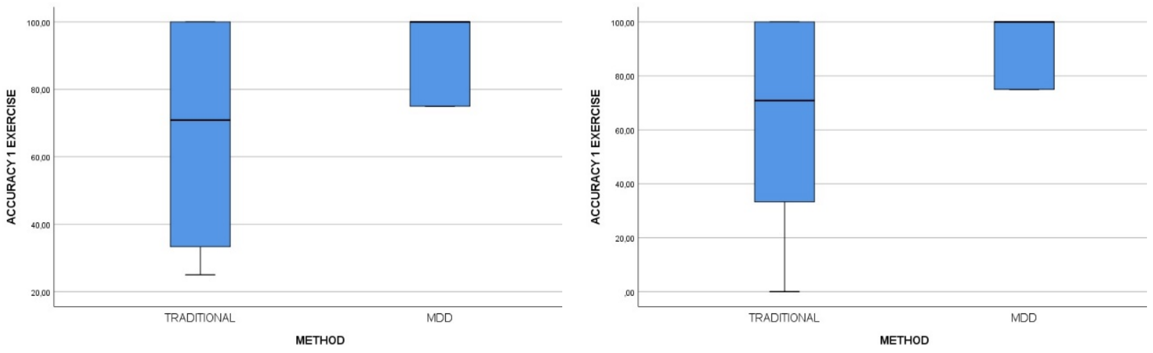

Figure 6: Box and Whiskers plot for Accuracy considering only Exercise 1 before and after corrections

After analyzing the comparison of results before and after applying corrections resulting from the DQMoS and DQMeS, we can conclude that there is an impact because of such corrections. Significances of response variables do not change but we found changes in the descriptive data. All in all we can state that the use of the DQMoS and DQMeS provides a seal of quality to raw data that provide confidence to the experimenter. Note that this example is based on a sample size of 10 experimental units, so the probability of finding errors is low. In spite of the small size of the sample, we have found errors that involved differences in descriptive data. This may lead us to think that in experiments with a larger sample size, where the amount of possible errors can be considerable, the impact of the corrections could involve important changes in the statistical results. Note that all identified errors have been detected with a post-hoc analysis of the data. In spite of their best efforts, original experimenters were unable to detect them and the results of the experiment (with the DQ problems) were published [44]. So, the use of a systematic method and a model shows improvements in finding DQ problems.

4 DQMOS AND DQMES APPLICATIONS

We applied the DQMoS and DQMeS to four SEE in order to measure and improve the quality of their data. In all the cases, the DQMeS was applied after the experiment execution and analysis. Thus, it was not possible to identify corrective actions in order to avoid DQ problems before the experiment execution or even before its analysis and publication of results.

The first experiment which DQ was analyzed was conducted at Universidad de la República (UdelaR), Uruguay, by the Software Engineering Research Group (GrIS) [25, 58]. Its aim was to know and compare the effectiveness of verification techniques and defect types detected. The second experiment was executed by the Empirical Software Engineering Research Group (GrISE) [28] at Universidad Politécnica de Madrid (UPM), Spain, following the same aim. Finally, we analyzed the data resulting from two experiments (base and replication) conducted by the Research Center on Software Production Methods (PROS) from Universidad Politècnica de València (UPV), Spain, aiming to compare the MDD paradigm with traditional software development methods [44]. The running example presented in the previous section corresponds to the UPV-Replication experiment (the last experiment studied).

Table 6 presents the general characteristics and results of each application of our method. We use the term "attribute" to refer to the set of data to which the metric is applied.

Table 6 SEE characteristics and results

Characteristics and results	Experiment			
	UdelaR	UPV-Base	UPV-Replication	UPM
Data repository	Relational Database	Spread-sheets	Spread-sheets	Spread-sheets
Data collection	Web tool	Manual (paper), web form	Manual (paper)	Manual (paper)
#Subjects	14	26	19	32
#DQ metrics applied	15	16	16	16
#DQ metrics instantiated in attributes	51	47	59	71
#Measurements executed	Auto: 38 Manual: 9	Auto: 32 Manual: 12	Auto: 44 Manual: 13	Auto: 58 Manual: 10
#DQ problems identified	11	9	10	9
#Attributes and data with DQ problems	Att: 20 Data: 7,379	Att: 13 Data: 55	Att: 17 Data: 43	Att: 37 Data: 219
#Attributes and data corrected	Obj: 19 Data: 7,353	0	Obj: 3 Data: 12	0

4.1 DQ Metrics instantiation in the experimental data

All 21 DQ metrics included in the DQMoS were applied to at least one of the four experimental data. Considering both UPV experiments, we observed that the same 16 DQ metrics were applied. We strongly believed this is due to the similarity in the experimental design and data registered. On the other hand, the UPM experiment had 15 DQ metrics in common with UPV, and only 11 with UdelaR. Although the experimental designs were analogous in UPM and UdelaR, we observe that there exist other factors (such as data collection procedures and tools, storage procedures and repositories used) that impact on the DQ metrics instantiation.

Both UPM and UPV experiments used spreadsheets to store data. This means that the data collection and storage processes impact on the DQ analysis of SEE data, as other authors also suggest [14, 21, 39]. In particular, Bosu [12] found as a result of a targeted systematic literature review, that both data collection reporting and data pre-processing activities were associated with the identification of DQ issues. Liebchen [31] claimed that the majority of papers analyzed during a systematic literature review, suggested to apply DQ improvement actions during the collection phase.

We conclude, as we had already supposed, that the DQMoS instantiation is context-dependent, taking into account experimental design and process, data collection and storage procedures, and the experimenter DQ requirements. The selection of which DQ metrics should be applied and to which specific data will vary with each experiment.

These results show that the DQMoS and DQMeS allowed finding DQ problems in four SEE conducted by three different research groups. These findings evidence the importance of using a systematic and disciplined method in order to analyze and improve DQ in SEE.

4.2 Corrective actions applied

We applied corrective actions in two of the four experiments. If some actions had a high implementation cost (i.e., it can only be implemented manually, or its automation is too complex) and a low benefit (i.e., the data affected is not critical for the experimental analysis), the cleaning was not executed.

In the UdelaR case we executed 99.6% of the corrections identified. In this case, a tool was used for collecting the data, and a relational database for its storage. Thus, corrective actions could be implemented automatically, cleaning a big amount of data at once. In the other experiments, the collection was manual and the storage was in spreadsheets, increasing the effort of implementing massive cleaning actions.

In UPV-Replication, we executed 27% of the corrections. For the remaining DQ issues we did not identify possible corrective actions to apply. For example, missing values could not be corrected, as the real values were not known. The corrections applied in this case as well as the impact in the experimental results were presented in Section 2 (along with the running example).

We estimated the cost of implementing corrective actions in UPV-Base and UPM and, even though this estimations were not registered, we can say it was higher than in the other experiments and very time-consuming as they required a manual validation from the experimenters.

4.3 Analysis of DQMoS applications results

Table 7 shows the DQ values obtained after instantiating the DQMoS to the four SEE data. The results show the presence of DQ problems in the data. This table also shows which DQ metrics were applied in each experiment. The term "N/A" means that the DQ metric was not applicable in that case, and "N/E" means that it could not be implemented. The DQ value for M1 to M17 is calculated as: *# attributes without DQ problems / # attributes measured*. For these measures, a DQ value less than 1.000 shows the presence of (at least one) DQ problem. For M18 to M21, whose granularities are the whole database, the DQ value could be: low, good or acceptable; according to the satisfaction criteria defined for each metric. A DQ value "low" shows the presence of (at least one) DQ problem.

Between 15 and 16 DQ metrics (from the 21 included in the DQMoS) were applied in each experiment. All metrics were applied at least once throughout the four experiments, and 10 of them were applied in the four cases. Besides, the DQ metrics applied considered all different DQ dimensions and factors of the DQMoS.

Considering the four cases, 11 DQ metrics showed the presence of DQ problems. In particular, 3 DQ problems appeared in the four cases: *Out of range value, Intra-relation integrity rule and Data structure*. They all belong to different DQ dimensions, showing once again the benefits of using our multifaceted approach.

Table 7: DQ Values obtained for each DQ metric

DQ Metric	DQ Value			
	UdelaR	UPV-Base	UPV-Replication	UPM
#M1	0.979	0.968	0.988	0.912
#M2	1.000	0.941	1.000	0.968
#M3	N/A	0.692	1.000	N/A
#M4	0.979	N/A	N/A	N/A
#M5	N/E	1.000	1.000	N/E
#M6	0.874	N/A	N/A	1.000
#M7	N/A	1.000	1.00	1.000
#M8	0.973	1.000	0.959	0.913
#M9	0.999	1.000	0.800	0.896
#M10	N/A	0.996	0.933	1.000
#M11	0.981	1.000	1.000	1.000
#M12	0.978	0.971	0.963	0.955
#M13	1.000	N/A	N/A	N/A
#M14	N/A	0.857	0.926	1.000
#M15	0.950	N/A	N/A	N/A
#M16	0.947	1.000	1.000	1.000
#M17	0.980	N/A	N/A	N/A
#M18	Low	Low	Low	Low
#M19	N/A	Good	Low	Low
#M20	N/A	Low	Low	Low
#M21	Acceptable	Low	Low	Low

4.4 Experimenters experience

We asked the experimenters from UdelaR, UPV and UPM if, after applying the DQMoS and DQMeS, they felt satisfied with the results achieved, the benefits obtained and the costs incurred. We developed a questionnaire based on the framework proposed by Moody [41, 42]. This framework assesses model quality in terms of three satisfaction variables. *Perceived Usefulness* (PU): the degree to which the experimenters believed that the DQMoS and DQMeS will be effective in achieving its intended objectives; *Perceived Ease Of Use* (PEOU): the degree to which the experimenters believed that using the DQMoS and DQMeS would be free of effort; and *Intention To Use* (ITU): the degree to which the experimenters intended to use the DQMoS and DQMeS in the future.

We defined 16 questions: 8 for PU, 6 for PEOU and 2 for ITU. The possible responses are values between 1 (totally disagree) and 5 (totally agree).

As example, we present one question for each variable.

- PU: "*I think that applying the DQ Metrics increases the probability of finding DQ problems in my experimental data, that would be more difficult to identify other way*."
- PEOU: "*The DQ metrics proposed are simple and easy to apply to my experimental data*."
- ITU: "*The DQMoS is easily understandable, could be instantiated and applied to my experimental data*."

Table 8 shows the result obtained for each satisfaction variable (for UPV and UdelaR experimenters). The values, calculated as the mode and median of the responses obtained, reflect a positive feedback for the three variables. Unfortunately, after some insistence, it was not possible to obtain the UPM experimenter's response.

The obtained results reveal that the experimenters understand the value and improvements introduced to the quality of the experimental data by applying the DQMoS and DQMeS, they intend to use them in future experiments, and agree that the method is simple to apply.

Table 8: Experimenters survey results

Experimenter	**Satisfaction variable (mode)**			**Satisfaction variable (median)**		
	PU	**PU**	**PEOU**	**ITU**	**PEOU**	**ITU**
UdelaR	5	5	5	4.5	5	5
UPV	4	4	4	4	4.5	4

4.5 Threats to validity

We have classified the threats that the application of our approach may be open to according to the classification provided by Wohlin [62]. For each threat we specify if it is avoided, incurred or mitigated.

- **Conclusion validity**. This threat is concerned with issues that affect the ability to draw the correct conclusions about relationships between the treatment and the outcome. Threats of this type are: (1) *Low statistical power*: this appears when the sample size is low. Note that all experiments used in the application of our framework have less than 30 subjects (except for the experiment conducted in UPM). This is mitigated replicating the application of our proposal in 4 different experiments. (2) *Random heterogeneity of subjects*: this appears when subjects are highly heterogeneous. We avoided this threat since subjects of all experiments used in our framework application have a similar background. All of them are students of computer science.
- **Internal validity**. This threat is concerned with influences that may affect the dependent variable with respect to a causality which the researchers are unaware of. Threats of this type that may appear are: (1) *History*: this appears when the experiments that compose our suite are conducted at different times, since the context may affect the results. We mitigated this threat using the same Data Quality Analyst in the framework application of all the experiments. (2) *Instrumentation*: this appears when errors on instruments used in each experiment may affect the results. Our framework application suffers this threat since we did not have access to instruments definition, we evaluated the data quality using instruments defined in each experiment by other persons. (3) *Suitability of subjects for experimental tasks*: this appears when subjects play a role in the experiment in which they are not experts.

- **Construct validity.** This threat is concerned with generalizing the results of the experiment to the concept, or theory, behind the experiment. (1*) Inadequate pre-operational explanation of constructs*: this appears when the theory being analysed is not clear. We avoided this threat since the framework was applied for a single Data Quality Analyst that is expert at the framework. This ensures that the framework was applied in the same way in all the experiments. (2) *Experimenter expectancies*: this appears when experimenters can bias the results based on what they expect. We have mitigated this threat since all defects detected after applying our framework have been corroborated with the experimenters of each experiment to check its validity.
- **External validity.** This threat is concerned with conditions that limit our ability to generalize the results of our experiments to industrial practice. (1) *Interaction of selection and treatment*: this appears when data of the experiments used in our framework are not representative of the population we want to generalize. We avoid this threat since all experiments focus on the software engineering topic. (2) *Restricted generalizability*: this appears when results can only be generalizable to a specific context. We suffer this threat because the validation has been done retrospectively, once experiments had been conducted. We have mitigated this threat using source data from 4 experiments conducted in 2 different contexts: model-driven and testing.

4.6 Applications conclusions

Our approach was useful in finding and cleaning DQ problems in the four experiments we applied it. We found that between 55% and 75% of the DQ metrics applied showed the presence of a DQ problem. This implies that DQ is a critical issue, which must be attended in the Empirical Software Engineering domain.

As we already mentioned, the DQMoS and DQMeS were applied after the experiments' results were published. This presents a problem to SEE, as experimenters did not detect the DQ problems during the collection or analysis of their data; and also shows that applying our proposal during the experimental process will increase the probability of finding and correcting more DQ problems.

After analyzing the comparison of results before and after applying corrections in UPV-Replication, we found DQ errors that involved differences in experimental data. Note that even when the results concerning hypotheses tests did not change, the DQ problems presented in the data had an impact on the experimental results after applying the corrections. Therefore, the fact that the hypotheses tests did not show different results is something specific to this experiment. Besides, there is an important novelty after applying the DQMeS: we can trust better the results once we apply the framework. The contribution of using the DQMeS is that together with the results, we can increase the probability that such results have been calculated with correct data. Even though experimenters can make their own manual corrections to improve the quality of the data, this does not assure that DQ problems would be found, as they do not follow a systematic approach like the one we propose in this paper.

Even though accuracy is the DQ dimension that is most considered and analyzed among the authors in the ESE area, is the one with less DQ errors found in these applications. The problems we found belonged to different DQ dimensions, showing the importance of using a multi-faceted quality approach.

In view of these results and the experimenters' opinions, we believe that our approach helps in finding DQ problems and cleaning experimental data, and could be used in order to obtain more reliable experimental analysis and results.

5 CONCLUSIONS

In this work we present an initial DQ Model (DQMoS) and a DQ Methodology (DQMeS) that can be applied to SEE data. The DQMoS defines DQ metrics as the instrument to measure the quality of the data and identify DQ problems. The DQMeS provides a systematic approach in order to analyze and improve DQ in this domain.

The results of applying the DQMoS and DQMeS to four SEE show that experimental data may contain different DQ problems, and that these problems are difficult to be identified in advance. This is the reason why we believe it is important to apply a structured and disciplined framework that allows the improvement of data quality. Researchers normally apply an ad-hoc approach to assess experimental data before obtaining its results, but this, in the four cases we applied our model and methodology, seems not to be enough. By following a systematic approach, we detected DQ problems that experimenters could not find and that impact on the experimental results.

The comparison of the experiment's results before and after applying the corrections to one of the experiments we analyzed, showed that in spite of the small amount of data, we found errors that involved differences in descriptive data, p-values and size effect. So, we identified important DQ problems that the experimenters did not discover on their own. By applying the proposed framework, we increased the confidence in the quality of data used, and we can trust that the experimental results are calculated based on better data quality.

We believe that the multifaceted approach proposed in this work contributes to identify and correct DQ problems that are not usually addressed in SEE; as we showed in four experiments. The positive feedback obtained from the experimenters' surveys showed they value the improvements introduced to the quality of the experimental data by applying the DQMoS and DQMeS. Moreover, they intend to apply it again in future experiences. We also believe that the DQMoS and DQMeS contribute to both DQ and ESE communities. The introduction of a systematic and thorough DQ analysis of SEE data motivates the consideration of DQ aspects that are not normally addressed by researchers in Software Engineering and could otherwise be neglected. We hope this will be a step towards the standardization in the assessment of DQ in the ESE domain. For the DQ community, we introduce a new field of application, by defining a DQ model for a domain that, to the best of our knowledge, has not been explored before.

As future work we will apply our approach to other SEE data from the beginning of the experimental process, and to other empirical studies such as case studies. We propose to apply our DQ model and methodology in SEE that have not been executed yet, in order to evaluate the results of applying our framework during the experimental conception and design. This will allow us to understand if the experimenters, together with the Data Quality Analyst, can carry out activities that help improving the quality of the data during the first phases of the experiment, preventing from finding DQ problems after obtaining the experimental results. We also intend to apply our framework to some of the popular datasets used in the software engineering domain. As a long term goal, we also intend to extend our proposal to the analysis of the data resulting of the execution of other SEE that do not involve humans (such as computational experiments), as well as of the execution of Software

Development Processes. Finally, we plan to analyse and review related works as how DQ is applied in other areas of interest, such as psychology and medical research.

ACKNOWLEDGMENTS

This research was partially funded by PEDECIBA, Uruguay, and has the support of the Generalitat Valenciana through project GV/2021/072.

REFERENCES

[1] Abedjan, Z., Chu, X., Deng, D., Fernandez, R. C., Ilyas, I. F., Ouzzani, M., Papotti, P., Stonebraker, M. and Tang, N. (2016) 'Detecting Data Errors: Where are we and what needs to be done?', Proceedings of the VLDB Endowment, 9(12), pp. 993–1004.

[2] Aranda, J. and Venolia, G. (2009) 'The secret life of bugs: Going past the errors and omissions in software repositories', in Proceedings - International Conference on Software Engineering, pp. 298–308. doi: 10.1109/ICSE.2009.5070530.

[3] Bachmann, A. (2010) Why Should We Care about Data Quality in Software Engineering ? University of Zurich.

[4] Bachmann, A. and Bernstein, A. (2009) 'Software Process Data Quality and Characteristics – A Historical View on Open and Closed Source Projects', in IWPSE-Evol '09 Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, pp. 119–128. doi: 10.1145/1595808.1595830.

[5] Bachmann, A. and Bernstein, A. (2010) 'When process data quality affects the number of bugs: Correlations in software engineering datasets', in Proceedings - International Conference on Software Engineering, pp. 62–71. doi: 10.1109/MSR.2010.5463286.

[6] Bachmann, A., Bird, C., Rahman, F., Devanbu, P. and Bernstein, A. (2010) 'The Missing Links : Bugs and Bug-fix Commits', in FSE-18, pp. 97–106.

[7] Basili, V. R. and Weiss, D. M. (1984) 'A Methodology for Collecting Valid Software Engineering Data', IEEE Transactions on Software Engineering, SE-10(6), pp. 728–738. doi: 10.1109/TSE.1984.5010301.

[8] Batini, C., Barone, D., Mastrella, M., Maurino, A. and Ruffini, C. (2007) 'A framework and a methodology for data quality assessment and monitoring', in In Proceedings of the 12th International Conference on Information Quality, pp. 333–346.

[9] Batini, C. and Scannapieco, M. (2016) Data and Information Quality - Dimensions, Principles and Techniques. Springer (Data-Centric Systems and Applications).

[10] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R. and Zimmermann, T. (2008) 'What Makes a Good Bug Report?', in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York, NY, USA: ACM (SIGSOFT '08/FSE-16), pp. 308–318. doi: 10.1145/1453101.1453146.

[11] Bird, C., Bachmann, A. and Aune, E. (2009) 'Fair and Balanced ? Bias in Bug-Fix Datasets Categories and Subject Descriptors', in ESEC-FSE09.

[12] Bosu, M. F. (2015) Data Quality in Empirical Software Engineering : An Investigation of Time-Aware Models in Software Effort Estimation. University of Otago, Dunedin, New Zealand.

[13] Bosu, M. F. and MacDonell, S. G. (2013) 'A taxonomy of data quality challenges in empirical software engineering', in 22nd Australasian Conference on Software Engineering, ASWEC, pp. 97–106. doi: 10.1109/ASWEC.2013.21.

[14] Bosu, M. F. and MacDonell, S. G. (2013) 'Data Quality in Empirical Software Engineering: A Targeted Review', in Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering. New York, NY, USA: ACM (EASE '13), pp. 171–176. doi: 10.1145/2460999.2461024.

[15] Bosu, M. F. and MacDonell, S. G. (2019) 'Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation'. Journal of Data and Information Quality. Vol. 11. pp. 1-38. doi: 10.1145/3328746.

[16] Caballero, I., Verbo, E., Calero, C. and Piattini, M. (2007) 'A Data Quality Measurement Information Model Based on ISO/IEC 15939', in 12th International Conference on Information Quality (ICIQ), pp. 393–408.

[17] Caro, A., Calero, C., Caballero, I. and Piattini, M. (2006) 'Defining a Data Quality Model for Web Portals', Web Information Systems–WISE 2006, pp. 363–374.

[18] Cartwright, M. H., Shepperd, M. J. and Song, Q. (2003) 'Dealing with missing software project data', in PProceedings of the Ninth International Software Metrics Symposium (METRICS'03), pp. 154–165. doi: 10.1109/METRIC.2003.1232464.

[19] Chen, K., Schach, S. R., Yu, L., Offutt, J. and Heller, G. Z. (2004) 'Open-source Change Logs', Empirical Software Engineering, 9(3), pp. 197–210. doi: 10.1023/B:EMSE.0000027779.70556.d0.

[20] Crosby, P. B. (1984) Quality without tears: the art of hassle-free management. McGraw-Hill.

[21] Disney, A. and Johnson, P. (1998) 'Investigating Data Quality Problems in the PSP (Experience Paper)', Sigsoft'98, pp. 143–152.

[22] Du, J. and Zhou, L. (2012) 'Improving financial data quality using ontologies', Decision Support Systems. Elsevier B.V., 54(1), pp. 76–86. doi: 10.1016/j.dss.2012.04.016.

[23] Embley, D. W., Liddle, S. W. and Pastor, O. (2011) 'Conceptual-Model Programming: A Manifesto', in Embley, D. W. and Thalheim, B. (eds) Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.

3–16. doi: 10.1007/978-3-642-15865-0_1.

[24] Etcheverry, L., Marotta, A. and Ruggia, R. (2010) 'Data Quality Metrics for Genome Wide Association Studies', in 2010 Workshops on Database and Expert Systems Applications, pp. 105–109. doi: 10.1109/DEXA.2010.40.

[25] Herbert, J. and Vallespir, D. (2009) 'Effectiveness and Cost of Verification Techniques: Preliminary Conclusions on Five Techniques', in 2009 Mexican International Conference on Computer Science (ENC), pp. 264–271. doi: 10.1109/ENC.2009.11.

[26] Juran, J. M. and Gryna, F. M. (1988) Juran's Quality Control Handbook. McGraw-Hill (Industrial engineering series).

[27] Juristo, N. and Moreno, A. M. (2010) Basics of Software Engineering Experimentation. 1st edn. Springer Publishing Company, Incorporated.

[28] Juristo, N. and Vegas, S. (2003) 'Functional testing, structural testing, and code reading: What fault type do they each detect?', Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2765, pp. 208–232.

[29] Kasunic, M., McCurley, J. and Zubrow, D. (2008) Can you trust your data? establishing the need for a measurement and analysis infrastructure diagnostic.

[30] Khoshgoftaar, T. M. and Rebours, P. (2007) 'Improving Software Quality Prediction by Noise Filtering Techniques', Journal of Computer Science and Technology, 22(3), pp. 387–396. doi: 10.1007/s11390-007-9054-2.

[31] Liebchen, G. A. (2010) Data cleaning techniques for software engineering data sets. Burnel University.

[32] Liebchen, G. A. and Shepperd, M. (2005) 'Software Productivity Analysis of a Large Data Set and Issues of Confidentiality and Data Quality', in 11th IEEE International Software Metrics Symposium (METRICS'05), pp. 46–49. doi: 10.1109/METRICS.2005.43.

[33] Liebchen, G. A. and Shepperd, M. (2008) 'Data Sets and Data Quality in Software Engineering', in Proceedings of the 4th international workshop on Predictor models in software engineering - PROMISE '08, p. 39. doi: 10.1145/1370788.1370799.

[34] Liebchen, G. A. and Shepperd, M. (2016) 'Data Sets and Data Quality in Software Engineering: Eight Years On', in Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering. New York, NY, USA: ACM (PROMISE 2016), p. 7:1--7:4. doi: 10.1145/2972958.2972967.

[35] Liebchen, G. A. and Twala, B. (2006) 'Assessing the quality and cleaning of a software project dataset: an experience report', in Proceedings of Evaluation and Assessment in Software Engineering (EASE 2006), pp. 1–7.

[36] Liebchen, G. A., Twala, B., Shepperd, M., Cartwright, M. and Stephens, M. (2007) 'Filtering, robust filtering, polishing: Techniques for addressing quality in software data', in Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, pp. 99–106. doi: 10.1109/ESEM.2007.48.

[37] Marotta, A., Fleitas, J. and Fagúndez, S. (2015) 'Data Stream Quality Evaluation for the Generation of Alarms in the Health Domain', Journal of Intelligent Systems. De Gruyter, 24(3), pp. 361–369.

[38] Menzies, T., Brady, A. and Ekrem, K. (2011) 'What is "Enough" Quality for Data Repositories?', Software Quality Professional (SQP), 13(4), pp. 42–51.

[39] Mockus, A. (2008) 'Missing Data in Software Engineering', Guide to Advanced Empirical Software Engineering, pp. 185–200. doi: 10.1007/978-1-84800-044-5.

[40] Moges, H. T., Dejaeger, K., Lemahieu, W. and Baesens, B. (2013) 'A multidimensional analysis of data quality for credit risk management: New insights and challenges', Information and Management. Elsevier B.V., 50(1), pp. 43–58. doi: 10.1016/j.im.2012.10.001.

[41] Moody, D. L. (2003) 'The method evaluation model: a theoretical model for validating information systems design methods.', in Ciborra, C. U., Mercurio, R., de Marco, M., Martinez, M., and Carignani, A. (eds) ECIS, pp. 1327–1336.

[42] Moody, D. L., Sindre, G., Brasethvik, T. and Sølvberg, A. (2003) 'Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework', in Spaccapietra, S., March, S. T., and Kambayashi, Y. (eds) Conceptual Modeling --- ER 2002. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 380–396.

[43] Otto, B., Hüner, K. M. and Österle, H. (2009) 'Identification of Bussiness Oriented Data Quality Metrics', in 14th International Conference on Information Quality (ICIQ 2009).

[44] Panach, J. I., España, S., Dieste, O., Pastor, O. and Juristo, N. (2015) 'In Search of Evidence for Model-driven Development Claims', Information and Software Technology. Newton, MA, USA: Butterworth-Heinemann, 62, pp. 164–186. doi: 10.1016/j.infsof.2015.02.012.

[45] Pastor, O. and Molina, J. C. (2007) Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer-Verlag New York, Inc.

[46] Pipino, L. L., Lee, Y. W. and Wang, R. Y. (2002) 'Data quality assessment', Communications of the ACM, 45(4), p. 211. doi: 10.1145/505248.506010.

[47] Redman, T. C. (1997) Data Quality for the Information Age. 1st edn. Norwood, MA, USA: Artech House, Inc.

[48] Rodriguez, D., Herraiz, I. and Harrison, R. (2012) 'On software engineering repositories and their open problems', in 1st International Workshop on Realizing AI Synergies in Software Engineering, RAISE 2012, pp. 52–56. doi: 10.1109/RAISE.2012.6227971.

[49] Rosli, M. M., Tempero, E. and Luxton-Reilly, A. (2013) 'Can We Trust Our Results? A Mapping Study on Data Quality', in Proceedings of the 2013 20th Asia-Pacific Software Engineering Conference (APSEC) - Volume 01. Washington, DC, USA: IEEE Computer Society (APSEC '13), pp. 116–123. doi: 10.1109/APSEC.2013.26.

[50] Rosli, M. M., Tempero, E. and Luxton-Reilly, A. (2016) 'What is in Our Datasets?: Describing a Structure of Datasets', in Proceedings of the Australasian Computer Science Week Multiconference. New York, NY, USA: ACM (ACSW '16), p. 28:1--28:10. doi:

10.1145/2843043.2843059.

[51] Rosli, M. M. (2018) A Framework for Understanding and Evaluating the Quality of Data Sets in Empirical Software Engineering. The University of Auckland, New Zealand.

[52] Rosli, M. M., Tempero, E. and Luxton-Reilly A. (2018) ‘Evaluating the quality of datasets in software engineering’. Advanced Science Letters. Vol. 24. pp. 7232-7239. doi: 10.1166/asl.2018.12920.

[53] Scannapieco, M. and Catarci, T. (2002) ‘Data Quality under the Computer Science perspective’, Computer Engineering, 2(2), pp. 1–12.

[54] Seo, Y., Yoon, K. and Bae, D. (2008) ‘An empirical analysis of software effort estimation with outlier elimination’, in Proceedings of Predictor Models in Software Engineering (PROMISE 2008), pp. 25–32.

[55] Shepperd, M. (2011) ‘Data quality: Cinderella at the Software Metrics Ball?’, in Proceeding of the 2nd international workshop on Emerging trends in software metrics, pp. 1–4. doi: 10.1145/1985374.1985376.

[56] Shirai, Y., Nichols, W. and Kasunic, M. (2014) ‘Initial Evaluation of Data Quality in a TSP Software Engineering Project Data Repository’, in ICSSP 2014 Proceedings of the 2014 International Conference on Software and System Process, pp. 25–29. doi: 10.1145/2600821.2600841.

[57] Strong, D. M., Lee, Y. W. and Wang, R. Y. (1997) ‘Data Quality in Context’, Communications of the ACM, 40(5), pp. 103–110.

[58] Vallespir, D., Apa, C. and León, S. De (2009) ‘Effectiveness of five verification techniques’, in Proceedings of the XXVIII International Conference of the Chilean Computer Society.

[59] Valverde, M. C., Vallespir, D., Marotta, A. and Panach, J. I. (2014) ‘Applying a Data Quality Model to Experiments in Software Engineering’, in Indulska, M. and Purao, S. (eds) Advances in Conceptual Modeling, QMMQ Workshop. Cham: Springer International Publishing, pp. 168–177.

[60] Wand, Y. and Wang, R. Y. (1996) ‘Anchoring Data Quality dimensions in Ontological Foundations’, Communications of the ACM, 39(11), pp. 86–95.

[61] Wang, R. Y. and Strong, D. M. (1996) ‘Beyond Accuracy: What Data Quality Means to Data Consumers’, Journal of Management Information Systems, 12(4), pp. 5–33. doi: 10.1080/07421222.1996.11518099.

[62] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A. (2000) Experimentation in Software Engineering: An Introduction. Norwell, MA, USA: Kluwer Academic Publishers.

[63] Yoon, V. Y., Aiken, P. and Guimaraes, T. (2000) ‘Managing Organizational Data Resources: Quality Dimensions’, IInformation Resources Management Journal. Hershey, PA, USA: IGI Global, 13(3), pp. 5–13. doi: 10.4018/irmj.2000070101.

[64] Brown, N., & Heathers, J. (2017). The GRIM test: A simple technique detects numerous anomalies in the reporting of results in psychology. Social Psychological and Personality Science, 8(4), 363--369.

[65] Schumm, W. R., Crawford, D. W., & Lockett, L. (2019). Using statistics from binary variables to detect data anomalies, even possibly fraudulent research. Psychology Research and Applications, 1(4), 112-118.

[66] Wicherts, J. (2021). Comparing the prevalence of statistical reporting inconsistencies in COVID-19 preprints and matched controls: A Registered Report. https://osf.io/2yn8c/download}