Applying a Data Quality Model to Experiments in Software Engineering

Carolina Valverde[1], Diego Vallespir[1], Adriana Marotta[1], and Jose Ignacio Panach[2]

[1]Universidad de la República, Montevideo, Uruguay
{mvalverde,dvallesp,amarotta}@fing.edu.uy
[2]Departament d'Informàtica, Universitat de València,Valencia, España
joigpana@uv.es

Abstract. Data collection and analysis are key artifacts in any software engineering experiment. However, these data might contain errors. We propose a Data Quality model specific to data obtained from software engineering experiments, which provides a framework for analyzing and improving these data. We apply the model to two controlled experiments, which results in the discovery of data quality problems that need to be addressed. We conclude that data quality issues have to be considered before obtaining the experimental results.

Keywords: data quality, software engineering, controlled experiments.

Introduction

Empirical Software Engineering collects data for predictions, discoveries, or to determine the effectiveness and impact of use in new techniques and strategies [14]. Data collected during the experimental activities becomes the primary source to obtain the experimental results. These results are assumed to be trusted, and the research community as well as the professionals use them to make decisions. However, the quality of the data used in the software engineering experiments is seldom questioned or analyzed, as Data Quality (DQ) issues have not received the attention it deserves in this area [16]. Thus, the quality of the experimental results could be unknown [13,14].

DQ research area has focused basically on defining different DQ aspects, as well as on proposing techniques, methods and methodologies for measuring and dealing with DQ problems [3,4], [7]. In many research areas, data producers and consumers have recognized DQ issues as an important matter that needs to be considered and attended [5], [8].

The importance of the DQ used by empirical studies has been recognized and studied in the last few years [13,14,15,16,17,18], [30,31]. According to Liebchen, there seems to be an increase over time in the amount of works that consider DQ issues, suggesting that the community is giving more attention to DQ [14]. In the few cases that DQ is considered, the analysis of the quality of the data is normally ad-hoc; i.e. nor a systematic neither a repeatable method is used. In order to change this situation, we propose a framework adapted specifically to this field of study: Experiments in

Software Engineering (ESE). To achieve this, we develop a DQ model and a systematic, disciplined and structured approach (that uses this model) in order to analyze and improve DQ in ESE that involves humans as subjects. Our DQ model defines DQ metrics that are based on techniques proposed in the DQ area [3,4,5,6,7].

In this work, we apply our DQ model to the data of two controlled experiments. We present the application of the model as well as the results obtained by applying the proposed DQ metrics. We found that data used by ESE present DQ problems that need to be addressed before obtaining the experimental results.

The paper is structured as follows. Section 2 presents a DQ meta-model as the conceptual base for our model. Section 3 describes the experiments we used to evaluate our model. Section 4 presents the DQ model and metrics. Section 5 shows the application of a DQ metric. Section 6 presents the results obtained by applying the defined DQ metrics to the experiments presented. Finally, section 7 compares related work, and section 8 presents the conclusions.

Background: Data Quality Meta-model

DQ is generally defined as "fitness for use" [9], [14,15], if data is suitable for its use or purpose. As the use of the data depends on every context, its quality will be evaluated in function of its specific purpose [4]. DQ is a multifaceted concept, as it is defined as a function of the dimensions it describes. Each dimension represents a different aspect (or facet) of DQ [3,4,5,6,7].

Our approach is based on a DQ meta-model [32] that includes the following concepts: a *quality dimension* is a concept that captures one facet of DQ, a *quality factor* represents a particular aspect of a DQ dimension. A dimension is seen as the join of factors having the same aim. A *quality metric* is a quantifiable instrument that defines the way a quality factor is measured, and will indicate the presence of a DQ problem. 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. DQ measurement can be done at different levels of granularity: cell (attribute value for a given tuple), tuple, column, table or even database.

We have chosen the most widely referenced DQ dimensions and their definitions according to [3,4,5,6,7]:

- ***Accuracy***. Specifies how accurate and valid the data is. It indicates if a correct association between the Information System (IS) states and the real world objects exist. Three quality factors are proposed. **Semantic accuracy** refers to how close is a real world value to its representation in the IS. **Syntactic accuracy** indicates if a value belongs to a valid domain. **Precision** refers to the detail level 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 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.
- ***Consistency***. Specifies if the semantic rules are satisfied in the IS. An inconsistency exists when more than one state in the IS is associated with the same object

in the real world. There are different kinds of integrity restrictions. **Domain restrictions** refer to rules about the attributes values. **Intra-relation restrictions** refer to rules that must be satisfied by one or more attributes in the same relation, while in the **inter-relation rules** the attributes entailed are from different relations.

- ***Uniqueness***. Specifies the duplication level of the data through two 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 is always represented in the same format and structure. We consider the 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 factors, ***ease of understanding*** and ***metadata***.

Experiments Description

In order to validate the DQ model defined in this work through a proof of concept, we have applied the DQ model to data collected from two executed experiments in Software Engineering. These experiments are briefly presented in this section.

The experiments aim to compare the Model-Driven Development (MDD) paradigm [12] versus a traditional software development method. MDD proposes focusing all analysts' effort on building a conceptual model that abstractly represents the system. The code is automatically generated through transformation rules. In a traditional development method, the code is manually implemented. The following variables are defined: *Software Quality:* degree of success after applying test cases; *Effort:* time spent to develop the system; *Productivity:* quality-effort ratio; *Satisfaction*: range of values from 1 to 5 (1: totally unsatisfied, 5:totally satisfied).

Subjects of the experiment are master students from the Technical University of Valencia. We have 2 sessions of 2 hours per development method (MDD and traditional) and 2 problems (P1 and P2) to develop a Web application from scratch.

The experiment starts with a demographic questionnaire that subjects have to fill in on paper to identify their background. Next, we apply the traditional development treatment to all the subjects. The experimenter records the start and end session times, and then calculates how much time each subject spent to develop the system (per session and total). Once the session has finished, each subject must fill in a satisfaction questionnaire. Finally, experimenters check the quality of the systems developed through test cases and write down the results. Each test case is defined as a sequence of items. We consider that a test case is 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 spread sheets. Finally, the process is repeated to apply the MDD treatment.

The experiment was carried out twice with the same design. The base experiment was executed in 2012 with 26 subjects, while the replication in 2013 had 20 subjects. In the replication, we extend the problems 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. At the end of each exercise, subjects have to write down on a paper the time spent to finish it. The experimenter then copies these data in the spread sheets.

Data Quality Model for Software Engineering Experiments

A DQ model represents DQ concepts (such us dimensions, factors and metrics) and the relation between them, which are defined for a specific domain. We define a DQ model for the ESE domain, you can find DQ models for other domains in [10,11]. This model provides a framework for the analysis and assessment of DQ. Fig. 1 shows how our DQ model is defined from the DQ meta-model presented in Section 2.

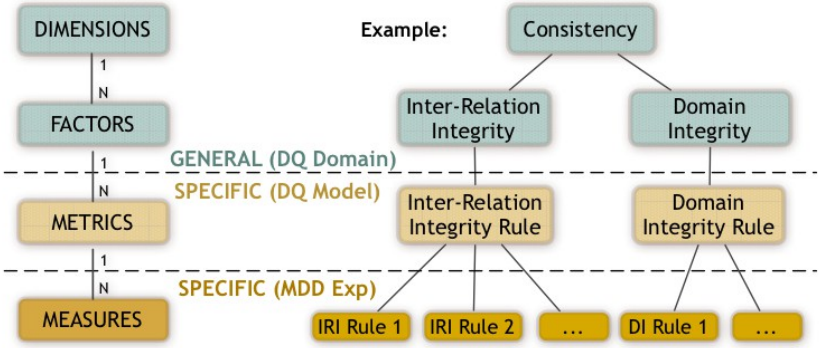

Fig. 1. DQ model defined and its application

Our model defines a set of DQ metrics that can be applied to ESE data, and are based on the DQ concepts presented in Section 2. They are shown in Table 1. We define the metrics by induction, as they are applied to specific experiments (from the particular) and adjusted such a way they could be applied to any ESE data collection (to the general). This model is the result of successive refinements to previous developed models [1,2]. DQ problems 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.

Due to space restrictions, it is not possible to present every metric. In the next section, we show the definition and results of a particular DQ metric of our model. We

choose the metric “Inter-relation Integrity rules” (M14), as it corresponds to a DQ aspect that is not generally addressed in ESE data. In Section 6 we present the results of the application of the whole DQ model to the experiments.

Table 1. DQ Metrics

DQ Dimension	DQ Factor	DQ Metric	Id
Accuracy	Syntactic Accuracy	Out of Range Value	M1
		Lack of Standardization	M2
		Embedded Value	M3
	Semantic Accuracy	Inexistent Record	M4
		Incorrect Record	M5
		Out of Referential Value	M6
	Precision	Lack of Precision	M7
Completeness	Density	Null Value	M8
		Omitted Information	M9
	Coverage	Omitted Record	M10
Consistency	Domain Integrity	Domain Integrity Rule	M11
	Intra-relation Integrity	Intra-relation Integrity Rule	M12
		Unique Value	M13
	Inter-relation Integrity	Inter-relation Integrity Rule	M14
		Invalid Reference	M15
Uniqueness	Duplication	Duplicate Record	M16
	Contradiction	Contradictory Record	M17
Representation	Data Structure	Data Structure	M18
	Data Format	Data Format	M19
Interpretability	Ease of understanding	Ease of understanding	M20
	Metadata	Metadata	M21

Data Quality Metric Example: Inter-relation Integrity Rule

This metric is implemented through integrity rules involving attributes that belong to different relations, and that must be satisfied in the database. Its measurement is defined as the verification of rules satisfaction in the data. 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.

In order to apply this metric to the experiments, rules are defined with the experiment responsible. We define 6 rules for the base experiment and 9 for the replication, but only 3 are presented here.

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.

Total execution time is the sum of the time spent in the two working sessions.

Time spent in making the first exercise has to be less or equal than the total execu-

tion time (only for the replication).

We measure the three cases presented through formulas implemented in a spreadsheet that verifies rules satisfaction. "Rule 1" was not satisfied in any of the experiments. In the base experiment, we found 2 test case records where one of its items result was 0 but the test case result was 1. We discovered that the item involved was not considered to calculate the final result, so it was not necessary to take corrective actions.

In the replication experiment, 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 a null value. This DQ problem impacts on the experimental results obtained (regarding the variable *software quality*).

"Rule 2" was satisfied in both experiments. "Rule 3" was only applied to the replication experimental data, where it 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, it did not exist one of the session time values. Note that these cases were also found by the metric "Omitted Information". Thus, the total execution time only considered the time of one 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 by less than 3 minutes.

Results

Table 2 shows the DQ value obtained for each metric applied. It is calculated as the aggregation of the DQ values obtained by applying the metric to the corresponding data. We applied 16 out of the 21 DQ metrics that are defined in the model (Table 1). We validated with the experiment responsible that the application of the remaining 5 metrics was not necessary for these experiments. Whereas in the base experiment the metrics were applied on 47 different domain objects, in the replication they were applied on 59. The amount of measurements is higher in the second case because more data are collected. As a result of applying our model, we found that in the base experiment 9 of the metrics showed the presence of a DQ problem that needs attention. These DQ problems correspond to 13 different objects. In the replication 10 metrics showed the presence of DQ problems, corresponding to 17 objects. Only 7 of the DQ problems found are common to both cases.

When possible, an automatic measurement method is applied (68% in the base experiment and 75% in the replication), through the implementation of formulas in a spreadsheet. When the measurement is subjective or it involves the comparison with data extracted from the real world, a manual method is used (26% in the base experiment and 22% in the replication). Remaining metrics 6% and 3% respectively, were not implemented since they require a great amount of time from the experimenters.

We propose preventive actions for every case in which a DQ problem was found in order to prevent its occurrence in future experiences. We then classified the DQ problems found. Table 2 shows which metrics correspond to each classification.

Data Errors (DE). In the base experiment, we found data errors for 4 of the 9 metrics. However, no corrective actions were taken. This is because the data affected by the quality problems were not used as a base to obtain the experimental results.

In the replication, we found 4 data errors. For 2 of them corrective actions were taken (1 described in the previous section). In the remaining cases it was not possible to apply corrections because real data was omitted and could not be known.

Questionable Value (QV). We found questionable values for 2 of the metrics in the base experiment and 2 in the replication. These cases correspond to unusual values. We did not identify any possible corrective actions after validation with the experiment responsible.

Improvement Opportunities (IO). We propose specific improvement opportunities for 3 subjective metrics in the base experiment and for 4 subjective metrics in the replication, all referred to the data structure, format and storage.

Table 2. Results of applying the DQ Metrics to the experiments

DQ Metric Id	DQ Value - Base	DQ Problem	# Data - Base	DQ Value - Repl.	DQ Problem	# Data - Repl.
M1	0.968	QV	7	0.988	QV	5
M2	0.941	DE	8	1.000		
M3	0.692	DE	16	1.000		
M5	1.000			1.000		
M7	1.000			1.000		
M8	1.000			0.959	DE	9
M9	1.000			0.800	DE	9
M10	0.996	DE	1	0.933	DE	4
M11	1.000			1.000		
M12	0.971	QV	3	0.963	QV	3
M14	0.857	DE	20	0.926	DE	13
M16	1.000			1.000		
M18	Regular	IO		Regular	IO	
M19	Acceptable			Regular	IO	
M20	Regular	IO		Regular	IO	
M21	Regular	IO		Regular	IO	

As Table 2 shows, the quantity of data with a DQ problem is low. This might be mainly because of the low amount of data collected during the experiment. However, the experimental results could anyway be affected due to the DQ problems found, regardless of the amount of data.

We can see some differences between the results obtained in each case, even though collected data are almost the same. While in the base case no corrective actions were taken, in the replication we found errors in data used to obtain the experimental results that needed correction. Contrary to the replication, in the base case the DQ analysis was carried out after the experimental statistical analysis. Thus, as the amount of data is low, the experimenters could have applied some manual corrections

during the analysis. However, this "ad-hoc" method cannot assure that the DQ problems will be found, as it depends on the person who is making the analysis (his knowledge and abilities), as well as on the complexity and amount of data. Neither has been established a systemic way to perform the DQ analysis, so it cannot be repeated in future experiences.

Another difference is in the metric with the lowest DQ value. While the first case corresponds to data entered by the subjects (M3), the second one is an omission in a calculation that made the experimenter. Moreover, the improvement in the metrics M2 and M3 is because preventive actions were taken during the replication.

The results obtained show that the application of the DQ model proposed to the experimental data allows the identification of DQ problems that could otherwise be ignored. Even though the data analyzed is of low complexity, the application of the DQ metrics uncovers the existence of bad quality data, as showed in the example presented. It is important to analyze and improve the quality of the data used, so that the experimental results can be trusted.

Related Work

The importance of the quality of data used by empirical studies has been acknowledged and assessed in the last years [13,14,15,16,17,18], [30,31], mostly due to the impact that it may have on the decisions taken. 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 [14], [16], [25]. However, we did not find in the literature any study that specifically analyses the quality of the data which source is a controlled experiment in Software Engineering, as we present in this work.

On the other hand, the results of a systematic literature review carried out by Liebchen and Shepperd [14,15] 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 the review recognize its importance (138 out of 161), little work has been done to deal with DQ problems. Liebchen also suggests the development of unified DQ protocols for the empirical software engineering community, since none of the works found proposes or applies one. Our DQ model aims to address this absence, by proposing a DQ framework including the definition of DQ metrics that could be applied to ESE, in order to assess and improve its quality.

Bachman [13], [17], [28] analyzed 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. Bachman 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.

Three literature reviews were carried out in this particular topic, showing interest and concern about how researchers are dealing with DQ problems [13,14], [30]. They all conclude that empirical software engineering community should pay more attention to this issue, which has long been neglected according to the results

We also found another set of related works that analyze the quality in software engineering historic datasets [19,20,21,22,23,24], [26,27], [29]. In these works, software project, process and product data are collected, stored, and then used to analyze strategies and methodologies, construct heuristics or prediction models, or apply statistical techniques. The works found also agree on the great significance of assessing the quality of the data used, as they have an impact on the obtained results.

Conclusions

We present a DQ model that can be applied to ESE data. This model defines DQ metrics as the instrument to measure the quality of the data and identify DQ problems. We show how to apply the DQ metrics defined in two controlled experiments. We found that experimental data has DQ problems, and that they can impact on the results of the experiment.

Our model allows us to predefine metrics for DQ in ESE, being an important contribution to both DQ and ESE communities. The introduction of a DQ analysis previous to the experimental statistical analysis motivates the consideration of DQ aspects that are not normally addressed by ESE researchers and could otherwise be neglected.

Our ultimate goal is to define a DQ model such that it can be useful for any ESE. As future work we will apply our approach to other experiments with a higher number of subjects and data. This way, our systematic approach will probably find a higher amount of DQ problems. As a result of these new applications we aim to adjust (again) our DQ model and achieve its generalization for the ESE domain.

References

Valverde, C., Grazioli, F., Vallespir, D.: A study of the quality of data gathered during the use of personal software process. In: Proceedings JIISIC 2012. Lima, Peru (2012)

Valverde, C., Vallespir, D., Marotta, A.: Data quality analysis in software engineering experimental data. In: Proceedings CACIC 2012, pp. 794-803, Argentina (2012)

Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques. Springer-Verlag Berlin Heidelberg (2006)

Strong, D.M., Lee, Y.W., Wang, R.Y.: Data quality in context. Communications of ACM 40, pp. 103–110 (1997)

Pipino, L., Lee, Y. W., Wang, R. Y.: Data quality assessment. Communications of ACM, vol. 45, no. 4, pp. 211–218 (2002)

Wang, R.Y., Strong, D.M.: Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management Information Systems 12(4), 5–33 (1996)

Scannapieco, M., Catarci, T.: Data quality under a computer science perspective. Archivi & Computer, vol. 2, pp. 1–15 (2002)

Redman, T.: Data Quality for the Information Age. Artech House (1996)

Crosby, P. B.: Quality without tears: The art of hassle free management. McGraw-Hill, New York, USA (1984)

Moranga, M.A., Calero C., Piattini, M.: Comparing different quality models for portals. Online Information Review 30(5): 555-568 (2006)

Etcheverry, L., Marotta, A., Ruggia, R.: Data Quality Metrics for Genome Wide Associa-tion Studies. In DEXA Workshops, pp. 105–109 (2010)

Embley, D. W., Liddle, S., Pastor, Ó.: Conceptual-Model Programming: A Manifesto, in Handbook of Conceptual Modeling, ed: Springer, pp. 3-16 (2011)

Bachmann, A.J.E.: Why Should We Care about Data Quality in Software Engineering? Ph.D. thesis, University of Zurich (2010)

Liebchen, G.A.: Data Cleaning Techniques for Software Engineering Data Sets. Ph.D. thesis, Brunel University (2010)

Liebchen, G.A., Shepperd, M.: Data sets and data quality in software engineering. In: Proceedings PROMISE '08, pp. 39–44, ACM, New York, USA (2008)

Liebchen G. A., Twala B., Shepperd M., Cartwright M., Stephens M.: Filtering, robust, filtering, polishing: Techniques for addressing quality in software data. ESEM'07, pp. 99–106, Madrid, Spain (2007)

Bachmann, A., Bernstein, A.: When Process Data Quality Affects the Number of Bugs: Correlations in Software Engineering Datasets. In MSR '10, pp. 62–71, Cape Town, South Africa. IEEE Computer Society (2010)

Chen, K., Schach, S. R., Yu, L., Offutt, J., Heller, G. Z.: Open-source change logs. Emp. Softw. Eng. 9(3), 197–210 (2004)

Liebchen, G. A., Shepperd, M.: Software productivity analysis of a large data set and issues of confidentiality and data quality. Proceedings of METRICS'05 (2005)

Bachmann, A., Bernstein, A.: Software process data quality and characteristics - a historical view on open and closed source projects. In IWPSE-Evol'09, pp. 119–128, Amsterdam, The Netherlands (2009)

Basili, V., Weiss, D.: A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering. 10(6), 728-738 (1984)

Kim, S., Zhang, H., Wu, R., Gong, L.: Dealing with Noise in Defect Prediction, Proc. of ICSE'11, Honolulu, Hawaii, pp. 481–490 (2011)

Strike, K., Emam, K. E., Madhavji, N.: Software Cost Estimation with Incomplete Data. IEEE Trans. on Software Engineering. 27(10), pp. 890–908 (2001)

Aranda J., Venolia. G.: The secret life of bugs: Going past the errors and omissions in software repositories. In ICSE'09, pp. 298–308 (2009)

Liebchen, G. A., Twala, B., Shepperd, M., Cartwright, M.: Assessing the quality and cleaning of a software project data set: An experience report. In Proceedings of EASE'06. British Computer Society (2006)

Cartwright, M. H., Shepperd, M. J., and Song, Q. Dealing with Missing Software Project Data. In Proceedings of METRICS'03, pp. 154, Australia. IEEE Computer Society (2003)

Rodriguez, D., Herraiz, I., Harrison, R.: On software engineering repositories and their open problems. In RAISE (2012)

Bachmann, A., Bird, C., Rahman, F., Devanbu, P., Bernstein, A.: The Missing Links: Bugs and Bug-Fix Commits. In ACM SIGSOFT / FSE '10, USA. ACM (2010)

Wu, R., Zhang, H., Kim, S., Cheung, S.: ReLink: recovering links between bugs and changes, Proceedings of the 19th ACM SIGSOFT, Szeged, Hungary (2011)

Bosu, M.F., MacDonell, S.G.: Data quality in empirical software engineering: a targeted review. Proceedings of EASE'13, Brazil, ACM Press, pp.TBC (2013)

Bosu, M.F., MacDonell, S.G.: A Taxonomy of Data Quality Challenges in Empirical Software Engineering. Australian Software Engineering Conference, pp. 97-106 (2013)

Etcheverry, L., Peralta, V., Bouzeghoub, M.: Qbox-Foundation: a Metadata Platform for Quality Measurement. DKQ'08 in EGC'08, Sophia-Antipolis, France, January 2008.