What is in our datasets? Describing a structure of datasets

Marshima Mohd Rosli, Ewan Tempero, Andrew Luxton-Reilly
Department of Computer Science
The University of Auckland
Auckland, New Zealand
mmoh603@aucklanduni.ac.nz, ewan@cs.auckland.ac.nz, andrew@cs.auckland.ac.nz

ABSTRACT

In order to facilitate research based on datasets in empirical software engineering, the meaning of data must be able to be interpreted correctly. Datasets contain measurements that are associated with metrics and entities. In some datasets, it is not always clear which entities have been measured and exactly which metrics have been used. This means that measurements could be misinterpreted. The goal of this study is to determine a useful way to understand what datasets are actually intended to represent. We construct precise definitions of datasets and their potential elements. We develop a metamodel to describe the structure and concepts in a dataset, and the relationships between each concept. We apply the metamodel to a number of existing datasets from the PROMISE repository. We found that of the 70 existing datasets we studied, 61 datasets contained insufficient information to ensure correct interpretation for metrics and entities. Our metamodel can be used to identify such datasets and can be used to evaluate new datasets. It will also form the foundation for a framework to evaluate the quality of datasets.

Keywords

data quality; modeling datasets; empirical studies; software engineering datasets

1. INTRODUCTION

Research has shown that the quality of datasets is critical to the results of empirical studies in software engineering [23]. For many years, researchers have been using publicly accessible repositories of datasets in their research. Although datasets play a central role in research, few studies consider the quality of their datasets [26, 20], which may lead to questionable results if the datasets contain quality issues. Studies have indicated that some datasets do have quality issues [10, 21]. These studies consider quality issues such as noise, missing data or incorrect data. However, an issue that has not received much attention is whether or not the data is ambiguous or not able to be interpreted correctly.

As an example of potential problems with correctly interpreting data, the `ant 1.3` dataset for *org.apache.tools.ant.util.IdentityMapper* from the PROMISE repository [19] contains a column labelled "loc". It would seem reasonable to interpret this as "lines of code", except that the measurements presented do not agree with our own measurements for the same metric[1]. The differences could be due to errors in our dataset, errors in the PROMISE dataset, the metrics are different, or what is being measured (the entities) are different. In fact based on the publication that the dataset comes from [12], we know that the metrics are different, but anyone using the dataset would not know that fact. Furthermore, we suspect that the entities being measured are also different. However, from public information, we have no way of confirming what they measured.

To assess the potential for this kind of problem, we have developed a metamodel to describe the structure of datasets to help identify what information may be missing. In this study, we present the metamodel and apply it to a number of public software engineering datasets and demonstrate that there is cause for concern, and also establish that there is value in a metamodel.

In general, software engineering datasets contain data and additional information. This additional information typically describes the purpose, meaning and context, thereby allowing researchers to better understand the data. Such information is referred to as "metadata". In addition, metadata helps researchers to reproduce the results based on the datasets. More importantly, the appropriate metadata will allow researchers to avoid misinterpretation problems that are similar to the *ant* example mentioned earlier.

The ultimate goal of our research is to evaluate the quality of data in software engineering datasets. As a first step, we need a useful way to understand what datasets are actually intended to represent. We begin by investigating existing datasets to identify the different ways datasets are structured. We then construct precise definitions for datasets and their potential elements. As mentioned earlier, we develop a metamodel that describes the structure and concepts in a dataset, and the relationships between each concept. We anticipate that the definitions and the metamodel will allow researchers to clearly identify whether a dataset has sufficient information to be usable for analysis in empirical research.

[1] qualitascorpus.com

ACSW '16 Multiconference, February 02 - 05, 2016, Canberra, Australia

ISBN 978-1-4503-4042-7/16/02...$15.00

DOI: http://dx.doi.org/10.1145/2843043.2843059

The contributions of this paper are:

- Constructing a precise definition for a measurement dataset and measurement metadata
- Developing a dataset metamodel to describe the structure and concepts in a dataset, and the relationships between each concept, and
- Applying the metamodel to existing public software engineering datasets.

The remainder of the paper is organised according to the following sections. In the next section, we present related work. Section 3 discusses the motivation for our work, which includes inspiration from examples of existing datasets. We then discuss our metamodel to describe the structure of a dataset in Section 4. Section 5 presents the evaluation of a metamodel with the existing datasets. Section 6 discusses the value of the metamodel. Finally, we present our conclusions in Section 7.

2. RELATED WORK

In this section, we start with related work that reviews studies on data quality in empirical software engineering. We then discuss data quality issues and type of information stored in software engineering repositories. Next, we present studies that discuss techniques to improve the quality of data. Finally, we discuss how our previous work informed the development of the metamodel presented in this paper.

Liebchen and Shepperd first raised the issue of data quality, and in 2008, reported that data quality had not received sufficient attention from the software engineering community as evidenced by the small number of studies that explicitly reported the quality of datasets [18]. In 2011, Shepperd again noted that data quality is critically important to empirical results and urged the research community to pay attention to this issue [23].

In 2013, Bosu and McDonell systematically reviewed data quality research in empirical software engineering. They focused on three key elements: data collection reporting, data pre-processing and data quality issues. They revealed only 23 studies which had considered the three key elements of data quality in the literature [4].

Gray et al. analysed software datasets from the NASA repository for defect prediction. They found a number of quality issues in the datasets that may lead to questionable results. For example, they found two identical attributes with the same values but having different labels in *KC4* dataset. These labels are: "number of lines" and "loc total". In the dataset documentation, both attributes are poorly defined and this makes it difficult to interpret the measurements. They suggested a data cleaning method for preprocessing the datasets to ensure the data is suitable for research purposes [10].

Shepperd et al. investigated two versions of datasets from the NASA Metric Data Program repository and PROMISE repository. They identified numerous data quality issues and classified them into five categories: identical features, constant features, features with missing values, features with conflicting values, and features with implausible values. They noted that the two versions of datasets have important issues with quality, namely implausible values and insufficient information about the preprocessing of datasets. This indicates that researchers need to consider quality issues in the preprocessing stage prior to applying any classification, algorithm or prediction with the datasets [24].

Rodriguez et al. surveyed 12 public software repositories, including NASA, PROMISE and ISBSG. They classified the repositories into several dimensions, including type of information stored (meta-information, low-level information and processed information) and the format of the information is stored (plain text, ARFF, CSV, databases). They also discussed common quality issues when dealing with the datasets in the repositories, in particular, from the perspective of machine learning [21].

Cheikhi et al. presented a structured overview of datasets in two public software repositories: ISBSG and PROMISE. They classified the content of datasets in the repositories into eight categories: the topics addressed, the source of the datasets, the year in which the dataset was originally donated, the availability, the attribute description, the type of software project used by the dataset, the number of attributes, and the number of instances. They found some PROMISE datasets do not provide descriptions of attributes and past usage information, which are necessary for replication studies. This illustrates that existing datasets may have their own particular way to describe the content of datasets [5].

The research community has responded to the quality issues with several studies on techniques to improve the quality of data. For example, Mockus [20] and Zhang [26] suggested imputation techniques to handle missing data problems. Kim et al [15] and Khoshgoftaar et al. [13] developed noise identification and filtering techniques to improve the quality of data. Bachman et al. [2, 3] enhanced a technique to prepare software engineering process data and constructed a framework to measure the quality of data. Although several techniques were proposed to resolve the challenges associated with data quality, we found there is still a lack of research in assessing the quality of datasets [22].

In our previous work, we reported that researchers used various terminologies to discuss data quality issues in empirical research, especially with respect to the kinds of quality issues a dataset might have. In addition, we also found that they used existing datasets from many different sources of data [22]. It seems possible the existing datasets may have different formats and structures, which create challenges with interpretation of datasets that will make it difficult to clearly report the quality of data. This indicates a need for a standard way of describing a structure of datasets to identify formal definitions for data quality issues. In this paper, we begin the process of developing such way by providing motivation for assessing data quality through observations of existing datasets from software repositories.

3. MOTIVATION

In order to evaluate the quality across different datasets in a standard way, we need a standard terminology to model the dataset. This standard will allow researchers to apply a common interpretation to understand the content of datasets.

Datasets from software repositories come in a great variety of formats and structures. Different formats provide different ways to represent the structure of data. For example, some datasets contain metadata that corresponds to measurement in external locations such as web pages or external files.

In this study, we selected six examples of datasets from real software repositories to illustrate the wide variety of datasets that exist. These examples are selected based on the common formats of datasets that appear in the software repositories.

3.1 Examples of dataset

3.1.1 Example 1: PROMISE Boetticher

Many existing datasets in software repositories are presented in tabular format. This is a simple structure for representing data in rectangular tables made up of columns and rows. The columns are almost always labelled and used to distinguish different properties of entities. The rows are sometimes labelled and each row typically contains values for a particular entity.

Figure 1 illustrates part of the Boetticher dataset [19], which is survey data presented in a typical tabular format. In this example, each row corresponds to values for the properties of entity. The rows are not labelled and there is no metadata in the dataset. However, there is metadata in the web page where the dataset comes from.

In this example, we need to be able to model the column headers that represent different properties of entities. We assume some of the properties are metrics because they look like metrics and are associated with integer values, while other properties do not look like metrics since they are associated with text. This assumption indicates that some of the values associated with the properties are measurements for the metrics.

Every row contains values corresponding to different properties of entities except the column headers. We assume that each row contains information about a particular entity because the conventions for rows in a dataset are each row has values for a different entity. This illustrates that we need to find a way to describe a concept that represents values for a particular entity.

Gender	Highest Degree	Numeric Degree	Comp Sc Undergrad Courses	Comp Sc Grad Courses	Hardware Undergrad Courses	Hardware Grad Courses
Male	Bachelors	3	6	0	6	0
Male	Masters	4	0	1	0	1
Male	Bachelors	3	5	3	2	0
Male	Masters	4	30	6	2	0
Male	Bachelors	3	0	0	0	0
Male	Bachelors	3	0	1	0	3
Male	HS	2	0	0	0	0
Male	Bachelors	3	2	0	3	0
Male	Bachelors	3	1	2	6	2
Male	Bachelors	3	4	0	2	0
Male	Masters	4	0	1	0	0
Male	Bachelors	3	18	1	3	0
Male	Bachelors	3	5	0	8	0
Male	PHD	5	1	0	0	0
Female	Masters	4	6	4	0	0

Figure 1: PROMISE Boetticher dataset.

3.1.2 Example 2 : PROMISE Datatrieve

Figure 2 illustrates part of the datatrieve dataset [19], which is defect analysis data presented in Attribute-Relation File Format (ARFF). In this particular example, there are no columns and it is organised differently to the first example. There are rows that contain properties and values, and there is no metadata in the dataset.

We notice that the elements after the *@attribute* seems similar to the column header in tabular data, but this example is not tabular data. For example, LOC6_0 looks like a column header that represents a property corresponding to values in the rows after the *@data*. This indicates that every element after the @attribute represents a property of entity, and some of these properties could be metrics associated with measurements.

```
@relation DATATRIEVE

@attribute LOC6_0 numeric
@attribute LOC6_1 numeric
@attribute Added_LoC numeric
@attribute Del_LoC numeric
@attribute Diff_Block numeric
@attribute Mod_Rate numeric
@attribute Mod_Know numeric
@attribute ReusedLoC numeric
@attribute Faulty6_1 {0, 1}

@data

127,126,6,7,2,10,1,120,0
458,441,50,67,17,23,1,391,0
182,178,10,14,3,13,1,168,0
270,270,14,14,3,10,1,256,0
107,104,11,14,4,21,1,93,0
892,869,40,63,16,11,1,829,0
816,800,80,96,26,20,1,720,0
685,669,64,84,22,20,1,601,0
2728,2757,73,44,22,4,1,2684,0
2924,2923,219,220,58,14,1,2704,0
875,862,95,108,32,21,1,767,0

{.....omitted.....}
```

Figure 2: PROMISE Datatrieve dataset.

In Figure 2, there is a number of elements that do not name properties, rows and values. In order to describe this kind of dataset, we need to capture these elements, namely: *@relation*, *@attribute*, *@data* and *numeric*. Some of this is metadata, such as *numeric*. The other elements which are not metadata tell us something related to the structure, and used to separate the elements corresponding to the measurements. This illustrates that we need to be able to find how to describe elements that provide information about the structure.

We observe that after the *@data*, every row contains a list of values separated by commas. We assume these values correspond to different properties of entities which are organised in rows after the *@attribute*. This illustrates that each row after the *@data* contains values for a particular entity that refers to the same concept in Example 1. In this case, we need to be able to model this concept in a standard way for different formats of data.

3.1.3 Example 3 : PROMISE KC2

Figure 3 illustrates part of the KC2 dataset [19], which is defect analysis data. The structure of the dataset is similar to Example 2 except there is metadata in the dataset. In this example, the lines of the file that contain metadata begin with a percentage character (%).

There are many different kinds of information in the metadata. Some of the information describes common information that does not relate to measurements (e.g the time the dataset file was created, and the author of the dataset file). Some of the information is about the population (e.g the number of properties, and the number of instances) while other information is about the properties of entities (e.g the description for the properties of entities). This indicates that we need to be able to model the different kinds of metadata because only some of the metadata correspond to the measurements.

```
% 1. Title/Topic: KC2/software defect prediction
% 2. Sources:
%    -- Creators:  NASA, then the NASA Metrics Data Program,
%    {....omitted...}
% 5. Number of instances: 522
% 6. Number of attributes: 22 (5 different lines of code measure,
%         3 McCabe metrics, 4 base Halstead measures, 8 derived
%          Halstead measures, a branch-count, and 1 goal field)
% 7. Attribute Information:
%      1. loc            : numeric % McCabe's line count of code
%      2. v(g)           : numeric % McCabe "cyclomatic complexity"
%      3. ev(g)          : numeric % McCabe "essential complexity"
%      4. iv(g)          : numeric % McCabe "design complexity"
%      5. n              : numeric % Halstead total operators + operands
%      6. v              : numeric % Halstead "volume"
%      7. l              : numeric % Halstead "program length"
%      8. d              : numeric % Halstead "difficulty"
%      9. i              : numeric % Halstead "intelligence"
%     10. e              : numeric % Halstead "effort"
%     11. b              : numeric % Halstead
%     12. t              : numeric % Halstead's time estimator
%     13. lOCode         : numeric % Halstead's line count
%     14. lOComment      : numeric % Halstead's count of lines of comments
%     15. lOBlank        : numeric % Halstead's count of blank lines
%     16. LOCodeAndComment: numeric
%    {....omitted....}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@relation KC2

@attribute loc                 numeric
@attribute v(g)                numeric
@attribute ev(g)               numeric
@attribute iv(g)               numeric
@attribute n                   numeric
@attribute v                   numeric
@attribute l                   numeric
@attribute d                   numeric
@attribute i                   numeric
@attribute e                   numeric
@attribute b                   numeric
@attribute t                   numeric
@attribute lOCode              numeric
@attribute lOComment           numeric
@attribute lOBlank             numeric
@attribute lOCodeAndComment    numeric
@attribute uniq_Op             numeric
@attribute uniq_Opnd           numeric
@attribute total_Op            numeric
@attribute total_Opnd          numeric
@attribute branchCount         numeric
@attribute problems            {no,yes}

@data

1.1,1.4,1.4,1.4,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,2,2,2,2,1.2,1.2,1.2,1.2,1.4,no
1,1.0,1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,yes
{..............omitted..............}
5,1.0,1,1,7,19.65,0.4,2.5,7.86,49.13,0.01,2.73,2,1,0,0,5,2,5,2,1,no
9,2.0,1,1,21,87.57,0.25,4,21.89,350.27,0.03,19.46,8,0,0,0,8,10,11,10,3,no
9,2.0,1,1,21,87.57,0.25,4,21.89,350.27,0.03,19.46,8,0,0,0,8,10,11,10,3,no
6,1.0,1,1,16,57.36,0.35,2.86,20.08,163.88,0.02,9.1,5,0,0,0,5,7,8,8,1,no
6,1.0,1,1,16,57.36,0.35,2.86,20.08,163.88,0.02,9.1,5,0,0,0,5,7,8,8,1,no
```

Figure 3: PROMISE KC2 dataset.

3.1.4 *Example 4 : Qualitas Weka 3.7.5*

Figure 4 shows part of the Weka 3.7.5 dataset [25], which is clone analysis data. This example contains tabular data and metadata in the dataset. It is more complex than the previous examples because this dataset contains data for different kinds of entities. The entities are clone pairs of source code files, clusters of the clone pairs and population data (groups of entities).

This dataset contains two groups of data for different entities. The two groups of data are the clone pairs and the clusters. Each group has column headers that represent different properties of entities.

We notice that every group of entities has a set of descriptions that represents the meaning for each column header. This can be seen in the lines that begin with number signs (#) followed by numbers and the descriptions. We assume that the descriptions are the metadata for the properties of entities because the column headers appear explicitly in the descriptions.

In this dataset, there is population data for the groups of entities. It describes data relating to the measurements applying to the two groups of entities. We assume the population data has the same form as the typical tabular data because it contains measurements, properties of entities, and descriptions for properties of entities. This indicates that we need to be able to model the elements of population data in the same way as we model the elements of regular data.

In Figure 4, there are properties on the highlighted row, but some of these properties are metrics while others are not.

```
Data from clone analysis.
Tool:mete-cmcd: 2013-01-30T1035
Timestamp:Wed Jan 30 22:49:23 NZDT 2013
{............omitted...........}
Clone Pair Information
# 1.Cluster    A unique ID identifying the cluster the clone pair belongs to.
# 2.File1      The name of the file (sans foldpath prefix) containing the lexically first method in the clone pair.
# 3.Method1    The name of the lexically first method in the clone pair.
# 4.Location1  The beginning and ending line numbers in the file where the lexically first method can be found.
# 5.ELOC1      The number of lines of code in the lexically first method.
# 6.Nodes1     The number of nodes in the AST for the the lexically first method.
# 7.File2      The name of the file (foldpath common prefix) containing the lexically second method in the clone pair.
# 8.Method2    The name of the lexically second method in the clone pair.
# 9.Location2  The beginning and ending line numbers in the file where the lexically second method can be found.
# 10.ELOC2     The number of lines of code in the lexically second method.
# 11.Nodes2    The number of nodes in the AST for the the lexically second method.
# 12.Diff      The normalised difference score between the two methods.
# 13.RawDiff   The raw difference score between the two methods.

Cluster  File1  Method1  Location1      ELOC1  Nodes1     File2  Method2  Location2  ELOC2  Nodes2  Diff  RawDiff
C530  /src/main/java/weka/associations/Apriori.java        weka.associations.Apriori.findLargeCarItemSets()  (1475,1517)  29
/src/main/java/weka/associations/Apriori.java weka.associations.Apriori.findLargeItemSets() (1374,1404)      22      238     29.03
C394  /src/main/java/weka/associations/AssociatorEvaluation.java         weka.associations.AssociatorEvaluation.makeOptionStrin
(64,91)         18       190      /src/main/java/weka/classifiers/functions/supportVector/KernelEvaluation.java
weka.classifiers.functions.supportVector.KernelEvaluation.makeOptionString(Kernel) (96,123)   18      190     0.00    0.00
C711  /src/main/java/weka/associations/AssociatorEvaluation.java         weka.associations.AssociatorEvaluation.toSummaryString
(241,250)       7        65       /src/main/java/weka/classifiers/functions/supportVector/KernelEvaluation.java
weka.classifiers.functions.supportVector.KernelEvaluation.toSummaryString(String) (321,330)   7       65      0.00    0.00
{............omitted...........}

Cluster Information
# 1.     Cluster         A unique ID identifying the cluster the clone pair belongs to.
# 2.     Pairs           Number of clone pairs in cluster
# 3.     Methods                  Number of distinct methods in cluster
# 4.     ELOC                     Sum of ELOC for all methods in cluster
# 5.     ELOC(cloned) ELOC for all but the smallest method

Cluster       Pairs   Methods      ELOC        ELOC(cloned)
C61           14           10           87           80
C62           1            2            18           9
C60           1            2            12           6
C59           44           10           85           77
{............omitted...........}
```

Figure 4: Qualitas Weka 3.7.5 dataset.

This illustrates we need to be able to capture the fact that we have these two different kinds of values that potentially can be a measurement or not a measurement. For example, we assume that *Method1* contains values that are not measurements because the description for *Method1* provides information about the entity.

3.1.5 *Example 5 : PROMISE Ant 1.3*

Figure 5 shows part of the Ant 1.3 dataset [19], which is code analysis data. This dataset is presented in a comma separated values (CSV) file. Similarly to Example 1, this dataset has column headers that represent different properties of entities and there is no metadata in the dataset. The metadata is provided in a web page from which the dataset comes.

In this dataset, we notice that some properties of entities contain values which are neither measurements nor identifiers for entities. The property *version* is one such example. This indicates that we need to be able to model the different kinds of values associated with different properties of entities.

```
name,version,name,wmc,dit,noc,cbo,rfc,lcom,ca,ce,npm,lcom3,loc,dam,moa,mfa
ant,1.3,org.apache.tools.ant.taskdefs.ExecuteOn,11,4,2,14,42,29,2,12,5,0.725,395,1,1,
ant,1.3,org.apache.tools.ant.DefaultLogger,14,1,1,8,32,49,4,4,12,0.835164835,257,1,0
ant,1.3,org.apache.tools.ant.taskdefs.TaskOutputStream,3,2,0,1,9,0,0,1,1,0,58,1,1,0.7
ant,1.3,org.apache.tools.ant.taskdefs.Cvs,12,3,0,12,37,32,0,12,12,0.858585859,310,1,
ant,1.3,org.apache.tools.ant.taskdefs.Copyfile,6,3,0,4,21,1,0,4,6,0.7,136,1,0,0.880952
ant,1.3,org.apache.tools.ant.util.GlobPatternMapper,5,1,0,3,15,0,2,1,4,0.5,137,1,0,0,0.
ant,1.3,org.apache.tools.ant.taskdefs.Move,4,4,0,6,32,6,0,6,1,2,325,0,0,0.963855422,
ant,1.3,org.apache.tools.tar.TarInputStream,16,3,0,3,40,84,1,2,16,0.658333333,604,1,
ant,1.3,org.apache.tools.ant.taskdefs.CompileTask,4,5,0,5,21,0,0,5,4,0.333333333,87,
ant,1.3,org.apache.tools.ant.types.PatternSet,17,3,0,10,55,76,5,6,11,0.59375,461,1,0,
ant,1.3,org.apache.tools.ant.taskdefs.Patch,9,3,0,9,29,0,0,9,9,0.583333333,168,1,1,0.
ant,1.3,org.apache.tools.ant.taskdefs.Mkdir,3,3,0,3,14,1,0,3,3,0.5,84,1,0,0.948717949,
ant,1.3,org.apache.tools.ant.taskdefs.Exec,10,3,0,5,41,21,1,5,7,0.793650794,360,1,0,
ant,1.3,org.apache.tools.ant.AntClassLoader,17,2,0,9,64,76,7,2,9,0.879166667,713,0.
ant,1.3,org.apache.tools.ant.types.EnumeratedAttribute,5,1,5,12,11,8,11,1,5,0.75,59,1,
ant,1.3,org.apache.tools.ant.XmlLogger,10,1,0,9,49,1,0,9,8,0.920634921,382,1,0,0,0.
{............omitted..........}
```

Figure 5: PROMISE Ant 1.3 dataset.

3.1.6 Example 6: PROMISE nrpClassic (Nrp1)

Figure 6 illustrates part of the nrpClassic (Nrp1) dataset [19], which is software requirement data. This dataset is in plain text format and contains only values. The values are organised in different rows and there is no metadata in the dataset. The metadata is provided in external files that can be accessed through the web page of dataset.

In this example, each row contains values organised in many different ways. To model this dataset, we need to be able to indicate which values correspond to properties of entities. We assume the values presented in pairs correspond to properties of entities because there is no information that tell us the meaning of the values. This illustrates that some of the values could be measurement for the metrics.

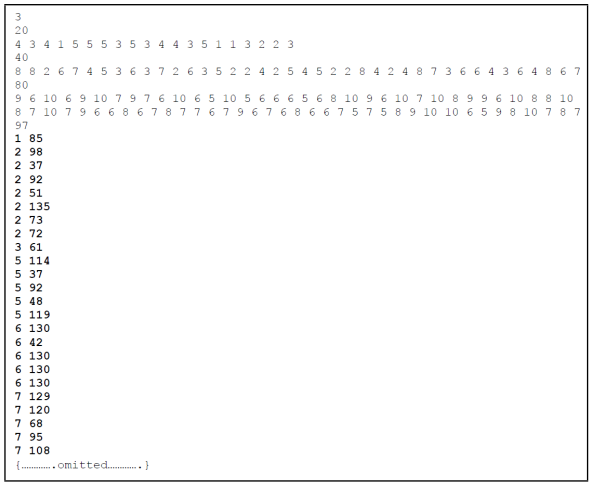

Figure 6: PROMISE nrpClassic (Nrp1) dataset.

3.2 The need for metamodel

Many approaches to conceptual and ontological modelling of data and metadata have been published in software engineering [17, 16, 6, 14, 9]. Although those approaches are mature enough in describing concepts for the purposes to understand the data, they assume that the data (eg. values and properties) stored in datasets are correct. However, some datasets do have quality issues [10, 21] and no models that deal with this kind of datasets exist yet. As mentioned in the introduction, this study proposes a metamodel that contains concept definitions for elements in a dataset and uses it for modelling datasets that have quality issues such as duplicate data. This metamodel is developed to capture the structure and concepts in the datasets and may facilitate the assessment of dataset quality by researchers.

In our previous work, we surveyed recent literature on data quality in software engineering and identified a range of quality issues in datasets [22]. We found many different definitions used to describe the quality issues in datasets. For example, Gray et al. used the term *duplicate data* as "data points (or instances) that appear more than once within a dataset" [11] and Galhardas et al. used the term *exact duplicate data* to refer to "different records have the same value in a field (or a combination of fields) for which duplicate values are not allowed" [8].

The definitions of duplicate data mentioned earlier use a variety of terminologies that can be interpreted in many different ways. For example, the terminology "record" is not necessarily same as a row or a record in a database. It also can be used to refer to a collection of values in a dataset or a basic data structure in programming languages. This indicates that we need to have a common agreement about the definition for record because it can have a different meaning in many circumstances such as databases and certain programming languages.

Therefore, we believe that creating common agreement about the terminology and concepts in datasets is a first step for constructing formal definitions for data quality issues. For the definitions to be useful for researchers practice, the terminology need to have a clear and consistent meaning, and their relationships can be explicitly represented. This is the purpose for dataset metamodel.

As we described in the subsection 3.1, it is clear that existing datasets from software repositories have many different structures to represent the content of datasets. We notice that these datasets do have some similar aspects of data that appear in a same way. For example, the properties are relatively easy to identify because their position in the structure of data. They are also not difficult to find although they do appear in many different positions, such as in tabular format and ARFF. This indicates that some aspects of data can be easily identified across the different structures of datasets. This suggests that the metamodel is possible.

4. DESIGNING A DATASET METAMODEL

In this section, we start with providing a precise definition for a *measurement dataset*, and *measurement metadata*. We then present a dataset metamodel to describe the structure and concepts in a dataset, and the relationships between each concept.

4.1 Definitions for measurement dataset and measurement metadata

Before presenting the definitions, it is necessary to clarify exactly what is meant by measurement. In this study, we adopt a definition of measurement suggested by Fenton and Pfleeger who define measurement as "the process where numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules" [7]. Thus, measurement represents a process that captures information about attributes of entities.

We consider a dataset to consist of a collection of elements. An element can be a single token, unit, word or phrase. Generally, a dataset can be stored in any number of ways, such as in a single file or across multiple files. For simplicity, our definitions are a dataset in a single file. All the definitions can be generalised into other cases. We provide the following definitions for a measurement dataset and measurement metadata.

- A measurement dataset is a collection of elements that includes one or more measurement values.
- Measurement metadata is information about entities and metrics.

4.2 Dataset metamodel

We are interested in datasets that contain data pertaining to entities from some population. Every entity has data

associated with them, and we called the data as "characteristics" of entity except in Section 3, where they were identified as properties. In this study, we divide the characteristics into attributes, which are concepts associated with metrics, and properties, which are concepts not associated with metrics, and we use the properties to distinguish what we mean from the attributes concepts for metrics.

Datasets may contain other data that do not come from attributes and properties. Some aspects of the dataset may be there simply to provide structure (e.g. a string that indicates the beginning of a section). Some may help to organise the values (e.g. column headings which effectively provide names for the characteristics). Some may help to understand what the characteristics are, or what the values mean (e.g. comments providing more detailed explanations of the characteristics than just the column headings). Such information may also provide detailed information about the entities involved, or the population the sample comes from.

To model a dataset, we need to identify all the concepts mentioned above. Some of the concepts are abstract (e.g. an attribute) and so do not explicitly appear in a dataset, but there are parts of the dataset that may correspond to those concepts. Other concepts do appear explicitly, but we want to be able to distinguish their different aspects (e.g. some values are measurement values while others are not).

Figure 7 shows our metamodel for a dataset, using UML notation. It captures the above discussion in more precise detail, as follows.

As mentioned earlier, a dataset is a collection of elements. An element can be a single token or unit or word or phrase. Elements are classified as one of *value*, *label*, *metadata*, *structural*, and *other*. These elements are highlighted as a physical structure in Figure 7.

The metamodel introduce the notion of *value* that we identify the need for in several examples of Section 3. The *values* are associated with the different characteristics of entities. We characterise *value* as *measurement value*, *record identifier*, and *other value* to correspond with each of the characteristics. The *measurement values* are values for attributes of the entities which are associated with metrics. We can see the need for the *record identifier* in Example 4 and *other value* in Example 5. *Record identifiers* are values which are used to identify different entities being measured, and these unique values distinguish one entity from another. *Other values* are values which neither *measurement values* nor *record identifiers*.

We classified elements associated with *values* as *labels* that we specify the need for in Example 1 and Example 4. These *labels* are used to identify the different characteristics of entities which are attributes and properties. We characterise them as *metric label*, *entity label* and *other label*. *Metric label* is used to represent a metric that measures the attributes of entities and is associated with *measurement value*. *Entity label* is used to represent a property that identifies the different entities and is associated with *record identifier*. *Other label* is used to represent a property of entity that is not identified with the entities, but is associated with *other value*.

One of the key aspects of data in our metamodel is *metadata*. We can see the need for the *metadata* in Examples 3 and 4. The *metadata* contain descriptions for different characteristics of entities. We characterise *metadata* as *metric metadata*, *entity metadata*, and *other metadata*. *Metric metadata* describe the metrics used to measure the entities. *Entity metadata* describe the properties of entities that identify the entities being measured. *Other metadata* describe the properties of entities that do not identify the entities being measured.

Elements classified as *structural* commonly provide the structure separation for elements in the dataset. We determine the need for the *structural* in Example 2. The elements for the *structural* appear explicitly as delimiters for an element and a group of elements in the dataset. A delimiter for an element is a sequence of one or more characters used to specify the boundary between each element. (e.g tab, colon, end of line and hashtag.) For a group of elements, the delimiter is an element that is used to specify the boundary between groups of elements within a dataset. (e.g a caption for a section in a dataset.)

We characterise elements that provide common information not related to measurement in the dataset as *other*. We consider these elements neither *values*, *labels*, *metadata* nor *structural*. We can see the need for such elements in Example 3.

We introduce the concept of *record* that we determined the need for in Example 1 and Example 2. We define a record as a list of values that contains information about a particular entity. This concept of *record* is not same as a row or a record in a database because the concept is associated with a single entity. The list of values in the record may consist of *measurement value*, *record identifier* and *other value*. We use the *record identifier* for entity identification to distinguish among the records.

Below, we provide precise definitions for the elements mentioned above.

(a) Label: An element that is used to represent a characteristic.

(b) Value: An element that is recorded about a characteristic.

(c) Structural: A set of elements that consist of delimiters that used to specify boundary between each element and a group of elements.

(d) Metadata: A set of elements that represents description for characteristics.

(e) Other: A set of elements that is not related to measurement.

(f) Measurement value: An element that obtained through the process of measurement for attributes of entities.

(g) Record identifier: An element that uniquely identify an entity.

(h) Other value: An element that is not a measurement value and not a record identifier.

(i) Metric label: An element that represents a metric that measures the attribute of entity.

(j) Entity label: An element that represents a property that identifies the entity being measured.

(k) Other label: An element that represents a property that does not identify the entity being measured.

(l) Metric metadata: A set of elements that describes the metric used to measure an entity.

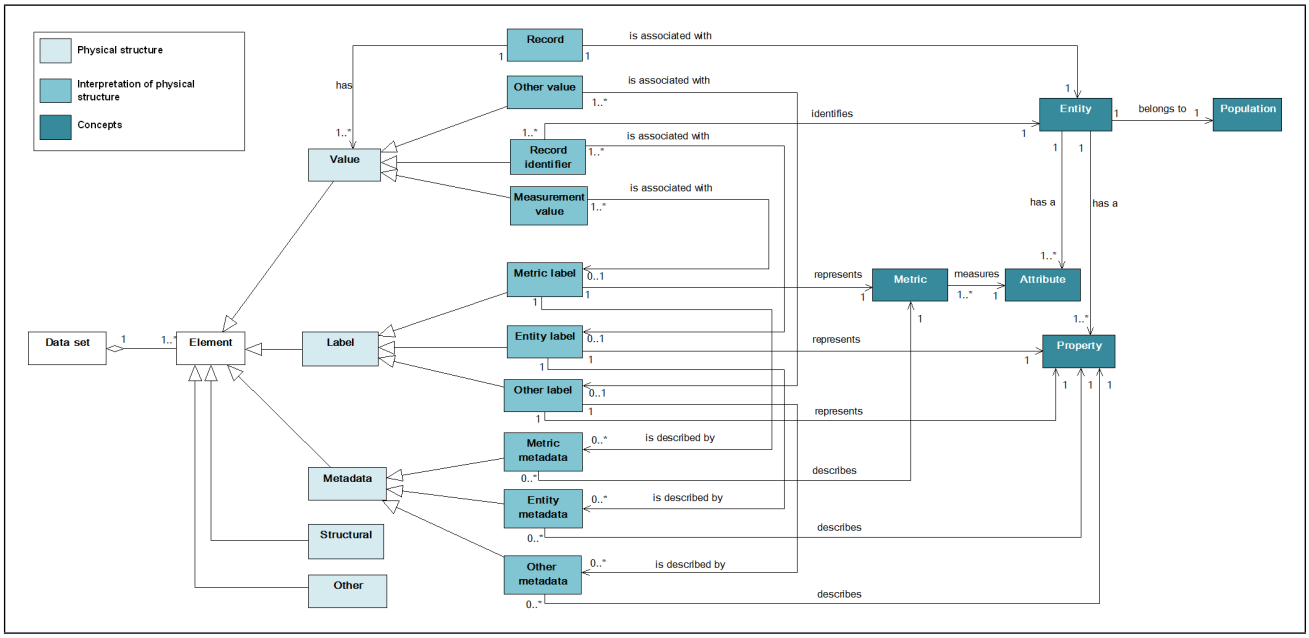

Figure 7: Dataset metamodel

(m) Entity metadata: A set of elements describing the property that identifies the entity being measured.

(n) Other metadata: A set of elements describing the property that does not identify the entity being measured.

(o) Record: A list of values that contains information about a particular entity.

The metamodel illustrates five concepts that do not appear explicitly in a dataset. The concepts are *metric*, *entity*, *attribute*, *property* and *population*. We adopt the definition for *metric* provided by IEEE Standard Glossary of Software Engineering Terminology [1] to enable a common understanding throughout this study. The following are precise definitions for the concepts.

(a) Metric: A metric is 'a quantitative measure of the degree to which an item possesses a given quality attribute [1].'

(b) Attribute: A characteristic of entity that associated with metric.

(c) Property: A characteristic of entity that are not associated with metric.

(d) Entity: A thing that can be measured by its attributes or properties.

(e) Population: A group of entities.

5. EVALUATION

In this section, we selected 64 real datasets from PROMISE repositories which have similar structure to the examples in Section 3. In total, we modelled 70 datasets with our metamodel, including the examples of datasets in Section 3. We started by mapping the data sets to the physical structure of data. We then determined the interpretation of the physical elements based on the degree of metadata provided in the dataset.

Below we present the evaluation of datasets in Section 3 in more precise detail.

5.1 Example 1: PROMISE Boetticher

To model this dataset, we highlighted the column headers in bold face in Figure 1. We modelled the column headers as labels. Based on our assumption in Section 3, we modelled some of the labels as *metric labels* associated with *measurement values*. For example, we modelled label *Comp Sc Undergrad Courses* as a *metric label* because it is associated with integer values in the fourth column which we assumed to be *measurement values*.

During modeling the dataset, we notice two labels that could have an issue with inconsistent data. The labels are *Highest Degree* associated with string values (e.g Bachelors, Masters) and *Numeric Degree* associated with integer values (e.g 3, 4, 5). We notice that every row contains value *Bachelors* for label *Highest Degree* has value *3* for label *Numeric Degree*. This indicates that label *Highest Degree* and label *Numeric Degree* could be modelled as the same metric associated with two different *measurement values* or two different metrics associated with two different *measurement values*. This illustrates that the metamodel is able to model the dataset even though there is a potential quality issue with the data.

As mentioned in Section 3, values in this dataset are organised into rows which contain information about the entities. We modelled every row as *record* because we assumed that every row contained a list of values for a particular entity. However, we have no information to confirm that every record is a different entity. This illustrates the issue that there could be missing metadata.

5.2 Example 2 : PROMISE Datatrieve

In this dataset, we highlighted the elements that we as-

sumed to be labels after the *@attribute* in bold face in Figure 2. This is because these elements look similar to the column header in tabular data. In this case, we modelled every label as a *metric label* because we assumed the values after the *@data* are *measurement values*.

As described in Section 3, there are elements other than metadata that provide the structure separation for the dataset. We highlighted these elements in italics in Figure 2. We modelled these elements as *structural* because they represented elements used to specify the boundary between elements in the dataset.

In this dataset, we modelled every row after the *@data* as *record* because we assumed that each row contained a list of values for a particular entity. This illustrates that we use the same method to model the record with Example 1, although the values are organised in a different format.

5.3 Example 3 : PROMISE KC2

The structure of the KC2 dataset is the same as the previous example, and it allows us to model all the same elements in a similar way. In this dataset, we want to focus on the use of *metric metadata*, *record* and *other*.

As mentioned in Section 3, this dataset contains many different kinds of information in metadata. We modelled the information that describes the common information as *other* because they are not related to the measurement in the dataset. We assumed that the information about the properties of entities described the meaning for each metric label in the dataset. On this assumption, we modelled this information as *metric metadata*.

The information about the population contains data that represents the properties of entities corresponding to the measurements. We modelled the population data in the same way as regular data because we consider a population is another kind of entity, which is a group of entities. For example, the population data contains a label (*Number of instances*) and a value (*522*). We modelled the *Number of instances* as the *metric label* and the associated value as the *measurement value*.

During modeling the records, we noticed there are four rows that contain identical data. If an ID or an *record identifier* is given for every row, it will be obvious that every row is different and indicates there is an entity being measured. In this particular case, we assumed that these rows belong to different entities, but we have no information to confirm it. For that reason, we were not able to distinguish between duplicate data due to the rows being accidentally being reproduced. This illustrates that, with less information describing the data, we were not able to completely model the dataset.

5.4 Example 4 : Qualitas Weka 3.7.5

The structure of Weka dataset is different from other examples but all the same elements as with the previous examples are modelled in the dataset. In this example, we want to focus on the use of the *entity metadata*, *entity label*, *record identifier*, *other value*, *other label* and *other metadata*.

As mentioned earlier in Section 3, we assumed that *Method1* in the clone pair data contains values that are not measurement but identify the entities. In this case, we modelled the values associated with *Method1* as *record identifiers* because the values represent the entities being measured. In addition, we modelled *Method1* as *entity label* because the description for *Method1* described the property of entity that identified the entity being measured. For that reason, we modelled the description for *Method1* as the *entity metadata*.

In the same clone pairs data, we assumed *File1* contained values that are neither measurement values nor record identifiers. This is because the description for *File1* describes the property of entity that does not identify the entity being measured. In this case, we modelled the values associated with *File1* as *other value*, *File1* as *other label* and the description for *File1* as *other metadata*.

We highlighted elements below the description *Global values* in italics in Figure 4 to illustrate the population data in the dataset. We modelled this population data in the same way as the typical tabular data. For example, we modelled the label (*Files*) as *metric label*, the value (*1006*) as *measurement value* and the description (*Number of files analysed*) as *metric metadata*.

5.5 Example 5 : PROMISE Ant 1.3

The structure of Ant 1.3 dataset is same as Example 1. We modelled *metric label* and *measurement values* the same as Example 1, and we focused on the use of *other value* and *other label*.

We highlighted the column headers in bold face in Figure 5, and we modelled them as labels. Based on our assumption in Section 3, some of the labels contain values that are neither measurements nor identifiers for different entities. We modelled these values as *other value* because they are not measurement values and not record identifiers. For example, we can see that the values for label *version* and label *name* in the first column represents values for properties of entity that do not identify the entities being measured. Therefore, we modelled these values as *other value* and the associated labels as *other label*.

During modeling the dataset, we noticed that there were two labels that are same but contain two different values. The same label is *name* and the two values represent *name of dataset* and *name of source code file*. Based on the two different values, we assumed that the labels represented two different properties of entities. As mentioned above, we modelled the label *name* in the first column as *other label*. For the label *name* in the third column, we modelled it as *entity label* because it is associated with the values that we assumed to be *record identifiers*. This illustrates that the process of modeling the dataset is able to mitigate the potential quality issues in a dataset.

5.6 Example 6: PROMISE nrpClassic (Nrp1)

This dataset contains only values that are organised in different rows. We highlighted the pairs of values in bold face in Figure 6. Based on our assumption in Section 3, we modelled the values in pairs as *measurement values*. However, we cannot confirm the assumption that we made because there is no metadata in the dataset. This illustrates that there is a potential issue for data quality.

5.7 Overview of evaluation

In order to present an overview of this evaluation, we present a matrix of dataset metamodel elements on the existing datasets in Figure 8. The columns of the matrix consist of 15 elements of metamodel. We exclude the concepts because they represent abstract concepts that do not appear

explicitly in a dataset. The rows contain 70 datasets, including the examples in Section 3. We group a number of datasets that have the same structure and column headers in a single row, for example Ant 1.3 dataset has the same structure and column headers with 32 datasets.

From the data in Figure 8, it is apparent that every dataset contain *values* and *structural* elements for the physical structure of data. However, 5 of 70 datasets contain no more elements. Additional 61 datasets contain *labels* without any other information, and the remaining of datasets contain *labels* with *metadata* elements that describe the *labels*.

We modelled every dataset with assumptions that we made based on the degree of metadata in the dataset. As can be seen in Figure 8, each dataset contains values that we modelled them as *measurement values* because we assumed every dataset contains measurement values. As mentioned earlier in Section 4, measurement values are associated with metric label that described by metric metadata. It seems a reasonable assumption for PROMISE KC2, Qualitas Weka, PROMISE cocomosdr and PROMISE nasa93 because they contain *metric metadata* in the datasets.

No.	Measurement data sets / Dataset metamodel elements	Label	Value	Structural	Metadata	Other	Record	Metric label	Entity label	Other label	Measurement value	Record identifier	Other value	Metric metadata	Entity metadata	Other metadata
1	PROMISE Boetticher	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
2	PROMISE Datatrieve	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
3	PROMISE KC2	√	√	√	√	√	√	√	–	–	√	–	–	√	–	–
4	Qualitas Weka 3.7.5	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
5	PROMISE Ant 1.3 (33)	√	√	√	–	–	√	√	√	√	√	√	√	–	–	–
6	PROMISE Nrp1 (5)	–	√	√	–	–	√	–	–	–	√	–	–	–	–	–
7	PROMISE PC1 - PC5 (5)	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
8	PROMISE KC3	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
9	PROMISE CM1	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
10	PROMISE MW1	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
11	PROMISE Mozilla4	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
12	PROMISE Kemerer	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
13	PROMISE China	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
14	PROMISE kitchenham	√	√	√	–	–	√	√	√	–	√	√	–	–	–	–
15	PROMISE coc81	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
16	PROMISE miyazaki	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
17	PROMISE bugreport	√	√	√	–	–	√	√	√	√	√	√	√	–	–	–
18	PROMISE mtdjedit	√	√	√	–	–	√	√	√	–	√	√	–	–	–	–
19	PROMISE PITSA - PITSF (6)	√	√	√	–	–	√	√	√	√	√	√	√	–	–	–
20	PROMISE cocomosdr	√	√	√	√	√	√	√	–	–	√	–	–	√	–	–
21	PROMISE nasa93	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
22	PROMISE MC1-MC2 (2)	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
23	PROMISE JM1	√	√	√	–	–	√	√	–	–	√	–	–	–	–	–
24	PROMISE AM1	√	√	√	–	–	√	√	√	–	√	√	–	–	–	–

Figure 8: Matrix of dataset metamodel.

6. DISCUSSION

This study set out with the aim of understanding what datasets are actually intended to represent. We constructed precise definitions of datasets and their potential elements, and developed a dataset metamodel to describe the structure and concepts in a dataset. We evaluated our metamodel with the existing datasets from software repositories. We found that any form of dataset can be modelled using our dataset metamodel.

We found that 61 of 70 datasets contained *value*, *label* and *structural* for the physical structure of data. It seems possible that this is due to the nature of the real datasets that they typically have measurements with multiple columns, and the column headers used to distinguish these columns were identified as *labels*, mainly for metrics and entities. In addition, although the *metadata* is not available for these datasets, this does not mean that all the *values* can be misinterpreted. In some cases, it could be that the *labels* are sufficient to uniquely identify the metrics and the entities.

We noticed that the process of modelling the datasets due to insufficient metadata in datasets may result in data quality issues. For example, PROMISE KC2 illustrates that without a *record identifier* in every record, we are unable to distinguish between duplicate data or different entities which might have the same data. In this particular case, we found the data quality issues by modelling the dataset. This indicates that the metamodel has a capability to identify the potential issues of data quality even though the purposes is to describe the structure of data.

The overall result indicates that very few datasets in the PROMISE repository actually contain adequate metadata to describe what the *labels* mean. In the introduction, we mentioned that the *ant* dataset example indicates an issue with interpretation of data because no information available to explain what the LOC means in the dataset. For that reason, other researchers using the same datasets can potentially misinterpret what the measurement means. We speculate that this could be a potential risk of misinterpretation for every dataset that have *labels* without *metadata* in the PROMISE repository.

7. CONCLUSIONS

In order for datasets to be useful, we need to understand what the measurement values in them are intended to represent. To do so, it is necessary to have adequate metadata about the entities and metrics used. This will allow us to interpret the datasets correctly and use them for analysis in empirical research.

This study has constructed precise definitions of datasets, and developed a metamodel that will allow researchers to apply a standard interpretation to understand the content of datasets. We evaluated datasets from real software repositories that have a variety of formats and structures to demonstrate the metamodel utility. Every dataset contains *values* and *structural* elements. However, 5 of 70 datasets contain nothing more than *values* and *structural* elements, and an additional 61 datasets contain *labels* without *metadata* elements. The remaining 4 datasets contain *labels* with *metadata* that describes what the *labels* mean.

As discussed in Section 5, the *labels* element are not always sufficient to completely determine the metric that has been used, or the entity being measured. Our evaluation suggests that there is a clear risk that most datasets could be used incorrectly if researchers access only the datasets.

If researchers look further than the datasets, there is more metadata available for entities or metrics in external locations. For these datasets, the risk of misinterpretation is lower. Our metamodel can model this information as metadata in the external locations, but as it is separate from the datasets there is a risk for anyone using it for which they will not be aware.

As discussed in Section 6, the process of modelling the dataset by itself may identify potential data quality problems. This is because every assumption that we need to make due to the limitation of metadata could have risks of

being misinterpreted. This indicates that our metamodel is able to support the determination for data quality issues that might occur in a dataset.

Overall, we can see that most of the datasets in the PROMISE repository have a risk of being misinterpreted. As a consequence, we believe that when a dataset is being created, it is important to provide complete metadata. The dataset metamodel defined in this paper embodies the necessary criteria for describing the elements of datasets to be usable for analysis in empirical research. This metamodel also contributes to unify the terminology of datasets, constituting a first step towards an ontology for measurement datasets in future research.

Acknowledgment

The research is funded by the PReSS Account of University of Auckland, Ministry Of Higher Education Malaysia and Universiti Teknologi MARA (UiTM).

8. REFERENCES

[1] *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990, 1990.

[2] A. Bachmann and A. Bernstein. Software process data quality and characteristics: A historical view on open and closed source projects. In *Int. and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) workshops*, pages 119–128. ACM, 2009.

[3] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein. The missing links: Bugs and bug-fix commits. In *Int. Symp. on Foundations of Software Engineering*, pages 97–106. ACM, 2010.

[4] M. F. Bosu and S. G. MacDonell. Data quality in empirical software engineering: a targeted review. In *Int. Conf. on Evaluation and Assessment in Software Engineering*, pages 171–176. ACM, 2013.

[5] L. Cheikhi and A. Abran. An Analysis of the PROMISE and ISBSG Software Engineering Data Repositories. *Int. Journal of Computers and Technology*, 13(5), 2014.

[6] L. Chirinos, F. Losavio, and J. Bø egh. Characterizing a data model for software measurement. *Journal of Systems and Software*, 74(2):207–226, Jan. 2005.

[7] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 2nd edition, 1998.

[8] H. Galhardas and J. Barateiro. A survey of data quality tools. *Datenbank-Spektrum*, 1:14:15–21, 2005.

[9] F. García, M. F. Bertoa, C. Calero, A. Vallecillo, F. Ruíz, M. Piattini, and M. Genero. Towards a consistent terminology for software measurement. *Journal of Information and Software Technology*, 48(8):631–644, Aug. 2006.

[10] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. The misuse of the NASA Metrics Data Program data sets for automated software defect prediction. In *Int. Conf. on Evaluation & Assessment in Software Engineering*, pages 96–103. IET, 2011.

[11] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. Reflections on the NASA MDP data sets. *IET Software*, 6(6):549, 2012.

[12] M. Jureczko and L. Madeyski. Towards identifying software project clusters with regard to defect prediction. In *Int. Conf. on Predictive Models in Software Engineering*, pages 9:1–9:10, New York, USA, 2010. ACM.

[13] T. Khoshgoftaar and J. Van Hulse. Identifying noise in an attribute of interest. In *Int. Conf. on Machine Learning and Applications*, page 6. IEEE, 2005.

[14] H. M. Kim, A. Sengupta, M. S. Fox, and M. Dalkilic. A Measurement Ontology Generalizable for Emerging Domain Applications on the Semantic Web. *Journal of Database Management*, 18(1):20–42, 2007.

[15] S. Kim, H. Zhang, R. Wu, and L. Gong. Dealing with noise in defect prediction. In *Int. Conf. on Software Engineering*, pages 481–490. IEEE, 2011.

[16] B. Kitchenham. Modeling software measurement data. *IEEE Transactions on Software Engineering*, 27(9):788–804, 2001.

[17] B. Kitchenham, S. L. Pfleeger, and N. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 21(12):929–944, 1995.

[18] G. A. Liebchen and M. Shepperd. Data sets and data quality in software engineering. In *Int. Workshop on Predictor Models in Software Engineering*, pages 39–44. ACM, 2008.

[19] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The PROMISE repository of empirical software engineering data, June 2012.

[20] A. Mockus. Missing data in software engineering. *Guide to Advanced Empirical Software Engineering*, page 185, 2008.

[21] D. Rodriguez, I. Herraiz, and R. Harrison. On software engineering repositories and their open problems. In *Int. Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, pages 52–56. IEEE, 2012.

[22] M. Rosli, E. Tempero, and A. Luxton-Reilly. Can we trust our results? a mapping study on data quality. In *Asia-Pacific Software Engineering Conf.*, volume 1, pages 116–123. IEEE, Dec 2013.

[23] M. Shepperd. Data quality: Cinderella at the software metrics ball? In *Int. Workshop on Emerging Trends in Software Metrics*, pages 1–4. ACM, 2011.

[24] M. Shepperd, Q. Song, Z. Sun, and C. Mair. Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, 39(9):1208–1215, Sept 2013.

[25] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble. The Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies. In *Asia Pacific Software Engineering Conf.*, pages 336–345. IEEE, Nov. 2010.

[26] W. Zhang, Y. Yang, and Q. Wang. Handling missing data in software effort prediction with naive Bayes and EM algorithm. In *Int. Conf. on Predictive Models in Software Engineering*, page 4. ACM, 2011.