Statistical Dataset on Software Metrics in Object-Oriented Systems

Tarcísio G. S. Filó and Mariza A. S. Bigonha
Department of Computer Science, UFMG, Brazil
{tfilo,mariza}@dcc.ufmg.br

Kecia A. M. Ferreira
Department of Computing, CEFET-MG, Brazil
kecia@decom.cefetmg.br

ABSTRACT

This paper presents a set of statistical data on the software metrics of object-oriented systems. The data were generated from the Qualitas.*class* Corpus, which gathered a large amount of metrics data from the 111 systems included in Qualitas Corpus. We used the R project for Statistical Computing to generate 6 statistical graphs, 4 summarization/aggregation tables and an R script, for each of the 21 metrics evaluated in each of the 111 systems. This amounted to 13,986 graphs, 8,800 tables and 2,200 R scripts. We also utilized EasyFit to fit a large number of distributions to each dataset, which in turn provided the best fitting statistical distribution, as well as the fit ranking. We also provide a MySQL database dump that normalizes the metric measures and facilitates data manipulation tasks such as filtering and aggregation. By making this set available, we intend to help researchers in their work on software metrics.

1. INTRODUCTION

Understanding the shape of existing software is the first step towards identifying good software. However, we know very little about the large-scale structures of the source code of existing software. [1]. In this context, metrics provide a quantitative way in which to measure software, allowing researchers to understand the kinds of structures present in current programs.

Barriers to measuring code and the interpretation of these measurements include access to the program code, as well as the selection of a suitable measurement tool [2]. Terra et al. [3] provided Qualitas.*class* Corpus: a compiled version of the 111 systems included in Qualitas Corpus [2]. They also gathered a large amount of metrics data from these systems. We were therefore able to provide, as a complement to Qualitas.*class* Corpus, a statistical dataset of software metrics in object-oriented systems.

We utilized the R Project for Statistical Computing[1] to generate 6 statistical graphs and 4 summarization/aggregation tables for each of the 21 evaluated metrics in each of the 111 systems included in the corpus. We also provide the R script which generates this information, thus allowing researchers to improve it according to their needs.

One of the most important problems in statistics is knowing the distribution of a dataset [4]. The statistical distribution that best fits the data, as well as the fit ranking for each system studied is therefore an additional contribution of this research. It is important to know basic features such as the distribution of the software structures [1]. We used the EasyFit Distribution Fitting Software[2] to select the statistical distribution that best fits the data. This reduces the effort when conducting empirical studies on software metrics, providing fundamental information to the understanding of the software structures.

This research provides a large amount of statistical data about the 111 systems of Qualitas.*class* Corpus. Our goal is to help researchers by reducing or eliminating the effort required to generate statistical information when conducting empirical studies related to object-oriented software metrics. We also provide a MySQL database dump that normalizes the metric measures, making data manipulation tasks such as filtering and aggregation easier.

This paper is organized as follow: Section 2 describes Qualitas.*class* Corpus, the collection of 111 system on which our statistical dataset is constructed. Section 3 shows the proposed relational database model of software metrics, which describes the MySQL database dump provided in our dataset. Section 4 describes all of the statistical data generated. Section 5 reports the structure of the information available in the dataset. Section 6 presents our conclusion.

2. QUALITAS.CLASS CORPUS

Qualitas.*class* Corpus is a compiled version of the Qualitas Corpus, provided by Tempero et al. [2]. It provides compiled Eclipse Java projects for the 111 systems included in the last release of the corpus[3]. The aim of Qualitas.*class* Corpus is to assist researchers by removing the need for compilation when conducting empirical studies. It contains more than 18 million lines of code, 200,000 compiled classes, and 1.5 million compiled methods.

The Qualitas.*class* Corpus contains a collection of systems, each of which may contain one or more projects. For instance, the AspectJ system is divided into four internal projects: matcher, rt, tools and weaver. Thus, the 111 systems of Qualitas.*class* Corpus are comprised of 802 internal projects, including 461 from NetBeans. Qualitas.*class* Corpus utilized Metrics[4] to compute the metrics.

An XML file is provided for each project, which contains the values of the source code metrics. The XML file structure is shown in detail in Appendix A. Qualitas.*class* Corpus contains the values of 23 source code metrics, distributed as follows:

- **Basic metrics:** total lines of code (TLOC), number of packages (NOP), number of classes (NOC), number of interfaces (NOI), number of methods (NOM), number of fields (NOF), number of overriden methods (NORM), number of

[1]http://www.r-project.org/
[2]http://www.mathwave.com
[3]The current release is 20120401
[4]Metrics 1.3.8, http://metrics2.sourceforge.net

DOI:10.1145/2659118.2659130 http://doi.acm.org/10.1145/2659118.2659130

parameters (PAR), number of static methods (NSM) and number of static fields (NSF).

- **Complexity metrics:** method lines of code (MLOC), specialization index (SIX), *McCabe* cyclomatic complexity (VG), nested block depth (NBD) and normalized distance (RMD).
- **CK metrics**: weighted methods per class (WMC), depth of inheritance tree (DIT), number of children (NSC) and lack of cohesion in methods (LCOM).
- **Coupling metrics:** afferent coupling (CA), efferent coupling (CE), instability (RMI) and abstractness (RMA).

Our dataset does not include TLOC and NOP, because these are metrics that contain only one value per system.

3. RELATIONAL DATA MODEL ON OBJECT-ORIENTED SOFTWARE METRICS

In order to read the XML files provided in Qualitas.*class* Corpus, it was necessary to develop a tool using JAXB. JAXB[5] provides a fast and easy way to read XML documents, performing the *unmarshalling* process, which is the conversion of the XML document to a tree of Java objects. The tool then uses these Java objects to insert the data into a MySQL database, in order to structure and normalize the measures. The proposed data model for storing the measures can be seen in the entity-relationship diagram of Figure 1.

After the extraction of the data from the XML files to the database, we have a reliable and fast way to aggregate and summarize the metrics data. For example, an absolute frequency table for the metric weighted methods per class (WMC) can be easily obtained by an SQL command. This kind of table counts the number of times a measure has occurred in the data. For example, there are 10 classes with $WMC = 10$ and 20 classes with $WMC = 5$. The following SQL command gets this table in the proposed relational database model:

```
SELECT
        value, count(value)
FROM
        measure
WHERE
        metric_id = 'WMC'
GROUP BY value;
```

In our dataset, we provide a MySQL database dump, whose structure is fully populated with the metrics data. Table 1 summarizes the number of measures per metric in the MySQL database dump. For instance, CA has 16,566 measurements, i.e., there are 16,566 values measured for this metric that originate from the 111 systems of Qualitas.*class* Corpus.

From a MySQL server instance, our dump file can be restored with the following command, where the *path_to_dump_file* variable should be replaced with the full path of the MySQL database dump file provided[6].

```
SOURCE path_to_dump_file;
```

Source is a mysql command that executes the MySQL database dump file in the database selected.

[5]http://docs.oracle.com/javase/tutorial/jaxb

[6]We tested it in Mysql 5.5.35-0ubuntu0.13.10.2 Version

Table 1: Measurement count per metric.

Id	Metric	Count
CA	Afferent coupling	16,566
CE	Efferent coupling	16,566
DIT	Depth in tree	247,396
LCOM	Lack of cohesion in methods	247,395
MLOC	Method lines of code	1,663,248
NBD	Nested block depth	1,663,526
NOC	Number of classes	16,566
NOF	Number of fields	247,481
NOI	Number of interfaces	16,566
NOM	Number of methods	247,481
NORM	Number of overriden methods	247,395
NSC	Number of children	247,396
NSF	Number of static fields	247,481
NSM	Number of static methods	247,481
PAR	Number of parameters	1,663,526
RMA	Abstractness	16,563
RMD	Normalized distance	16,564
RMI	Instability	16,566
SIX	Specialization index	247,395
VG	*McCabe* cyclomatic complexity	1,663,526
WMC	Weighted methods per class	247,482

4. STATISTICAL DATASET

We employed the R tool for statistical computing to generate tables and graphs that describe the metrics data, for each of the 21 evaluated metrics and each of the 111 systems. The tables and graphs are well documented in R Tutorial [5].

The generated tables are as follows:

1. **Cumulative Frequency Distribution Table:** the cumulative frequency distribution of a quantitative variable is a summary of data frequency below a given level. In our study, the cumulative frequency distribution shows the total number of software artifacts (methods, classes or packages) whose metric values are less than or equal to a set of chosen values. This table shows two columns. The first displays the metric values and the second the total number of software artifacts whose metric values are less than or equal to the corresponding chose value.
2. **Cumulative Relative Frequency Distribution Table:** the cumulative relative frequency distribution of a quantitative variable is a summary of frequency proportion below a given level. The relationship between cumulative frequency and relative cumulative frequency is: $Cum.\ Rel.\ Freq. = Cum.\ Freq./Sample\ Size$. In our study, this table has two columns. The first one displays the metric values and the second the *Cumulative Relative Frequency* below the corresponding metric value.
3. **Frequency Distribution Table**: the frequency distribution of a data variable is a summary of the data occurrence in a collection of non-overlapping categories. In our study, the frequency distribution is the summary of metric values according to a proposed classification, for instance, range between the values. This table has two columns. The first one exhibits the range of metric values and the second one the summary of metric values in the corresponding range.
4. **Relative Frequency Distribution Table**: the relative frequency distribution of a data variable is a summary of the

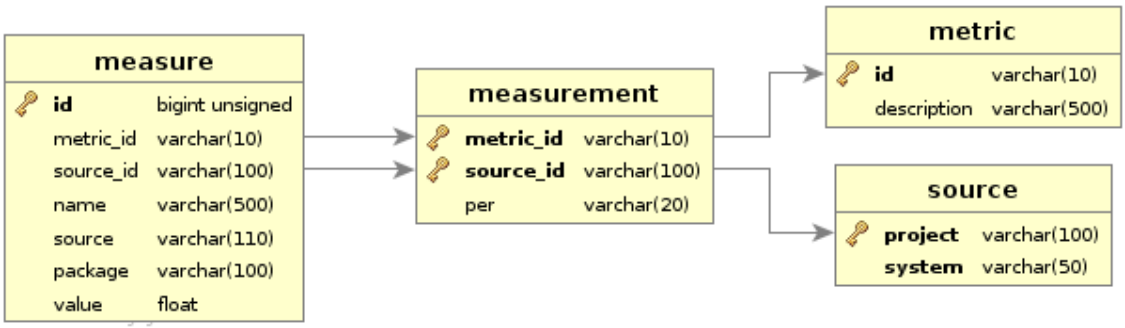

Figure 1: Entity-relationship diagram - Storage measures, measurements, metrics and projects available in *Qualitas.class Corpus*

frequency proportion in a collection of non-overlapping categories. The relationship between frequency and relative frequency is: $Relative\ Frequency = Frequency/Sample\ Size$. In our study, the relative frequency distribution is a summary of the proportion of metric values in each metric range value. This table has two columns. The first one shows the range of metric values and the second the proportion of the corresponding range in the dataset.

The generated graphs are as follows:

1. **Histogram:** it consists of parallel vertical bars that graphically show the frequency distribution of a quantitative variable. The area of each bar is equal to the frequency of the item classes. In our study, the histogram of the metric values is a collection of parallel vertical bars that show the number of occurrences classified according to their range of values.
2. **Scatter Plot:** a scatter plot pairs up the values of two quantitative variables from a dataset and displays them as geometric points in a cartesian diagram. In our study, we pair up the metric value and the number of occurrences of this value as (x,y) coordinates.
3. **Log-Scaled Histogram:** shows the same data of a histogram in doubly logarithmic scale.
4. **Cumulative Frequency Graph:** shows the cumulative frequency distribution table.
5. **Cumulative Relative Frequency Graph:** shows the cumulative relative frequency distribution table.
6. **Probability Histogram**: shows the relative frequency distribution table, as a histogram.

Therefore, for each metric, and for each system in Qualitas.*class* Corpus, we provide four tables and six graphs, generated with the R tool. Figures 2a to 2f show the six graphs generated for the number of parameters metric in the Eclipse system.

In addition, we provided the R script used to generate the described tables and graphs. Our aim is to show the commands used in the tool to generate the statistical data, which shall in turn allow other researchers to adapt it according to their needs. In total, there are 13,986 graphs, 9,324 tables and 2,331 R scripts.

We also provided the same statistical data described above to the 95th, 96th, 97th, 98th and 99th percentiles. This was achieved by removing the last 5%, 4%, 3%, 2% and 1% of the data. For example, the 97th percentile represents 1%, 2%, ..., 97% of the ordered metric values data in an ascending way. If such information is counted, there is a total of 69,930 graphs, 46,620 tables and 11,655 R scripts.

4.1 Data-Fitting

When the EasyFit tool carries out a Kolmogorov-Smirnov test, it selects the appropriate distribution for the data. This selection involves choosing a distribution that has the best fit to a dataset. The purpose of this is to set the appropriate distribution for each software metric from each system. Exploring the distributions of software metric values is crucial to increasing the understanding of the internal structures of source codes [1].

We provide a fit result file, named *fitResult.txt*, for each systems' metric, which shows the distribution ranking returned by the EasyFit tool. The first line of the file shows the sample size. The second line shows the statistical distribution that best fits the data. The next lines displays the parameters of the statistical distribution, which shows the best fit. The number of lines detailing the parameters varies, because the distributions have different parameters. They are shown in a $key = value$ scheme, where each line shows the parameter name (key) and its respective value (value). The next lines of the file show the fit ranking in the following format: the first line shows the distribution position in the fit ranking, the second line shows the distribution name and the last line shows *True* if the distribution could be fitted to the data or *False*, if not.

The EasyFit tool has two additional important uses. Firstly, it plots the *pdf* (probability density function) graph, which describes the probability of a variable assuming a value x: $f(x) = p(X = x)$ [6, 7]. This graph shows the dataset modelled with the appropriate distribution, allowing us to see important features of the curve. Secondly, it plots the *cdf* (cumulative density function) graph, which describes the probability of a variable assuming a value x: $f(x) = p(X \leq x)$ [6, 7], with an adjustment line of the data to the distribution. Figures 2g and 2h illustrate the *pdf* and *cdf* graphs for the number of parameters metric in Eclipse.

As part of the data-fitting process, we provide two more graphs, the *cdf* and *pdf* graphs, as well as the fit result files. In total, there are 2,331 fit result files and 4,662 *cdf* and *pdf* graphs. We did not generate this information in other percentiles, because the purpose of this process is to establish the distribution that best fits the data. Removing values of the dataset, in this case, has no purpose, even for data visualization. Therefore, when including the *cdf* and *pdf* graphs, we provide 18,648 graphs.

DOI:10.1145/2659118.2659130 http://doi.acm.org/10.1145/2659118.2659130

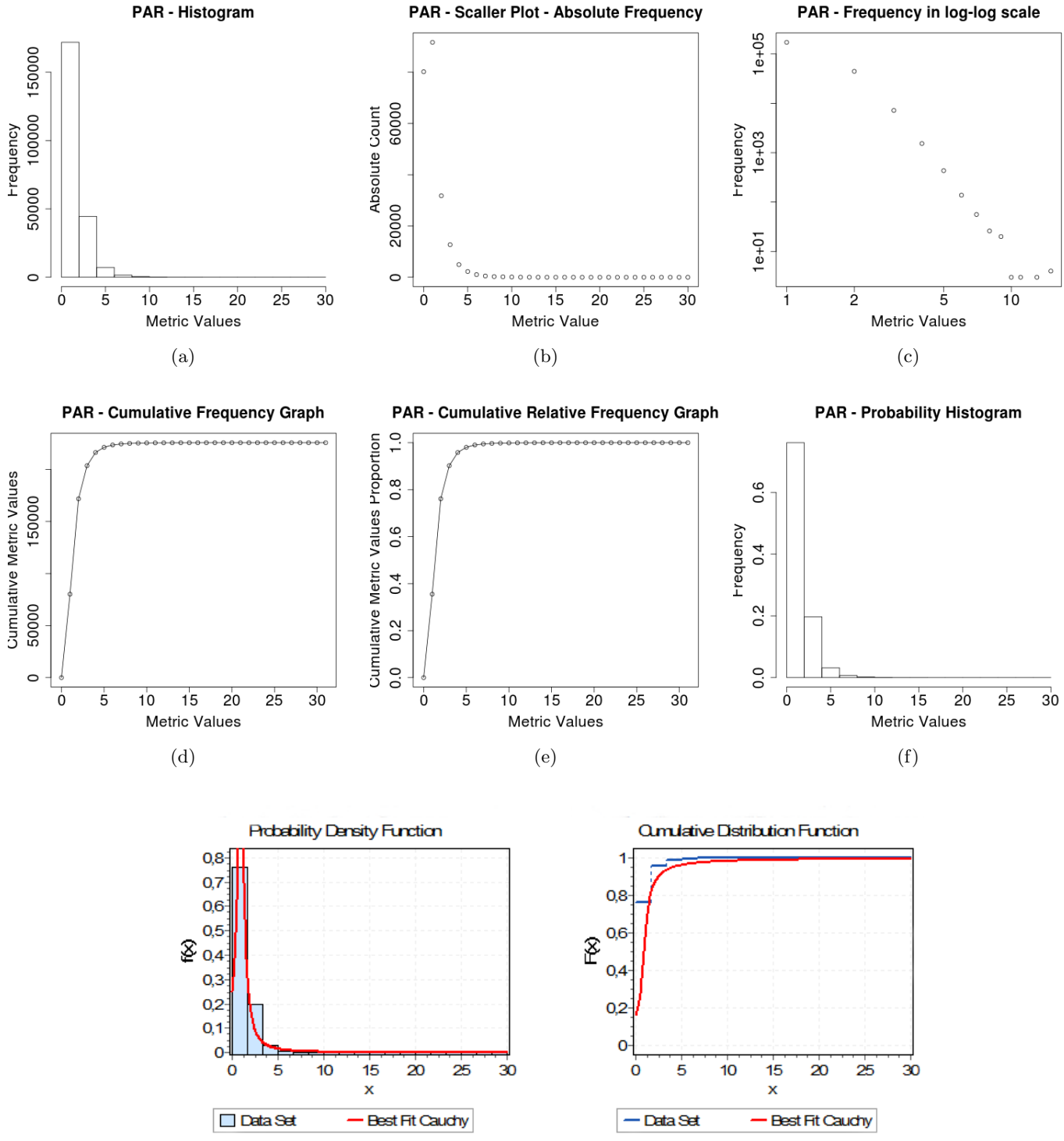

Figure 2: Eclipse: (a) Histogram (b) Scatter Plot (c) Histogram in logarithmic scale (d) Cumulative Frequency Graph (e) Cumulative Relative Frequency Graph (f) Probability Histogram (g) Probability Density Function (h) Cumulative Density Function.

DOI:10.1145/2659118.2659130 http://doi.acm.org/10.1145/2659118.2659130

5. DATASET STRUCTURE

We provide our dataset in a gzip compressed archive (.tar.gz), whose directory structure contains the statistical data. A simplified view of the directory structure of the dataset is shown in Figure 3. The folder named dataset is the hierarchy's parent folder. This folder contains 21 subfolders, which represent the 21 metrics shown in Table 1. These folders are named according to the Id column of this table. Figure 3 shows the CA folder, which relates to the afferent coupling metric. Within each of the metrics folders are 6 folders called 0.95, 0.96, 0.97, 0.98, 0.99 and 1.0, which contain the data generated for the 95th, 96th, 97th, 98th and 99th percentiles. The 1.0 folder shows the whole dataset and is shown in Figure 3 as a sample. Within each of these percentile folders, are 111 folders, which correspond to each of the systems used. Within each system folder, are two other folders, named images and tables, and the fitResult.txt file as mentioned in Section 4.1. The images folder contains the 6 graphs described in Section 4 and the *pdf* and *cdf* graphs presented in Section 4.1. The tables folder contains 4 text files which show the tables presented in Section 4. The files are clearly named according to the graphs or tables they contain. As pointed out in Section 4.1, the *fitResult.txt* file only exists in the 1.0 folder. The gzip compressed archive also contains the MySQL database dump file next to the dataset folder, named *dump.sql*.

The dataset is public available at:

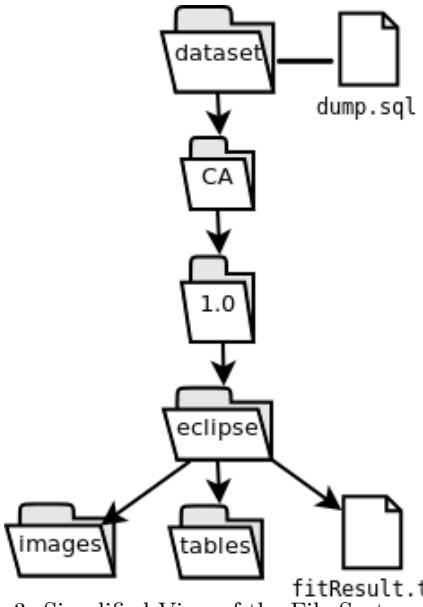

Figure 3: Simplified View of the File System Structure

`http://www.dcc.ufmg.br/~tfilo/statisticaldataset.tar.gz`

6. CONCLUSION

This paper describes how we provided a statistical dataset for software metrics in object-oriented systems. We believe such information is valuable when conducting empirical studies, in order to understand the internal structures of existing source codes. With our contributions, researchers can go straight to the data interpretation stage, for example, looking for valid cause-effect relationships related to software metrics, or explaining why some distributions exist for a metric in some applications and not others.

In practice, the process of generating all of the statistical data provided is by no means easy. By making our dataset available, we hope to eliminate this arduous task, which was previously unavoidable when conducting research related to software metrics. In addition, by using statistical data generated from different tools, related studies may lead to entirely different results. Our dataset also helps to increase the ability to reproduce the studies. For example, the information provided can be used to reproduce the work performed by Baxter et al. [1], who investigated claims that many important relationships between software artifacts follow a power-law distribution.

7. REFERENCES

[1] G. Baxter, M. Frean, J. Noble, M. Rickerby, H. Smith, M. Visser, H. Melton, and E. Tempero, "Understanding the shape of java software," in *OOPSLA '06*. New York, NY, USA: ACM, 2006, pp. 397–412.

[2] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, "Qualitas corpus: A curated collection of java code for empirical studies," in *APSEC2010*, Dec. 2010, pp. 336–345.

[3] R. Terra, L. F. Miranda, M. T. Valente, and R. S. Bigonha, "Qualitas.class Corpus: A compiled version of the Qualitas Corpus," *Software Engineering Notes*, vol. 38, no. 5, pp. 1–4, 2013.

[4] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference (Statistics: a Series of Textbooks and Monogrphs)*, 4th ed. CRC, May 2003.

[5] R. (2014) An r introduction to statistics. [Online]. Available: http://www.r-tutor.com/

[6] EasyFit. (2013) Easyfit. [Online]. Available: http://www.mathwave.com/products/easyfit.html

[7] Matlab. (2014) Matlab. [Online]. Available: http://www.mathworks.com/help/stats

APPENDIX

A. STRUCTURE OF THE XML FILE METRICS

A sample XML file for the Batik project is shown in Figure 4. The file was reduced in the figure in order to show the structure of the available files.

Line 1 contains the Metrics element. It is identified by the scope attribute, which is the project name. This element is composed of n Metric elements, which contains the id attribute that identifies the metric. For example, in line 2, the reported metric is *McCabe* cyclomatic complexity, identified by $id = VG$. In Line 7, the reported metric is efferent coupling, identified by $id = EC$. Line 12 states that the reported metric is depth of inheritance in tree, identified by $id = DIT$.

Each element of Metric type is composed by a Values type element. The Values type element contains an attribute called per, which identifies whether that metric is given in the method scope (method, line 3), class scope (type, line 13) or package scope (packageFragment, line 8).

The Values element is composed of n Value elements, that show the actual measures. The Value element is composed of name, source, package and value attributes. Therefore, the cases of measurements per method, class or package are included as follows:

- method: the name specifies the method name, source specifies the class file which contains the method, and package specifies the package where the class belongs. A sample is show in line 4;

```
1<Metrics scope='batik-1.7_samples'>
2 <Metric id='VG'>
3  <Values per='method'>
4   <Value name='run' source ='
        UntrustedScriptHandler.java'
        package ='com.untrusted.script'
        value ='6'/>
5  </Values>
6 </Metric>
7 <Metric id='CE'>
8  <Values per='packageFragment'>
9   <Value name='com.test.script' package
       ='com.test.script' value='1' />
10  </Values>
11 </Metric>
12 <Metric id='DIT'>
13  <Values per='type'>
14   <Value name='
        EventListenerInitializerImpl'
        source='
        EventListenerInitializerImpl.java
        ' package='com.test.script' value
       ='1' />
15  </Values>
16 </Metric>
17 <Metric id = 'NOP'>
18        <Value value='2'/>
19    </Metric>
20</Metrics>
```

Figure 4: Sample XML file for batik-1.7_samples project.

- class: the name specifies the class name, source specifies the file class which contains the method, and package specifies the package to which the class belongs. An example is show in line 14.
- package: the name and package show the package name. Source is not used. A sample is show in line 9.

There are measurements that refer to the whole system. This is the case of the number of packages and total lines of code. As shown in line 17, it contains only one Value element, not having the parent Values like the other metrics.