

# Qualitas.class Corpus: A Compiled Version of the Qualitas Corpus

Ricardo Terra, Luis Fernando Miranda, Marco Tulio Valente, Roberto S. Bigonha

Department of Computer Science, UFMG, Brazil

{terra,luisfmiranda,mtov,bigonha}@dcc.ufmg.br

## Abstract

This paper documents a compiled version of the Qualitas Corpus named *Qualitas.class Corpus*. We provide compiled Java projects for the 111 systems included in the corpus. We also gathered a large amount of metrics data (such as measurements from complexity, coupling, and CK metrics) about the systems. By making *Qualitas.class Corpus* public, our goal is to assist researchers by removing the compilation effort when conducting empirical studies.

## 1 Introduction

*Qualitas.class Corpus* is a compiled version of the Qualitas Corpus proposed by Tempero et al. [6]. In short, it provides compiled Eclipse Java projects for the 111 systems included in the last release of the corpus.<sup>1</sup> Table 1 provides an overview on our compiled corpus. It contains more than 18 million LOC, 200,000 compiled classes, and 1.5 million compiled methods. As another contribution, we have gathered a large amount of metrics data (such as measurements from size, coupling, and CK metrics) about the systems (see Section 3).

Table 1: Overall Total

Systems	111
Lines of Code (LOC)	18,548,026
Internal Projects (NOIP)	802
Packages (NOP)	16,509
Classes (NOCL)	202,052
Interfaces (NOI)	22,115
Methods (NOM)	1,464,893

The original Qualitas Corpus provides a huge contribution for experimentation in software engineering. However, there are several scenarios—e.g., experiments that rely on Abstract Syntax Trees (AST) or bytecode—in which researchers need to import and compile the source code. Since this task is not trivial in the case of systems with many external dependencies, this work addresses such effort by providing a compiled variant of the Qualitas Corpus.

## 2 Compilation Process

The corpus contains a collection of **systems**, each of which contains *one or more projects*. For instance, the AspectJ system is divided in four internal projects: **matcher**, **rt**, **tools**, and **weaver**. Thus, considering the 111 systems,

the *Qualitas.class Corpus* has a total of 802 internal projects (including 461 projects from NetBeans).

Since the Qualitas Corpus provides us with the source code of the systems, our main work consisted in organizing the code in well-defined Java projects according to the following guidelines:

- In the case of outdated systems, we compiled their most recent version, instead of thoses in the Qualitas Corpus.
- Source code distributions usually do not include third-party libraries, which are required by the compilation process. We hence searched for these libraries using Maven repositories<sup>2</sup> or version control systems.
- Some projects rely on very particular libraries. For instance, JTOpen 7.8 relies on the IBM AS/400 library, which is not publicly available. In these cases, we created *stub* JAR files to simulate the internal structure of the libraries (e.g., `ibm_as400_stub.jar`).
- Some projects presented compilation errors. In these cases, we fixed the error and explained our fixing procedure in a comment.<sup>3</sup> For instance, package `etc.testcases` on Ant 1.8.2 has some classes whose file name differs from the public class name. Therefore, we renamed the class names and inserted comments into the code to explain the changes.

## 3 Measurements using Metrics

*Qualitas.class Corpus* also includes the values of the following 23 source code metrics measured at the level of classes [1, 4, 3, 5]:

- *Basic Metrics*: Number of lines of code (LOC)<sup>4</sup>, Number of packages (NOP), Number of classes (NOCL), Number of interfaces (NOI), Number of methods (NOM), Number of attributes (NOA), Number of overridden methods (NORM), Number of parameters (PAR), Number of static methods (NSM), and Number of static attributes (NSA).
- *Complexity Metrics*: Method lines of code (MLOC), Specialization index (SIX), McCabe cyclomatic complexity (VG), Nested block depth (NBD), and Normalized distance (RMD).

<sup>2</sup>Maven repositories: [search.maven.org](http://search.maven.org) and [mvnrepository.com](http://mvnrepository.com)

<sup>3</sup>Comments like `//Qualitas.class:...` to be easily identified.

<sup>4</sup>Our metric counts non-blank and non-comment lines of codes.

<sup>1</sup>The current release is 20120401

- *CK Metrics*: Weighted methods per class (WMC), Depth of inheritance tree (DIT), Number of children (NOC), and Lack of cohesion in methods (LCOM).<sup>5</sup>
- *Coupling Metrics*: Afferent coupling (CA), Efferent coupling (CE), Instability (I), and Abstractness (A).

Table 2 presents a subset of the metrics gathered for the systems in the corpus. As can be noticed, the corpus is very heterogeneous. For example, systems' size ranges from 3.5 KLOC (fitjava) to 2,500 KLOC (Eclipse). There are lowly- (e.g., jasml, LCOM 0.08) and highly-cohesive systems (e.g., frees, LCOM 0.57). Analogously, there are lowly- (e.g., xmojo, CE 0.6) and highly-coupled systems (e.g., megamek, CE 38).

We relied on Google CodePro Analytix<sup>6</sup> and Metrics<sup>7</sup> to compute the metrics. For each project  $P$ , the *Qualitas.class Corpus* provides a XML file with a `Metric` element for each metric  $M$  (identified by the attribute `id`). For example, the element `<Metric id="NOM" avg="4.04" ... >` contains the average value of the metric *Number of Methods*.

In a summarized perspective, Figure 1 illustrates the distribution of the average value for a subset of metrics. Basically, each circle represents a system and the values in the vertical indicate the overall average for each metric. For example, the MLOC metric ranges from 3.35 (fitlibraryforfitnesse) to 23.4 (jparse), but the overall average is  $7.88 \pm 2.70$ .

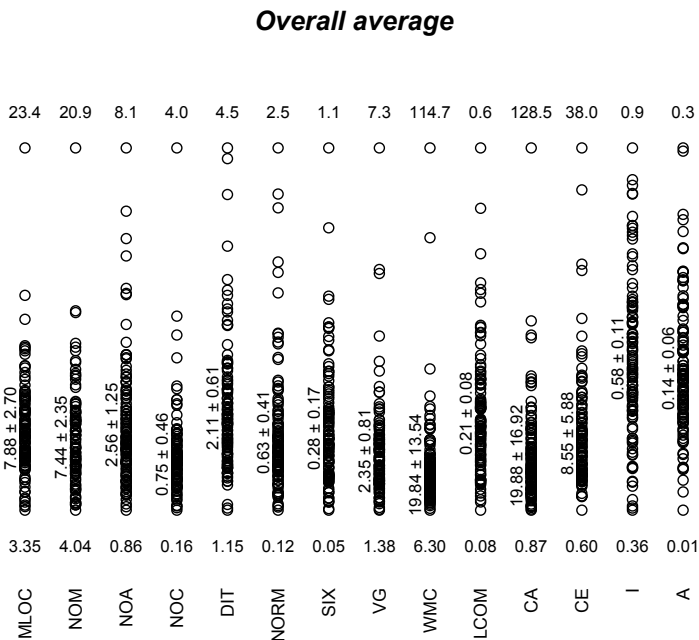


Figure 1: Distribution of the systems (average)

<sup>5</sup>Our metric relies on the LCOM HS (Henderson-Sellers) method [3].  
<sup>6</sup>CodePro Analytix 7.1.0, <http://developers.google.com/java-dev-tools>  
<sup>7</sup>Metrics 1.3.8, <http://metrics2.sourceforge.net>

## 4 Final Remarks

The original Qualitas Corpus provides the source code of a large curated collection of open source Java systems. However, there is a broad spectrum of scenarios—e.g., experiments that rely on Abstract Syntax Tree (AST) or bytecode—in which researchers need to import and compile the source code. In practice, this compilation process is not trivial and may impact other researchers' ability to replicate the study. This work therefore addresses such hard-working and time-consuming tasks by providing a compiled version of the original Qualitas Corpus. For instance, we used the *Qualitas.class Corpus* to evaluate a refactoring recommendation system we are working on [8, 7]. As future work, we intend to integrate the *Qualitas.class Corpus* with our previous data set with historical source code metrics values [2].

The *Qualitas.class Corpus* is publicly available at:

<http://java.labsoft.dcc.ufmg.br/qualitas.class>

*Acknowledgments*: This research was supported by grants from CNPq, CAPES, and FAPEMIG.

## References

- [1] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.
- [2] Cesar Couto, Cristiano Maffort, Rogel Garcia, and Marco Tulio Valente. Comets: A dataset for empirical research on software evolution using source code metrics and time series analysis. *Software Engineering Notes*, pages 1–3, 2013.
- [3] Brian Henderson-Sellers. *Object-oriented metrics: measures of complexity*. Prentice-Hall, 1996.
- [4] Michele Lanza, Radu Marinescu, and Stéphane Ducasse. *Object-Oriented Metrics in Practice*. Springer-Verlag, 2005.
- [5] Robert Martin. OO design quality metrics – an analysis of dependencies. In *Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA'94*, pages 1–8, 1994.
- [6] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. The Qualitas Corpus: A curated collection of Java code for empirical studies. In *17th Asia Pacific Software Engineering Conference (APSEC)*, pages 336–345, 2010.
- [7] Ricardo Terra, João Brunet, Luis Miranda, Marco Tulio Valente, Dalton Serey, Douglas Castilho, and Roberto Bigonha. Measuring the structural similarity between source code entities. In *25th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 753–758, 2013.
- [8] Ricardo Terra, Marco Tulio Valente, Krzysztof Czarnecki, and Roberto Bigonha. Recommending refactorings to reverse software architecture erosion. In *16th European Conference on Software Maintenance and Reengineering (CSMR), Early Research Achievements Track*, pages 335–340, 2012.



