An Embedded Multiple-Case Study on OSS Design Quality Assessment across Domains

Ampatzoglou, Apostolos; Gkortzis, Antonios; Charalampidou, Sofia; Avgeriou, Paraskevas

Published in:
EPRINTS-BOOK-TITLE

2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement

An Embedded Multiple-Case Study on OSS Design Quality Assessment across Domains

Apostolos Ampatzoglou, Antonios Gkortzis, Sofia Charalampidou, Paris Avgeriou
Department of Mathematics and Computer Science
University of Groningen
Groningen, The Netherlands
e-mails: a.ampatzoglou@rug.nl; a.gkortzis@rug.nl; s.charalampidou@rug.nl; paris@cs.rug.nl

Abstract - **Context: Investing on Open Source Software (OSS) as a "code reuser", involves certain risks, such as the difficulty in understanding the level of OSS design quality Aim: We investigate the levels of design quality of OSS projects, across different application domains. Method: We conducted a case study, which is the most fitting research method for observing a phenomenon in its real context, which is active for a long period of time, and for which variables cannot be controlled. Results: We present the values for seven design quality metrics of 546 OSS projects, as well as the statistically significant differences across application domains. Conclusions: The results of the study suggest that OSS application domains correlate with several design quality characteristics, in the sense that projects within one application domain appear to have similar levels of design quality. In addition to that, the results reveal application domains with high and low levels of design quality.**

Keywords — design quality; open source; application domain

I. INTRODUCTION

Open source software (OSS) projects can be utilized in many ways, one of the most important being the white-box reuse of open source code. Mockus suggests that 53% of OSS projects have performed reuse activities in 30% of their development process and that 49% of projects have reused more than 80% of their code. Additionally, it is suggested that most reused units have gone through major or minor modifications in order to be adopted in the target project [1]. However, investing on OSS, as a "code reuser", involves certain risks, partly because understanding OSS projects' quality and predicting the way this quality evolves is extremely difficult [2]. The main reasons for this is that developers are not under strict supervision, there are many feature requests and bug reports that come from different people, who might use the software under different perspectives and there are many changes in the community.

In this paper we attempt to shed some light on the level of OSS projects' *design quality*, across different application domains, using an established metric suite for assessing the quality of object-oriented designs [3]. With the term design quality we refer to quality characteristics of classes, objects and relationships that are measurable at implementation level, such as size, coupling, complexity and cohesion. To achieve our goal, we conducted an embedded multiple-case study that included 546 OSS projects from eight different application domains. To the best of our knowledge, no other work compares the quality of OSS projects across application domains. The rest of the paper is organized according to a variation of the linear analytic reporting structure presented by Runeson et al. [4].

II. CASE STUDY DESIGN

In order to assess OSS projects' design quality we performed a case study for two reasons. First, Runeson et al. suggests that case studies, in the software engineering domain, are observational and used for monitoring projects and activities in a real-life context [4]. Second, case studies are used in environments that cannot be controlled, where the attribute under study is related to many factors [4]. This study fits the above reasons, because OSS projects are examined in a real-life context, they cannot be monitored in isolation and their environment cannot be controlled. The case study of this paper has been designed according to the guidelines from Runeson et al. [4].

A. Objectives and Research Questions

The goal of this study, described using the Goal-Question-Metrics (GQM) approach [5] is to *analyze open source projects for the purpose of evaluation with respect to the level of design quality on multiple open source application domains from the point of view of software developers in the context of OSS white-box reuse.*

According to the abovementioned goal we extracted and stated two research questions that will guide the case study design and the reporting of the results. The metrics used to answer the questions are discussed in detail in section II.D.

RQ_1: What are the levels of design quality across open source software domains?

RQ_2: Are there statistically significant differences among the levels of design quality of the studied open source software domains?

B. Case and Unit Analysis

Case studies are distinguished between single-case or multiple-case studies and holistic or embedded case studies [4]. In this paper we performed an embedded multiple-case study, where the involved contexts are the OSS domains, the cases are the OSS projects and the units of analysis are their classes.

C. Case Selection

As mentioned in II.B the cases of our study are open source projects. In order to gather as many cases as possible, we have chosen to use a web repository that documents

978-0-7695-5056-5/13 $26.00 © 2013 IEEE
DOI 10.1109/ESEM.2013.48

design quality measurements, namely *percerons*[1] which was created by one of the authors [6 and 7]. The repository was initially created in 2009, as a catalogue of design pattern instances, which were accompanied by quality assessment data for the design pattern participating classes. Later, the repository was extended by adding a search engine for open source software components and by exploring the design quality of additional OSS projects. At this moment, the repository is available through a list of web services that help open source software developers and software development companies in general to adopt open source software in their product development. At this point there are two main service categories that are available inside Percerons: *Percerons Source Code Search Engine* and *Percerons Open Source Software Quality Assurance.*

From that repository, we have been able to extract data about eight different open source software application domains[2]. In total, the repository shares data on 546 open source projects, of different types of release (alpha, beta, stable) spread across the eight software application domains[2]. The selected cases of our study conform to the four propositions on case selection by Verner et al. [8]: measurements on the design quality of an OSS project can be taken at any time; they are precise, because of the clarity in the definition of used metrics and they are clearly relevant to the case study research questions.

D. Data Collection

The dataset that was created after selecting the cases consisted mainly of numerical data. Each project was characterized by 10 variables: [A01] name, [A02] version, [A03] application domain[2] and [A04] - [A10] metric scores on design quality characteristics. We adopted the metrics on design quality from the Chidamber & Kemerer suite [4], which is one of the best known metric suites for measuring object-oriented design quality. The suite involves one coupling (CBO: Coupling between objects), one cohesion (LCOM: Lack of Cohesion of Methods), two inheritance (DIT: Depth of Inheritance Tree, NOCC: Number of Children Classes) and two complexity metrics (WMC: Weighted Method per Class, RFC: Response For a Class), that are described next in this order.

CBO measures the number of classes that the class is connected to, in terms of method calls, field accesses, inheritance, arguments, return types and exceptions. High coupling is related to low maintainability and understandability [9]. *LCOM* measures the dissimilarity of pairs of methods, in terms of the attributes being accessed. High Lack of Cohesion is an indicator of violating the single responsibility principle [10], which suggests that each class should provide the system with only one functionality. *DIT* and *NOCC* quantify the maximum depth and the average width of each hierarchy in the system. High values of these metrics imply structures that are difficult to understand and maintain [9]. On the other hand, low or zero metric scores, imply that the designers have not benefited from one of the main object-oriented advantages. *WMC* is calculated as the average *Cyclomatic Complexity (CC)* among methods of a class. High WMC results in difficulties in maintaining and understanding the system [9]. *RFC* calculates how many methods an object can call (including the methods of the class itself), as an indication of the amount of functionality that this method can provide. High RFC values, indicate classes that provide much functionality to the system, but are considered complex [11].

Finally, we examine an additional design metric related to size, i.e. *Number of Classes (NOC)*, in the sense that it provides an estimation of the amount of functionality offered by the system [11]. The size of the system needs to be taken into account, since smaller systems are expected to be less coupled, less complex, to have less classes as leafs in hierarchies and use less inheritance trees. For example, the range of values of the CBO metric is [0, NOC-1] and the range of values for the RFC is [0, class methods + coupled class methods], where the range of value of coupled class methods is again [0, NOC-1]. Thus, assessing quality characteristics (apart from cohesion), without taking into account the size of the system would be unfair for application domains with larger projects.

E. Data Analysis

The analysis phase of our study has employed descriptive statistics and non-parametric hypothesis testing, because none of the variables [A04] – [A10] follow a normal distribution or are homogenous and because non-parametric tests are less affected by outliers. In order to explore RQ_1, we use descriptive statistics on variables [A03] (for grouping) and [A04]-[A10]. In order to make the charts more readable, variables [A04]-[A10] have been normalized. More specifically, variables have been transformed in order for their range value to be [0, 1], by subtracting the minimum value and dividing with the maximum value. Finally, in order to investigate the existence of possible statistical differences in design quality across different domains, we performed Kruskal-Wallis tests, with the domain [A03] as grouping variable, and the metric scores [A04] – [A10] as testing variables.

III. RESULTS

In this section we present and discuss the results of the case study. In Figure 1 we present error bars for every design quality metric that has been considered. This provides indications on the mean value of each metric across different application domains and at the same time depicts information on the 95% confidence interval (CI) of the corresponding variables.

Next, in order to provide a more holistic presentation, we combine the seven metrics into one radar chart per application domain. In the radar chart of Figure 1 the axes represent the normalized value of the metric scores, whereas the two series represent the OSS application domain under study and the average values from all studied OSS projects.

[1] http://www.percerons.com

[2] The software domain names are derived from sourceforge.net, where the open source projects are being developed and maintained. The domains are (in the parenthesis we remark the number of OSS projects considered from each application domain): Audio and Video (50), Games (135), Business & Enterprise (50), Home & Education (32), Science & Engineering (40), Communications (59), Development Tools (125) and Graphics (55).

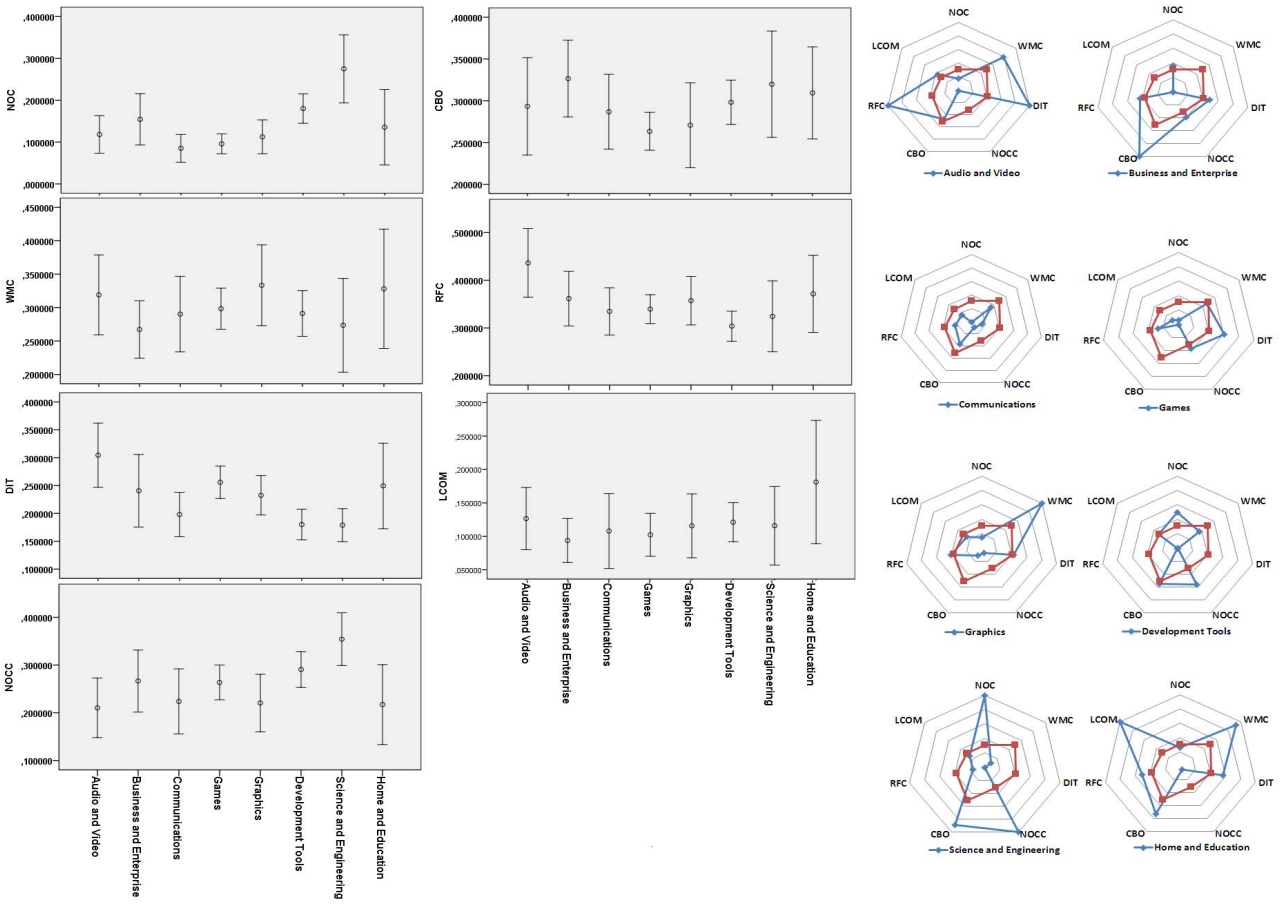

Figure 1 Design Quality Metrics Across different OSS Domains

The most important results concerning RQ_1 are:

- The application domain with the best levels of design quality appears to be *Communications,* whose average metrics scores are better (i.e. lower) than the average OSS project. However, *Communication* projects are among the smallest in terms of size.
- The application domain with the worst levels of design quality appears to be *Home and Education,* whose projects, although not among the largest, appear to be the most complex, the least coherent and among the most coupled systems. However, it consistently had the largest variance in metric values, which is an indicator that this category is too broad and should be decomposed for further analysis. In fact in Sourceforge the category consists of four very diverse sub-categories.
- The largest systems are *Science and Engineering*, *Development Tools* and *Business and Enterprise* projects. Among them, the best levels of design quality are provided by *Development Tools*, which can be due to the fact that they are created by developers with strong software engineering background.
- *Games*, *Graphics*, *Audio and Video* applications that are all in the field of entertainment, and are expected to share some common modules, present similar metrics (size, complexity and inheritance).
- OSS *Games* seems to exhibit good design quality metrics, with respect to the average OSS project, which is rather surprising, in the sense that literature suggests that games are usually poorly designed [12].

The above observations are only valid for our sample and not the population of OSS projects, since the statistical significance has not been examined. In order to identify if the above mentioned differences are statistically significant we conducted seven independent Kruskal-Wallis tests, and present their results in Table I.

TABLE I. STATISTICAL SIGNIFICANCE (KRUSKAL-WALLIS TESTS)

Null Hypothesis (H_0)	**Sig**	**Decision**
Distribution of NOC is the same across different domains	0.000	Reject H_0
Distribution of WMC is the same across different domains	0.696	Retain H_0
Distribution of DIT is the same across different domains	0.000	Reject H_0
Distribution of NOCC is the same across different domains	0.005	Reject H_0
Distribution of CBO is the same across different domains	0.089	Retain H_0
Distribution of RFC is the same across different domains	0.012	Reject H_0
Distribution of LCOM is the same across different domains	0.692	Retain H_0

The main findings from RQ_2 are summarized below:

- OSS projects classified under the same application domain are similar with respect to some design quality attributes, e.g. *size*, *inheritance* and *response for a class*.

- *Complexity*, *cohesion* and *coupling* appear to be uniform across different OSS application domains.
- The main design quality metrics that are related to *maintainability* and *understandability* [9 and 10], i.e. CBO, WMC, LCOM appear not to differ across domains.
- The main design quality metrics that are related to the provided *functionality* [11], i.e. NOC, RFC appear to differ across domains. This result makes sense as projects within the same application domain should more or less handle similar requirements. For example, in terms of functionality a CRM is expected to be more similar to an ERP, than a First Person Shooter game.

IV. Discussion

The results of this study can provide useful information both to researchers and practitioners. Concerning researchers, more and more empirical studies use OSS projects as objects. The results of the study can guide researchers in searching for appropriate projects as objects in their studies. For example, a research on refactoring opportunities can use objects from the *Home and Education* application domain, as they are more likely to be poorly designed. Furthermore, the results point out potential weaknesses in the open source development process, that are uniform in almost all projects, i.e. high complexity, low cohesion and high coupling. Methods that enhance these quality characteristics, such as design patterns and refactorings should be further studied by researchers. Concerning practitioners, the results indicate application domains, where it is more possible to extract more maintainable and reusable components. Moreover, the results indicate application domains, whose projects are more understandable, and can be used from developers as a way for understanding past design solutions. For example, a software engineer can use the know-how inside a well-designed development tool, so as to understand how java byte code can be accessed in a metric tool.

V. Validity Evaluation

This section discusses construct validity, reliability, and external validity for this study. Internal validity is not applicable as the study does not examine causal relations.

The main threat concerning construct validity has to do with the fitness of selected metrics for assessing design quality characteristics of OSS projects. The selected metric suite has been rigorously validated with professional software engineers and has been cited in over thirty five hundred scientific articles; thus the suite is credible as an indicator of the quality of object-oriented designs [5]. An additional threat in this category, concerns OSS projects' classification, as it is a central part of the RQs. To mitigate this threat we did not derive the application domains on our own but we reused the classification schema from Sourceforge, the most well-known OSS repository. A final threat to construct validity is the version type of release, i.e. alpha, beta, stable. The different release strategies that might be used among OSS projects, might provide different levels of design quality. However, the large number of software examined in this work provide a representative sample of the population, with respect to the percentage of each release type per domain.

In order to mitigate reliability two different researchers were involved in the data collection and one double-checked the results of the other. Furthermore, one researcher double-checked the results of the data analysis performed by another researcher. Finally, all primitive data are publicly available in the percerons repository and the study design is documented in the paper. Concerning external validity we identify two threats. Firstly, we investigated OSS projects hosted in only one repository. Projects of different repositories, such as github, might have different levels of quality. Additionally, all the OSS that have been investigated have been written in Java and there is a possibility that results are different for other OO languages. An additional threat is that the projects have been selected for the percerons repository according to their success in the community. In that sense, unsuccessful projects have not been examined. However, we believe that unsuccessful projects are highly unlikely to be reused, so they can safely be excluded from the study.

VI. References

[1] A. Mockus, "Large-scale code reuse in open-source software", *1st International Workshop on Emerging Trends in FLOSS Research and Development,* IEEE, pp. 7-12, 20-26 May 2007, Minesota, USA.

[2] A. Mavridis and I. Stamelos, "Real options as tool enhancing rationale of OSS components selection", *3rd International Conference on Digital Ecosystems and Technologies*, IEEE, pp.613-618, Instabul, Turkey, 1-3 June 2009.

[3] S. R. Chidamber, and C.F. Kemerer, "A metrics suite for object oriented design" *Transactions on Software Engineering,* IEEE, 20 (6), pp.476-493, June 1994.

[4] P. Runeson, M. Host, A. Rainer and B. Regnell, "Case Study Research in Software Engineering: Guidelines and Examples", *John Wiley & Sons*, 2012 .

[5] V. R. Basili, G. Caldiera and H. D. Rombach, "Goal Question Metric Paradligm" inside book: Ensyclopedia of Software Engineering, *John Wiley & Sons*, pp. 528-532, 1994.

[6] A. Ampatzoglou, A. Gortzis, I. Deligiannis and I. Stamelos, "A methodology on extracting reusable software components from Open Source Games", *16th International Academic MindTREK Conference*, ACM, pp. 93-100, 3-5 Octomber 2012, Tampere, Finland.

[7] A. Ampatzoglou, O. Michou and I. Stamelos, "Building and Mining a Repository of Design Pattern Instances: Practical and Research Benefits", *Entertanment Computing*, Elsevier, 4(2), pp. 131-142, April 2013.

[8] J. M. Verner, J. Sampson, V. Tosic, N. A. A. Bakar, B. A. Kitchenham, "Guidelines for industrially-based multiple case studies in software engineering," *3rd International Conference on Research Challenges in Information Science*, IEEE, pp. 313-324, 22-24 April 2009, Fes, Morocco.

[9] A. Van Koten and A.R. Gray, "An application of bayesian network for predicting object-oriented software maintainability", *Information and Software Technology,* Elsevier, 48 (1) pp. 59–67, January 2006.

[10] R.C. Martin, "Agile Software Development: Principles, Patterns and Practices", *Prentice Hall*, Upper Saddle River, USA, 2003.

[11] M. Morisio, I. Stamelos, V. Spahos, D. Romano, "Measuring functionality and productivity in Web-based applications: A case study", *6th International Symposium on Software Metrics*, IEEE, pp.111-118, 4-6 November 1999, Florida, USA.

[12] A. Ampatzoglou and I. Stamelos, "Software engineering research for computer games: A systematic review", *Information and Software Technology*, Elsevier 52 (9), pp. 888–901, September 2010.