ASSESSING THE MODIFIABILITY OF TWO OBJECT-ORIENTED DESIGN ALTERNATIVES – A CONTROLLED EXPERIMENT REPLICATION

Ignatios Deligiannis[1], Panagiotis Sfetsos[1], Ioannis Stamelos[2], Lefteris Angelis[2]
Alexandros Xatzigeorgiou[3], Panagiotis Katsaros[2]

[1]Dept. of Informatics, Technological Education Institute of Thessaloniki, Greece
{igndel, sfetsos@it.teithe.gr} http://sweng.csd.auth.gr/htmls/people_files/deligiannis.html
[2]Department of Informatics, Aristotle University of Thessaloniki, Greece
{stamelos, lef@csd.auth.gr}
[3]Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
{achat@uom.gr}

ABSTRACT

This paper presents a replication study of a controlled experiment, investigating the impact of many design characteristics on one of the most desirable quality factors, *modifiability*. Two alternative design structures were used; a responsibility-driven (RD) versus a control-oriented "mainframe" (MF) design. Two groups of undergraduate students participated, each performing on one of the two designs. The subjects designed, implemented in Java, and tested a set of three maintenance tasks in order to assess the degree of their understanding, effort, and performance. The results indicate that the RD version due to its delocalised structure, exhibited higher correctness, better extensibility, and design stability, than the MF version. In order to provide an objective assessment of the differences between the two versions, a considerable number of metrics were used on the delivered solutions, quantifying separately each produced design's characteristics.

KEYWORDS

Object-Oriented, experiment, maintainability, design, metrics.

1 INTRODUCTION

The necessity for change is indissolubly related with the development process, and it results from the fact that with the progress of time, additional knowledge is accumulated both from software developers, as well as from customers using the software. The Object-Oriented (OO) paradigm, by providing a number of mechanisms, such as inheritance and encapsulation, has been claimed to enhance understandability, due to increased cohesion of class structures, and manage more efficiently complexity. Hence, it is considered having particular affinity to *modifiability*, which is one of the most important quality factors by which the OO paradigm aims at reducing the maintenance costs of a system. This was also supported by a number of empirical studies [1].

Hence, a considerable number of researchers and practitioners in Software Engineering field, have turned their concerns in even more effectively applying the OO mechanisms and techniques, aiming at producing even more understandable and maintainable software, i.e. more qualitative systems. Particularly the designers, often facing the dilemma of choosing among different available design structure solutions, have focused on design quality and the various

design characteristics from which it stems. They aim mainly at some most desirable quality factors, such as understandability, performance, correctness and reusability.

The role of empirical research is to provide empirical evidence with proper grounding, whether and to what extent the above-mentioned aims and quality factors have been met. Experimental replication, that is an empirical study conducted under identical conditions as a basic experiment, is desirable in empirical investigation for several reasons: a) it can help us to know how much confidence we can place in the results of the experiment; b) it enables us to estimate the mean effect of any experimental factor [2]; c) the results of an experiment can be more easily generalised than those of a case study.

2 DESIGN OF THE EXPERIMENT

The research reported in this paper is a replication study of a controlled experiment carried out by Arisholm et. al. [3]. The main goal of the study is to investigate to what extent design characteristics affect the *modifiability* of OO software. Thus, to investigate how design characteristics affect modifiability, we need mainly to focus on structural characteristics, since they describe more precisely the structure of a system. In order to achieve an assessment of these characteristics, we need a set of proper metrics, since only them can provide an objective assessment.

In this study, two alternative designs of a system were used, implementing the same functionality. They were written in exactly the same manner, namely, programming style, naming conventions, and documentation. Similar skilled subjects performed a given set of change tasks. The experiment investigates the same hypotheses as the initial study, using similar settings. However, it deviates from the original experiment in three points: a) *Java* was used, instead of *Mocca* programming language; b) the subjects carried out the modification tasks realistically; implementing in code, compiling and running, separately for each requested task; c) a wide range of metrics were applied upon the produced code. This was considered particularly useful, since we believe by applying a considerable number of metrics, available in the literature, and capturing major design characteristics, a more accurate and detailed assessment of the delivered solutions could be obtained. This provides an in depth understanding of what design characteristics mostly affected the subjects' behavior and therefore modifiability.

The ***hypotheses*** tested were the following:

(H1) *Change effort*: The RD design requires more change effort than the MF design.
(H2) *Learning curve*: The RD design has a stronger learning effect (one learns more easily from previous tasks) than the MF design.
(H3) *Correctness*: The solutions for the RD design contain more errors than the MF design.
(H4) *Change complexity*: The RD design has higher change complexity than the MF design.
(H5) *Structural stability*: The RD design has better structural stability than the MF design.

The statistical test will attempt to reject the null hypothesis, which is the opposite of H1 to H5.

Forty-three students were used as participants to perform modification tasks on the two system designs, MF and RD. All were undergraduate at the second semester of their studies, at the Dept. of Information Technology of the Technological Education Institute of Thessaloniki, Greece, enrolled in the class of OO Programming. Each lecture was supplemented by a practical session using the Java programming language. They were randomly assigned to two groups, 22 performed in MF and 21 in RD design.

*Experimental material***:** The application used was a system implementing a 'Coffee Machine' system [4]. Each system documentation included: a) a description of its functionality, followed by a figure showing the logical parts of the machine; b) the Java code, consisting of 11 and 16 classes, and 286 and 391 lines of code, for the MF and RD version respectively. The two

designs were classified according to Coad and Yourdon's quality design principles [5, 6]. Thus, the MF design, not adhering to them, is considered to be the "bad" design, while the RD design, which adhered to the principles, is considered to be the "good" design.

Experimental tasks: The participants were asked to perform three modification tasks. Each task was coded, compiled, and tested realistically, as it happens in practice. This was actually one of the differences with the original study. The main purpose on the test was to motivate the subjects to produce solutions of good quality, before proceeding to the next task.

Procedures: No time-constraint was specified to complete all the tasks. However, they were told to perform as quickly and correctly as possible, hence they were requested to mark their performance time. Additionally, they were given a debriefing questionnaire to complete, expressing their subjective opinion concerning a number of issues.

Experimental Design: We applied statistical analysis on the collected data to interpret the results in order to draw meaningful conclusions from the experiment. The choice of the design affects the data analysis and vice versa. In this study, the type of design applied is a *randomized within-subjects design* [7]. The independent variable is the *design principles* under investigation. The experiment context is characterized as *multi-test within object study* ([7] p. 43). An advantage of using such a design is that it simplifies the statistical analysis. The participants were assigned into two groups as mentioned above.

Dependent variables: For hypotheses testing the following variables were examined:

- *Change Effort* – Time in minutes for each completed task.
- *Correctness* – The completed task was examined whether it worked correctly or not.
- *Learning Curve* – The normalised difference in effort to understand the last change (*c3*) versus the first change (*c1*) for each subject, for design MF and RD respectively.

$$\text{Learning Curve}(d) = \frac{\text{Understand}(d, c1) \ - \text{Understand}(d, c3)}{\text{Understand}(d, c1) + \text{Understand}(d, c3)}, d \in \{\text{RD, MF}\}$$

- *Subjective Change Complexity* – Measured from the questionnaire by two of participants' subjective answers, regarding *Solution difficulty,* and *Solution confidence.*
- *Structural Stability* – The impact changes have on the design, measured by the differences in the average values of the measures before and after each change.

3 EXPERIMENTAL RESULTS

To interpret the results, we consider examining the following factors: a) participants performance time; b) each task required modifications based on measurements extracted from participants' solutions. Hence, eight structural metrics, capturing structural design characteristics, were applied, provided by the Together© CASE tool.

Considering Task 1, all data shows that the two designs required similar treatment, hence both groups performed almost similarly. Both groups had to add similar trivial code in two classes, a method call, and a new method. Task2, required adding new functionality mainly based on inheritance. However, it differed between the two groups. MF design, with its centralised structure, was more demanding, mainly due to its procedural approach. RD group, by its OO structure, providing more reuse efficiency, led to better performance. Task 3, which required performing a 'check' before proceeding to 'produce' a requested drink, was more difficult for RD group to accomplish. MF group was forced to apply a procedural approach in a centralised structure, as in task2, where the most of the functionality was captured in one class by using many *if*-statements. On the other side, RD group with its decentralised structure faced a 'delocalised plan' [9] approach (OO technology's "weak point"), since the required 'check' captured a long message path between five classes.

To make a scientific statement with a reasonable degree of confidence, the significance level for the hypotheses test were set to α=0,1.

Change Effort (H1): *Hypothesis H1 is not supported.* Total effort shows no significant difference between the two groups (*t*-test, *p*=0,69) (Fig.1). However, considering each task separately, task2 showed a significant difference against MF group, while task 3 against RD group.

Learning Curve (H2)**:** *Hypothesis H2 is not supported.* Data shows no significant difference between the two groups (*t*-test, *p*=0,863) (Fig.2). In total, MF group required more time to understand than RD group. Particularly, task2 is considerably more difficult to understand for MF group, mainly due to procedural structure (many 'ifs') in one method. In contrast, RD group understood more easily due to reusability efficiency. Considering task 3, MF group understood more easily because they had to work with the same method as in task2 (learning effect), while RD group's functionality was split in 5 classes.

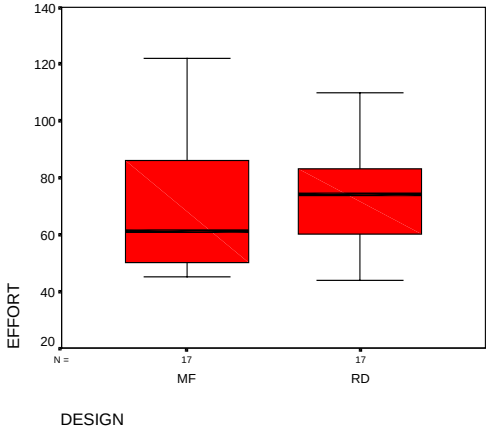

Figure 1. Performance effort

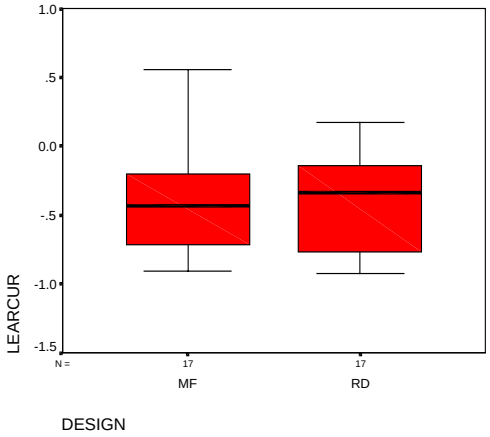

Figure 2. Learning curve

Correctness (H3)**:** *Hypothesis H3 is not supported.* RD group performed significantly more correctly than MF group (*t*-test, *p*=0,09) (Fig.3).

Subjective Change Complexity (H4)**:** *Hypothesis H1 is supported.* Considering *confidence,* Task1 showed no difference between both groups. In Task2 and 3, MF group participants indicated greater confidence than RD group. Considering *difficulty,* Task1 and 2 showed no considerable difference, while Task 3 showed significant difference against RD group (*t*-test, *p*=0,669) (Fig.4).

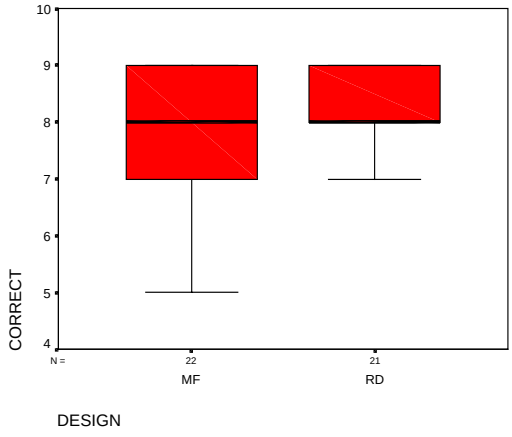

Figure 3. Correctness

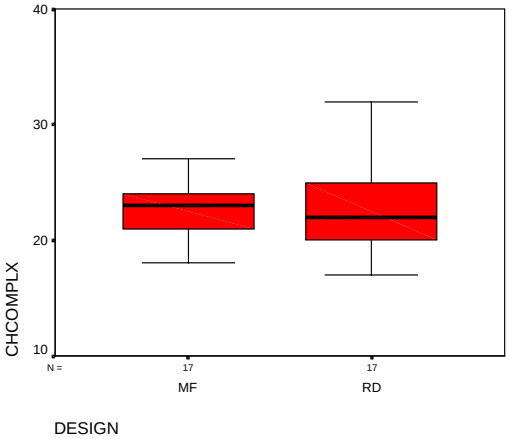

Figure 4. Subjective Change Complexity

***Structural Stability (H5)*:** *Hypothesis H1 is supported.* Data extracted from 8 metrics values indicated a significant difference between the two groups (Wicoxon, p=0,069, t-test, p=0,114) (Fig.5). The selected metrics were the following: Coupling Between Objects (CBO), Response for a Class (RFC), Weighted Method Per Class (WMPC1 &2) [10], Lines Of Code (LOC), Method Invocation Coupling (MIC) [8], No Of Attributes (NOA) [11], No Of Operations (NOO) [12]

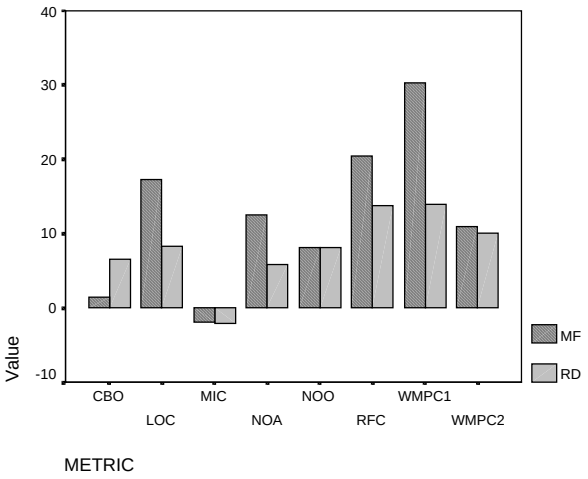

Figure 5. Structural Stability

.

3.1 Comparing the results with the original study

To compare the results between the two studies, we first have to consider the conditions under which they were conducted, since their impact determines subjects performance and therefore to results. Three were these differences, discussed later in section 4: a) programming language used, b) implementation manner, and c) metrics set applied to solutions. However, the two studies differ in hypothesis 1.

4 THREATS TO VALIDITY

This section discusses threats to validity and the attempt to alleviate them. The *external validity* of this study depends basically on the population selection, the choice of the design alternatives, and the choice of modification tasks. Considering the *internal validity,* this may be threatened for example by, unclear experimental materials, ambiguous questions, and skewed group assignments. An ultimate means to improve the validity of the original study, according to its authors, is by replication. Hence, we focused on those threats that were related to the differences between the original and our study. These are the following:

Programming language, Java vs. Mocca: In this study *Java* was used, while in the original study *Mocca* programming language was used. We consider Java currently is the most widely used programming language [13]. Also the participant students in this study had good knowledge of it. However, this could lead to differences between the two studies.

Coding realistically vs. Pen and Paper: The changes were coded using an advanced editor, while in the original study pen and paper were used. As the original study authors mentioned, using a computer one provides a number of advantages. We consider the main advantage is that subjects perform closely to realistic situations that happen in industry. However, there is a risk, due to students' not professional experience. Namely, one could spend a long time in a trivial task, when coding, compiling, and running a program, mainly caused by a "misunderstandable" compiler message. This practice could also lead to differences between the two studies. To alleviate this threat, an instructor was available

Selection. This is the effect of natural variation in human performance. Volunteer students were used, considered more motivated and suited for the task than the whole population. Hence the selected group is not representative for the whole population. Our concern was to select the most capable of the students from the course, offering them a degree bonus for their participation. An additional reason for this kind of selection was that we considered excluding those not really willing to participate, because they might not perform properly.

5 CONCLUSIONS

Examining the results, the following conclusions could be drawn: In general, design structure seems to play a major role in modifiability; Since, it is a creative process, guided by design principles could lead to quality solutions; a) *Extensibility*, in this study captured by task2, is more efficiently and effectively applied on a decentralised structure, better providing understanding and reusability conditions. b) 'Delocalised plan' is a potential drawback within OO paradigm. However, we believe that it could be to some extent alleviated by applying proper design techniques. As such, we consider some design patterns found in literature [14] [15], which could guide us to more effective members placement, hence yielding a more cohesive and less coupled design; c) RD design supports correct solutions.

The results give us the motivation for further research on a task solution basis. This could provide the necessary information to gain a better understanding of each task structure and implementation difficulty. Hence, we will consider first separating each task implementation, where a proper set of metrics would be applied providing metric values before and after task completion. These metrics values compared to both objective performance values (time spent), as well as subjective values (questionnaire), could help us to more precisely explain which structural design characteristic and to what extent affect subjects understanding and therefore the ability to perform efficiently and effectively.

ACKNOWLEDGEMENTS

The authors thank the students who participated in this study, at the Dept. of Informatics at Technological Education Institute of Thessaloniki. Also, we thank the anonymous reviewers for providing useful comments.

REFERENCES

1. Deligiannis, I., Shepperd, M.,Webster, S.,Roumeliotis, M., *A Review of Experimental Investigations into Object-Oriented Technology.* Empirical Software Engineering Journal, 2002. 7(3): p. 193-231.
2. Fenton, N. and S.L. Pfleeger, *Software Metrics, A rigorous & Practical Approach, Second Edition.* International Thompson Computer Press, 1997.
3. Arisholm, E., D. Sjoberg, and M. Jorgensen, *Assessing the Changeability of two Object-Oriented Design Alternatives - a Controlled Experiment.* Empirical Softw. Engineering, 2001. 6: p. 231-277.
4. Cockburn, A., *The Coffee Machine Design Problem: Part 1 & 2.* C/C++ User's Journal, 1998(May/June).
5. Coad, P. and E. Yourdon, *Object-Oriented Design*. first ed. ed. 1991: Prentice-Hall.
6. Coad, P. and E. Yourdon, *Object-Oriented Analysis*. second ed. 1991: Prentice Hall.
7. Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., Wesslen, A., *Experimentation in Software Engineering - An introduction*. 2000: Kluwer Academic Publishers.
8. Marinescu, I., R., *An Object Oriented Metrics Suite on Coupling. Universitatea*. 1998.
9. Wilde, N., Mathews, P., and Ross, H., *Maintaining Object-Oriented Software.* IEEE Software, 1993 (Jan): p. 75-80.
10. Chidamber, S. and C. Kemerer, *A Metrics Suite for Object Oriented Design.* IEEE Trans. Softw. Eng., 1994. 20(6): p. 476-493.
11. Henderson-Sellers, B., *Modularization and McCabe's Cyclomatic Complexity.* Communications of the ACM, 1992. 35(12): p. 17-19.
12. Chen, J.-Y. and J.-F. Lu, *A new metric for object-oriented design. Information and Software Technology*, 1993. 35(April): p. 232 - 240.
13. Goodley, S., *Java on course to dominate by 2002*. 1999, http://www.vnunet.com/News/87054.
14. Gamma, E., Help, R., Johnson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. 1995: Addison -Wesley.
15. Larman, C., *Applying UML and Design Pattrens*. 2002: Prentice Hall PTR.