

# The Maintainability of Object-Oriented Software

John M. Hall  
University of Idaho  
john-cs581@johnmhall.net

## Abstract

*Object-oriented processes are often touted as solutions to software development problems. The popularity of object-oriented method has generated many favorable reviews, particularly in the popular media. Object-oriented development promises to reduce the maintenance effort. However, these promises are not based on reliable experimentation. The software engineering literature does include some experimental results. These studies do not show that object-oriented techniques reduce maintenance costs. Most of these experiments show that object-oriented approaches actually increase maintenance difficulties.*

Key Words: Object-Oriented, Experimental Results, Software Engineering, Software Maintenance, Procedural, Design Methodologies

## Introduction

Object-oriented development methods model system components as objects. These objects allow the designer to separate the interface from the implementation. The popular thought is that this allows a developer to ignore many aspects of the system while focusing on those at hand. Another conjecture is that this object model is more natural in terms of how we model systems. The popular trend seems to be a shift from typical procedural development methods to object-oriented methods. Unfortunately, software engineering is a field that even in the research is traditionally poor with respect to experimental verification [9].

Maintenance is the development phase that occurs to a system after it has been delivered for its original purpose. This is an important phase for software projects because most development effort goes toward maintenance [5]. The maintenance phase is worth studying because of this significant contribution to development effort.

## Problem Statement

Because of the importance of the maintenance phase, a simple definition of a good development method is one that requires less effort to maintain. The choice of development method has some impact on the amount of effort needed for the maintenance phase [2]. Object-oriented development methods claim to improve maintainability. However, this hypothesis is claimed without any experimental data. Therefore, it is important to examine the software engineering

literature to find some experimental data to either back or refute this claim. However, before this hypothesis can be investigated, the null hypothesis must first be shown or rejected. The null hypothesis states that there is no difference between the maintainability of procedural and object-oriented methods. Investigating these two hypotheses would allow more informed decisions when choosing development methods.

## Literature Review

There are relatively few complete experimental results comparing the effectiveness of object-oriented design methods. This is consistent with findings that experimental validation is universally used within the field of software engineering [9]. Some experimental results are available in [2]. This experiment compared the effectiveness of subjects when maintaining both procedural and object-oriented designs. In general they found that object-oriented designs were more difficult to maintain. In particular, they found strong results that bad object-oriented designs were more difficult to maintain than procedural designs. They also found statistically weaker results suggesting that good procedural designs were easier to modify than good object-oriented designs. This experiment was based on a small sample size of 20 students. The conclusion is that, especially if developers are experienced with procedural methods, the shift to object-oriented methods will cause a substantial drop in productivity. The study conducted by [8] was looking at a smaller subset of object-oriented maintainability characteristics. In order to better control the experiment, they looked only at whether inheritance, a fundamental object-oriented construct, made programs more difficult or less difficult to maintain. They studied two groups of approximately 57 students. They found that for complex programs, inheritance makes maintenance more difficult, not less.

There are also several examples in the literature that find that object-oriented approaches have some advantages with respect to maintainability. Most of these, such as [5], do not use experimental data to back this claim. However, the study described in [3] does a fairly complete experiment. This project evaluates the comprehension of both procedural and object-oriented code. This experiment used 30 professional developers in two phases that were at least a week apart. Comprehension was measured as the number of files each subject opened. The results of this experiment showed that the procedural group had a larger scope of comprehension. This means that those subjects maintaining a procedural project needed to open a larger percentage of the source files than subjects maintaining an object-oriented project.

In general the literature finds that object-oriented design methodologies require more effort to maintain than traditional procedural methodologies. [4] provides an overview of eight studies regarding the effectiveness of object-oriented design methodologies in general. This document suggests several problems with the experiments. In general most experiments use inexperienced students as subjects, most experiments deal with small projects, most experiments provide insufficient documentation to subjects, and most experiments focus on a small number of the available constructs [4]. These variables introduce some uncertainty into the results. These variables must either be explicitly studied individually, or controlled for in future experiments.

## Conclusion

The software engineering literature does not support the null hypothesis investigated. There is a difference between the maintainability of procedural and object-oriented methods, at least among the groups studied. However, the experimental results as a whole do not show that the maintainability of one method is better or worse than the other. In order to either support or reject this claim, the contradicting experimental studies must be refuted.

The experiment described in [3] showed that maintenance of a procedural system required comprehending a greater percentage of the system than maintenance of an object-oriented system. The largest gap in this study is regarding the definition of comprehension. What is not clear from the experimental design is the number of files for each version of the system. If the number of files in the object-oriented version was larger, then defining comprehension as a percentage of the files that was viewed does not seem to be an accurate measure. This study warrants further research.

The field of software engineering needs to do more research to answer the question of whether object-oriented methods are better than procedural methods in terms of maintainability. In addition, to be applied to larger groups, these studies must control attributes such as experience of the subjects.

## Bibliography

- [1] Briand L., Bunse C., Daly J. *An Experimental Evaluation of Quality Guidelines on the Maintainability of Object-Oriented Design Documents*, In Proceedings of Empirical Studies of Programmers, Oct. 1997. Available at: <http://citeseer.nj.nec.com/briand97experimental.html>.
- [2] Briand L., Bunse L., Daly J., Differding C. *An Experimental comparison of the Maintainability of Object-Oriented and Structured Design Documents*, Empirical Software Engineering - An International Journal, 1997. Available at: <http://citeseer.nj.nec.com/12374.html>.
- [3] Corritore C., Wiedenbeck S. *Direction and Scope of Comprehension-Related Activities by Procedural and Object-Oriented Programmers: An Empirical Study*, Proceedings of the 8th International Workshop on Program Comprehension (IWPC'00). 2000. Available at: <http://www2.umassd.edu/SWComprehension/compdocs/nebraska/corritoreiwpc00.pdf>.
- [4] Deligiannis, I. Shepperd, M.J. "A Review of Experimental Investigations into Object-Oriented Technology", Proc. of 5th IEEE Workshop WESS99, Oxford, Sept. 3-4, 1999. Available at: [http://dec.bournemouth.ac.uk/ESERG/Technical\\_Reports/TR02-05/TR02-05.pdf](http://dec.bournemouth.ac.uk/ESERG/Technical_Reports/TR02-05/TR02-05.pdf).
- [5] Engels G., Kappel G. *Object-Oriented System Development: Will the New Approach Solve Old Problems?* IFIP 1994 Congress, Vol. 3, K. Duncan and K. Krueger (eds.), North-Holland, September 1994. Available at: <http://citeseer.nj.nec.com/engels94objectoriented.html>.
- [6] Miller J. Replicating software engineering experiments: a poisoned chalice or the Holy Grail (DRAFT), Reading for Trends in Software Engineering at the University of Calgary. Available at: <http://sern.ucalgary.ca/courses/SENG/693/F00/readings/JamesMiller.pdf>.
- [7] Pressman, R. S. *Software Engineering: A Practitioner's Approach (Fourth Edition)*. McGraw-Hill. ISBN 0073655783. 1997.
- [8] Unger B., Prechelt L. *The impact of inheritance depth on maintenance tasks: Detailed description and evaluation of two experiment replications*. Technical Report 19/1998, Universität Karlsruhe, Fakultät für Informatik, Germany, July 1998. Available at: <http://www.ipd.uka.de/~prechelt/Biblio/inheritTR.pdf>.
- [9] Zelkowitz M., Wallace D. *Experimental validation in software engineering*, Conference on Empirical Assessment in Software Engineering, Keele University, Staffordshire, UK, March, 1997. To appear in Nov. 1997 Information and Software Technology. Available at: <http://hissa.nist.gov/exper/ease.html>.