Using Refactoring Techniques to Reduce Duplication in Object-Oriented Programming Languages

Simon Baars
University of Amsterdam
Puiflijk, Netherlands
simon.mailadres@gmail.com

Ana Oprescu
University of Amsterdam
Amsterdam, Netherlands
A.M.Oprescu@uva.nl

Abstract

Duplication in source code can have a major negative impact on the maintainability of source code. There are several techniques that can be used in order to merge clones, reduce duplication and potentially also reduce the total volume of a software system. In this study, we look into the opportunities to aid in the process of refactoring these duplication problems for object-oriented programming languages. Measurements that have been conducted so far have indicated that more than half of the duplication in code is related to each other through inheritance, making it easier to refactor these clones in a clean way. More measurements will be conducted to get a detailed overview of in what contexts clones occur, and what this means for the refactoring processes of these clones. As a desired output, we strive to construct a model that automatically applies refactorings for a large part of the detected duplication problems and implement this model for the Java programming language.

1 Introduction

Refactoring is used to improve quality related attributes of a codebase (maintainability, performance, etc.) without changing the functionality. There are many methods that have been introduced to help with the process of refactoring [**?**, **?**]. However, most of these methods still require manual assessment of where and when to apply them. Because of this, refactoring takes up a signification portion of the development process [**?**, **?**], or does not happen at all [**?**]. For a large part, refactoring requires domain knowledge to do it right. However, there are also refactoring opportunities that are rather trivial and repetitive to execute. In this thesis, we take a look at the challenges and opportunities in automatically refactoring duplicated code, also known as "code clones". The main goal is to improve maintainability of the refactored code.

Duplication in source code is often seen as one of the most harmful types of technical debt. In Martin Fowler's "Refactoring" book [**?**], he exclaims that *"Number one in the stink parade is duplicated code. If you see the same code structure in more than one place, you can be sure that your program will be better if you find a way to unify them."*. However, this statement is not accepted by everyone. Several papers argue that not each type of duplication is harmful [**?**].

2 First Level Heading

First level headings are all flush left, initial caps, bold and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

2.1 Second Level Heading

Second level headings must be flush left, initial caps, bold and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

2.1.1 Third Level Heading

Third level headings must be flush left, initial caps and bold. One line space before the third level heading and 1/2 line space after the third level heading.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

Fourth Level Heading

Fourth level headings must be flush left, initial caps and roman type. One line space before the fourth level heading and 1/2 line space after the fourth level heading.

2.2 Citations In Text

Citations within the text should indicate the author's last name and year[**?**]. Reference style[**?**] should follow the style that you are used to using, as long as the citation style is consistent.

2.2.1 Footnotes

Indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page they appear on. Precede the footnote with a vertical rule of 2 inches (12 picas).

2.2.2 Figures

All artwork must be centered, neat, clean and legible. Do not use pencil or hand-drawn artwork. Figure number and caption always appear after the the figure. Place one line space before the figure, one line space before the figure caption and one line space after the figure caption. The figure caption is initial caps and each figure is numbered consecutively.

Make sure that the figure caption does not get separated from the figure. Leave extra white space at the bottom of the page to avoid splitting the figure and figure caption.

Figure **??** shows how to include a figure as encapsulated postscript. The source of the figure is in file `fig1.eps`.

Below is another figure using LaTeX commands.

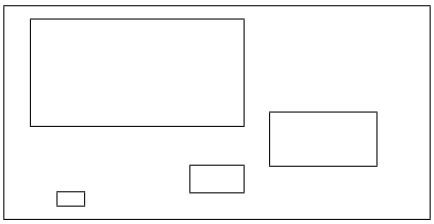

Figure 1: Sample Figure Caption

2.2.3 Tables

All tables must be centered, neat, clean and legible. Do not use pencil or hand-drawn tables. Table number and title always appear before the table.

One line space before the table title, one line space after the table title and one line space after the table. The table title must be initial caps and each table numbered consecutively.

Table 1: Sample Table

A	B	1
C	D	2
E	F	3

2.2.4 Handling References

Use a first level heading for the references. References follow the acknowledgements.

2.2.5 Acknowledgements

We would like to thank the Software Improvement Group (SIG) for their continuous support in this project.

[1] This is a sample footnote