LEWIS UNIVERSITY

Structural Quality & Software Evolution

A Thesis

By

Alison Major

Department of Computer and Mathematical Sciences

Submitted in partial fulfillment of the requirements

for the degree of
Master of Science in Computer Science,

Concentration in Software Engineering

February 12, 2022

The undersigned have examined the thesis entitled '**Structural Quality & Software Evolution**' presented by **Alison Major**, a candidate for the degree of **Master of Science in Computer Science (Concentration in Software Engineering)** and hereby certify that it is worthy of acceptance.

TODO: Add lines for signatures here. Justify above paragraph.

Abstract

Some software engineering projects fail to evolve, which makes them obsolete. This topic is interesting and important to developers because the software that fails to evolve will fail to generate user engagement, leading to revenue loss. We review a number of projects and resources to understand the correlation of software structure quality and its impacts on a system's ability to evolve. With this understanding, we explore ways to improve the evolution of a software system through tools and suggestions.

TODO: Update the abstract with each iteration of this paper.

Acknowledgements

TODO: Update this section with my own acklowedgements.

Gratitude is a great virtue, though revenge is profitable

It's customary and good manners to say thank you however, where do you draw the line? In some of the theses that I've read, and I write this after having read thousands, literally, the following and more have been acknowledged: God, one's advisor, one's better half, parents, children, friends, classmates, lab-mates, lab technicians, lab assistants, pets, fav. Prof, neighbors, physicians, exercise trainer(s), wiki, the maintenance guy, landlord, the school hockey team, secretary, department head, driver, dentist, chauffer, the police, fav. presidential candidate, one's chef, Led Zeppelin, the pastor, one's biggest crush, the cable man, the mani/pedi girl, hair stylist, the best/worst/fav bar tender(s), the janitor, one's obs/gyn, one's mentor, and in a more recent thesis, Michael Phelps (8 gold medals at the 2008 Olympic games in Beijing, China, way to go...)

Keep in mind that one has to use one's own words when writing an acknowledgement. Plagiarism is unauthorized.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

TODO: Chapter 1 - introduction into what I did

When building software systems, we have several areas of concern: cost, delivery timeline, quality, etc. The cost and time-to-market are often the two problems given the highest priority in a project. However, engineers must consider the software quality to preserve the system's longevity. Despite its importance, the code and architecture quality can be challenging to understand and measure.

When we think about projects, we can assume that as time goes on and changes and additions occur within a system's source code, the complexity of that system will grow ("Fig. 1.1"). However, when we manage the code structure, we can keep the complexity in check, allowing systems to evolve. Developers can maintain this structure through simple steps like having readable code and more complex considerations, like how coupled and cohesive a

system is.

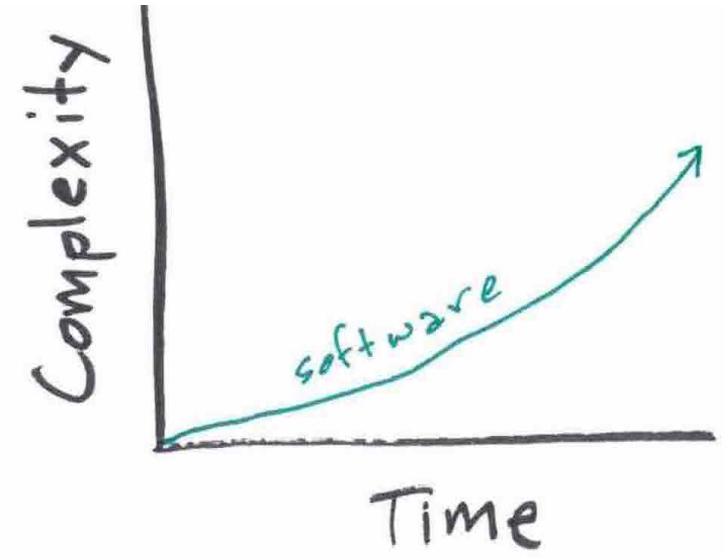

Figure 1.1: Generally speaking, a software system will get more complex as it grows over time.

One way to understand the quality around a system is to discuss its "maintainability," the ease of receiving new features or resolving bugs. For example, developers may find that adjusting one area to add a new feature requires touching several other code areas in tightly coupled systems. Some code measuring systems provide a Maintainability Index, a well-known quality measure. However, its effectiveness in quantifying software quality is debated [1].

On the other hand, code smells are used extensively by practitioners to identify low-quality spots in the software system. These areas would need the teams' attention and are good candidates for refactoring.

Pylint is a static analysis tool that identifies several classes of code quality

concerns, particularly relevant to our study, refactor violations, which report on various code smells. We can assume that there must be some correlation between Maintainability Index and the type and number of code smells in a software system, quantified by the Pylint refactor score.

This study explores such assumptions and systematically investigates any correlation between the Maintainability Index metric and the Pylint Refactor score. Furthermore, we perform analysis on specific refactor violations to reveal and shed light on the relative effectiveness of the different refactor violations and their relationship to Maintainability Index.

The structural quality of a software system will impact the software evolution. If the project has poor structural quality, the architecture will minimize its ability to evolve, and the software system will eventually "die-off" so to speak.

Why should we care about whether the code is maintainable? It is assumed that a large amount of the cost over the lifetime of a project is attributed to maintainability. Fred Brooks, in his book "The Mythical Man-Month" even claimed that over 90% of the costs for a typical software system come up in the maintenance phase [2]. Once the bulk of the system is off the ground and live worldwide, how well the team can improve the system with new features and fix bugs, even working on different parts in parallel, can be impacted by its maintainability. Any successful piece of software will inevitably need to be maintained.

We will look at many open-source Python systems using Pylint and at-

tempt to correlate the data from the Pylint scores to the level of ease in adding new features to the system. This will determine if a system is more maintainable with better Pylint scores.

In this study, we will be using Pylint and will be focused on the values of the Refactor score regarding a set of open-source Python systems. To understand the scores we will be working with; we must understand what Pylint itself is doing.

Through the documentation of Pylint, we can understand how to use it and the scores it will provide [3]. The Pylint score itself is calculated by the following equation [4]:

```
10.0-((float(5*e+w+r+c)/s)*10)
```

Numbers closer to `10` reflect systems that have fewer errors, fewer warnings, and have overall better structure and consistency. In the above equation, we are using the following values [5]:

- **statement** (`s`): the total number of statements analyzed
- **error** (`e`): the total number of errors, which are likely bugs in the code
- **warning** (`w`): the total number of warnings, which are python specific problems
- **refactor** (`r`): the total number of refactor warnings for bad code smells

- **convention** (`c`): the total number of convention warnings for programming standard violations

The Refactor score is of particular interest to us and considers many features that are meticulously outlined on the Pylint site [6]. These types of warnings include several checks, such as when a boolean condition could be simplified, or a useless `return`, and so on. This score, in particular, will be part of our focus.

To calculate the Refactor score, Pylint will check the code for code smells based on the definitions for checks that have been documented. For every infraction, the score increases by one count.

> "In computer programming, a **code smell** is any characteristic in the source code of a program that possibly indicates a deeper problem." [7]

We can use these Refactor scores to help us spot architecture smells. After all, code smells can point the way to deeper problems in our system. There are fundamental design principles that have been established that we should consider when creating software; code smells alert us to areas that have deviated from these principles. These smells are drivers for refactoring and when addressed, can help us maintain the integrity of our architecture rather than creating a patchwork construction.

Work done by Dr. Omari and Dr. Martinez involves collecting a sub-set of Python projects that we can use for further research. The bulk of the effort they have provided is determining which classifiers to use to pare down the public set of Python systems into a good collection for further analysis [8]. The work that they have provided will be used to select appropriate Python systems for review by collecting meta-data on these code systems.

From their subset of repositories, we will then be able to collect current Pylint scores from each of our selected systems. This will give us a sampling of data that we can dig deeper into, comparing similar systems (similar size, similar number of contributors, etc.) and their evolution process by reviewing past commits rather than merely the current state of the system.

TODO: Conclude with a summary of the organization of the thesis, including identification of the general content of specific chapters and appendices.

Chapter 2

Background and Literature Review

TODO: Chapter 2 - why I did it in the context of what was previously known

The background and literature review section needs to provide sufficient fundamental background information about the subject to support your objectives, hypothesis (or research questions) and methods, and review the pertinent literature related to the specific problem / hypothesis you are addressing. In Johnson (1991), some of the questions that he listed that the literature review should be to answer include:

- what are the fundamental science, math, engineering concepts related to your research (scope),
- what part of your research work has ever been investigated before and what has not, (some of this may have been included in the introduction)

- how does your research work relate to that done by others,
- how have others defined/measured/identified the key concepts of your research,
- what data sources have you used or have other researchers used in developing general explanations for observed variations in a behavior or phenomenon in a concept in your thesis etc.

The lit review (20 pages or more) should not be limited to the above questions only. Ingeniousness and creativity is expected of a grad student. Bullets can be single spaced. The above bullets are in the style "thesis-bullets." When you type bulleted text, highlight the bulleted text and then select "thesis-bullets" from under the format, style menu to automatically change their formatting as above.

2.1 Section header

Given the length of each chapter, it is required to use headers and sub headers (possibly sub-sub headers). These can be numbered or one can just rely on different formats. The section headers in this document are labeled "heading 2" ("heading 1" was used for chapter titles). The heading styles formats should be consistent throughout the document as it helps significantly in creating the automatic table of contents.

2.1.1 Sub Heading

The subheadings here have a different format ("heading 3") than the section headers.

Sub-Sub Heading

You can even get to another level of headers, defined here as "heading 4." The table of contents, however, is currently set up to just include three levels of headers.

2.1.2 Equations

Equations can be created in MS WORD equation editor or they can be created with other software. Equations should be numbered. They can be numbered within each chapter (e.g., 2.1, 2.2) or they can be numbered sequentially throughout the entire thesis. Equations should be indented or centered with the equation number to the right. The example below and associated "thesis-eqn" style can be used for all your equations.

Include example of an equation here.

This equation was written with the equation editor. Found through "insert, object, equation editor 3.0. The equation editor can also be found through "tools, customize, commands", and in categories, look for insert and in the commands section, look for equation editor, drag and drop the icon onto the toolbar. This editor is fine for relatively simple equations, other

options are available for more complex equations.

2.1.3 Tables

Tables should have meaningful information with descriptive headers. You can use the "thesis-table caption" style to define your captions and refer to the table in the text with a "cross reference" (Table 1). MS Word re-numbers table captions automatically when new tables inserted. But you need to right click on any cross references and "update field" if there are changes.

Step #	Instruction
Create table caption	Insert, reference, caption, table
Format the caption	Format, style, "thesis-table-caption"
Create table	Table, insert...
Format the table	The formatting of the table can vary, including use of single space as appropriate. Most journals require that tables are formatted using table style "Table Simple 1" format.
Reference the table from text	With the cursor at the location you want to cite the table: insert, reference, cross reference, table, label and number only.

Table 2.1: Table 1: Steps in creating a table

2.1.4 Figures

Figures and illustrations are a necessary means of communicating technical information. Often times, figures included in the background/lit review section are copied from existing copyrighted information. In all cases, this is technically inappropriate without also receiving permission from the copyright owner. Citing the source of the figure is not sufficient. This rule is enforced for PhD dissertations because they are submitted to ProQuest for electronic access by others. The enforcement of this rule for MS theses is dependent on the specific committee members.

Resolution of figures is often a problem in theses. Resolution should be ¿300 dpi, preferably 600dpi (2.1). You should note that saving images as jpeg files is a sure way to lower the resolution to an unacceptable extent. From experience, a good way is to copy your graphic (for example from PowerPoint or excel) and when pasting it into word, use the "paste special" "as an "enhanced metafile" (2.2). This also substantially reduces the resulting file size in comparison with pasting graphs in as excel graphics.

Sub heading (heading 3) The subheadings here have a different format ("heading 3") than the section headers.

Sub-sub heading (heading 4) You can even get to another level of headers, defined here as "heading 4." The table of contents, however, is currently set up to just include three levels of headers.

Equations Equations can be created in MS WORD equation editor or they can be created with other software. Equations should be numbered.

Figure 2.1: Figure 1: Example photo with high resolution. Caption created with "insert, reference, caption, figure" and the style changed to "thesis-figure caption."

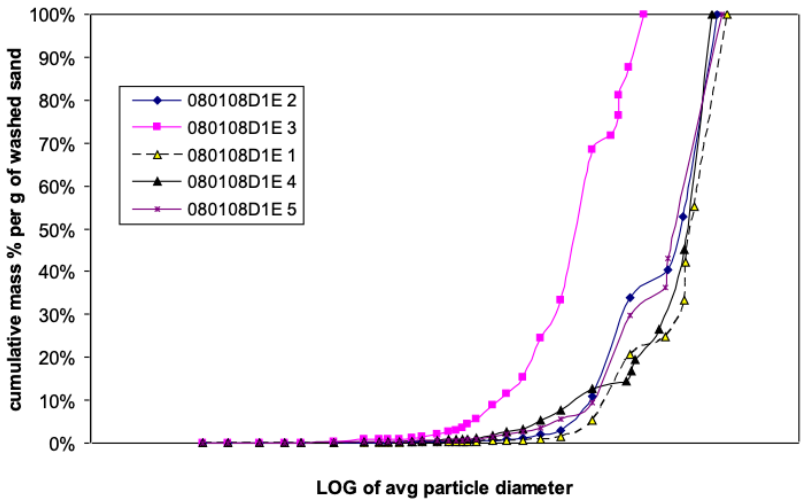

Figure 2.2: "Figure 2: Example of high resolution graphic inserted with "paste special, as enhanced metafile"

They can be numbered within each chapter (e.g., 2.1, 2.2) or they can be numbered sequentially throughout the entire thesis. Equations should be indented or centered with the equation number to the right. The example below and associated "thesis-eqn" style can be used for all your equations. [1] This equation was written with the equation editor. Found through "insert, object, equation editor 3.0. The equation editor can also be found through "tools, customize, commands", and in categories, look for insert and in the commands section, look for equation editor, drag and drop the icon onto the toolbar. This editor is fine for relatively simple equations, other options are available for more complex equations.

Tables Tables should have meaningful information with descriptive headers. You can use the "thesis-table caption" style to define your captions and refer to the table in the text with a "cross reference" (Table 1). MS Word re-numbers table captions automatically when new tables inserted. But you need to right click on any cross references and "update field" if there are changes. Table 1: Steps in creating a table

Figures Figures and illustrations are a necessary means of communicating technical information. Often times, figures included in the background/lit review section are copied from existing copyrighted information. In all cases, this is technically inappropriate without also receiving permission from the copyright owner. Citing the source of the figure is not sufficient. This rule is enforced for PhD dissertations because they are submitted to ProQuest for electronic access by others. The enforcement of this rule for MS theses is

dependent on the specific committee members. Resolution of figures is often a problem in theses. Resolution should be ¿300 dpi, preferably 600dpi (Figure 1). You should note that saving images as jpeg files is a sure way to lower the resolution to an unacceptable extent. From experience, a good way is to copy your graphic (for example from PowerPoint or excel) and when pasting it into word, use the "paste special" "as an "enhanced metafile" (Figure 2). This also substantially reduces the resulting file size in comparison with pasting graphs in as excel graphics.

Figure 1: Example photo with high resolution. Caption created with "insert, reference, caption, figure" and the style changed to "thesis-figure caption."

Figure 2: Example of high resolution graphic inserted with "paste special, as enhanced metafile"

Chapter 3

Methodology

TODO: Chapter 3 - how I did it

In addition to the detailed methods you need to describe in this section, you need to provide specific objectives and an overview of your approach if they have not already been presented in the introductory chapters. The best place to put those items can vary among theses. Sometimes the background and lit review is really necessary to justify and substantiate the specific objectives and approach and, therefore, it is best to save those details for the beginning of this chapter.

These paragraphs are in "thesis-body text." Other styles including captions, headers etc. can be used as presented in the previous chapter. Table 2 summarizes all of the styles that can be used with this template.

Style Name	When Used
Heading 1	Chapter Titles
Heading 2	Primary Headers
Heading 3	Sub Headers
Heading 4	Sub-Sub Headers
Thesis-body text	All Paragraphs
Thesis-bullets	Bullets
Thesis Figure Caption	All figure captions.
Thesis Table Caption	All table captions.
Thesis-eqn	Equations
Thesis-reference	Reference list at end of thesis

Table 3.1: Table 2: Styles used in this template

Chapter 4

Results

TODO: Chapter 4 - what I found

Results, findings, discussion of results OR manuscripts. It is best to also reiterate information in your literature review to help substantiate the findings of your research.

This template is best used for directly typing in your content.

Chapter 5

Conclusion

TODO: Chapter 5 - what it all means, putting the pieces together (what is my contribution to the research field)

This chapter could also be called "Conclusions and Recommendations" or "Conclusions and Implications." In general, there should be no new information presented here. It should be a synthesis of information that you've already discussed.

References

Includes all references: articles, media facts, books, reports, regulations, internet articles, papers that you referenced from the text. In the text, citations can be (Smith and Jones, 2007) or Smith et al., 2007) (if more than two authors) if you wish to present your references alphabetically. Alternatively, you can include the citations in the text as a number [1] or 1 if you wish to present your references numerically. The MS WORD tools – "insert, reference, footnote, endnote" (or "cross reference" if you refer to the same reference more than once) should be used to help you organize and manage your references.

References can be written in single space with extra space between references as in the format below. There are many different ways to arrange the information and punctuation in a reference listing. The most important thing is to make sure all references are complete and that the format of your references is consistent throughout.

Example, S.Z. (2008). How to cite a complete journal reference. J. Complete Thesis. 1(2): 47-52.

Example, S.Z., Second, W.S. (2007). How to cite a complete conference proceedings paper. In: Proceedings, 2nd International meeting of Masters Students, Paper # XW15 (Potsdam NY, November, 2007).

If you use the "thesis" reference" style you will get the proper line spacing and indent style without further changes. Above are examples to show

complete citation, other formats also acceptable.

Alternative format:

An alternative format for references is to use IEEE format. You can find a reference on IEEE format here: http://www.ijssst.info/info/IEEE-Citation-StyleGuide.pdf

You can use either the option provided above in the template or use the IEE format.

We should cite something [8].

Bibliography

[1] A. V. Deursen, "Think twice before using the "maintainability index"," 2014, https://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/ [Online; accessed 23-Jan-2022]. [Online]. Available: https://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/

[2] F. Brooks, *The Mythical Man-Month.* Addison-Wesley, 1975.

[3] Logilab and contributors, "Pylint," Logilab, 2020, https://pylint.org/ [Online; accessed 14-December-2021]. [Online]. Available: https://pylint.org/

[4] P. Logilab and contributors, "Pylint features," Logilab and PyCQA, 2021, https://pylint.pycqa.org/en/latest/technical_reference/features.html#reports-options [Online; accessed 14-December-2021]. [Online]. Available: https://pylint.pycqa.org/en/latest/technical_reference/features.html#reports-options

[5] R. Kirkpatrick, "A beginner's guide to code standards in python - pylint tutorial," 2016, https://docs.pylint.org/en/1.6.0/tutorial.html [Online; accessed 18-December-2021]. [Online]. Available: https://docs.pylint.org/en/1.6.0/tutorial.html

[6] P. Logilab and contributors, "Pylint features," Logilab and PyCQA, 2021, https://pylint.pycqa.org/en/latest/technical_reference/features.html#refactoring-checker [Online; accessed 14-December-2021]. [Online]. Available: https://pylint.pycqa.org/en/latest/technical_reference/features.html#refactoring-checker

[7] W. contributors, "Code smell — Wikipedia, the free encyclopedia," 2021, https://en.wikipedia.org/wiki/Code_smell [Online; accessed 18-December-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Code_smell

[8] S. Omari and G. Martinez, "Enabling empirical research: A corpus of large-scale python systems," 2018, [Provided by Dr. Omari].

Appendix A

Type or paste your appendices here. Appendices are a place to organize and include all of the "extra" material that is important to your research work but that is too detailed for the main text. Examples can include: specific analytical methods, computer code, spreadsheets of data, details of statistical analyses, etc. But, these materials do not speak for themselves. There should be a reference to these materials from the main chapters (complete details included in Appendix A) and there should be some text at the beginning of each appendix to briefly explain what the information is and means that is included in that appendix.