Report Assignment 1
Software Evolution

Quinten Heijn - 10374442
Dylan Bartels - 10607072

22 November 2017

Introduction

In this report we give a short explanation of our submission for assignment 1 for Software Evolution. We begin with addressing some general choices that apply for all the metrics. We continue with giving some insight on how we calculated each metric and conclude with a summary of our results.

A key concept in this assignment is the *unit*. This concept is used in multiple of our metrics. We defined an unit as the smallest piece of testable code[1]. Because we're using Java, an unit will refer to a single method.

Most of our choices are based on the paper *"A Practical Model for Measuring Maintainability"*[1]. We also used the textbook *"Building Maintainable Software"*[2].

Volume

Method

To calculate the volume of a project we counted the number of lines of code for all the files in the project that ended with *".java"*. *Lines of code* are all the lines in the relevant files excluding empty lines and comments.

To translate the lines of codes to a ranking we used the table provided in the paper[1].

Motivation

We chose for this method due to the literature being recommended as the primary guideline for the metrics.

Unit Size

Method

The calculation of the unit size was very straightforward. We extract the full file of the M3 model, which is created with the standard library *m3*. Again, we remove the empty lines and comments and count the remaining number of lines.

The lower boundaries for the ranking are based on the literature on from Building Maintainable Software, Java Edition: Ten Guidelinesfor Future-Proof Code[2, p45]

16-30	31-60	61-100	rank
25	5	0	++
40	20	0	+
50	30	5	o
60	40	10	-
-	-	-	- -

Motivation

This method is chosen due to it fulfilling exactly the result we wish to achieve. Given that the volume method is correct than we can assume the unit size will also be.

Complexity

Method

The method used to calculate the code complexity is based upon the unit complexity method found in A practical model for measuring maintainability[1]. With this method the code complexity is measured per unit for each function by incrementing the counter by one when the following cases occur:

- do (Statement) while (expression)
- while (expression) (Statement)
- if (expression) (Statement)
- if (expression) (Statement) else (Statement)
- for (expression) (Statement)
- case (expression)
- catch (Parameter) (block)
- (expression) and (expression)

- (expression) or (expression)

Based upon the same literature the interval chosen to distribute the calculated unit complexity into is 0-11, 12-20, 21-50, 50+. The total lines of code corresponding with the unit complexity are compared as a percentage of the total lines of code.

Motivation

Complexity greatly influences the readability of an unit, resulting in a developer preference for lower code complexity.

Duplication

Method

We used the definition given in the paper[1] for duplicate code. A block of code, larger than 6 lines, that is similar to another block of code larger than 6 files. However, we made a few exceptions. We did not look at empty lines, comments and brackets. A lot of closing brackets can cause false positives, in the sense that the block of code is a duplication, but it can not be resolved and it says little about the maintainability of the software.

Motivation

Duplication makes a system larger than it needs to be. Also, it often means when changes are made a programmer has to be aware of all the copies of a piece of code, making software with many duplication a lot harder to maintain.

Results

SmallSql

Volume:
24048 lines of code

Unit Size with interval [0-15, 16-30, 31-60, 60+]:
86%, 9%, 4%, 1%

Complexity with interval [0-10, 11-20, 21-50, 50+]:
77%, 7%, 11%, 5%

Duplication:
3.96%

Metric	Rank
Volume	++
Unit Size	o
Unit Complexity	--
Duplication	+

SqlBig

Volume:
168836

Unit Size: [0-15, 16-30, 31-60, 60+]:
76%, 13.7% , 6.7%, 3.7%

Complexity:
For complexity we encountered a problem with parsing the java templates that were used. The parseTree library did not seem to support this java feature. We tried filtering out the lines of code that made use of template, but not succeed because of an error with reading the files when using the IO function readFiles. We got a "Unsupported scheme unknown" error, which we could not resolve with only the online documentation.
Duplication:
Because we were using an algorithm of quadratic complexity and made use of lists, instead of maps, the time it took to calculate duplication became too long for us to get the result.

Metric	Rank
Volume	++
Unit Size	o
Unit Complexity	?
Duplication	?

References

[1] Ilja Heitlager, Tobias Kuipers, and Joost Visser. A practical model for measuring maintainability. In *Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the*, pages 30–39. IEEE, 2007.

[2] Joost Visser, Sylvan Rigal, Rob van der Leek, Pascal van Eck, and Gijs Wijnholds. *Building Maintainable Software, Java Edition: Ten Guidelines for Future-Proof Code.* " O'Reilly Media, Inc.", 2016.