Software Metrics as Indicators for Effort of Object-Oriented Software Projects

Wouter F. A. Bos
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
w.f.a.bos@student.utwente.nl

ABSTRACT

In the world of software development, it is essential to decrease the effort developers need to change or add functionality to software. A lower effort lead to lower cost, both in human and financial resources. One way to achieve this may be increasing the agility of software. It would be useful to know if there are existing measurable software properties (software metrics) which could indicate the agility of the software. Recent research has found a way to find software metrics which could indicate the agility of software. Furthermore, it found eight software metrics which can be considered. Unfortunately, more metrics were not in the scope of this research.
This research uses new data to show if any suitable indicators of software agility are consistent. Furthermore, we consider more software metrics. Significance tests will validate the results. Next, we review relevant research to rule out the mathematical cause for unexpected results of recent research.

Keywords

Software Agility, Software Metrics, Code Properties, Complexity, Cohesion, Coupling, Estimation of Effort, Object Oriented-Software, Pearson's Correlation Coefficient, Spearman's Correlation Coefficient

1. INTRODUCTION

In the world of software development, it is essential to decrease the effort developers need to change or add functionality to software. A lower effort lead to lower cost, both in human and financial resources. One way to achieve this may be increasing the agility of software.

Currently, it is not very clear what the agility of software is. It would be useful to know if there are existing measurable software properties which could show the agility of the software. Based on these properties, developers may be able to improve their software's agility. Thus, reducing the needed effort to change or add functionality at a later stage of the project. Furthermore, these indicators may help to improve the accuracy and speed of effort-estimation based on the history of a software project. Since effort-estimation is often a large part of software project management [12] this may further reduce the cost.

Nowadays, software developers often use software metrics to evaluate software properties. One or more software metrics can contribute to evaluating one software property. For example, the Lines of Code[1] metric contributes to evaluating the complexity property. Other properties can be flexibility and maintainability.

Our goal is to find out which software metrics of Object-Oriented Systems are suitable indicators of agility in software. Recent relevant research [9] shows that many metrics can be considered as an indicator for the agility of software. Examples are the Weighted Methods per Class (WMC, see section 2.1.1) and the Depth of Inheritance Tree (DIT, see section 2.1.2). Furthermore, there is a way to test the suitability of those metrics. This method uses Pearson's and Spearman's correlation coefficients (see section 2.3).

Based on recommendations of recent relevant research [9], this research looks if it is possible to consider two more metrics: Afferent Coupling and Efferent Coupling (see sections 2.1.6 and 2.1.7). The research will also investigate if different data consistently show suitable indicators of software agility. Furthermore, the research focuses on finding other ways to test the suitability of software metrics as indicators of software agility.

To reach our goals, we will perform a measurement experiment on an agility testbed. We will analyse the results using Pearson's and Spearman's correlation coefficients. Finding other testing methods for the suitability of software metrics as indicators is primarily done by reviewing other relevant research.

The rest of this paper has the following structure. Section 2 elaborates on the background of this research proposal. Next, section 3 discusses the research setup. Following is the methodology in section 4. After that, section 5 shows the results and section 6 will discuss them. In section 7 we conclude our research, after which we recommend future work in section 8.

2. BACKGROUND

This chapter will provide the information essential for understanding the research. The chapter will start by discussing relevant software metrics and properties. After that, section 2.2 explains the agility testbed in more detail. Section 2.3 gives information about Pearson's and Spearman's correlation coefficients.

.

[1]The size of software based on the number of code lines.

30th Twente Student Conference on IT Febr. 1st, 2019, Enschede, The Netherlands.

2.1 Software metrics and properties

There are many different metrics, used for many different applications. In general, metrics are used to evaluate, classify and identify flaws in software.

This research will use ten different metrics:

- WMC, DIT, RFC and AMC for complexity
- CBO, CA and CE for coupling
- CAM and two versions of LCOM for cohesion

The metrics are selected based on other relevant work (see section 9) and availability of measurement tools (see section 3.2). The following subsections will elaborate on the selected metrics by defining them. Formal definitions can be found in relevant work [3, 4] and the source of the tooling we use[2].

2.1.1 Weighted Methods per Class (WMC)

WMC is the sum of all the complexities of the methods in a class and is the first metric described in the metrics suite by Chidamber and Kemerer (CK) [3]. This metrics suite consists of six metrics. In the initial definition of the WMC, all methods only had a complexity of one, which made it the number of methods per class [4]. Nowadays, there are other definitions of a method's complexity. The amount of branches is an example of this. In our research, we will use the initial definition.

2.1.2 Depth of Inheritance Tree (DIT)

The DIT is the number of a class' ancestors. This includes more than the parent classes. In Java, the *Object* class is an ancestor of all other classes. This results in a DIT value of at least one. It is the second metric described in the CK metrics suite.

2.1.3 Response For a Class (RFC)

The Response For a Class is the size of a class' response set. This set is the union of the class' methods that can be called when a method of a class' instance is invoked and the class' methods. Ideally, all deeper nested calls are included. Since this can be expensive to carry out correctly, often only once-nested calls are counted. This follows the proposed definition as the fifth metric in the CK metrics suite and gives a rough indication of the ideal RFC value.

2.1.4 Average Method Complexity (AMC)

The Average Method Complexity is comparable to the WMC. The difference is that the AMC calculates the average of all complexities, where the WMC calculates the sum.

2.1.5 Coupling Between Object classes (CBO)

The CBO looks for classes acting on other classes. Examples are using methods or variables of another class.

2.1.6 Afferent Coupling (CA)

The Afferent Coupling of a class counts the number of classes which depend on it [6].

2.1.7 Efferent Coupling (CE)

The Efferent Coupling of a class counts the number of classes on which it depends [6].

2.1.8 Cohesion Among Methods of a class (CAM)

The CAM looks at method parameters and class attributes to measure a class' cohesion.

[2]http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/metric.html

2.1.9 Lack of Cohesion in Methods (LCOM/LCOM3)

Lack of Cohesion in Methods measures the similarity of methods within a class. This research uses two versions. Generally, software projects use the LCOM metric. Yet, Object-Oriented software projects use the LCOM3 most often.

For the LCOM metric, we use the definition of the CK suite. It subtracts the number of pairs of distinct methods in a class that do not share instance variables with those which do. If the value of LCOM is negative, it evaluates to 0.

Handerson et al. [8] defines our LCOM3 metric. The definition of is as follows:

$$LCOM3 = \frac{(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)) - m}{1 - m}$$

Here, m is the number of the class' methods, a is the number of the class' variables and $\mu(A)$ is the number of methods that access a variable. The LCOM3 varies between 0 and 2 and it does not take constructors, getters and setters into account.

2.2 Agility Testbed

Two Agility Testbeds were made with the aim of simplifying the study of software agility. Recent research [9] has used the first testbed[3] to perform a measurement experiment. We will use a newer testbed[4] to perform our measurement experiment.

In this testbed, thirteen different developers implemented a banking software system (API) in Java. All have the same functionality at predetermined points of progress in the project. This allows for better comparison between different systems than when comparing two different software projects.

In the first phase, the developers were divided into groups of two or three. Each group implemented a base system, which resulted in seven usable repositories for the second phase. These repositories are marked with letters (A - G).

The second phase consisted of six extensions which added functionality to the base system. Each individual developer extended the base system they developed in the first phase on their own. The order of implementing the extensions was predetermined. The developers only could continue with a next extension when they completed the previous one. It was not mandatory to complete all extensions, which resulted in a progress difference between all thirteen repositories. Table 1 shows the progress of each repository. The progress is indicated by the extension a developer finished (e.g. X1 indicates that the developer finished the first extension). Due to different reasons, the developers who worked on repository E in the first phase decided to stay together for the second phase. Repository G split off repository A.

To determine the effort spent by the developers, they recorded the hours they spent. The developers could mark those ours with three categories:

1. Primary development: Time worked on developing the actual functionality, thinking about the architecture and writing the code.

2. Secondary development: Writing documentation, testcases and tests.

[3]https://agilitytestbed.github.io
[4]https://agilitytestbed.github.io/2018

Table 1. Progress of each repository of the Agility Testbed

Repository	Progress.
A1	Base (B)
A2	B
B1	X5
B2	X6
C1	B
C2	X3
D1	X6
D2	X4
E	X2
F1	X4
F2	X4
G	B

3. Other tasks: Time spent on other things like looking into libraries and joining in meetings.

The testbed also contains which study the developers were following during the project and the study-year they were in. This might help eliminate the skill level of a developer when determining the effort put into developing.

2.3 Statistical Methods

This research will use Pearson's r and Spearman's ρ. These indicate if there is a correlation between two data sets.

2.3.1 Pearson's r

Pearson's r indicates the strength of a linear relationship between two variables. An r of 1 indicates a perfect positive relationship. This means that for any two variables X and Y if X increases then Y linearly increases. An r of -1 indicates a perfect negative relationship. If X increases, then Y linearly decreases. If the r is 0 it indicates that there is no correlation.

The definition of Pearson's r is as follows:

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}}$$

Here, n is the number of pairs of values X and Y. Pearson's r assumes that the variables X and Y are normally distributed.

2.3.2 Spearman's ρ

Spearman's ρ indicates the strength of a parameter-independent monotonic relationship between two variables. A ρ of 1 indicates a direct relationship. This means that for any two variables X and Y if X increases then Y increases. A ρ of -1 indicates an indirect negative relationship. If X increases, then Y decreases. If the ρ is 0 it indicates that there is no correlation. Not every monotonic relationship is linear, but every linear relationship is monotonic. The coefficient uses ranks instead of variable values. For example:

$$Let\ X := [4, 7, 1, 3, 9, 6, 7],$$
$$Then\ the\ ranks\ R(X) := [2, 4.5, 0, 1, 6, 3, 4.5]$$

If multiple values have the same rank, we use the average of these ranks. The definition of Spearman's ρ is as follows:

$$\rho = 1 - \frac{6\sum d^2}{n(n^2-1)}$$

Here, n is the number of pairs of values X and Y. d is the difference between the ranks $R(x)$ and $R(y)$ of a pair of values. Spearman's ρ does *not* assume a normal distribution of variables X and Y.

2.3.3 Significance tests and hypotheses

The statistical significance of the calculated correlation coefficients can validate the results of this research. To determine this, the independence can be tested by formulating hypotheses. The first hypothesis is the null-hypotheses H_0. This hypothesis proposes that no significant relationship exists between two tested variables. This is usually the opposite of the hypothesis H_1. For example, for Pearson's r:

$$H_0 : r = 0$$
$$H_1 : r \neq 0$$

For H_1 to hold, the p-value should be less than the significance a. The p-value represents the chance that H_0 is accepted. The a is the risk we are willing to take that H_0 is true. The level of certainty we want that H_1 holds is 95%. This is the confidence interval. Now, the sum of a and the confidence interval is 1, so a equals 0.05.

3. RESEARCH SETUP

This chapter will elaborate on the preparation and selection of the testbed's data as well as the used tooling. It also discusses a prediction of the influence of metrics on developer effort. We begin with establishing the scope of the research by making a selection of data. Then, we will discuss the used tooling. In section 3.3 a prediction is made of the influence of metrics on effort.

3.1 Selection of Data

It is not trivial to choose which data to use in the statistical analysis. We here establish which extensions we take into account. The preparation and selection of time data will be discussed as well as the option to scale this data.

3.1.1 Extensions

The first choice we can make is between which extensions we want to compare. Our sample size is the main consideration here. The possibilities are B-X1, X1-X2, X2-X3, X3-X4, X4-X5 and X5- X6. Since the sample size decreases significantly after X4 (from six to three), we chose to eliminate the last two possibilities. To keep the sample size as big as possible, we chose to use all other possibilities.

3.1.2 Reported Time Data Set

Creating the new testbed finished recently. This meant that the administered times were not ready for our research yet. So, we had to calculate the actual effort put into the base and the extensions ourselves. We decided to take the sum of time developers spent on primary or secondary development since these times reflect the effort put in development.

While most effort was administered carefully, there were some developers who made small mistakes. Relevant mistakes were as follows:

- Repository E did not assign a category to three of their recorded hours. Since they did keep track of the date of administration, we could determine it was part of the development of the Base. We chose to assign the three hours in proportion with the earlier assigned hours.
- Repositories B1 and B2 both assigned two categories to respectively 10.5 and 18.75 hours. Most of them were marked for both primary and secondary development. In this case, it did not have any effect since we need the sum of categories one and two. Only

Table 2. Times Scaled on B

	B-X1	X1-X2	X2-X3	X3-X4
B1	0.152	0.065	0.203	0.100
B2	0.359	0.294	0.156	0.026
D1	0.203	0.170	0.184	0.116
D2	0.499	0.443	0.366	0.201
E	0.185	0.343		
F1	0.363	0.316	1.148	0.281
F2	0.156	0.133	0.141	0.094

Table 3. Times Scaled on B and Preceding Extensions

	X1-X2	X2-X3	X3-X4
B1	0.056	0.167	0.070
B2	0.217	0.094	0.014
D1	0.141	0.134	0.075
D2	0.295	0.188	0.087
E	0.290		
F1	0.232	0.684	0.099
F2	0.115	0.109	0.066

for B2, we did have to divide one hour between categories two and three. The developer of B2 did keep track of the date of administration too, we were possible to proportionally assign the time within the relevant extension.

When discussing time scaling and selection, there are four options:

1. Select the reported times for reaching the extension, raw.
2. Select the reported times for reaching the extension combined with preceding extensions (and the base), raw
3. Select the reported times for reaching the extension, scaled on the base.
4. Select the reported times for reaching the extension, scaled on the base and all preceding extensions.

There are two sub-choices: use raw or scaled times and put times together or not. Since scaling should help to eliminate the factor of developer skill, we decided to scale the reported times. However, we are not sure how this factor can be eliminated best. While scaling on the base already factors in the developer skill per group, adding all preceding extensions would also include some individual skill. So, we chose to do both scaling on the base and scaling on the base plus all preceding extensions.

The selected times are shown in tables 2 and 3.

3.2 Tooling

Our research needs tools that can analyse software projects, retrieve software metrics and perform statistical analysis on the metrics. To retrieve the software metrics, the research will use the CKJM Extended[5] tool. This is an extended version of the Chidamber and Kemerer Java Metrics tool. The source code has been edited slightly to accommodate for Java 8, which was used in all projects of the testbed. The tool analyses the compiled bytecode to export the resulting values for the software metrics per class. This then can be used for further statistical analysis. Unfortunately, it was not possible to compile the bytecode of repository C2 because of a dependency issue. Since solving this would take time and affect the time data we chose to ignore C2.

For the statistical analysis, we use Python with the Scipy, Numpy and Matplotlib modules. We managed to save time using earlier work by Hollander [9]. The code can be found on GitHub[6].

The retrieved metric data is shown in tables 5, 6, 7 and 8.

[5]http://gromit.iiar.pwr.wrod.pl/p_inf/ckjm/

Table 4. Metric predictions and result

metric	Pred.	Res.	New Pred.
WMC	+	-	-
DIT	+	-	-
RFC	+	-	-
AMC	+	-	-
CBO	+	-	-
CA	N.A.	N.A.	-
CE	N.A.	N.A.	-
CAM	-	+	+
LCOM	+	-	-
LCOM3	+	Inc.	-

3.3 Prediction Of Metric Signs

Table 4 shows an overview of the metrics described in section 2.1. The first column shows the metric. Column two and three state Hollander's prediction and result for each metric [9]. The last column a prediction for this research. A plus means a direct or positive relation. This means that a higher metric value amounts to more effort. A minus means an indirect or negative relation. Thus meaning a lower metric value amounts to more effort. "N.A. " stands for "Not Available".

The result of LCOM3 was inconclusive. Pearson's correlation coefficient indicated a positive relation. Yet, Spearman's indicated an inverted relation. Since the resulting sign of the LCOM metric was negative, we predict the sign of LCOM3 to be negative too.

Since the methodology won't differ much from Hollander's research, the new prediction is based on the result for all metrics. For CA and CE it is logical to predict that a higher value amounts for more effort, based on the CBO. Since Hollander's results were all reversed, we predict that the sign will be negative.

Hollander's research also found that the CAM and WMC metrics are highly correlated. Thus, indicating that they are suitable indicators of software agility. Since these are not coupling metrics, we expect that the CA and CE are not highly correlated.

4. METHODOLOGY

This research will consist of three parts:

1. Measurement: retrieve relevant software metrics from the testbed
2. Statistical Analysis: analyse software metrics and possible correlations
3. Literature Research: investigate Pearson's and Spearman's correlation coefficients

The first two parts will be done after each other, with the measurement part as the first one. In this part, the

[6]https://github.com/wouwouwou/2018-2019_AgilityResearch

Table 5. Metrics B

	WMC	DIT	RFC	AMC	CBO	CA	CE	CAM	LCOM	LCOM3
B	8.778	2.111	21.111	22.518	3.444	1.667	2.444	0.603	80.556	1.479
D	7.214	1.214	15.357	11.141	4.429	2.000	2.571	0.573	9.429	1.164
E	4.394	1.182	9.121	7.052	5.848	2.091	3.758	0.559	4.455	1.133
F	4.538	0.923	17.846	36.706	4.538	1.385	3.154	0.611	8.846	1.495

Table 6. Metrics X1

	WMC	DIT	RFC	AMC	CBO	CA	CE	CAM	LCOM	LCOM3
B1	9.455	1.909	21.909	23.047	4.273	2.091	2.727	0.479	89.909	0.916
B2	10.636	1.909	24.455	17.293	3.455	1.636	2.364	0.604	117.091	1.459
D1	9.400	1.200	20.267	11.493	4.667	2.133	2.667	0.542	18.000	1.140
D2	9.667	1.200	20.200	11.446	4.733	2.133	2.733	0.530	18.667	1.144
E	5.073	1.146	10.634	7.256	6.488	2.512	3.976	0.557	7.049	1.145
F1	4.933	0.933	19.267	44.188	4.533	1.400	3.133	0.603	10.267	1.486
F2	5.438	0.938	21.688	31.895	6.000	1.812	4.188	0.572	12.562	1.522

Table 7. Metrics X2

	WMC	DIT	RFC	AMC	CBO	CA	CE	CAM	LCOM	LCOM3
B1	10.077	1.846	22.923	17.691	4.308	2.077	2.692	0.477	102.769	0.914
B2	10.500	1.786	25.286	16.903	3.500	1.714	2.357	0.570	110.714	1.407
D1	9.056	1.444	20.111	13.612	4.611	2.111	2.611	0.568	16.833	1.156
D2	10.471	1.176	22.059	11.678	4.529	2.059	2.588	0.530	20.647	1.103
F1	5.111	0.944	20.444	48.818	4.611	1.389	3.222	0.611	10.667	1.347
F2	5.222	0.944	22.278	33.777	6.222	1.889	4.333	0.551	10.000	1.511

Table 8. Metrics X3

	WMC	DIT	RFC	AMC	CBO	CA	CE	CAM	LCOM	LCOM3
B1	11.429	1.786	25.000	17.390	4.286	2.071	2.643	0.471	141.714	0.903
B2	11.375	1.688	25.000	14.887	3.562	1.750	2.312	0.547	153.875	1.403
D1	10.526	1.421	22.895	13.400	5.000	2.316	2.789	0.545	25.789	1.140
D2	11.889	1.167	24.778	11.681	4.722	2.167	2.667	0.518	26.944	1.088
F1	6.323	0.935	18.452	17.467	6.710	2.452	4.323	0.544	16.161	0.961
F2	5.429	0.952	24.143	34.997	7.619	2.143	5.476	0.545	10.429	1.522

CKJM Extended tool analyses the bytecode and generates the values for every metric. The Statistical Analysis will process these results and determin the characteristics of the data.

4.1 Measurement

In this part, we use Python to execute command line commands. This makes it possible to automatically clone the Git repository to the local machine. For every extension, the head is reset to the corresponding extension version hash. After compiling, the CKJM Extended tool is run on the compiled bytecode, which results in an XML file. This file contains the values for the metrics, per class in the project. Once the CKJM Extended tool has been used for all projects and all extensions (including the base), an XML file exists for every extension in a project. Using these files, we can carry out the statistical analysis.

4.2 Statistical Analysis

The statistical analysis part will process the data gathered in the measurement part. It calculates the correlation coefficients using this data. The first thing that we did was creating a CSV file containing the developers' reported times. The Python code contains classes for projects, their versions, the time data and the metric data. We loaded all data from XML files into the class structure. For most metrics, we took the average of its values for each class to get its value for a whole project. Exceptions were the Lines Of Code (LOC) and the Cyclomatic Complexity (CC) metrics. Here, we took the summed total (LOC) and the bare value (CC).

Once we populated the class structure of a project, we calculated the correlation coefficients. In the first run, we took the times scaled on the base version. In the second run, we included the preceding exceptions in these times.

To calculate the correlation coefficients per project, we needed pairs of the metric of an extension, and the time it took to get to the next one. Furthermore, to show the statistical significance of the found coefficients we calculated the p-values.

4.3 Literature Research

To determine if there are other ways to test the suitability of metrics as indicators of software agility, we perform a literature research. Furthermore, we contacted teachers and scientists of the University of Twente with this question.

4.4 Summary

We compare the metric values of WMC, DIT, RFC, AMC, CBO, CAM, CA, CE, LCOM and LCOM3 for the projects B1, B2, D1, D2, E, F1 and F2 at B, X1, X2, X3 to the time it took for the developers to reach the next extension, scaled to B **and** scaled to B plus preceding extensions. The CKJM Extended tool measures the metric values. We use Python to, per metric, calculate the Pearsons and Spearmans correlation coefficients of the data sets of the times and means. Literature research should determine if there are other ways to test the suitability of the software metrics as indicators of software agility.

5. RESULTS

Appendix A shows all tables which contain the results. Tables 9 up to and including 15 show per metric the values for the two correlation coefficients and the sign of these coefficients. Tables 9, 10, 11 and 12 use the times scaled on B and tables 13, 14 and 15 use scaling based on B plus preceding extensions. The values of the coefficients will

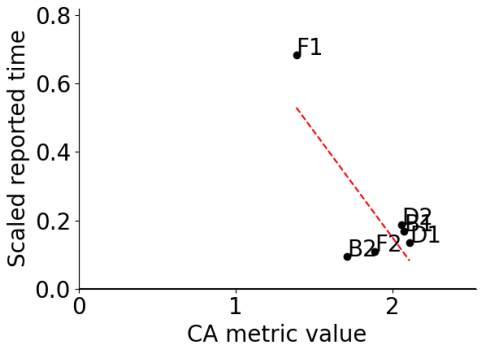

Figure 1. Scatter plot for CA metric (X2-X3, times scaled on B plus preceding extensions)

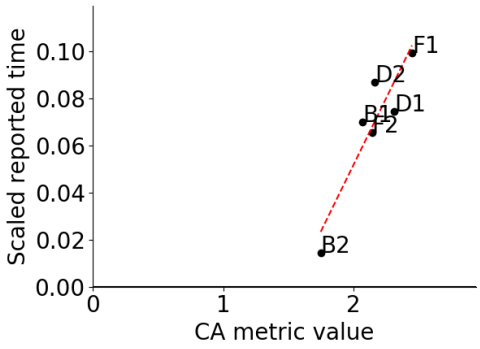

Figure 2. Scatter plot for CA metric (X3-X4, times scaled on B plus preceding extensions)

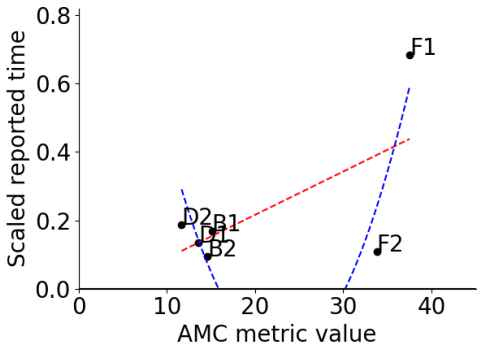

Figure 3. Scatter plot for AMC metric (X2-X3, times scaled on B plus preceding extensions)

be discussed in section 5.1. After that, we will discuss the sign in section 5.2.

5.1 Coefficient Values

When we look at the values of the coefficients, there is a small amount of statistically significant values (taking $\alpha = 0.05$) In tables 12 and 15 the p-values for both Pearson's r and Spearman's ρ are lower than 0.05 when looking at the CA metric. Tables 11 and 14 show that the p-values for Pearson's r for the CA and AMC come close to the threshold.

Furthermore, we generally see lower p-values for the coefficients over X2-X3 and X3-X4. This is the result of a smaller sample size of $n = 6$, instead of $n = 7$ over B-X1 and X1-X2. If the sample size is smaller, then the correlation has to be higher to be found statistically significant. Correlation coefficients can be interpreted using the following loose rules:

0.0 - 0.3 No correlation
0.3 - 0.5 Weak correlation
0.5 - 0.7 Moderate correlation
0.7 - 1.0 Strong correlation

Several metrics would have a strong correlation based on these rules. However, the p-values show that these are not statistically significant. Other than the CA, the RFC is the only metric with a hint towards a strong correlation across multiple result tables.

If we look at the scatter plots for the CA for X2-X3 and X3-X4 with the times scaled on B plus preceding extensions, we can see a clear linear relationship. This is visible in figures 1 and 2 by looking at the red linear regression lines.

For the ACM metrics, figure 3 shows a less clear linear relationship. However, it is not as bad as the monotonic relationship as reflected by the blue linear regression line, which partly falls below the x-axis. This is also reflected by the values of both Pearson's and Spearman's correlation coefficients: Pearson's r is higher than Spearman's ρ.

5.2 Coefficient Signs

The Sign column in table 9 - 15 show the sign of the correlation coefficients for each metric. In section 3.3 we made a prediction for these signs.

We can see that across all tables the signs are pretty inconsistent. Furthermore, tables 9, 11 and 14 have three or more signs which are inconclusive. Especially interesting is that for the significant values of the CA metric, the signs contradict each other. This is also visible in the discussed scatter plots shown in figures 1 and 2.

5.3 Literature Research

During our literature research we first searched for the original papers which define the correlation coefficients [14, 13]. We also found several works using and / or discussing one or both correlation coefficients [1, 7, 11, 15].

We also contacted assistant professor Doina Bucur and Dick Meijer from the University of Twente. They both acknowledged that Pearson's and Spearman's correlation coefficients are valid methods for our goal of finding metrics as suitable indicators of software agility. Bucur did suggest Machine Learning as an alternative. This way, the research takes multiple variables into account, instead of comparing one metric with the effort spent by developers.

6. DISCUSSION

In the previous section we found some interesting and fascinating results. In this section, we will reflect on these results.

Most notable is that the p-values across all measurements are relatively high. This means that the measured values for the correlation coefficient are generally not significant enough. Interesting is that the measurements with a sample size of $n = 7$ (tables 9, 10 and 13) showed higher p-values than the measurements with a lower sample size of $n = 6$.

This can mean that there is not any metric which could be used as an indicator for software agility. However, we did find the CA metric to be highly correlated and with a low enough p-value to be significant. This confirmed that Hollander's interest in the metric [9] was valid Yet, the found correlation was with a positive sign in two measurements (tables 12 and 15) and negative in two others (tables 11 and 14). A plausible explanation for this may be that the requirements for the extensions are a big factor in the resulting values for the correlation coefficients.

Unfortunately, leaving out the data of the C2 repository was necessary. If there was more time to resolve the dependency issue, this may have given a bit more data to get statistically significant results.

In figures 1 and 3 we can see that repository F1 is an outlier. The reported times seem too high. Therefore, we have taken a look at the descriptions given to the time administration for repository F1 over X2-X3. From those descriptions, it became clear that the developer refactored a significant part of his code to include the use of an Object-Relational Mapping (ORM) framework.

7. CONCLUSION

The goal of our research is to find out which software metrics of Object-Oriented Systems are suitable indicators of agility in software. We looked if it is possible to consider two more metrics (CA and CE) and investigated if different data consistently show suitable indicators of software agility. A small literature research focused on finding other ways to test the suitability of software metrics as indicators of software agility.

Next to CBO as metrics for the coupling property of software, CA and CE can be considered as indicators of software agility. CA was even the metric with the highest significance and highest values for its correlation coefficients. However, the results showed an opposing relationship between the CA metric and developer effort.

Where Hollander's research showed that the CAM and WMC metrics are highly correlated [9], our research pointed to the CA metric. As mentioned, this metric showed an opposing result, which means that this contributes to a higher inconsistency. This, combined with the low significance of the found values for our correlation coefficients, lets us conclude that the data over different extensions and different testbeds do not consistently show suitable indicators of software agility.

Our research also found that Pearson's and Spearman's correlation coefficients are suitable mathematical tools to find out if there is a relationship between a software metric and developer effort. However, one different method was suggested in the form of Machine Learning.

All in all, we did not find any software metrics of Object-Oriented Systems which are suitable indicators of agility in software.

.

8. FUTURE WORK

Because the scope of our research was limited, it may be interesting to consider the following considerations for future work:

- In our research, we scaled our times in two different ways to accommodate for developer skill. For a new research, we suggest investigating if there are other ways to improve this. One option could be including years of experience as a factor. Another is to categorise the developer skill based on experience with the programming language.

- We found out that using correlation coefficients only finds relationships between two variables. It would be interesting to use Machine Learning to look if there are relationships between multiple software metrics on the one side and the developer effort on the other. However, this would need a big training set of data and a validation set. This is a problem, since creating Agility Testbeds as used in our research takes a lot of time.

- Instead of looking at software metrics as indicators of software agility, future work could look at other factors. We may be able to learn from the way we commit our changes to software projects and the frequency our commits. Or it could be possible to investigate the effect of branching policies within software projects. It would be very beneficial to know how our behaviour with version control systems would affect the needed effort to create or change features in software projects.

9. RELATED WORK

Our research builds on Hollander's recent work [9], which had the same goal as our research: finding out which software metrics of Object-Oriented Systems are suitable indicators of agility in software. No other related work was found on this goal. We have performed the same measurement experiment and tooling Hollander has used, while using different data. The tooling has been adapted to accommodate for the new data while retaining the possibility to run with the Agility Testbed of 2017. We also added the functionality to run the tools on Ubuntu, since this was not the case yet.

Relevant to software metrics are Ferreira et al. [6] and Riaz et al. [5]. Furthermore, Chidamber and Kemerer [4] have defined a well-established suite for measuring software metrics.

Papers of Arifin et al. [2] and Mendes [12] are relevant because they confirm that the development effort of software can be measured by using the time reported by developers. Furthermore, M. Jørgensen [10] gave a summary of general existing knowledge about effort estimation.

10. ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Arend Rensink for his valuable input and feedback during the research project. He and David Huistra also provided me with the Agility Testbed, for which I am very grateful. This research would not have been possible without the honours students, who worked so hard to create the testbed.

Secondly, the fact I could use code from René Boschma and Han Hollander saved a lot of time in this research.

Finally, I want to thank my family and friends for helping me and supporting me during my study. It was a wonderful trip and I hope to continue it with studying for the Master Computer Science.

11. REFERENCES

[1] J. Adler and I. Parmryd. Quantifying colocalization by correlation: The pearson correlation coefficient is superior to the Mander's overlap coefficient. *Cytometry Part A*, 2010.

[2] H. H. Arifin, J. Daengdej, and N. T. Khanh. An Empirical Study of Effort-Size and Effort-Time in Expert-Based Estimations. In *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, pages 35–40. IEEE, 2017.

[3] S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[4] C. F. Chidamber, S. R., & Kemerer. *Towards a metrics suite for object oriented design*, volume 26(11). ACM, 1991.

[5] D. Datyal and A. Kaushik. A Systematic Review of Software Maintainability Prediction and Metrics. In *Third International Symposium on Empirical Software Engineering and Measurement*, volume 3(1), pages 217–219. IEEE Computer Society, 2016.

[6] K. A. M. Ferreira, M. A. S. Bigonha, R. S. Bigonha, L. F. O. Mendes, and H. C. Almeida. The Journal of Systems and Software Identifying thresholds for object-oriented software metrics. *The Journal of Systems & Software*, 85(2):244–257, 2012.

[7] J. Hauke and T. Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones Geographicae*, 2011.

[8] B. Henderson-Sellers, L. L. Constantine, and I. M. Graham. Coupling and cohesion (towards a valid metrics suite for object-oriented analysis and design). *Object oriented systems*, 3(3):143–158, 1996.

[9] J. P. Hollander. Using Software Metrics as Indicators for Agility of Object-Oriented Projects. In *29th Twente Student Conference on IT (TSCIT)*. University of Twente, 2018.

[10] M. Jorgensen. What we do and don't know about software development effort estimation. *IEEE Software*, 31(2):37–40, 2014.

[11] J. Lee Rodgers and W. Alan Nice Wander. Thirteen ways to look at the correlation coefficient. *American Statistician*, 1988.

[12] E. Mendes. Improving Software Effort Estimation Using an Expert-Centred Approach. In M. Winckler, P. Forbrig, and R. Bernhaupt, editors, *Human-Centered Software Engineering*, pages 18–33, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] K. Pearson. Mathematical contributions to the theory of evolution. III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 187:253–318, 1896.

[14] C. Spearman. Demonstration of Formulae for True Measurement of Correlation. *The American Journal of Psychology*, 18(2):161, 1907.

[15] J. H. Zar. Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association*, 1972.

APPENDIX

A. RESULT TABLES

Table 9. Coefficients B-X1, Times Scaled on B

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	0.169 (p=0.717)	-0.018 (p=0.969)	Inc.
DIT	-0.053 (p=0.910)	-0.092 (p=0.845)	-
RFC	0.083 (p=0.860)	-0.239 (p=0.606)	Inc.
AMC	-0.079 (p=0.866)	-0.092 (p=0.845)	-
CBO	-0.143 (p=0.759)	0.018 (p=0.969)	Inc.
CA	0.091 (p=0.847)	0.092 (p=0.845)	+
CE	-0.294 (p=0.523)	0.018 (p=0.969)	Inc.
CAM	-0.051 (p=0.913)	-0.092 (p=0.845)	-
LCOM	-0.076 (p=0.871)	-0.018 (p=0.969)	-
LCOM3	-0.126 (p=0.787)	-0.092 (p=0.845)	-

Table 10. Coefficients X1-X2, Times Scaled on B

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	-0.057 (p=0.903)	-0.071 (p=0.879)	-
DIT	-0.280 (p=0.544)	-0.273 (p=0.554)	-
RFC	-0.359 (p=0.429)	-0.679 (p=0.094)	-
AMC	-0.251 (p=0.588)	-0.536 (p=0.215)	-
CBO	0.040 (p=0.932)	0.357 (p=0.432)	+
CA	0.042 (p=0.928)	0.288 (p=0.531)	+
CE	-0.078 (p=0.868)	0.143 (p=0.760)	Inc.
CAM	0.392 (p=0.384)	0.143 (p=0.760)	+
LCOM	-0.278 (p=0.546)	-0.321 (p=0.482)	-
LCOM3	0.180 (p=0.699)	0.143 (p=0.760)	+

Table 11. Coefficients X2-X3, Times Scaled on B

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	-0.536 (p=0.273)	-0.200 (p=0.704)	-
DIT	-0.529 (p=0.281)	-0.116 (p=0.827)	-
RFC	-0.485 (p=0.330)	-0.429 (p=0.397)	-
AMC	0.758 (p=0.081)	0.029 (p=0.957)	+
CBO	-0.046 (p=0.931)	-0.116 (p=0.827)	-
CA	-0.776 (p=0.070)	-0.143 (p=0.787)	-
CE	0.091 (p=0.864)	-0.029 (p=0.957)	Inc.
CAM	0.586 (p=0.222)	0.086 (p=0.872)	+
LCOM	-0.395 (p=0.438)	0.029 (p=0.957)	Inc.
LCOM3	0.129 (p=0.807)	-0.600 (p=0.208)	Inc.

Table 12. Coefficients X3-X4, Times Scaled on B

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	-0.334 (p=0.518)	0.086 (p=0.872)	Inc.
DIT	-0.663 (p=0.151)	-0.543 (p=0.266)	-
RFC	-0.769 (p=0.074)	-0.667 (p=0.148)	-
AMC	-0.196 (p=0.710)	-0.200 (p=0.704)	-
CBO	0.435 (p=0.388)	0.371 (p=0.468)	+
CA	0.822 (p=0.045)	0.886 (p=0.019)	+
CE	0.279 (p=0.592)	0.371 (p=0.468)	+
CAM	0.043 (p=0.935)	-0.486 (p=0.329)	Inc.
LCOM	-0.633 (p=0.177)	-0.371 (p=0.468)	-
LCOM3	-0.593 (p=0.214)	-0.600 (p=0.208)	-

Table 13. Coefficients X1-X2, Times Scaled on B plus Preceding Extensions

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	-0.176 (p=0.705)	-0.071 (p=0.879)	-
DIT	-0.328 (p=0.472)	-0.273 (p=0.554)	-
RFC	-0.512 (p=0.240)	-0.679 (p=0.094)	-
AMC	-0.297 (p=0.518)	-0.536 (p=0.215)	-
CBO	0.177 (p=0.704)	0.357 (p=0.432)	+
CA	0.131 (p=0.780)	0.288 (p=0.531)	+
CE	0.051 (p=0.914)	0.143 (p=0.760)	+
CAM	0.433 (p=0.332)	0.143 (p=0.760)	+
LCOM	-0.338 (p=0.458)	-0.321 (p=0.482)	-
LCOM3	0.185 (p=0.692)	0.143 (p=0.760)	+

Table 14. Coefficients X2-X3, Times Scaled on B plus Preceding Extensions

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	-0.571 (p=0.237)	-0.429 (p=0.397)	-
DIT	-0.503 (p=0.309)	-0.319 (p=0.538)	-
RFC	-0.504 (p=0.308)	-0.543 (p=0.266)	-
AMC	0.789 (p=0.062)	0.143 (p=0.787)	+
CBO	-0.021 (p=0.968)	0.174 (p=0.742)	Inc.
CA	-0.771 (p=0.072)	-0.086 (p=0.872)	-
CE	0.123 (p=0.817)	0.257 (p=0.623)	+
CAM	0.560 (p=0.248)	-0.029 (p=0.957)	Inc.
LCOM	-0.374 (p=0.465)	-0.257 (p=0.623)	-
LCOM3	0.110 (p=0.836)	-0.543 (p=0.266)	Inc.

Table 15. Coefficients X3-X4, Times Scaled on B plus Preceding Extensions

metric	**Pearson's r**	**Spearman's ρ**	**Sign**
WMC	-0.321 (p=0.535)	0.086 (p=0.872)	Inc.
DIT	-0.602 (p=0.206)	-0.543 (p=0.266)	-
RFC	-0.582 (p=0.225)	-0.667 (p=0.148)	-
AMC	-0.036 (p=0.946)	-0.200 (p=0.704)	-
CBO	0.518 (p=0.292)	0.371 (p=0.468)	+
CA	0.921 (p=0.009)	0.886 (p=0.019)	+
CE	0.343 (p=0.506)	0.371 (p=0.468)	+
CAM	-0.175 (p=0.740)	-0.486 (p=0.329)	-
LCOM	-0.718 (p=0.108)	-0.371 (p=0.468)	-
LCOM3	-0.620 (p=0.190)	-0.600 (p=0.208)	-