Maintainability metrics for object oriented systems

Jonas Lindell

Mats Hägglund

Abstract

This paper will mainly focus on the metrics for maintainability proposed by Wei Li and Sallie Henry in their research Object-Oriented Metrics that Predict Maintainability [1]. There they suggest metrics that is relevant for predicting maintainability of an object-oriented (OO) system. We will analyze their work from the viewpoint of software metrics given by Fenton and Pfleeger in Software Metrics [2].

Metrics

"You cannot control what you cannot measure". Metrics are built up from measurement, for example the number of pupils in a class can be measured and said to be the metric Pupils in Class. In software industry metrics is useful in predicting cost, maintainability efforts etc., if we don't have any metrics of what we are doing, it will be if not impossible at least very hard to predict the previous mentioned attributes. There are two main categories of measurements; direct and indirect. The direct ones are e.g. the number of pupils in class whilst the indirect could be the percentage of pupils absent. If we use a metric it is important that it is widely adopted and well defined, or else we have to make an unambiguous definition so that every one interprets the metric in the same way. Software metrics measure certain aspects of software [1,112].

Maintainability

Most software is exercised repeatedly, and some of us are responsible for the upkeep of delivered software [2,354]. Maintainability is the attribute for how easy/hard and time consuming the process of maintaining the software is. For the software to be maintainable we want it to be easy to understand, enhance or correct. When analyzing maintenance we can in general terms choose between a product based or a process based approach. In this paper we will mainly focus on the product based analyzes.

The research by Li and Henry

Li and Henry conducted a case study with the goal to find OO metrics that correlated with maintainability. They have studied two commercial systems by validating the maintenance effort put in to them during three years. The systems they have studied are:

UIMS™ (User Interface System) and QUES™ (Quality Evaluation System). Both systems where built using Classic-Ada™ which is an object oriented design/programming language developed by Software Productivity Solutions Inc. The maintenance effort is measured by the number of lines that have changed in a class during the three years they have collected the measurement. Their definition of a line change is: deletion of a line, addition of a line or a change in a line. This measurement they use in their case study to estimate the maintainability of the OO systems. The metrics they decided to be used in the estimation are:

- DIT, depth in the inheritance tree. The DIT metric measures the position of a class in the inheritance tree [1,113]. In this metric the root class will get a DIT of zero.
- NOC, number of children. NOC looks at a class and measures how many direct children it has.
- MPC, message-passing coupling. Is defined as the number of send statements defined in a class.
- RFC, response for class. RFC is measured as number of local methods + number of methods called by local methods. RFC metric captures the cardinality of the response set of a class.
- LCOM, lack of cohesion of methods. Captures the measure of how many disjoint sets of local methods.
- DAC, data abstraction coupling. Captures the concept of abstract data types. And it is calculated as the number of abstract data types in a class.
- WMC, weighted method complexity. Measures the static complexity of a class [1,113]. WMC is measured as the sum of McCabe's cyclomatic complexity of every local method.
- NOM, number of local methods. Is simply measured as the number of local methods in a class.
- SIZE1, number of semicolons per class. Is the lines of code (LOC) measurement. Here defined as the number of semicolons in a class.
- SIZE2, number of methods + number of attributes. Measures the number of properties in a class.

We will list the reasons hypothesis why they chose these metrics in the following; DIT, “One may hypothesize that the larger the DIT metric, the harder it is to maintain the class” [1,113], because the higher DIT a class has the more superclass properties it may access and may hence violate the encapsulation. NOC, the more direct children a class has, the more classes it may potentially effect because of inheritance [1,113], and hence be harder to maintain. RFC, “the larger the response set for a class, the more complex the class” [1,113], and hence the higher RFC the harder to maintain because of it being more difficult to trace with a lot o method calls and responses. LCOM, “it seems logical that the more cohesive a class, the easier the class is to maintain” [1,113], so if LCOM is high it may be that the class is badly designed and partitioned and hence hard to maintain. WMC, they felt that it is logical that the more methods in a class the more complex the class is, and hence harder to maintain. NOM, “The more methods a class has, the more complex the class’s interface” [1,114]. DAC, “The more ADTs a class has the more complex the coupling of that class with other classes” [1,114].

Their goal with the research was to with the aid of statistical analyses to identify any relationship between these metrics and maintenance effort and also as well find a compact model that performs essentially as adequate job as the full model. They had two hypotheses that they tested in their statistical analyses; “The first hypothesis states that there is a strong relationship between object-oriented metrics and the maintenance effort as measured” [1,116]. “The second hypothesis states that there is redundancy among all the metrics used in the preliminary analyses section” [1,116]. The first hypothesis they tested in Analysis 1, Analysis 2 and Analysis 3, those analyses where designed to determine if the maintenance effort can be predicted from metrics and to determine if the size metrics are the sole major predictors. In their study they used “change” as the dependant variable, change is measured as the number of line changes in a class.

In analysis 1 they conducted a regression analysis using change as dependent and all the metrics as independent variables. They used the R-square and Adjusted R-square statistical methods and came to the conclusion that most of the variance in the dependant

variable change was accounted for by the metrics used in the test. “The analysis concludes that the prediction of the maintenance effort as measured by change is possible from the metrics” [1,116].

In analysis 2 they used change again as the dependant variable but this time only the two size variables as independent variables. Again the conducted a regression analysis using the R-square and Adjusted R-square methods, the conclusion they could draw from this analysis was that a large portion of the variance in maintenance effort can be predicted from size metrics. “Analysis 2 concludes that the size metrics are important predictors” [1,116].

Analysis 3 was used to determine if size metrics are the sole major predictors in the regression model. Analysis 3 they used to test the null hypothesis: “the size model is not essentially as good as the full model” or “there is no difference between the full model and the size model” [1,116]. To test the hypothesis they used an F test. They came to the conclusion that “the size metrics are not the sole predictors in the models” [1,117].

To test the second hypothesis which stated that there is redundancy among the metrics they conducted a refined analysis. To begin with they had to eliminate some of the independent variables, to determine which of the redundant predictors to be removed they used the variation inflation factor (VIF) of each predictor. They set up the criteria that no none of the used metrics in the analyses are allowed to have a VIF value > 50. To determine which of the metrics to be removed to achieve this goal they looked at correlations between the metrics as well as the ease to collect the metrics to determine which ones should be removed. They ended up removing Size1 and Size2, by removing these 2 metrics and recalculating the VIF every metrics VIF value is less than 50. And hence a compact model has been identified. Upon this compact model they conduct a regression analysis and the result was that with this model they can predict maintenance effort with a high degree of confidence, and also that the quality of the prediction is quite reliable. “We conclude that the prediction of maintenance effort is possible from the compact code model” [1,119].

As an addition they also performed a cross validation analysis of the compact prediction models (Analysis 5) It was designed to determine if the conclusion drawn from Analysis 4 is valid in the entire SPS software development environment. The null hypothesis for this analysis was:

H0:“there is no relationship between the predicted change and the observed change.” [1,119]

And the alternative hypothesis was:

H1: “there is a positive relationship between the predicted change and the observed change.” [1,119]

To test this cross validation they used a one-sided t-test. The results they got led to the rejection H0. “We conclude that the compact code model prediction equation is valid in the population” [1,119]. So the result is that the compact model is valid for the whole population not only for the specific system it was made for.

Analysis of the study

Case study

Since [1] conducted a case study we will start by analyzing the way of their work according to the guidelines given by [2]. “Before you decide on how to investigate, the first step in your investigation is deciding what you want to investigate”, “the goal for your research can be expressed as an hypothesis you want to test ” [2,120]. The major goal for the case study was to find out if there exists a relationship between OO metrics and the maintenance effort. This shows that [1] has the goal clearly defined and thereby the most essential attribute for a case study is fulfilled. [1] stated two hypotheses to be tested in the statistical analyzes (described before) so the goal has been expressed in the way it should.

According to [2] every case study requires conception, design, preparation, execution, analysis, dissemination, and decision making. Above it was shown that the conception was clear. Design means that you translate the objective into a formal hypothesis, which is usually done by dividing it into two different hypotheses, the null hypothesis and the

alternative hypothesis. The general hypothesis by [1] does not follow this proposed standard since no null hypothesis was defined, but their study was divided into five different analyzes and within these the null hypothesis was used. The case study was obviously going to confirm that a relation exists between metrics and maintainability so no null hypotheses were given for general hypotheses. It might have been a good idea to also include a null hypothesis so that proposed procedures are followed, e.g. one null hypothesis could have been: there is no relation between OO metrics and maintainability.
To the preparation part belongs that they have contacted a software company and received permission to use their software for the study. It is hard to comment on their execution since it is not described thoroughly in the paper, but to some extent it is clear that execution has been done if we look into the fact that data about maintenance has been collected for about 3 years. Analysis part was to some extent left out in the research, especially the part of making sure that measurements taken are valid and useful. The second part of analysis is to analyze the sets of data by using statistical principles. This was done and the results were given in the paper for each of their five different analyzes. In order to make outcome of the experiment available in a useful form, the last stage of analysis is to make descent documentation and conclusions. In the case study information about how the analyses were performed as well as the data collected is presented so anyone could redo study and come to the same conclusion. So they have fulfilled the last part: “Dissemination and decision making” and end result will be available for other interested. Hence we feel they have fulfilled the requirements on a case study given by [2].

Metrics reduction

In the research it is mentioned that besides the main goals of the statistical analysis, finding a compact model that performs essentially as adequate a job as the full model is a desirable property. [2] discuss this matter concerning measurement and how developers report dissatisfaction because they sometimes feel overwhelmed with data. This has lead to efforts trying to reduce large numbers of measures in a meaningful way. Principal component analysis [2,219] is used in some studies to reduce the dimensionality of many related metrics to a smaller set of “principal components”, while retaining most of the

variation observed in the original metrics. In the journal [1] analyses are divided into preliminary analyses and refined analyses to investigate this matter. The full model consisted of all the metrics explained earlier: DIT, NOC, MPC, LCOM, RFC, DAC, WMC, NOM, SIZE1 and SIZE2. To identify the sole major predictors for the change different sets of independent variables was chosen to distinguish if same result of the analysis could be retrieved with a reduced model. To accomplish this, the variation inflation factor (VIF) was used as well as determining the ease of collecting the metrics and previous research result showing correlations between metrics. The way of reducing numbers of measures in the case study was not exactly done as suggested by [2], but none the less the course of action was stated clearly.

Aspects on chosen metrics

"Formally, we define measurement as a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute" [2,28]. Another aspect on a measurement is that it should be well defined and meaningful for our analysis. Are the metrics that [1] has chosen well defined, meaningful and do they map the empirical world to the formal so that the measure assigns a number or symbol to an entity?

We begin by examining DIT (depth in inheritance tree), this measure [1] maps to an integer ranging from 0 to N, where N is a positive integer. What DIT captures is the level of inheritance that a class has, so the superclass has a DIT of 0, the lower down in the inheritance tree a class is the higher it's DIT. The measure we feel is well defined and maps a number to an entity in the empirical world. The measure can be argued to be a valid measure for the analysis since one can ponder that the more inheritance a class has the more properties of a classes above it can access, and hence become harder to maintain. DIT is also mentioned as an OO metric by [2,318] .

Next we have NOC, Number of direct Children. Quite similar to DIT, but NOC captures the idea that when changing something in a superclass it can affect the child classes in a none desirable way. Hence NOC will affect the maintainability of a class, which also makes this a valid measurement in analyzing maintainability. NOC is also mentioned as an OO metric by [2,318] .

MPC message-passing coupling. This measure captures the property of how many method calls to other classes there is in a class. Since higher coupling leads to higher complexity it will also make the class more difficult to maintain and understand. So this metric is essential when analyzing maintainability.

LCOM lack of cohesion metric. A meaningful measurement in the sense that a poorly defined class and without clearly defined substance will be harder to maintain. Hence it is a meaningful metric. LCOM is also mentioned as an OO metric by [2,319] .

RFC, response for class. Same idea as MPC but also contains the number of local methods. Since it also takes size into consideration it captures the idea that a class containing more methods makes maintenance more complex, especially if it consists of a large number of method calls. RFC is also mentioned as an OO metric by [2,318] .

DAC, data abstraction coupling. The structure of an abstract data type should not have critical properties available for any other class in a way that it can alter these in an incorrect way. This is what [1] is worried about and hence considering this to be a crucial metric. We feel that this maybe is not that relevant metric for maintainability as long as encapsulation for the abstract data types holds.

WMC, weighted method complexity is the summation of McCabe's cyclomatic complexity measure for each local method. Since McCabe captures the complexity of a program [2,293] it also can be considered to affect the maintainability in the way that higher complexity renders more difficult maintainability. WMC is also mentioned as an OO metric by [2,318] .

NOM, number of local methods. More methods means more to maintain and more parts that can break. This was partly covered by RFC.

SIZE1, number of semicolons in a class. This captures LOC which in this sense is clearly defined. The greater size the more to maintain which means it affects maintainability.

SIZE2, number of attributes + number of local methods. The greater size the more to maintain which means it affects maintainability.

In the reduced model SIZE1 and SIZE2 was removed and it is quite obvious since these are already more or less covered by other metrics.

Other measures

Here we give some measures that could affect maintainability but is not covered by the study.

Comment Percentage. It can be reasoned that the comment percentage of the code can predict maintainability, since comments will make the code easier to understand and also to maintain. The Comment Percentage metric can be calculated as the total number of comments divided by the total number of code less the number of blank lines [3,3]. “The SATC has found a comment percentage of about 30% is most effective. Since comments assist developers and maintainers, this metric is used to evaluate the attributes of Understandability, Reusability, and Maintainability” [3,3].

References

1. *Object-Oriented Metrics that Predict Maintainability*, Wei Li and Sallie Henry Journal on Systems Software 1993;23:111-122
2. *Software Metrics Second Edition Revised Printing,* Norman E. Fenton and Shari Lawrence Pfleeger. PWS PUBLISHING COMPANY ISBN: 0-534-95425-1
3. *Software Quality Metrics for Object-Oriented Environments* Linda H. Rosenberg and Lawrence E. Hyatt. http://satc.gsfc.nasa.gov/support/CROSS_APR97/oocross.PDF