Journal of Scientific Research and Development 2 (13): 52-58, 2015
Available online at www.jsrad.org
ISSN 1115-7569

Empirical validation of object-oriented software metrics as changeability indicators

Amos Orenyi Bajeh *, Shuib Basri, Low Tang Jung, Faizal Ahmad Fadzil

Department of Computer and Information Sciences, University Technology PETRONAS, Tronoh, Malaysia

Abstract: Changeability is a desirable attribute of software product since they are subject to modifications throughout their life cycle. The measurement of this attribute at early stage of software development can facilitate the development of software products which can be easily modified during maintenance. The changeability/modifiability measurement models proposed in the literature are subjective with varied set of metrics and other subjective factors such as regression coefficients. Subjective models yield disparate results leading to different conclusion about the quality of the same software product. Thus there is a need to develop an objective measurement model for changeability. As a first step towards developing an objective model for changeability, this paper presents an empirical validation of four Object-Oriented High Level Design metrics, individually validated using data collected from real world software products undergoing maintenance.

Key words: Object-oriented software; High-level design; Software metrics; Empirical validation

1. Introduction

Measurement is one of the crucial aspects of all engineering disciplines. It facilitates the management of both the engineering process and the engineering of quality into final products. If software engineering must be considered as an engineering discipline, software metrics must be robust and valid both theoretically and empirically (Sheldon et al., 2002).

Software development process artifacts developed at the design and implementation phases can be measured to determine their attributes. The design phase produces the initial artifact often referred to as High-Level Design (HLD) which basically depicts the skeletal architecture of the software been developed. The implementation phase produces the Low-Level Design (LLD) which is the translation of the HLD into computer source code with more detail provided using a programming language. To facilitate the development of software products with good external quality attributes such as changeability, the measurement of the early software artifact, the HLD, is beneficial. This provides early information about the quality of the product been developed and thus serve as early indicator for the need to refine software design in order to improve the software product that will be eventually produced after implementation.

Changeability is defined in the ISO/IEC 9126 (SQuaRE) standard as the capability of the software to be modified (ISO/IEC, 2001, 2015). This is a very important quality attribute that software product need to have because it is a measure of the level of difficulty or effort required to make necessary changes to software as they evolve over time. For software to be maintainable and remain operational, it must be modifiable.

Given that changeability is an external quality attribute (Genero et al., 2005) that cannot be directly measured from the software, there is a need to identify metrics that are directly measurable from the software and that can serve as good indicators of the changeability of the software. In other words, product metrics need to be empirically validated in order to identify those that can be used as indicators of changeability. This paper presents an empirical validation of some Object-Oriented (OO) software HLD metric as changeability indicators.

The remaining part of this paper is organized as follows: Section 2 discusses related works that use OO metrics in the development of changeability measurement models. Section 3 presents the research methodology used for the empirical validation of the HLD metrics considered in this study. The results of the empirical validation are presented and discussed in Section 4. Section 5 presents a conclusion of the paper and highlight some future works relating to this study.

2. Related works

Studies have been conducted on the measurement of software changeability/modifiability of both software HLDs and LLDs. This section present some of the notable studies reported in the literature. Some of the studies are wholly on changeability/modifiability while others have changeability/modifiability as part of their study or proposed measurement model.

* Corresponding Author.

Sheldon et al. (2002) proposed a modifiablity measurement metric, Average Modifiability (AM), as part of OO inheritance hierarchy maintainability estimation metric. The metric is defined as follows:

$$AM = AU + \frac{\sum_{i=1}^{t}(SUCC(C_i)/2)}{t} \quad (1)$$

where AU is Average Understability of inheritance hierarchy and it is defined as:

$$AU = \frac{\sum_{i=1}^{t}(PRED(C_i)+1)}{t} \quad (2)$$

$SUCC(C_i)$ is the number of classes that succeed or inherit from class C_i.

$PRED(C_i)$ is the number of classes that precede class C_i or number of classes from which class C_i inherits it properties

Antonellis, Antoniou, Kanellopoulos, Makris, Theodoridis, Tjortjis, and Tsirakis, (2007) proposed a maintainability measurement model in which changeability is one of the attributes of maintainability. Changeability, as other attributes, was measured from nine OO metrics: 4 LLD metrics and 5 HLD metrics. These metrics were arbitrarily chosen and linked to the attributes without any empirical validation of the metrics to ensure that they are good indicators of changeability and other attributes.

Kiewkanya, Jindasawat, and Muenchaisri (2004) proposed a set of regression based modifiability measurement model using weighted-predicted-level technique. The models are composed of OO class diagram and sequence diagram metrics. Some of the OO HLD class diagram metrics in the models are Number of Class (NC), Average Number of Association (ANAssoc), Number of Aggregation Hierarchy (NAggH), Maximum Aggregation Hierarchy (MaxHAgg), Number of Generalization Hierarchy (NGenH) and Maximum Depth of Inheritance (MaxDIT).

Genero, Manso, Visaggio, Canfora, and Piattini (2007) proposed two separate modifiability measurement models developed using data collected from two controlled experiments with students as subject. The two models are regression equations composed of NC, Number of Method (NM), MaxDIT and NGenH. Thus, this implies that these metrics are good indicators of modifiability.

Rizvi and Khan (2010) used regression analysis, an exploratory approach, to develop a modifiability measurement model for OO HLD. The model comprises NC, Number of Generalization (NGen), NGenH, NAggH and MaxDIT metrics. Thus, this also implies that these metrics are good indicators of modifiability.

These regression based changeability/modifiability measurement model are subjective to the set of software used in developing the models. The set of metrics and the regression coefficients in the models varies as a result of the different software used in their development. Due to their subjective nature, when these models are used to measure the same software, they yield disparate results which lead to different conclusion about the quality of the same software (Lincke Gutzmann and Löwe, 2010)

The concept of change impact has been used in developing changeability measurement metrics. Change impact entails the measurement of the potential extent to which a change to a component or element of a design affects other components of the design often due to interrelationships between the components. In other words, it is a measure of the ripple effect that a change has on a design. Ajrnal, Kabaili, Keller, Lustman, and Saint-Denis (2000) used this concept to propose a method for measuring the impact of change. For instance, the expression $impact(Cl_i, Cl_j) = S.H' + G$ implies that the impacted classes are those associated (S) with but not inherited (H') from Cl_i, or those aggregated (G) with Cl_i. In this form of expression, S represents association, G represents aggregation and H represents inheritance. Other symbols are I representing Invocation, L representing Local impact and F representing Friendship-for C++ friendship concept. Based on this form of expressions, the authors conducted an empirical study and identified CBO_IUB (Coupling between Object Is Used By) and WMC (Weighted Method per Class) as good indicator of changeability. These are OO LLD metrics.

Briand, Wust, and Lounis (1999) also identified the following LLD metrics as good indicators of change ripple effect: CBO (Coupling between Objects), PIM (the number of method invocation in class A of methods in class B, taking polymorphism into account) and INAG (Indirect Aggregation coupling).

In this paper, we argue that there is a need to first of all validate the individual metric to show that each of them is empirically valid as good indicator of changeability. The use of regression analysis to develop measurement models is exploratory and thus subjective to the attributes of the software from which the data for the regression analysis are collected from. This can be seen in the different regression model presented above; they have different set of OO metrics and the regression coefficients in the models are determined from the set of software used in the analysis. This assertion is corroborated by the result of the study reported by Lincke et al. (2010) in which different regression models give different measurement values for the same software product and these values lead to different conflicting conclusions about the quality of the measured software. A more objective model can be developed by, first, empirically validating the OO metrics to identify each of them as suitable indicators of changeability. Thereafter the metrics can be objectively aggregated to form an objective changeability measurement model. This paper focuses on empirical validation of OO HLD metrics that are measurable from UML class diagram, which is the de facto diagram for OO software design

architecture. The HLD metrics will give early indication of the level of changeability of OO software design thereby facilitating the engineering of changeability quality into software design.

3. Methodology

This section presents the OO HLD metrics validated, the tested hypothesis and the approach used for the empirical validation. The approach used for the empirical validation of OO HLD metrics is the case study method of empirical software engineering research. This method allows the study of real world software products that are undergoing maintenance at hand. Thus, the changes committed to the software are realistic modifications. The software samples for the empirical validation were collected from GitHub software repository (GitHub). Empirical data are extracted from the software by using a java-based measurement tool developed for this study.

3.1. High-level design metrics

Given that OO HLD depicted in UML class diagrams provide limited detail about the software i.e. UML class diagrams shows only classes, their attributes and method declarations and interclass relationship such as association, dependency and inheritance; the concept of change impact discussed in Section 2 will be suitable for the identification of HLD metrics that can be empirically validated as changeability indicators. The concept is based on the interrelatedness of design components. Components that are interrelated are most likely going to be affected by change(s) to any of the components. Thus the OO HLD metrics that measures the interrelationships between the classes of a software design is considered for empirical validation as changeability indicators. Table 1 presents the OO HLD metrics that measure the various dimensions of interclass relationship that can be measured from HLD.

Table 1: OO HLD Metrics

Metric	Description
Number of Association ($NumAssoc$)	This is the count of the number of association relationship in a class diagram. It is the same as NAssoc defined by Genero et al. (2000; 2002; 2005)
Number of Dependency ($NumDep$)	This is the number of dependency relationship in a class Diagram. It is the same as NDep defined by Genero et al. (2000; 2002; 2005)
Number of Generalization ($NumGen$)	This is the count of the number of parent-child relationship in a class diagram. Same as NGen by Genero et al. (2000; 2002; 2005)

Also considered for empirical validation is the Sheldon et al.'s modifiability metric AM defined in equation (1). This metric measures the modifiability of inheritance hierarchy in a design but have not been empirically validated in the literature.

3.2. Changeability response variable

In order to validate the OO HLD metrics identified above, a metric that can serve as a surrogate measure of the level of difficulty of modifying a design needs to be devised. Using the concept of change impact, the difficulty of making changes to a design gets higher as the number of impacted components increases. That is, the higher the ripple effect of a change, the more difficult to effect the change on the software because the number of components potentially affected by the change increases and thus, the number of components to examine for possible regression error(s) increases.

In GitHub software repository, changes made to software are stored as issues and there are two categories of issues: open issues and closed issues. Open issues are reported issues that have not been addressed while closed issues are issues that have been addressed in the software by making necessary changes to the software. Each closed issue in the software repository, consist of the files that have undergone changes as a result of addressing the issue. The number of files affected gives a measure of the ripple effect of the change made to the software. Thus, this number of file is used as the response variable for changeability and it is referred to as Number of Co-changed class files ($NumCoC$) in this study. The $NumCoC$ is the count of the number of java source files that have been modified as a result of addressing a closed issue in the repository. Figure 1 shows the UML class diagram of an instance of a closed issue collected from the GitHub repository. Each closed issue contains classes that co-changed when the necessary changes were made on the software to address a reported issue.

3.3. Hypothesis formulation

The empirical validation of the OO HLD metrics is to investigate the relationship between the metrics and the response variable, $NumCoC$. It seeks to empirically observe the level of significance of the interclass relationship in determining the number of classes that co-changed as a result of making changes to software components. Thus, for each of the metrics defined in Table 1 and the modifiability metrics AM, the following hypothesis is tested: ***There is a significant and positive empirical relationship between the OO HLD metrics and the number of co-changed classes (NumCoC) in closed issues***.

A positive correlation between the metric and the $NumCoC$ implies that as the metric increases in value, the difficulty in making changes to a design increases as well. The higher the number of association, dependency and inheritance in a design the higher the impact of change and thus the more difficult it

becomes to successfully modify a design without regression errors due to changes. In other words, software maintainers will have to examine more classes for ripple effect of change(s) made to a component if the component has more association, dependency and inheritance relationships with other classes in the design.

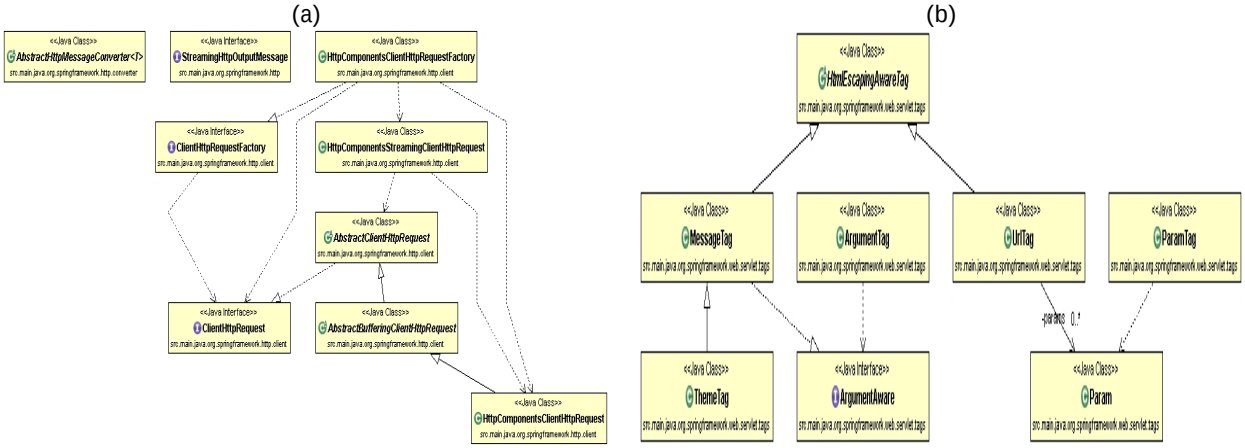

Fig. 1: UML class diagrams of closed issues (a) A closed issue with nine co-changed java files (b) A closed issue with eight co-changed java files.

3.4. Data collection

The closed issues of three real world open source OO software undergoing maintenance are collected from the GitHub software repository. This software is of different sizes and different functional domains. Table 2 presents a description of the three software samples used for the empirical validation of the OO HLD metrics. The numbers of closed issues for each of the software samples in the table are the available information as at the time the closed issues were collected for this study.

Table 2: Open source object-oriented software sample

Software	Description	No. of Closed Issues
J Unit	A framework for the development of repeatable tests for java classes. It is an instance of the xUnit architecture for unit testing frameworks. (http://junit.org/ or https://github.com/junit-team/junit)	665
Spring-framework	The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications on any kind of deployment platform. (https://github.com/spring-projects/spring-framework)	319
elastic search	A distributed RESTful search engine built for the cloud. Some of its features include: • Distributed and Highly Available Search Engine. • Each index is fully shared with a configurable number of shares. • Read / Search operations can be performed on either one of the replica shard. (https://github.com/elasticsearch/elasticsearch/)	4,308

Not all closed issues are suitable for the empirical validation. Some of the closed issues involve changes to non-source code (non-java) files and some changes are merely program comments addition or removal. Such closed issues can serve as a threat to the validity of the empirical results because they can constitute noise in the data. In order to mitigate possible threats to the validity of the empirical validation, the following selection criteria are used to collect the closed issues from their *GitHub* repositories:

- The closed issue must involve change(s) to the java class files: this is to ensure that the changes made to the software in the closed issues actually affect the source code files and not only other file types such as xml or doc files.
- The change(s) on the java files must be made on the executable lines of code and not program comments and addition/removal of blank lines:

changes to comments and addition/removal of blank lines are not due to software design defects but just internal documentation and code readability.

- Closed issues with newly added program files (classes) are not collected for the study. This is to ensure that the measured class relationships precede the changing of the software, and thus to ensure internal validity of the study
- More than one java class file must be involved in the change(s): HLD change impact requires two or more classes affected by change(s) made to one or more of the classes, therefore there should be more than one java class file changed in each of the selected closed issues.

A java-based measurement tool was developed for the automatic measurement of the OO HLD metrics defined in Table 1 and the Sheldon et al.'s average modifiability metrics, AM.

4. Results and discussion

This section presents the results of the measurement of the OO HLD metrics from the closed issues of the software samples and the correlation analysis to test the hypothesis. The results are then discussed.

4.1. Results

Table 3, Table 4 and Table 5 present the descriptive statistics of the OO HLD metrics collected from the closed issues of the three software sample J Unit, Spring-framework and elastic search respectively, using the java-based measurement tool developed for this study.

Table 3: Descriptive statistics of metrics for J Unit software

Metric	Min	Max	Mean	Median	Mode	Std
NumAssoc	0	22	1.05	0	0	2.71
NumDep	0	13	0.84	0	0	2.19
NumGen	0	6	0.31	0	0	0.90
AM	0	14.08	2.49	1.75	1.75	2.42
NumCoC	2	47	4.81	3	2	5.67

Table 4: Descriptive statistics of metrics for Spring-framework software

Metric	Min	Max	Mean	Median	Mode	Std
NumAssoc	0	5	0.52	0	0	0.96
NumDep	0	24	0.97	0	0	3.20
NumGen	0	11	0.82	0	0	1.97
AM	0	27.40	3.78	2.2	1.75	3.92
NumCoC	2	50	5.7767	3	2	7.02

Table 5: Descriptive statistics of metrics for elastic search software

Metric	Min	Max	Mean	Median	Mode	Std
NumAssoc	0	11	0.52	0	0	1.23
NumDep	0	43	1.88	0	0	4.97
NumGen	0	37	1.10	0	0	3.52
AM	0	27.95	6.122	4.52	3.5	4.25
NumCoC	2	45	5.87	4	2	5.82

In order to determine the type of correlation analysis to apply on the empirical data, the statistical distribution of each of the metrics was determined using the Kolmogorov-Smirnov (K-S) test for normality in MatLab. The K-S test for each and every metric returned an h value of 1. This implies that the metrics distributions do not follow a normal distribution. This is typical of software metric distribution. Thus, the non-parametric statistical correlation test is applied on the empirical data.

The descriptive statistics of all metrics of the three software samples showed that there are ties in the data given that the min and median values of majority of the metric are the same or very close in value with a lot of zeros. Thus, the Kendal Tau's non-parametric test was used for the correlation analysis. Table 6 presents the correlation of each of the OO HLD metrics with the changeability response variable, $NumCoC$.

Table 6: Result of the correlation analysis of the metrics

Metric	J Unit		Spring-framework		elastic search	
	rho	*pval*	*rho*	*pval*	*rho*	*pval*
NumAssoc	0.403	2.68E-07	0.296	0.000394	0.439	2.86E-14
NumDep	0.522	4.59E-11	0.495	3.21E-09	0.559	8.26E-24
NumGen	0.514	2.08E-10	0.553	2.70E-11	0.531	1.30E-20
AM	0.485	1.09E-10	0.466	4.28E-10	0.671	1.30E-40

4.2. Discussion

The results in Table 6 show significant and moderately strong relationships between the change impact metrics or response variable $NumCoC$, and each of the OO HLD metrics: $NumAssoc$, $NumDep$, $NumGen$ and AM. All the metrics have a p-value less than 0.05 which implies that the level of confidence on the observed correlation is above 95%. Thus, for these metrics the formulated hypothesis is supported by the empirical data from the software samples. This implies that there is a significant correlation between change impact and each of the three interclass relationships: association, dependency and generalization relationships.

The Sheldon et al.'s (2002) modifiability metric for inheritance hierarchy, AM has a significant and moderately strong correlation ship with the $NumCoC$ metric. Thus, the empirical data collected from the software samples also support the formulated hypothesis for AM.

These results corroborate the findings of study reported in Genero, Piattini, and Calero (2002) which state that the UML class diagram metrics are good indicators of changeability. However, unlike in their study where a moderately strong correlation is observed between the dependency metric (NDep) and modifiability, a stronger correlation is observed between the dependency metric (NumDep) and the change impact metric $NumCoC$ in our study. As stated by the author, the correlation observed in their study could be as a result of the low variance of the dependency metric in the software sample that they used. The dependency metric in our study has a higher variance which explains the stronger correlation that we observed.

The OO HLD metrics have positive correlation with the response variable, $NumCoC$. This implies that the impact of change increases when the HLD metrics increase in value. Since increase in change impact implies more difficulty in modifying the software and as the difficulty of modification increases the changeability of the software decreases, it implies that the validated OO HLD metrics are negatively correlated with changeability. That is, when the values of these metrics increase, the changeability of the software been measured decreases.

We believe that this type of empirical validation of individual metrics should be a prerequisite for the development of measurement models for external quality attributes such as changeability. Establishing the suitability of each metric before aggregating the suitable ones will lead to the development of objective measurement models. This is in contrast with the common approach in the literature where exploratory methods such as regression analysis are used. Exploratory approach leads to subjective models which are dependent on the characteristics of the set of software used in the development of the models.

5. Conclusion and future works

This paper presented an empirical validation of four OO HLD metrics that are suitable for measuring the changeability attribute of OO HLDs. The OO HLD metrics are measurable from UML class diagrams, the de facto diagram for representing the architecture of OO software. HLD artifacts consist of limited information about the software since they are the skeletal representation of the software at the design phase of software development. Thus, the concept of change impact is used in identifying the OO HLD metrics for the empirical validation conducted in this study. The investigated metrics are metrics that measure the interclass relationship: association, dependency and generalization. Also, a modifiability metrics AM proposed by Sheldon et al for measuring the modifiability of OO inheritance hierarchy but not empirical validated is validated in this paper.

The case study method of empirical software engineering research was employed for the investigation of the suitability of the four metrics as changeability indicators. As a result, the four metrics showed a very significant relationship with the surrogate metrics used to represent the difficulty of modifying software. The level of significance of the observed relationship between the four OO HLD metrics and the surrogate metric is above 95% - the de facto threshold for confidence level.

We argue that the development of measurement models for software external quality attributes (e.g. changeability, maintainability, etc.) requires the initial validation of individual attribute measurement metrics in order to identify the metrics that are suitable for measuring the external attribute. Thus, as a first step towards proposing an objective changeability measurement model, four OO HLD metrics (NumAssoc, NumDep, NumGen and AM) are empirically validated individually as changeability indicators.

Further study will be carried out to investigate the empirical validity of OO HLD cohesion and encapsulation metrics such as Cohesion Among Methods of a Class ($CAMC$) (Bansiya and Davis, 2002; Bansiya, Etzkorn, Davis, and Li, 1999), Similarity based Class Cohesion (SCC) (Al Dallal and Briand, 2010), Number of Attribute ($NumAttr$) and Number of Method ($NumMeth$). Thereafter, all the empirically valid OO HLD metrics will be objectively aggregated to form a measurement model for changeability. Also, we intend to investigate the empirical validity of OO LLD metrics using real world software products as research context and also develop an objective changeability measurement model for OO software LLDs.

References

Ajrnal Chaumun, M., Kabaili, H., Keller, R. K., Lustman, F., & Saint-Denis, G. (2000). Design

properties and object-oriented software changeability. *Software Maintenance and Reengineering, 2000. Proceedings of the Fourth European*, 45-54.

Al Dallal, J., & Briand, L. C. (2010). An object-oriented high-level design-based class cohesion metric. *Information and Software Technology, 52(12),* 1346-1361.

Antonellis, P., Antoniou, D., Kanellopoulos, Y., Makris, C., Theodoridis, E., Tjortjis, C., & Tsirakis, N. (2007). A data mining methodology for evaluating maintainability according to ISO/IEC-9126 software engineering–product quality standard. *Special Session on System Quality and Maintainability-SQM2007*.

Bansiya, J., & Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *Software Engineering, IEEE Transactions on, 28(1)*, 4-17.

Bansiya, J., Etzkorn, L., Davis, C., & Li, W. (1999). A class cohesion metric for object-oriented designs. *Journal of Object-Oriented Programming, 11(8)*, 47-52.

Briand, L. C., Wust, J., & Lounis, H. (1999). Using coupling measurement for impact analysis in object-oriented systems. *Software Maintenance, 1999. (ICSM'99) Proceedings. IEEE International Conference on*, 475-482.

Genero, M., Manso, E., Visaggio, A., Canfora, G., & Piattini, M. (2007). Building measure-based prediction models for UML class diagram maintainability. *Empirical Software Engineering, 12(5)*, 517-549.

Genero, M., Piattini, M., & Calero, C. (2002). Empirical validation of class diagram metrics. *Empirical Software Engineering, 2002. Proceedings*. 2002 International Symposium n, 195-203.

Genero, M., Piattini, M., & Calero, C. (2005). A survey of metrics for UML class diagrams. *Journal of Object Technology, 4(9)*, 59-92.

GitHub.GitHub software repositories. Visited in Nov. 14, Retrieved from https://github.com/

ISO/IEC. (2001). ISO/IEC 9126-1 standard, software engineering, product quality, part 1: Quality model. Geneva,

ISO/IEC. (2005). Software engineering---software product quality requirements and evaluation (SQuaRE), ISO/IEC25000.

Kiewkanya, M., Jindasawat, N., & Muenchaisri, P. (2004). A methodology for constructing maintainability model of object-oriented design. *Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on*, 206-213.

Lincke, R., Gutzmann, T., & Löwe, W. (2010). Software quality prediction models compared. *Quality Software (QSIC), 2010 10th International Conference on*, 82-91.

Sheldon, F. T., Jerath, K., & Chung, H. (2002). Metrics for maintainability of class inheritance hierarchies. *Journal of Software Maintenance and Evolution: Research and Practice, 14(3)*, 147-160.