Master thesis in Software Engineering

The impact of design complexity on software cost and quality

European Master in Software Engineering

Kaiserslautern, Germany, 2010

Nguyen Duc Anh

Technical University of Kaiserslautern

Supervisors:
Prof. Dr. Dr. h.c. H. Dieter Rombach
Marcus Ciolkowski

Technical University of Kaiserslautern

EMSE Co-supervisors:

Sebastian Barney

Blekinge Institute of Technology

Author's Declaration

I hereby certify that all of the work described within this thesis is the original work of the author. Any published (or unpublished) ideas and/or techniques from the work of others are fully acknowledged in accordance with the standard referencing practices. I understand that my thesis may be made electronically available to the public.

September, 2010

Nguyen Duc Anh

Abstract

Context: Early prediction of software cost and quality is important for better software planning and controlling. In early development phases, design complexity metrics are considered as useful indicators of software testing effort and some quality attributes. Although many studies investigate the relationship between design complexity and cost and quality, it is unclear what we have learned from these studies, because no systematic synthesis exists to date.

Aim: The research presented in this thesis is intended to contribute for the body of knowledge about cost and quality prediction. A major part of this thesis presents the systematic review that provides detail discussion about state of the art of research on relationship between software design metric and cost and software quality.

Method: This thesis starts with a literature review to identify the important complexity dimensions and potential predictors for predicting external software quality attributes are identified. Second, we aggregated Spearman correlation coefficients and estimated odds ratios from univariate logistic regression models from 59 different data sets from 57 primary studies by a tailored meta-analysis approach. At last, it is an attempt to evaluate and explain for disagreement among selected studies.

Result: There are not enough studies for quantitatively summarizing relationship between design complexity and development cost. Fault proneness and maintainability is the main focused characteristics that consume 75% total number of studies. Within fault proneness and maintainability studies, coupling and scale are two complexity dimensions that are most frequently used. Vote counting shows evidence about positive impact of some design metrics on these two quality attributes. Meta analysis shows the aggregated effect size of Line of code (LOC) is stronger than those of WMC, RFC and CBO. The aggregated effect sizes of LCOM, DIT and NOC are at trivial to small level. In subgroup analysis, defect collections phase explains more than 50% of observed variation in five out of seven investigated metrics.

Conclusions: Coupling and scale metrics are stronger correlated to fault proneness than cohesion and inheritance metrics. No design metrics are stronger single predictors than LOC. We found that there is a strong disagreement between the individual studies, and that defect collection phase is able to partially explain the differences between studies.

Keywords: Design metric, Design complexity, Software measurement, Meta-analysis, Vote counting, Systematic review, Fault proneness, Maintainability.

Acknowledgements

This thesis would not have been possible without the sincere help and contributions of several people. I would like to use this opportunity for expressing my sincere gratitude to them.

First of all, I would like to thank my thesis supervisors Prof. Dr. Dr. h.c. H. Dieter Rombach and Marcus Ciolkowski for offering me an interesting topic and providing me invaluable guidance, continuous supports and advices throughout the thesis. I am also thankful to my EMSE co-supervisors Dr. Richard Torkar and Sebastian Barney from the Blekinge Institute of Technology for their reviews and thoughtful advices. Besides, I would like to express my honest appreciation to Michael Klaes from department Process and Measurement, Fraunhofer Institute for Experimental Software Engineering (IESE) for fruitful discussions and helpful comments.

Moreover, I would like to thank department Process and Measurement, Fraunhofer IESE for giving me the opportunity to conduct the thesis in an industrial context. The thesis was supported by the BMBF project SPES 2020 (Grant No. 01IS08045, IKT 2020) and cooperation with Robert Bosch, Germany.

Last but not least, I am deeply grateful to my family in Vietnam and EMSE friends for their support, encouragement and to go through all the challenges I faced.

Table of Contents

List of Figures

List of Tables

Preface

Es ist eigentlich rätselhaft, was einen antreibt, die Arbeit so verteufelt Ernst zu nehmen. Für wen? Für sich? – Man geht doch bald. Für die Mitwelt? Für die Nachwelt? Nein, es bleibt rätselhaft.

(A. Einstein)

…a physical theory is just a mathematical model and ... it is meaningless to ask whether it corresponds to reality. All that one can ask is that its predictions should be in agreement with observation.

(S. Hawking)

This page intentionally left blank.

Chapter 1
Introduction

Software metrics, as a tool to measure software development progress, has become an integral part of software development as well as software research activities. Referring to the famous statement of Tom DeMarco [5], "You cannot control what you cannot measure". Measurements are – as in any other engineering discipline – also in software engineering a cornerstone for both improving the engineering process and software products. Measurement not only helps to visualize the abstraction of software development process and product but also provide an infrastructure to perform comparison, assessment and prediction of software development artifacts.

The large part of software measurements is to, in one way or another, measure or estimate software complexity due to its importance in practices and research. As computers software grows more powerful, the users demand more reliable and powerful software, software became larger and seemingly more complex; the needs for a controlled software complexity and software development process emerged. Software complexity has been shown to be one of the main contributing factors to the software development and maintenance effort [45]. Most experts today agree that complexity is a major feature of computers software, and will increasingly be in the future. Measurement of software complexity enhances our knowledge about nature of software and indirectly assesses and predicts final quality of the product.

Among complexity metrics, there are also two types of complexity, namely code complexity and design complexity. Code complexity measures the complexity of the source code in a software product while design complexity measures the complexity of design artifacts, such as design diagram. Design complexity metrics are becoming more and more popular due to the overwhelming of object oriented paradigm in practices nowadays [22]. The primary advantage of design complexity metrics over code complexity metrics is that they can be used at design time, prior to code implementation. This permits the quality of designs to be analyzed as the designs are developed, which allows improvement of the design prior to implementation. While classic code complexity metrics are well-studied in a long period of time, design complexity appears quite recently and leaves much of room for research.

Among studies about design complexity there are two major tendencies. One focuses on defining new metrics or method to capture better different aspects of design complexity. Papers that focus comparison between complexity metrics, complexity measurement framework, measurement tool or comparison among metrics also belong to this category. The second trend

composes studies that empirically validate design metrics by investigating its relationship to other software attributes. As true for all software metrics, design complexity metric is only meaningful if it provides an indication of important attributes like cost and quality. The software metric literature have shown many case studies of successful usage of design metrics as a predictor of others software attributes, such as software development costs [1], software reliability [3], or maintainability [7].

There is a large number of design complexity metrics have been proposed in literature. However, it is not obvious to select appropriate metric set that can predict given software attributes. Many different dimensions of complexity are measured as well as many metrics capture the same complexity dimensions. Which dimension of complexity is a good indicator of a given quality attribute? Which metric will be useful for conducting a prediction model? These questions are non-trivial and can only be answered by proper empirical studies. Typically, such studies validate the usage of metrics by building a prediction model. The predictive power depends not only on the selection of metric suite but also prediction techniques and evaluation method. The available data and feature of the data set also influences the predictive result.

While prediction techniques and evaluation method has been well studied and systematically aggregated into body of knowledge [4, 8, 13], there is very limited number of papers summarizing evidence-based knowledge about software complexity metrics [2]. Among the prediction models with the same prediction techniques, some studies yield different predictive results of same metric suite, even contradictory ones. This adds to the impression that, despite a large number of design metrics used in quality prediction models, it is currently still unclear whether we have learned anything from these studies. Kitchenham mentioned that there are numerous papers about software metrics and an opportunity for integrating these studies [10].

The thesis initially aims towards aggregating empirical studies about the ability of software design complexity metrics to predict cost and quality attributes. With our analysis, we want to answer three main questions: (1) What are design complexity dimensions/ metrics that are helpful for quality prediction? (2) Is there an agreement across the studies about the predictive ability of the design complexity metrics; that is, on how strongly a design metric influences software quality? (3) If there is disagreement, can we identify conditions under which studies agree? The body of knowledge about complexity metrics would be helpful for researchers and practitioners. It could provide a concrete guideline to select complexity metrics for conducting prediction model in a specific context. Also, it is important to notice that the thesis will be conducted in cooperation with one particular organization. Therefore, it is ensured that the scope of the project is appropriate for a master thesis.

1.1 Aims and objectives

The aim of this master thesis is to study the impact of complexity in cost and quality in specific contexts. Table 1 shows an adapted Goal-Question-Metric (GQM) template to form the research objective. The target objective of this research is software design complexity, in particular design complexity of OO system. The purpose is to externally validate design complexity by investigating its statistical relationship to external software quality. The topic is considered in the viewpoint of measurement researcher as well as practitioner. The study population is software engineering literature.

Table 1: GQM template of research goals

Analyze	Software design complexity
For the purpose of	Validation
With respect to their	Statistical relationship to cost and external software quality. Possible improvement of quality model
From the view point of	Measurement practitioner, researcher
In the context of	Software engineering literature

In order to achieve this target, the following tasks need to be done:

- Identify classification of complexity design metrics
- Identify possible complexity metrics that can be used as indicator of external quality attributes
- Assess the significance level of complexity metrics in statistical analyses
- Identify the predictive power of combination of different metrics
- Identify the overall effect size of design metrics on software quality attributes
- Identify the possible moderator variables that influence the relationship
- Find out the suitable metrics and method for purpose of prediction

1.2 Research questions

In the following, a set of research questions is listed. The major research question (RQ) is divided into several sub questions (SQ). Answering the SQs allows us to answer the main research question RQ:

RQ: Which design complexity metrics are indicators of software cost and quality?

- SQ1: Which quality attributes are predicted using design complexity metrics?
- SQ2: What kind of design complexity metrics is most frequently used in literature?
- SQ3: Which design complexity metrics is most frequently used in literature?
- SQ4: Which design complexity metrics are potentially predictors of quality attribute?
- SQ5: Which design complexity metrics are helpful in constructing prediction model?

- SQ6: Is there an overall influence of these metrics on external quality attributes? What are the impacts of those metrics on those attributes?
- SQ7: Do studies agree on these influences? If no, what explains this inconsistency? Is this explanation inconsistency across different metrics?

1.3 Structure of thesis

Figure 1 shows the overall structure of this thesis. The content is logically divided into three main parts, namely background research, research design and research contribution.

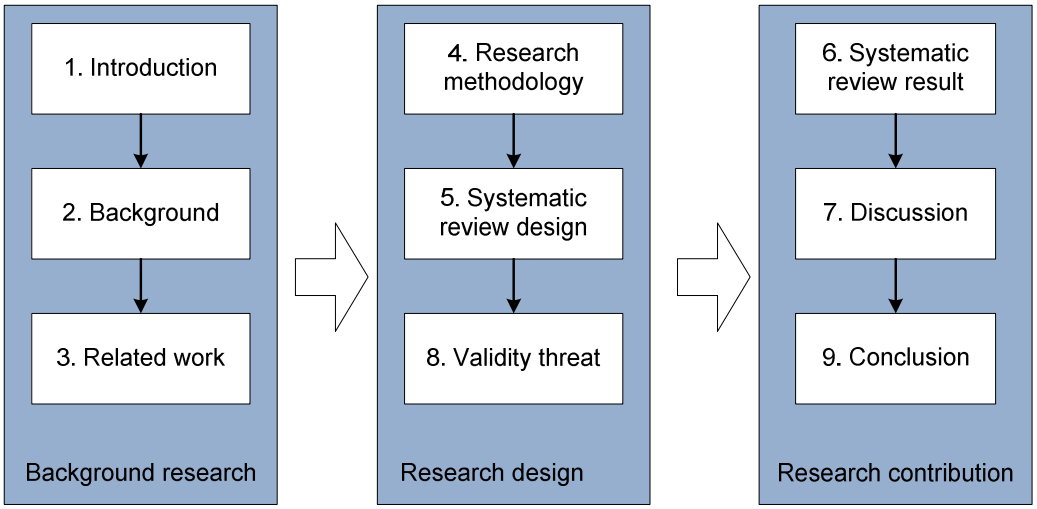

Figure 1: Thesis structure

In Background Research the reader is equipped with the essential information to follow the topics in the following parts. Chapter 1 (Introduction) present briefly the topic area, the necessity of the work and research questions. Chapter 2 (Background) introduces the software metrics, software design complexity, empirical study, statistical method and software quality attributes. Chapter 3 (Related Work) presents a brief summary of the relevant researches regarding to the aggregation of software measurement and quality prediction models. The chapters in Background Research can be skipped if the reader is already familiar with these topics.

In Research Design the implementation details of the conducted research are presented. Chapter 4 (Research Methodology) depicts the strategy how the stipulated research questions will be answered. Furthermore, the design of the systematic review is illustrated in Chapter 5 (Systematic Review Design). It includes a detailed review protocol which aims to add traceability to the review and to support a possible replication in the future. In Chapter 8, (Validity Threats) threats to validity regarding the research work are discussed.

In Research Contribution the original work of this thesis is presented. Chapter 6 (Systematic Review Results) analyses the gathered information through the systematic review and presents the findings. Chapter 7 (Discussion) provided detail of inference and comments on research question findings. The thesis work closes with Chapter 9 (Conclusion) which contains a conclusion and future work.

Chapter 2
Background

This chapter is used to summarize the concepts that are relevant to the research topic. In particular, definition about software metrics, design complexity, empirical research methodology and statistic methods are given in this section.

2.1 Software metrics

Software metrics provide a measurement for software and processes of software production. As a key concept in software engineering, there are many definitions of software metrics in literature. Here is one of the definitions of software measurement that highlights the role of measurement goal and process:

> *"Software measurement provides continuous measures for the software development process and its related products. It defines, collects and analyzes the data of measurable process, through which it facilitates the understanding, evaluating, controlling and improving the software product procedure" [6]*

Software metrics are categorized in three main groups, namely product metric, process metric and resource metrics [23]. Product metrics involve measurement of different feature of documents and programs generated during the software development process. Process metrics relate to measurement of activity happened during software development life cycle, such as software design, implementation, test, and maintenance. Resource metrics measure supporting resources such as programmers, and cost of the product and processes, etc [23].

Validation of software metrics is very important in order to determine their effectiveness in practices. Fenton distinguished between two type of metric validation, namely internal validation and external validation [6]. Internal validation attempts to theoretically prove that a metric is a true numerical characterization of the property it claims to measure. External validation involves empirically demonstrating that a metric is associated with some important external metrics (such as measures of changeability or testability) [6]. Although a design metric may be correct from a theoretical perspective, it may not be useful in practices. Metrics may be difficult to collect or may not really measure the intended quality attribute. Therefore, empirical validation of is crucial to the usage of a software metric in practice.

1.1. Software complexity

The first challenge when talking about software complexity is to answer the question: "What is complexity?" Unfortunately, there is no consensus on how to define software complexity. The early understanding of software complexity is a time or resource consumed by the system [6]. In computer science, complexity of a program is referred to as the algorithmic complexity and measured by the efficiency of an algorithm (the big O) [6]. Later, with the development of program methodology and language, complexity is interpreted as the difficulty level to understand, program and run the source program. The complexity of a program is thus referred to as structural attributes of software such as: control flow, data flow, data structure, cohesion, coupling and modularity [6].

IEEE standard defines software complexity as "*the degree to which a system or component has a design or implementation that is difficult to understand and verify*" [3]. The definition differentiates two different kind of complexity: complexity of design artifacts, such as UML diagram and complexity of implementation like source code. In the scope of this thesis, we only focus on design complexity.

1.2. Software design complexity

Basically, design complexity is structural features of design artifacts that can be measured prior to implementation stage. This information is important for predicting quality of final products in early phase of software development life cycle. There is a theoretical basic for investigation the relation between design complexity and software external quality attributes, namely theory of cognitive complexity [25]. Briand et al. hypothesized that the structural properties of a software component have an impact on its cognitive complexity [25]. In his study, cognitive complexity is defined as "*the mental burden of the individuals who have to deal with the component, for example, the designer, developers, testers and maintainers*" [25]. High cognitive complexity leads to the increasing effort to understand, implement and maintain the component. As the results, it could lead to undesirable external qualities, such as increased fault-proneness and reduced maintainability.

Prior to object oriented paradigm, design complexity involved the modeling of information flow in the application, Hence, graph theoretic measures and information content driven measures were used for representing design complexity. Nowadays, with the dominant of Objected oriented (OO) programming [24], certain integral OO design concepts such as inheritance, coupling, and cohesion have been argued to significantly affect complexity [97]. Those design features have been implicated in reducing the understandability of object-oriented programs, hence raising cognitive complexity. Figure 2 illustrates the theory with complexity compositions, namely coupling, cohesion, inheritance, scale and polymorphism.

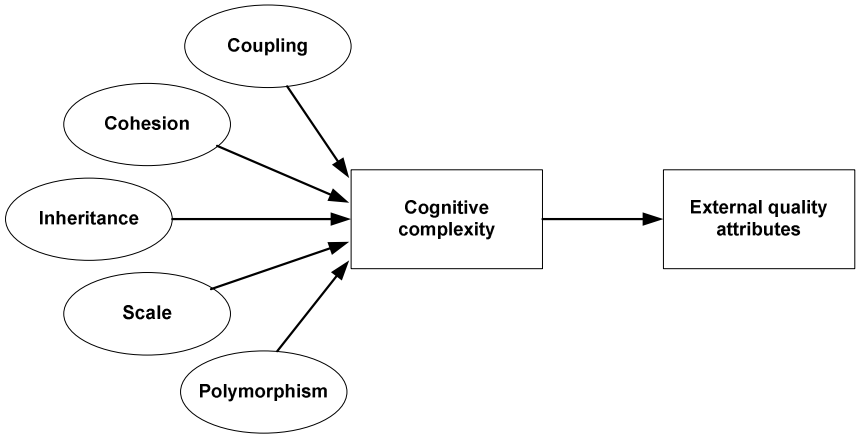

Figure 2: Theory of cognitive complexity (adapted from [129])

2.1.1 Coupling

Coupling is defined by Stevens as "the measure of the strength of association established by a connection from one module to another" [26]. In other words, coupling is a degree to which a software unit (a component, a module or a class) relies on each one of the other unit. Software engineering literature reveals some measurement frameworks that define different types of coupling in structured program. Briand et al. unified them in a comprehensive framework for coupling measure in OO system [27]. Basically, coupling metrics are defined by:

- Type of coupling: the mechanism that constitutes coupling between two classes, such as method invocation, attribute reference, type of attribute, type of parameters, passing of pointer to method.
- Locus of impact: the method, attribute are used or use other classes or attributes
- Granularity: level of detail where information is collected, such as method, class, module or system level.
- Counting direct or indirect connection
- Inheritance-based or non inheritance based coupling
- Polymorphism based or non polymorphism based coupling

It is an argument that the stronger the coupling between modules is, (such as the more inter-related they are), the more difficult these modules are to understand, change, and correct and thus the more complex the resulting software system [97]. Alternatively, a strong coupling between classes can increase the complexity of the software system and hence may impact on external quality such as maintainability and reliability. Low coupling has been considered as an important characteristic of good software systems because it allows individual modules in a system to be easily modified with relatively little worry about affecting other modules in a system [28].

2.1.2 Cohesion

Stevens also gave the first definition of cohesion as "*a measure of the degree to which the elements of a module belong together*" [26]. Alternatively, cohesion is a measure of how strongly-related is the functionality expressed by a unit of software. In general, cohesion is opposite with coupling. Low cohesion often correlates with high coupling, and vice versa. There are some measurement frameworks which defines different types of coupling in structured program. Briand et al. unified them in a comprehensive framework for cohesion [29]. Cohesion metric is defined by:

- Cohesion type: mechanism that makes a class cohesive, such as sharing of attributes, method invocations, attribute usage, type usage,
- Granularity: level of detail where information is collected, such as method, class, module or system level.
- Counting direct or indirect connection
- Inheritance-based or non inheritance based cohesion
- Polymorphism based or non polymorphism based cohesion

It is argued that in a highly cohesive module, all elements are related to the performance of a single function. Such modules are hypothesized to be easier to develop, maintain, and reuse, and to be less fault-prone. Therefore the impact of coupling on external quality is the target of many empirical validations [30].

2.1.3 Inheritance

Inheritance is a feature of OO programming that allows attributes and behaviors of objects be constructed and inherited from previously created objects [31]. This concept introduces a chain of inheritance classes with parent, child, ancestor and descendent classes. It is argued that the number of levels above a class in the inheritance hierarchy may, to some extent, influence the complexity of a software module. One reason is that behaviors of a class deep in the hierarchy could depend not only on the behavior of its own methods but also on the behavior of methods inherited from their parents and ancestor classes [33]. This inheritance adds complexity in the class. Chidamber et al. claimed that the deeper a class is placed in the hierarchy, the greater the difficulty in predicting the behavior of the class. This uncertainty about behaviors of the class may lead to difficulty in testing all the class interfaces and maintaining the class [32]. Wilde indicated that to understand the behavior of a method, a maintainer has to trace inheritance dependencies that are considerably complicated due to dynamic binding [34]. In another research, Mikhajlov claimed that a child class is more fragile due to changes in its parent classes [35]. These arguments seem to bring an impression that the more feature classes inherit the more difficult for tester and maintainer to understand, test and maintain them.

2.1.4 Polymorphism

Polymorphism is a property of OO software by which an abstract operation may be performed in different ways in different classes. Polymorphism involves the combination of message passing, inheritance and substitutability in OO programming languages which allows code sharing and reuse. It is found that there is no consensus on the polymorphism terminology in the OO programming languages community [115]. Benlarbi et al. classified polymorphism into five types, namely pure polymorphism, overriding, deferred methods, overloading and generics [115].

The main advantages of using polymorphism is the ability of objects belonging to different types to respond to method, attributes, or property calls of the same name, each one according to an appropriate type-specific behavior. Therefore, polymorphism has a direct impact on coupling cohesion and inheritance. As a consequence, it could indirectly affect external software quality. For instance, a programmer does not have to know the exact type of the object in advance, and so the exact behavior is determined at run-time (dynamic binding) [36]. This uncertainty can increase the effort to understand and maintain original source of method, attributes, which can impact on software maintainability and reliability.

2.1.5 Scale

Scale is the inherent property of any software system just like weight is an inherent characteristic of a tangible material. In our context, the scale of design means the size of a software or design artifacts in term of number of module, class, method, attributes and so on. The reason for including scale in the complexity model is its impact on ability of software developers to comprehend and control the source code. Therefore, the larger a software artifact is, the more difficult for a developer to remember and connect different parts of the artifact to achieve a comprehensive understanding [127]. Besides, it is intuitive that the larger a class is, the more time it takes to understand and visualize the logicality of the class. Therefore, the impact of scale on external quality seems to be unavoidable. Furthermore, scale is claimed to have a confounding impact on other complexity property such as coupling and cohesion, which can extend the difficulty in comprehending a system's functionality [113, 127]. Therefore, scale need to be explicitly controlled in studying the relation between complexity and external quality attributes.

2.2 Empirical study

Rombach et al. has stated that "*Human-based methods can only be studied empirically*" [39]. The relationship between cognitive complexity and software external quality depends on comprehending ability of software developer, tester or maintainer. This factor is not deterministic and hence, cannot be investigated any other ways than empirically.

In software engineering, empirical study involves introducing assumptions or hypotheses about observed phenomenon, investigating of the correctness of these assumptions and evolving it into body knowledge [40]. Empirical study is always attached to an environment context in which the study is performed. The formal definition of empirical software engineering is given as below:

> *"Empirical software engineering involves the scientific use of quantitative and qualitative data to understand and improve the software product, software development process and software management" [136]*

The definition differentiates two approaches in empirical studies. Qualitative approach attempts to interpret a phenomenon, problem or object based on explanation that people bring to them [41]. Quantitative approach involves quantifying a relationship or to compare two or more groups [41]. Therefore, quantitative investigations are common in empirical studies about relationship between design complexity and external software quality. In general, any quantitatively empirical study can be mapped to the following main research steps [54]:

- Definition: formulating an hypothesis or question to test
- Planning: designing, selecting suitable sample, population, participants
- Operation: executing the design, collecting data, variables, materials
- Data Analysis & interpretation: abstracting observations into data and analyzing data
- Conclusions: drawing conclusions and significance of the study.
- Presentation: report the study.

The work within the steps differs considerably depending on the type of empirical study. Quantitative empirical studies are differentiated due to the context and control level of experiment variable. Zelkowitz et al. [42] describes three main groups of validation model as below:

- An observational method: collects relevant data as a project develops. There is relatively little control over the development process [42].
- An historical method: collects data from projects that have already been completed. The data already exists; it is only necessary to analyze what has already been collected [42].
- A controlled method: provides for multiple instances of an observation in order to provide for statistical validity of the results [42].

In general, the controlled methods provide the more reliable results due to the well-control of experiment variables. However it is only possible to conduct controlled experiment for small case or laboratory situation that does not reflect the real case in industry. In practice, observational method or historical method is more common with the data collected from real projects. The drawback of these methods is the little or no control of experiment variable, which could seriously affect the reliability of the empirical result [42]. In the population of

studies about relation between design complexity and software external quality, it is found that historical and observational methods are overwhelming in study design [43].

2.3 Statistics methods

Quantitative empirical studies include the use of statistical methods to assess and quantify the relationship between treatment groups. There are two frequently used statistical approaches to investigate the relationship between design complexity and external quality, namely correlation analysis and regression analysis. From now on, we use the term "Correlation and regression analysis" to represents for this study area. Figure 3 shows the common methods to investigate the relationship between design complexity and external software quality attributes.

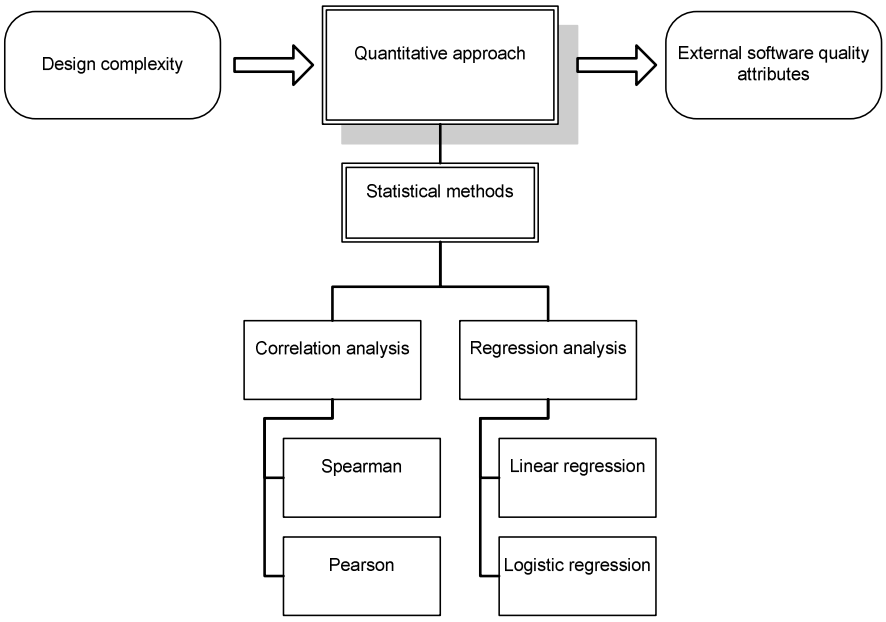

Figure 3: Empirical methods for investigating complexity-quality relationship

2.3.1 Correlation analysis

Correlation analysis investigates the extent to which changes in the value of an attribute (such as value of complexity metric in a class) are associated with changes in another attribute (such as number of defect in a class). The intensity of the correlation is expressed by a number called the coefficient of correlation and normally expressed by the letter "r" [44]. An correlation coefficient represents a numerical summary of the degree of association between two variables, e.g., to what degree do high values of coupling of a class go with high number of defect in that class [44]. Among study about relationship between design complexity and external quality, the two most common correlation analyses are Pearson correlation coefficient and Spearman rank correlation coefficient.

2.3.1.1 Pearson correlation coefficient

Definition: Pearson correlation coefficient ($r_{pearson}$) assesses how well the relationship between two variables can be described using a linear function. $r_{pearson}$ is obtained by dividing the covariance of the two variables by the product of their standard deviations:

$$r_{pearson} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[X-\mu_X][Y-\mu_Y]}{\sigma_X \sigma_Y} \qquad (1)$$

The value of $r_{pearson}$ ranges from -1 to +1. The value of +1 represents a perfect positive (increasing) linear relationship and value of −1 represents a perfect decreasing (negative) linear relationship. The value of 0 means there is no relationship at all between two variables.

Assumptions: The Pearson correlation coefficient can be applied to ordinal as well as interval or ratio data. The value of Pearson correlation suffers from outlier and data range [46]. The Pearson's r has two assumptions [46]:

- There is a linear relationship between two variables.
- The treatment variables are distributed normally.

Interpretation: The interpretation of $r_{pearson}$ is given through its square value. $r^2_{pearson}$ represents the amount of variability in the dependent variable that is associated with differences in the independent variable.

2.3.1.2 Spearman correlation coefficient

Definition: When the Pearson correlation is applied to ranking ordering, it is call Spearman's rank correlation coefficient ($r_{spearman}$). It assesses how well the relationship between two variables can be described using a monotonic function. The $r_{spearman}$ is calculated by the formula (2):

$$r_{Spearman} = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \qquad (2)$$

where *di* is the difference in the ranks given to the two variable values for each item of data

Assumption: Since the data (i.e. the ranks) used in the Spearman test are not drawn from a bivariate normal population Spearman's test is non-parametric and distribution-free. It is also not restricted by the assumption of linear relationship. The only assumption of Spearman test is the appearance of monotonic relation between two variables [46].

2.3.1.3 Correlation coefficient and causation relationship

It is noticed that correlation does not imply causation [47]. In other words, correlation analysis cannot be used to infer a causal relationship between the variables. The change in the dependent variable can be caused by an unknown variable that also causes the change in the independent variable. This unknown variable is a called confounding variable [48]. Therefore, a correlation can be taken as evidence for a possible causal relationship, but cannot indicate what the causal relationship, if any, might be.

Though correlation between two variables does not necessarily result in causal effect, it is still an effective method to select candidate variables for causation relationship. In order to achieve a meaningful casual relationship, it is necessary to control all confounding variables.

2.3.2 Regression analysis

In statistics, regression goes beyond correlation by adding prediction capabilities. R**egression analysis** includes techniques for modeling and analyzing several variables to identify the relationship between a dependent variable and one or more independent variables [46]. Among large amount of regression techniques has been used in software engineering literature, the most common techniques are linear regression model and logistics regression model [3].

2.3.2.1 Linear regression

Definition: Linear regression is the most popular statistical method to establish the relationship model between the explanatory variables and the dependent variable. Given a data set for dependent variable Y and several data sets for independent variable X1, X2 ... Xn, linear regression finds out the line that best fit with data of Y from data of X:

$$Y = a + b_1 x_1 + b_2 x_2 + ... + b_n x_n \qquad (3)$$

The widely used technique to estimate parameter b1, b2 ... bn for the best fit line is Ordinal Least Square (OLS), in which the parameter is calculated so the sum of squared distances between the observed values of Y from the predicted value based on the model is minimized [51].

Assumption: the main assumption of linear regression model is that the relationship between two variables has a form of linear equation. Besides, the standard error is assumed to follow a normal distribution with mean zero [52].

2.3.2.2 Logistic regression

Definition: Logistic regression is different from linear regression in that the dependent variable in logistic regression model is binary or dichotomous [57]. In logistic regression, the dependent variable is classified into one or two classes and the independent variables can take any form.

The expected value of Y with given the value of X is the conditional probability of Y with given X. When the logistics distribution is used, the probability is represented as follows:

$$\Pr(y \mid x) = \pi(Y = y, X = x) = \frac{e^{a+bx}}{1+e^{a+bx}} \qquad (4)$$

By a logit transformation, we have:

$$Logit(Y) = \ln\left(\frac{\pi}{1-\pi}\right) = a + bX \qquad (5)$$

Where Y is the binary dependent variable, pi is the probability that Y=1 happens and 1-pi is the probability that Y=0 happens. To estimate the model parameters, there is a technique to achieve this minimum OLS, namely log likelihood estimators [57]. The procedure to calculate the parameter is similar in linear regression model but it is applied to the logit function (formula 5).

Assumption: The logistic regression model does not require assumption of linearity between independent and dependent variables, normally distributed variables or homoscedasticity [57].

Interpretation: univariate logistic regression gives an useful interpretation for studying correlation between variables. Particularly, the regression coefficient b can give an estimation of odds ratio, one of common used effect size in statistics. Indeed, odds ratio is defined as the ratio of the odds for x=n+1 to the odds for x=n is given by the equation:

$$OR_{x,x+1}(Y=1) = \frac{Odds(Y=1 \mid x+1)}{Odds(Y=1 \mid x)} = e^{b} \qquad (6)$$

Replacing pi by logistics distribution function and conduct equivalence transformation, we have:

$$OR = e^{b} \qquad (7)$$

Formula 7 shows that the odds ratio between Y=1 and Y=0 with one unit change in X is estimated by exponential of regression coefficient b. Therefore the univariate logistic regression coefficients not only assess the predictive relationship between two variables but also identify the odds ratio between two data sets.

In case of multivariate regression model we have:

$$\Pr(y \mid x) = \pi(Y = y, X = x) = \frac{e^{a_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n}}{1 + e^{a_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n}} \qquad (8)$$

Among many mathematic distributions, logistics distribution is commonly used because of it is a mathematically flexible and easily used function. Besides, it lends itself to a meaningful interpretation by the direct derivation of odds ratio. Furthermore, in case of lacking variation in the dependent variable, logistic regression is useful.

2.3.2.3 Mutivariate regression model

In multivariate model, two stepwise selection methods – forward selection and backward elimination are used [44]. The forward stepwise procedure starts with 0 variables and examines and include satisfied variable one by one. The backward elimination method includes all variables in the beginning and delete unsatisfied variable one by one until a stopping criteria is fulfilled.

It is noticed that univariate regression models suffer from confounding variables while multivariate regression models suffer from multicollinearity. Multicollinearity refers to the degree to which any variable effect can be predicted by the other variables in the analysis. This will affect the result of prediction model and the interpretation of the model becomes difficult because of compounding influence. There are several ways to solve this problem, such as Principle component analysis (PCA), multicollinearity test and so on [57].

2.4 Software cost and quality

IEEE standard 1061 gives a definition of software quality:

> *"Software quality is the degree to which software possesses a desired combination of attributes such as maintainability, testability, reusability, complexity, reliability, interoperability, etc" [58].*

Software quality represents for wide range of desired non functional features of a software system. Software quality attributes are classified into two categories, namely external quality and internal quality [6]. External quality characteristics are those parts of a product that face its users, i.e. maintainability, reliability, usability, etc, where internal quality characteristics are those that do not, i.e. understandability, complexity, etc. Ideally, the internal quality determines the external quality and external quality determines quality in use [6]. In our model, dependent variable is design complexity, a type of internal quality. Independent variable is external quality attributes, such as maintainability and reliability.

Chapter 3
Related work

A preliminary study on the literature indicates that no systematic reviews on impact of design complexity on cost and quality attributes have been conducted before. However, several aggregation studies that are related to the complexity metrics and prediction of specific external quality attributes are identified. Although the studies reside in the same area, they do not focus on the issues addressed in this thesis work. A summary of those studies are given in this chapter.

Riaz et al. conducted a systematic review on maintainability studies [11]. The study attempts to find evidences on predictive ability of software maintainability. The author searches in Scopus and 8 electronic databases to select 15 papers. The work summarizes 34 maintainability prediction models. Furthermore, the work reports list of most commonly predictor of maintainability, such as Halstead metrics, cyclomatic complexity or average LOC. However, due to the small amount of found papers, it is not able to provide a statistical view of the investigated metrics. The paper also does not report the experiment context which is an important factor to judge the prediction results. Besides, the definition of successful indicator is not given.

Catal et al. conducted a systematic review on fault prediction studies [4]. The study tried to assess the common data set, method and metrics for constructing fault prediction model. 74 primary studies are selected from several journal article, proceeding, and book chapters in the period between 1990 and 2007. The synthesis of papers by characteristics of dataset (public, private or partial), method (statistics, statistics and expert opinions, machine learning or machine learning and statistics) and metric (method, class, process, file, quantitative values or component) is given. The study result suggests an increase of using public data set, machine learning methods, and the remaining stable of usage of class level metrics over year.

Gomez et al. performed a systematic review on measurement in software engineering [8]. This study tried to answer three questions: “What to measure?”, “How to Measure?’ and “When to Measure?”. 78 primary studies are selected via 4 electronics databases. For answering the first question, the study accumulates the metrics based on entities where the measures are collected (process, project, and product) and measured attributes (complexity, size, cost, etc). The second question is answered by presenting the percentage of metrics that have been validated (empirically, theoretically or both) and the focus of the metrics (object-oriented, measurement

focusing in process, quality, etc). The answer for the third question is presented by quantifying the metrics based on the phases when they were collected in the project lifecycle. The paper infers the domination of studies of product-level metrics, in particular, complexity metrics. These metrics are collected mostly in design and implementation phase.

Kitchenham conducted a mapping study on software metric [10]. The work aims at identifying research trends in software metric area and assesses the possibility of using secondary studies to integrate research results. The studies is classified by citation, study source and study topic. 87 studies are selected via Scopus and 25 papers are reviewed. The paper confirms an opportunity to aggregate primary studies in similar topic with a careful critical review and considering impact of data set.

Briand et al. conducted a literature survey of empirical studies that investigate the relationship between design metrics and quality attributes in OO systems [18]. The study summaries 39 empirical studies in term of dependent variables, independent variables, data set, data analysis method and model evaluation method. The common techniques in data analysis and prediction model and model evaluation are given. The paper also provides a summary of univariate and multivariate analysis extracted from collected studies. The author draws some conclusions about interrelationship between design measure, indicator of fault-proneness, effort and predictive power of models. The application of model across environment and usage of cost benefit model are also discussed.

Succi et al. empirically investigated the inter-relation among metrics of the Chidamber – Kemerer metric suite by using meta-analysis in a large open source data set [38]. Although their methodology is thus similar to ours, their study focuses on different goal (i.e., using meta-analysis to summarize to what degree CK metrics are correlated with each other, not with fault-proneness). Particularly, we want to investigate relationship between design metrics and external quality attributes.

While studies of Riaz [11], Catal [4], Gomez [8] and Kitchenham [10] focus on discovering the trend in the research field, our work attempts to answer more detail questions about complexity metrics. What particular dimension of complexity, metric set are useful for prediction software cost and quality will be investigated. Our focus topic is somehow similar to Briand's work [18] but will be conducted in larger scope and draws more detail conclusions. The question such as: "Which metrics set should be used in which given context?" will be answered in our work. The summary of a comparison is given in Table 2.

Table 2: Comparison to related studies

Paper	Year	Method	Objective	Aspect	DB-Range-No	Study type
Kitchenham [10]	2010	Mapping study	Software metric	Study goal, source & citation	Scopus; 2000-2005 87 classified, 25 reviewed	Empirical Theoretical
Riaz [11]	2009	Systematic review	Maintainability prediction	Prediction model techniques, metric & evaluation method	Scopus and 8 electronic databases; 1985-2008; 14 reviewed	Empirical
Catal [4]	2009	Systematic review	Fault prediction	Studies source, prediction method, metric and data set	11 electronic databases; 1990-2007; 74 reviewed	Empirical
Gomez [8]	2006	Systematic review	Software metric	What to measure, how to measure & when to measure	4 electronic databases 78 reviewed	Empirical Theoretical
Briand [18]	2002	Literature review	Quality prediction in OO system	Prediction metric, target attributes, data set, univariate model, multivariate model, model evaluation method	ad hoc literature review; 39 reviewed	Empirical
Succi [38]	2005	Meta analysis	Internal attribute correlation	C&K metrics, Pearson coefficient efficient	200 open source projects	Empirical
Our focus		Systematic review + meta analysis	Quality prediction	Prediction metric, target attributes, data set, univariate model, multivariate model, C&K metrics, Pearson coefficient efficient	Scopus, IEEE, ACM, 1985-2010	Empirical

Chapter 4
Research methodology

This chapter devotes to discuss the choice of research methods and how these approaches help to answer stated research questions (Section 4.2 to Section 4.5). Prior to that, a motivation that leads to the selection are given in Section 4.1.

4.1 Research method selection

Figure 4 shows the selection of research method to answer each research questions with the order of execution. At first, the literature review is used to obtain a quick impression about what type of software cost and quality attributes are investigated in design complexity studies. The detail about the method is given in Section 4.2. The literature review is selected instead of Systematic literature review because we want to achieve a broad view of research topics to shape the research focus. The review results in few costs and quality attributes that are meaningful and investigated in efficient number of study for the further comprehensive investigation.

To the best of our knowledge, no systematic summary in this topic have been done before. Besides, efforts to aggregate the knowledge from previous studies (i.e. an explorative review done by Briand [18]) have been insufficient. Hence this work aims to consolidate the large body of work in this area. The systematic and quantitative summary includes a systematic literature review and a meta analysis.

Second, a systematic review is executed to answer sub question 2, 3, 5. With the focus on few quality attributes, the comprehensive investigation of their predictors is performed via a systematic literature review. The design complexity dimension, design complexity metrics that are used in the selected studies will help to answer sub question 2 and 3 satisfactorily. Besides, the multivariate regression models reported in these studies will be synthesized to find the answer for sub question 5. Last but not least, the reported significance level, effect size and context factors in these studies will be exacted for further synthesis. Systematic is suitable for this investigation since it provides a complete result with the least bias in study selection. The detail about the method is given in Section 4.3.

The sub question 4, 6 and 7 are answered by research synthesis methods. There are two available quantitative synthesis methods in Software Engineering literature, namely vote

counting and meta analysis [62]. In order to find the potential predictors of software cost and quality, we firstly identify an efficient amount of study that investigates the usage of design metrics as predictors of software quality. Secondly, the design metric that has an evidence of non-zero impact on external quality is considered as a potential predictor of this quality attribute. To answer this question, only information about significance level and direction of the impact is required. Therefore, vote counting is applied for such a case that only little information is available. The detail about the method is given in Section 4.4.

The sub question 6 and 7 require and quantification and synthesis of effect size of design complexity metrics on software external quality. Due to the fairly large amount of data required for global effect size estimation and subgroup analysis, this method is only applicable for some design metrics. Besides, the number of reported effect sizes should be efficient. These reasons make meta analysis procedure unable to apply for investigated metrics in sub question 4. To identify the overall influence of some design complexity metrics on some software quality attribute, the reported influence (or effect size) from single study will be aggregated. It can be claimed as an overall influence if this global effect size represents for the whole population. If there are subgroups within the population whose effect sizes are different from each other, we cannot conclude about the overall influence. In this case the test for heterogeneity and subgroup analysis is important to answer the sub question 7. The detail about the method is given in Section 4.5.

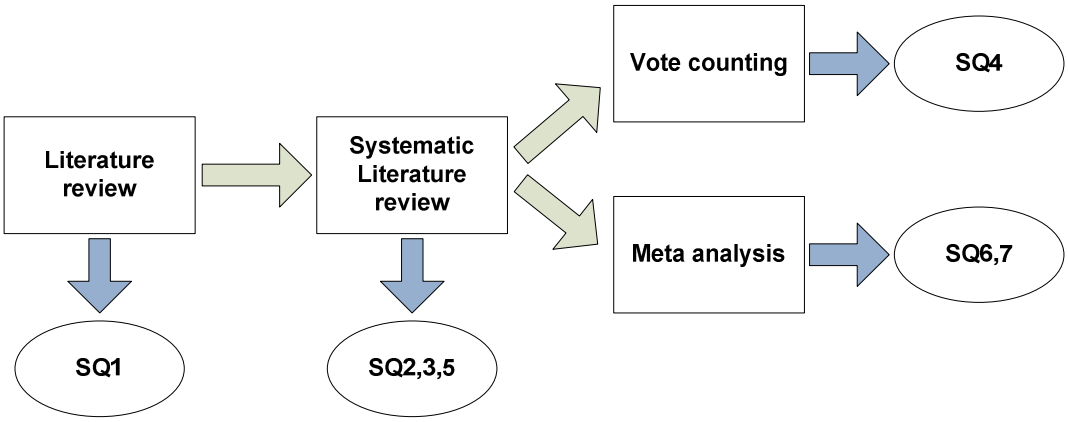

Figure 4: Research methodology

4.2 Literature review

A literature review is always performed before conducting any research. Researchers use literature reviews to achieve quick and broad knowledge about software complexity, prediction model and software cost and quality models. A literature review is different from a systematic literature review that we search for the relevant papers by the most effective way. In the very beginning of the research, when the main task is to search for scope and objective of research, this method is fit to explorative nature of the step. Ad hoc literature review help to find the

relevant studies in short time and build up knowledge of surrounding research area. However it does not ensure the completeness and systematicity. Therefore it is very likely that result of literature review is incomplete, bias and not repeatable. So we only use this method in the first phase to understand the research background, form the scope and objective of the research and identify some key papers for next phases.

4.3 Systematic literature review

A systematic literature review (often referred to as a systematic review) is a mean of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest [9]. Individual studies contributing to a systematic review are called *primary* studies; a systematic review is a form of *secondary* study [9]. There are many reasons for undertaking a systematic literature review. The most common reasons are to summarize the existing knowledge about interested research questions, to identify any gaps in current research in order to suggest areas for further investigation and to provide a framework/background in order to appropriately position new research activities. In the scope of this work, we select systematic review for relationship between software design complexity and external quality because:

- Its well-defined methodology enables reduction of bias in primary study selection result.
- Its systematic procedure enables consistency in study selection and evaluation of quality of primary study.
- Its outcome serves as input for meta-analytic techniques.

The systematic review involves the following steps [9]:

- Specifying research questions
- Data source selection
- Search strategy development
- Search string formation
- Study selection criteria identification
- Study quality assessment identification
- Study extraction strategy identification

The detail of each step is described in Chapter 5. In this thesis, we conducted a systematic review iteratively. The first iteration is a trial search and a test for review materials, such as assessment quality checklist and data extraction form. The second and third iteration search in different electronic databases for relevant studies. The last iteration searches for studies in gray literature. Detail description of these iterations is described in the search protocol in Chapter 5.

It is noticed that systematic review also has some disadvantages. Firstly, it cannot recognize publication bias so the result of systematic review can only reflect what is published but not the true nature of the problem. Besides, the procedure requires much more effort than ad hoc literature review [9].

4.4 Vote counting

There are two approaches to quantitatively summary results from systematic review, namely vote counting and meta analysis [62]. Vote counting is an alternative of meta analysis when there is not enough of statistical reported data. The main advantage of vote counting is that it is conceptually simple and can be used with very little information [62]. Besides it does not require actual value of effect sizes for aggregation [62]. The only assumption of the method is that studies use the same kind of statistical test.

Vote counting is used to reject the null hypothesis about the zero-effect in any treatment result. Kitchenham [62] reports the procedure of vote counting applied in software engineering as follows:

- Firstly, the null hypothesis is stated, i.e. the effect size between two variables is significantly positive (or negative).
- Secondly, the primary studies are categorized into three groups based on characteristics of their effect sizes. The groups are significant positive effect, significant negative effect and non-significant effect.
- Thirdly, the ratio between the number of significantly positive (or negative) effect sizes and total effect sizes are calculated.
- Lastly, the ratio is compared with a specific threshold to accept or reject the null hypothesis. Light et al. [63] suggests the cutoff value of 0.5 for two outcomes, which is appropriate in our case. In other word, we reject the null hypothesis if more than 50% of studies the reported effect size is significantly positive

We can expect that primary studies are various in the report of independent and dependent variables, statistics test and the choice of effect size. Therefore, vote counting is a suitable approach for the quantitative aggregation when there is not enough data.

Vote counting, however, is not free from error. Vote counting can be strongly biased towards the conclusion of zero effect size in populations of studies with small effect size and a low number of subjects. Besides, this bias is not reduced as the number of studies increase [64]. Therefore, the result of vote counting should be confirmed by other statistical methods, such as meta analysis.

4.5 Meta analysis

Meta-analysis is a statistical technique for identifying, summarizing, and reviewing results of quantitative studies to arrive at conclusions about a body of research [65, 66]. A typical meta-analysis is to summarize all the research results on one topic and to discuss reliability of the summary. Meta analysis is selected because of:

- Its well-founded mathematic background and statistical validity
- Its ability to provide a numerical aggregation value of studies. The outcome of a meta-analysis is an average effect size with an indication of how variable that effect size is between studies.
- Its ability to deal with study conflict. The heterogeneity test can assess the variation in reported effect sizes and subgroup analysis can help to explain this variation.
- Its ability to deal with publication bias. The "fill and trim" method can identify the appearance of publication bias and adjust the aggregated effect size with the awareness of publication bias.

It is noticed that meta analysis is only recently introduced in software engineering [38, 55, 56, 62, 70]. Besides, the original meta analysis procedure is applied for controlled experiments. In the context of this thesis, we applied a tailored meta analysis to observational and historical studies. Our approach includes seven main steps, which are adapted from [38, 62, 70]:

1. Study selection
2. Data extraction
3. Effect size estimation
4. Heterogeneity test
5. Explanation for heterogeneity
6. Sensitivity analysis
7. Publication bias check

The detail of each step is described in Chapter 5. It is noted that meta analysis requires each individual study reports same type of experiment variables and effect sizes. Besides, it is necessary to have efficient number of common moderator variables reported in each study to conduct a subgroup analysis.

Chapter 5
Systematic review planning

In this chapter, the systematic review protocols are developed and described in Section 5.1 and 5.2. After that, the tailored meta analysis approach is described in detail in Section 5.3.

5.1 Specifying the research questions

The first step in systematic review procedure is to construct the research question, which come from RQ1 in this thesis. The question structure is given in Table 1 by an adapted GQM template. The set of research questions for the systematic review is taken from the sub questions of the main research question: "RQ1: Which software complexity metrics are indicators for software cost and quality?" Table 3 shows the questions and their rationales.

Table 3: Question-goal mapping

Questions	Aim
SQ1: Which quality attributes are predicted using design complexity metrics?	To identify the possible independent variable for further quantitative aggregation
SQ2: What kind of design complexity metrics is most frequently used in literature?	To identify the most investigated complexity design dimension in literature
SQ3: Which design complexity metrics is most frequently used in literature?	To identify the list of useful complexity metrics for predicting external quality.
SQ4: Which design complexity metrics are potentially predictors of quality attribute?	To identify the most investigated metrics as inputs for further quantitative aggregation
SQ5: Which design complexity metrics are helpful in constructing prediction model?	Support SQ3 to identify the list of useful complexity metrics for predicting external quality.
SQ6: Is there an overall influence of these metrics on external quality attributes? What are the impacts of those metrics on those attributes?	Quantify the impact of complexity metrics on external quality attributes.
SQ7: Do studies agree on these influences? If no, what explains this inconsistency? Is this explanation inconsistency across different metrics?	To identify the moderator factors.

5.2 Developing a review protocol

A review protocol is an essential element in systematic review as it documents the detailed steps that will be used to carry out a particular systematic review with the aim to reduce the possibility of researcher bias [9]. This section presents the review protocol and the planned steps for the implementation of the review. The review protocol includes data source, search field, search strategy, search string, study selection criteria, quality assessment checklist and quality extraction form.

5.2.1 Electronic database and search field

To select the appropriate electronic database and search field, we conduct a survey through a list of systematic reviews in software engineering literature.

A review through systematic reviews in software engineering shows several choices for electronic database, such as Scopus, IEEE Explore, Current Contents, Computer Database, Science Direct, Springer Link, Inspec, ACM Digital Library and so on. To select the database, we determine the selection criteria as below:

- Coverage [19]**:** The database should provide efficient number of paper in software engineering and computer science field. Ideally, the database should index papers from different electronic data sources.
- Familiarity: The database should be easy to use or is familiar to the reviewer.
- Reputation: the database is well known or recommended or frequently used for systematic review.
- Advanced search [19]: the data source should facilitate the construction of complex search string.
- Exportability [19]: the data source should provide the function that support paper meta data synthesis and extraction.

Compendex, Inspec and Scopus are index electronic databases that we are familiar with. They also provide papers from different data sources, used in some previous systematic reviews [4, 8, 11, 15, 16]. However, in Scopus, the advanced search feature is more flexible since it allows unlimited number of search string clauses. Therefore, Scopus is selected for the main search. Besides, IEEE Explore and ACM Digital Library are selected due to its reputation and familiarity to the reviewer. These two databases are used in iteration 2 in order not to miss any important papers.

As common search fields in systematic reviews [4, 8, 11, 15, 16], article title, keyword and abstract will be used in this work.

With the objective to retrieve all relevant studies, we do not restrict the search range in any specific period of time. The search option for *Publication year* in IEEE Explorer or *Data range* in Scopus is *"All available year"*. The only limitation is that we only search in publication written in English.

5.2.2 Search Strategy:

As mentioned in [19], it is impossible for any search strategy to retrieve 100% of the relevant papers due to the shortage of reporting standards in the community. In addition, our review is restricted in human resource (only one main reviewer) and time (the research is performed within 6 months). Therefore, our objective is an effective search strategy with the balance between retrieved relevant papers and search effort. We aims at maximizing the amount of relevant material retrieved (high recall search) and minimizing the non-relevant material retrieved (high precision search) [19].

The search strategy is described in Figure 5, which includes two main phases: search term construction and search iteration.

5.2.2.1 Search term construction

In the first phase, the key papers will be identified after a literature review in relevant topics. In particular, the key papers can be introduced by our supervisor, results from an explorative search, or from reference list of other key paper. From the list of key papers, common terms are identified to initialize a search string. After that, several trial searches are conducted to determine which search string is appropriate. The criteria to select search string are given as:

- **Coverage**: is the ratio between number of key papers found in the search results and total number of key papers. The expected coverage is 100%. In other word, we would expect the search string covers all key papers.
- **Relevancy**: is the ratio between number of relevant papers and total number of retrieved papers from the search. We would expect an effective search with high relevancy. So by experience of the reviewer, the relevancy of 5% is reasonable. It means that we expect to have one relevant paper in every 20 papers from the search result pool.

5.2.2.2 Search process iterations

After confirmed by the supervisor, the search string is ready for conducting an actual search. In the second phase, the search procedure is executed iteratively. There are 3 iterations: pilot search, main search which both run in Scopus database and a compliment search which run in IEEE and ACM database. These iterations are summarized in Table 4.

Firstly, the pilot search and selection is conducted. The pilot procedure aims at refining the review protocol (inclusion, exclusion criteria, and quality assessment checklist and data extraction form) and perform test-retest check. The pilot search is conducted in Scopus. Due to restriction of time and resource, we only conduct a pilot search for a specific year (i.e. 2005) to limit the number of retrieved papers. The result pool from the pilot search will be used to:

- Check the self-agreement on study selection. The results of pilot selection will be compared against the results of an actual selection. If the actual selection pool does not fully cover the pilot selection pool, there is an inconsistency in primary studies selection.
- Check the self-agreement on quality assessment. The quality mark assigned for these studies will be compared with the quality mark for them in actual search and procedure to check the self-agreement on study quality judgment.

This so-call test-retest strategy is applied to reduce disadvantages of a single reviewer in selection bias [9]. Secondly, the procedure for the actual search is similar with the pilot search but with refined quality assessment check list and data extraction form. In the end, the result of this step is a whole pool of electronic selected papers from Scopus. Thirdly, the compliment search is performed. After the main search, we identify the main focus of the review and search for more relevant papers in that direction. The compliment search use IEEE and ACM databases to ensure quality of papers.

Table 4: Search iteration

Iteration	Description	Outcome
Iteration 1	Pilot search	Studies pool of year 2005 in Scopus with data extraction and quality assessment list
Iteration 2	Scopus	Studies pool of in Scopus with data extraction and quality assessment list
Iteration 3	IEEE, ACM	Focused studies pool in IEEE and ACM with quality assessment list.

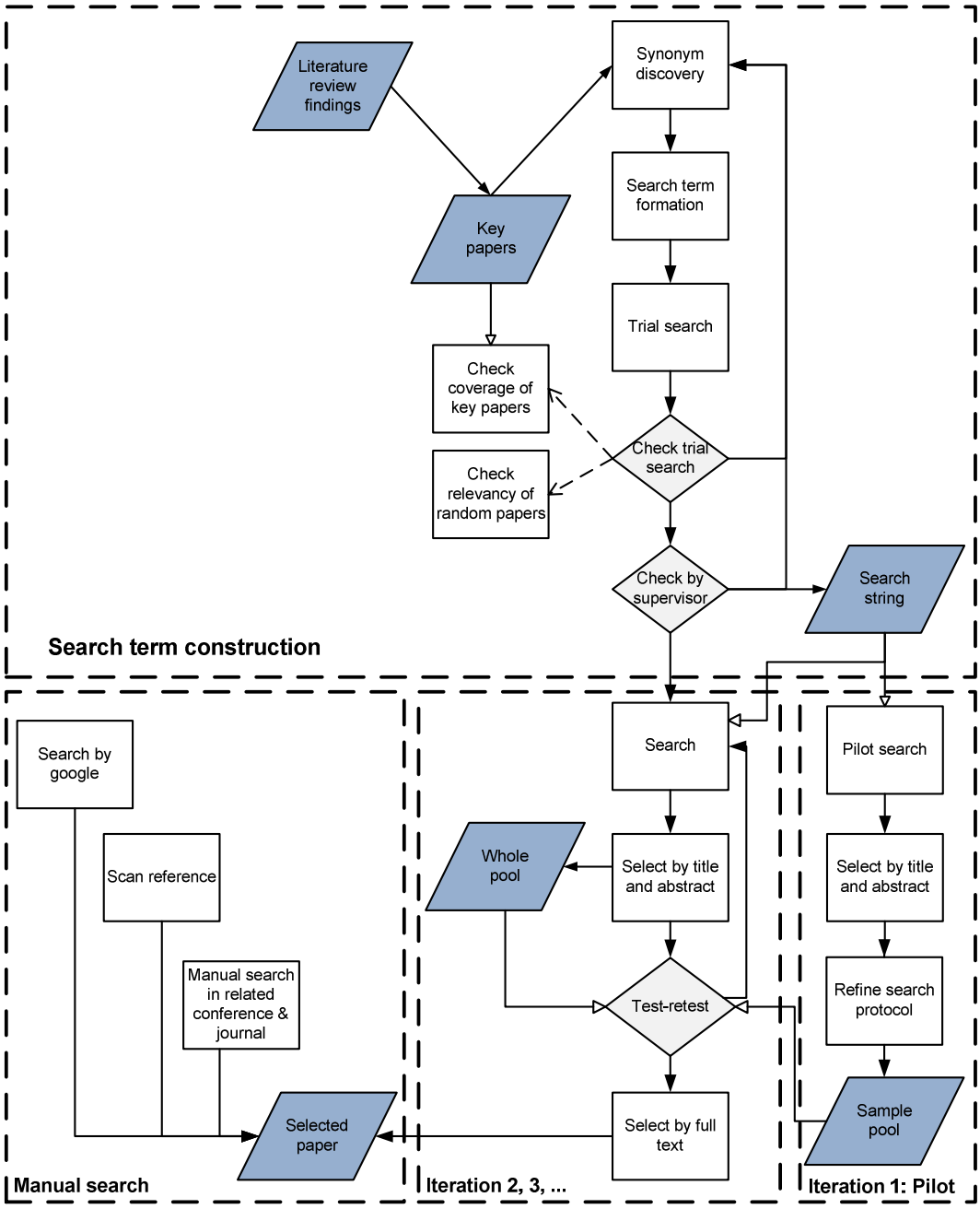

Figure 5: Search strategy

At last, some additional search strategies are performed to retrieve more relevant papers and to search for gray literature. The reference list of selected papers will be scanned in order to find more relevant papers. The manual search in related conference and journal will also be conducted. The search process will be applied for Google search engine. In the end, the additional manual search will add some extra papers that are not indexed by Scopus, IEEE Explore or ACM Digital library or not published.

5.2.3 Search string formation:

Derived from the main research question, the search string includes three main parts: "software design complexity", "indicator" and "software cost and quality". Table 5 presents the construction of the search string from these parts.

Table 5: Search term formation strategy

Search attributes	Steps	Rationale
Attribute 1	Identification of synonyms for "software design complexity" Identification different common types of "software design complexity"	to minimize the effect of heterogeneous terminology to increase the coverage of the search string
Attribute 2	Identification kinds of effect from complexity to other attributes Identification of synonyms for "indicator"	to minimize the effect of heterogeneous terminology
Attribute 3	Identification of synonyms and different type for "software cost" Identification of synonyms and different attributes of "software quality"	to increase the coverage of the search string
Background	Introduction of word "software" and synonyms Introduction of word "empirical study" and synonyms	to limit the search results in software engineering and empirical study

After identify the main elements of search string with the synonyms, the elements are joined by AND operator:

Search string = (Attribute1 AND Attribute 2 AND Attribute 3 AND Background) IN (Title or Abstract or Keyword)

The complete search term is shown in Table 6. These keywords resulted from several modifications to achieve desired coverage and relevancy. The search strings are applied on the Title/Abstract/Topic search fields to maximize number of relevant papers. We only consider papers written in English and limited in computer sciences and engineering domain.

Due to limitation of advanced search in IEEE and ACM electronic databases, we simplify the search string for these 2 databases as shown in Table 7.

Table 6: Search string for Scopus

Element	String part
Attribute 1	*("software complexity" OR "design complexity" OR "complexity metric*" OR "structural complexity" OR "structur* metric*" OR "complexity measure" OR "OO metric*" OR "object oriented metric*" OR "object-oriented metric*" OR "design metric*" OR "product metric")*
Attribute 2	*(impact OR implication OR assess* OR evaluat* OR influence* OR effect* OR relation* OR model* OR predict* OR estimat* OR measure* OR indicat* OR validat* OR quantif* OR correlat* OR relate OR investigat* OR spearman)*
Attribute 3	*(cost OR effort OR quality OR reliab* OR defect* OR maintain* OR usabilit* OR efficien* OR reusabilit* OR safety OR error* OR fault* OR prone* OR effectiveness OR testability)*
Background	*(software OR empirical OR experimental)*
Search string	*(TITLE-ABS-KEY("software complexity" OR "design complexity" OR "complexity metric*" OR "structural complexity" OR "structur* metric*" OR "complexity measure" OR "OO metric*" OR "object oriented metric*" OR "object-oriented metric*" OR "design metric*" OR "product metric") AND TITLE-ABS-KEY(impact OR implication OR assess* OR evaluat* OR influence* OR effect* OR relation* OR model* OR predict* OR estimat* OR measure* OR indicat* OR validat* OR quantif* OR correlat* OR relate OR investigat* OR spearman) AND TITLE-ABS-KEY(cost OR effort OR quality OR reliab* OR defect* OR maintain* OR usabilit* OR efficien* OR reusabilit* OR safety OR error* OR fault* OR prone* OR effectiveness OR testability) AND TITLE-ABS-KEY(software OR empirical OR experimental)) AND (LIMIT-TO(SUBJAREA, "COMP") OR LIMIT-TO(SUBJAREA, "ENGI") OR LIMIT-TO(SUBJAREA, "MULT")) AND (LIMIT-TO(LANGUAGE, "English"))*

Table 7: Search strings for IEEE Explore and ACM Digital Library

Element	String part
Attribute 1	*"object oriented metric*" OR "design metric*" OR DIT OR NOC OR LCOM OR WMC OR RFC OR CBO*
Attribute 2	*correlation or spearman or pearson*
Attribute 3	*Fault or defect or bugs or maintain**
Search string	*("object oriented metric*" OR "design metric*" OR DIT OR NOC OR LCOM OR WMC OR RFC OR CBO) AND (correlation or spearman or pearson) AND (fault or defect or bugs or maintain*)*

5.2.4 Study selection criteria:

The study selection criteria are decided based on content reported in paper's title and abstract. If some or a combination of the research questions are discussed in the paper, it is included in the selection. Particularly, the included paper has to address all of the following issues:

- Report the correlation or/and regression analysis between design complexity metrics and external software quality
- Report a value of correlation coefficient or regression coefficient
- Report experiment environment and data collection procedure
- Using one of empirical methods to answer the target research questions: case study, observation, controlled experiment, report data or empirical analysis.

If after reading the title and abstract the decision cannot be made to include the particular paper, the conclusion is read to determine its relevance. The following research issues are not our objective, so they are exclusion criteria. We exclude the papers that:

- Do not relate to software engineering or system engineering development
- Do not consider complexity of software/ system as a product design metric
- Do not report correlation of complexity metrics to other software project or quality attributes
- Define a new software complexity metric without empirical validation of metrics
- Critique existing software complexity metrics
- Develop or validate measurement tool or framework
- Investigate relation among complexity metrics (inter-correlation)
- Investigate cost or quality attributes of artifact that is not software product, e.g. process, test case, requirement document …
- Propose a method or procedure for managing complexity: specification, reduction …
- Assess or predict size of software artifacts
- Use other software attributes to estimate or predict software complexity
- Use software size metric, such as Line Of Code, Function Points as a only metric
- Validate or evaluate a software metrics empirically
- Predict or assess internal software quality attributes
- Use software complexity metrics for other purpose that doesn't relate to validation metrics or prediction of external attributes.
- Estimate or predict software attributes that do not use software complexity metrics.
- Estimate or predict software attributes that do not use software structural complexity metrics.
- Estimate or predict software attributes that use software complexity metrics that are specific for some application domain (web, prolog, etc).
- Investigate impact of code smell on external quality attributes

- Investigate combined impact of code metrics and design metrics on external quality attributes

5.2.5 Study quality assessment

Table 8: Study quality assessment (adapted from [6])

Criteria	Y/N/P
Problem statement	
Is the aim of the research sufficiently explained?	
Is the research hypotheses clearly stated?	
Research design	
Is the context of study clearly stated?	
Is the sample representative of the population to which the results will generalize?	
Was there a comparison or control group?	
Data collection	
Are the measures used in the study adequately described?	
Are the measures used in the study the most relevant ones for answering the research question?	
Is the data collection methods sufficiently described?	
Data synthesis	
Is there a data pre processing method?	
Is there an outlier data analysis method?	
Are the dataset characteristic sufficiently described?	
Are the statistical method sufficiently described?	
Are potential confounders adequately controlled for in the analysis?	
Meta analysis	
Are the effect size reported?	
Was statistical significance assessed?	
Conclusion	
Are all study questions answered?	
Are negative findings presented?	
Is the study experiment replicable?	

Quality of included papers is assessed during full-text reading. The defined quality criteria are tabulated in a quality assessment checklist as in Table 8. The quality criteria are rated by a "Yes", "No" or "Partially" scale.

5.2.6 Study extraction strategy:

After selecting primary studies, the data extraction process is performed. The selected studies share the typical structure of quantitatively empirical study, which includes the following parts:

- Problem statements: describe research objectives by research questions or hypothesis [50].
- Background: state the theoretical foundation of the investigated variable, metrics and reasoning for its hypothesized relation to external quality attributes.
- Experimental methodology: describe the research methodology to answer the research questions. Typical statistical methods are correlation analysis and regression analysis [50].
- Object and instrumentation: select the dependent and independent variable and its measure. The choice of measure depends on the availability of data and tool support [50].
- Context [50]: describe the project or data set where the data are collected. The information about case study can be various in level of detail. Observational study normally describes carefully staff assignment, project factors and time for repeatability [42]. Historical study normally describes the public data set and any available information attached to it [42].
- Data collection procedure: involve the tool, material used to collect data [50].
- Dataset reduction [50]: involve outlier analysis to remove the outliers, data transformation to apply the later statistical method, data classification for the specific research goals (i.e. classification of defect by severity) and data filtering (i.e. only select bugs from pre-release phase).
- Correlation analysis: either Spearman or Pearson correlation test to identify potential predictor of quality.
- Univariate regression analysis: assess the predictive ability of single metric
- Factor identification analysis: attempt to avoid the redundancy in metric set and remove collinearity for multivariate regression.
- Multivariate regression analysis: assess the predictive ability of combination metric
- Validity procedure: evaluating performance of prediction models.
- Interpretation and conclusion: interprets the finding from analysis part and presents the summary of the study.

Sections marked by a star indicate a compulsory section of the paper. Based on this structure, a data extraction form is constructed to extract useful information. The detail of data extraction form is given in Appendix C.

5.2.7 Review protocol evaluation:

Throughout the planning phase, the review protocol has been refined several times. The review team found that the review protocol needs continuous refinement in order to make sure that every step is precisely documented and focused. The components of the review protocol that

involved the major refinements were the research questions, search keywords and the data extraction form. The initial review protocol was sent to the review team supervisor.

The review team updated the review protocol accordingly and produced a second version of the review protocol. Aside from the evaluation, the review team performed their own verification on the review protocol through a series of piloting on the study selection and the data extraction. The details of the piloting will be presented in the coming sections.

5.3 Meta analysis planning

Our tailored meta analysis approach consists of eight parts, which are study selection, data extraction, global effect size estimation, heterogeneity check, sensitivity analysis, publication bias check, sub group analysis and cluster analysis. The sequence procedure is given as in Figure 6. Detail of each step is described in the following sub sections.

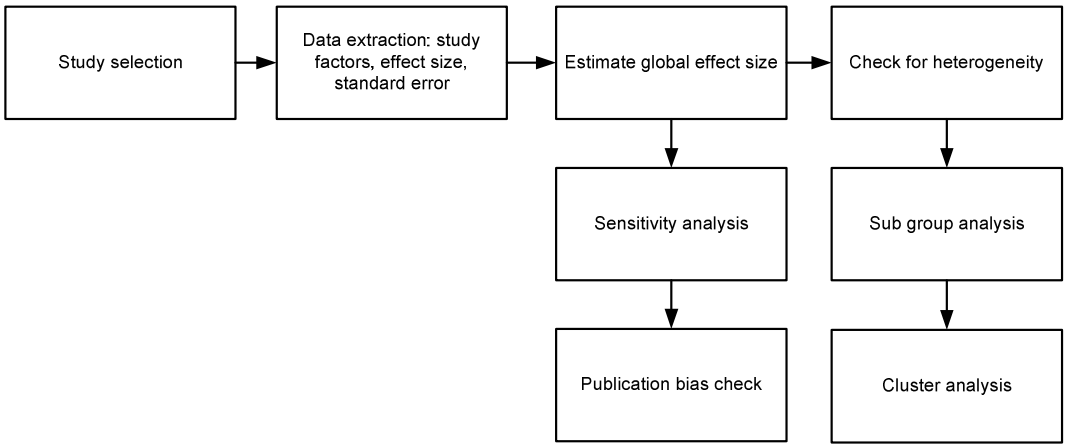

Figure 6: Meta analysis process

5.3.1 Study selection

Study selection is done as part of the systematic review. The studies selected from the review are filtered one more time to ensure that all papers report relevant effect sizes and information about moderator variables.

5.3.2 Data extraction

The information extracted from primary studies is effect size and moderator variables. There are four kinds of effect size reported in retrieved studies: Spearman correlation coefficient, Pearson correlation coefficient, linear regression coefficient and logistics regression coefficient

(detail is given in Section 2.3.1). Spearman coefficient and logistics regression coefficient are selected for further analysis because they are reported in the efficient number of studies. Besides, the moderator variables are also extracted from studies. Their categories are standardized and described in Table 9:

Table 9: Description of moderator variables

Properties name	Description
Data set size	Property of data set. Dataset is actually the set of class in which metric data is reported. The number of class that are used for correlation analysis or regression.
Dataset type	Property of data set. There are three types: public, private industry and private academic. Public dataset contain metric data that are distributed freely or open source code from which metric data can be calculated. The example sources of this type are NASA [153] and PROMISE [153]. Private data set belongs to private organization (can be industry organization or academic organization) and they are not distributed as public datasets.
Programming language	Property of project. The programming language used to develop the investigated classes. One data set can involve one programming language and also multi programming language.
Business domain	Property of project. The business domain of the data set. There are seven main identified domains: Internet software (browser and utilities), software development (environment, editor …) communication (satellite, telecommunication …) information management system, document processing, graphic, control system (simulation system, OS system …)
Defect collection phase	Property of project. The defect associated with classes has three types: pre-released defect which are collected during testing phase, post-release defect which are reported during maintenance time and mixed type.
Empirical type	Property of study. Historical method collects data from projects that have already been completed using existing data. Observational method collects relevant data as a project develops, such as industry projects. In our context it is academic projects where many context factors are well controlled.

5.3.3 Effect size estimation

After collecting all reported effect sizes (Spearman's rank correlation coefficients and estimated Odd ratio), the global effect size calculation process is performed to estimate the

average effect sizes for all primary studies and its confidence interval. The common method for this aggregation is Weighted Estimators of a Common Correlation, which is described in [65] and applied in software engineering context in [116].

The technique requires effect sizes are Pearson correlation coefficients and come from normally-distributed population. Since our original effect sizes are Spearman correlation coefficient and Odd ratio, it is necessary to convert those to Pearson coefficient. Therefore, the calculation process includes five steps:

1. Conversion of effect size to Pearson correlation of coefficient.
2. Normalization of Pearson effect size with Fisher z transformation [65,116].
3. Computation of the weighted point estimation of the transformed correlation [65,116].
4. Computation of the required confidence interval for the transformed correlation [65,116].
5. Application of the inverse transformation to obtain value range of original effect size [65,116].

5.3.3.1 Converting to Pearson correlation coefficient

Regarding to Spearman coefficient, we can convert it to Pearson correlation coefficient by using Formula 10:

$$r_{pearson} = 2\sin\left(\frac{\pi}{6} r_{spearman}\right) \qquad (10)$$

This transformation has error at approximately 0.005. The error decreases when the sample size and the actual population correlation increase as our case [38].

The odds ratios (OR) associated with one unit increase of design metric X is the ratio of the odds that Y =1 when X= x+1 to the odds that Y=1 when X= x and is approximated by the exponential of coefficient b (as given in Formula 7). Normally, the conversion of odds ratio require information from the 4-cell table [65]. Unfortunately, this information is not reported. Thanks to the advance in statistics method, Bonett introduced some alternative approaches to estimate Pearson correlation coefficient from Odds ratio [153]. The conversion formula is given as Formula 11:

$$r_{pearson} = \cos\left(\frac{\pi}{1+\sqrt{OR}}\right) \qquad (11)$$

5.3.3.2 Converting to Fisher Z

After having effect sizes in shape of Pearson correlation coefficient, the next step is to normalize Pearson correlation coefficient by using Fisher z transformation. The transformation is done by using formula 12:

$$z(r_{pearson}) = \frac{1}{2}\log(\frac{1+r_{pearson}}{1-r_{pearson}}) \qquad (12)$$

5.3.3.3 Calculating weighted point estimator

After normalizing effect sizes, we have all effect sizes with normal distribution. The next step is to decide the weight for each effect size and calculate the weighted mean effect size. There are many options for choosing a weight, such as data set size, standard error, data set quality and so on. In our approach, dataset size is selected due to its availability and calculative simplicity. Given a set of transformed effect size $z_1, z_2, ..., z_n$ and set of weight for each effect size $w_1, w_2, ...w_n$ the average weighted effect size z is calculated by Formula (13):

$$\bar{z} = \frac{\sum_{i=1}^{k} w_i z_i}{\sum_{i=1}^{k} (n_i - 3)} \qquad (13) \qquad SE(\bar{z}) = \sqrt{\frac{1}{\sum_{i=1}^{k} (w_i)}} \qquad (14)$$

Formula 14 give us a standard error of the average weighted effect size. The two approaches to calculate $\bar{z}$ and $SE(\bar{z})$ are fixed effect model and random effect model. In the fixed effect model, the weighted is given as:

$$w_i = n_i - 3 \qquad (15)$$

with ni is size of data set ith.

Average weighted effect size $\bar{z}$ and its standard error is calculated by formula 15, 16 respectively:

$$\bar{z} = \frac{\sum_{i=1}^{k} w_i z_i}{\sum_{i=1}^{k} (n_i - 3)} = \frac{\sum_{i=1}^{k} (n_i - 3) z_i}{\sum_{i=1}^{k} (n_i - 3)} \qquad (15) \qquad SE(\bar{z}) = \sqrt{\frac{1}{\sum_{i=1}^{k} (n_i - 3)}} \qquad (16)$$

Fixed effect model assumes that effect sizes come from a homogeneous population and are only different due to sample variability. In case effect sizes come from different populations or they are heterogeneous, fixed effect model is not applicable [65]. In such cases, random effects model is appropriate since it allows for variability due to an unknown factor influencing in the effect sizes for different studies and produces a larger estimate of the population variance than the fixed effect model [65].

In the random effect model, the weighted is calculated using Formula (17):

$$w_i = \frac{n_i - 3}{1 + (n_i - 3)\tau^2} \qquad (17)$$

With t and c is calculated by Formula 18, 19:

$$\tau^2 = \begin{cases} \dfrac{Q - (k-1)}{c} & if\ \mathrm{Q} > (k-1) \\ 0 & if\ \mathrm{Q} \le (k-1) \end{cases} \qquad (18)$$

$$c = \sum_{i=1}^{k}(w_i) - \frac{\sum_{i=1}^{k}(w_i)^2}{\sum_{i=1}^{k}(w_i)} = \sum_{i=1}^{k}(n_i - 3) - \frac{\sum_{i=1}^{k}(n_i - 3)^2}{\sum_{i=1}^{k}(n_i - 3)} \qquad (19) \qquad [45]$$

Since we are not sure about the homogeneity among primary studies, we decided to use both models. A heterogeneous test will be performed to decide which model is suitable. If heterogeneity is non-significant, a fixed effects model is a suitable choice. Otherwise, random effects model is selected.

5.3.3.4 Calculating 95% confidential interval

The 95% confidence interval of average weighted effect size $\bar{z}$ is calculated by the formula:

$$z^- = z - \frac{1.96}{\sqrt{\sum_{i=1}^{k}(n_i - 3)}} \qquad z^+ = z + \frac{1.96}{\sqrt{\sum_{i=1}^{k}(n_i - 3)}} \qquad (20)$$

5.3.3.5 Inverse transformation

In order to obtain effect size in original type (odds ratio or spearman coefficient) from standard effect size z, we firstly apply z inverse transformation as in Formula 21:

$$r_c = \frac{e^{2z} - 1}{e^{2z} + 1} \qquad (21)$$

We use formula 22 to retrieve Spearman coefficient and formula 23 to get Odds ratio from Pearson coefficient:

$$r_c = \frac{6}{\pi} \arcsin\left(\frac{r_s}{2}\right) \qquad (22)$$

$$OR = \left(\frac{\pi}{\arccos(r_c)} - 1\right)^2 \qquad (23)$$

5.3.4 Heterogeneity test

To assess the heterogeneity of the population, the Q test is applied [65]. Q index is calculated by formula 24 with a weighted from fixed effect model:

$$Q = \sum_{i=1}^{k} (n_i - 3)(z_{ri} - \bar{z}_r)^2 \quad (24) \quad [21]$$

A significant Q means there is variability among the effect size is greater than what is likely to have resulted from subject level variability alone. I square statistic can show the heterogeneity in percentages:

$$I^2 = \frac{Q - (k-1)}{Q} \times 100\% \qquad (25)$$

A value 0 of I square indicates absence of homogeneity, <40% means trivial or small, between 40% and 70% means moderate level and more than is high level of heterogeneity [68].

5.3.5 Explanation for heterogeneity

When heterogeneity appears, the desired action is to find the cause, or another word, to explain variations among the datasets. This activity typically takes the form of investigating a moderator variable that is able to account for a significant part of the observed variation. The available methods are subgroup analysis, cluster analysis and regression [69]. We select subgroup and cluster analysis due to its simplicity and availability of support tools.

5.3.5.1 Sub group analysis

In case of appearance of heterogeneity, sub group analyses will help to identify possible immediate sources of heterogeneity. The same meta analysis procedure is applied for each sub group. The summarized effect sizes of each group is examined and compared for consistency. The consistency of effect size in sub group is investigated by two tests: within-class goodness-of-fit statistics Qw and between-class goodness-of-fit statistics Qb. The two tests are given by the formula:

$$Q_W = Q_{W_1} + Q_{W_1} + ...Q_{Wn} \qquad (26)$$

with Qwi is Q statistics over all levels of the moderator variable. Qw follows the chi square distribution with k-p degrees of freedom, where k is the total number of studies in the sample and p is the total level of the moderator variable. Qb is the difference between the total amount of variation in the sample Qt and Qw. Qb follows the chi square distribution with p-1 degrees of freedom. The percentage of variance explained by the moderator variable is given as:

$$ve = \frac{Q_B}{Q_T} \qquad (27)$$

5.3.5.2 Cluster analysis

Sub group analysis is only feasible when we already have the known factors that are suspected to influence effect sizes [71]. Sometimes, it is the case that some unknown characteristics of the study are the actual factors of variation among studies. In this situation, cluster analysis is more suitable. The basic idea is to assign the set of effect sizes into a subset so that the heterogeneity of the subset is minimized. In our context it means the assignment of studies with similar value of effect sizes. The cluster analysis is conducted by using R package.

5.3.6 Sensitivity analysis

Sensitivity analysis is used to investigate how individual studies influence the aggregation results. Typically, sensitivity analysis involves comparing the results of two or more meta-analyses calculated using different set of input data [72]. In this work, we use CMA software which already provides Sensitivity analysis with leave one out strategy.

5.3.7 Publication bias test

Publication bias captures the idea that studies that report relatively large treatment effects are the ones that were most likely to be published. The estimated effect size from a biased collection of studies will tend to overestimate the true effect.

Firstly, the appearance of publication bias is identified by a funnel plot. A funnel plot is a scatter plot of treatment effect against a measure of study size [73]. It is used primarily as a visual aid to detecting bias or systematic heterogeneity. A symmetric inverted funnel shape arises from a 'well-behaved' data set, in which publication bias is unlikely. An asymmetric funnel indicates a relationship between treatment effect and study size [74]. This suggests the possibility of either publication bias or a systematic difference between smaller and larger studies ("small study effects").

Secondly, the corrective action is performed to assess the influence of publication bias on global effect size. "trim and fill" method is performed, in which we (1) remove (trim) the smaller studies causing funnel plot asymmetry, (2) use the trimmed funnel plot to estimate the true centre of the funnel, then (3) replace the omitted studies and their missing counterparts around the centre (filling) [73]. As well as providing an estimate of the number of missing studies, an adjusted intervention effect is derived by repeating the calculating of global effect size for filled studies [73].

5.3.8 Software tool

In this thesis, meta analysis procedure is conducted with the support of statistical tools, namely R package and Comprehensive meta analysis (CMA). The tools are selected due to the prior knowledge of the researcher about tools and their ease to use. R package is an open source project that provides a language and environment for statistical computing and graphics. The more information about R is provided in [75]. CMA is a powerful commercial software for meta analysis. The program combines ease of use with a wide array of computational options and sophisticated graphics. The more information about CMA is provided in [76]. In this research, we use R package to calculate global effect sizes, draw forest plots, check for heterogeneity, and conduct sub group analysis and cluster analysis. Sensitivity analysis, publication bias test are performed by CMA.

5.4 Conducting review

This section devotes to describing the intermediate results of review processes, which are study selection plotting, main study selection, quality assessment and data synthesis.

5.4.1 Study selection plotting

To validate that the search string is good enough to find the required primary studies, the search string result was validated against some pre-selected known relevant papers. Table 10 shows the papers result of 4 trial search strings against key papers. The search string 4 is formed to include all of known relevant papers.

Table 10: Coverage test for search strings

Times	No of identified papers	No of papers found in key pool	No of key papers
Search string 1	171	15	22
Search string 2	185	17	22
Search string 3	232	19	22
Search string 4	352	22	22

Table 11 shows result of test and retest. A sample pool with studies selected in 2005 is checked by abstract twice at different time. The selection pool from two times is compared. The test shows the overlap portion between 2 pools is 87% which is not bad for a single reviewer [154].

Table 11: Result of test and retest

Times	No of identified papers	No of selected papers
Key papers	5	5
First plotting (test)	23	4
Main selection (retest)	20	5

5.4.2 Primary study selection

The next step in the systematic review is to retrieve papers for the selection of primary studies. We utilized a reference management tool, namely Zotero [77], to handle resources such as bibliography generation, categorization of the papers, storage of downloaded full-text papers, sorting of papers in alphabetical order to identify duplicates, etc. Figure 7 summaries the paper selection result.

The first iteration resulted in a total of 906 papers as shown in Figure 7. We excluded 625 papers out of a total of 906 primarily based on title and abstract, which were clearly out of scope and did not relate to any research questions. The remaining 281 papers were subject to detailed exclusion criteria. The paper is scan to find out whether any exclusion criteria are met. This stage resulted in remaining 85 papers, which were further filtered out by reading full text. The studies are carefully read to find whether paper's objective, study design, data collection and analysis are relevant to answer our questions. This step remains 31 papers.

The second iteration resulted in a total of 265 papers. It is noticed that many papers in this set is duplicated with previous pool because Scopus also index papers in IEEE Explore and ACM Digital Library. After eliminating duplicated papers, there were left 85 papers. The title and abstraction exclusion results in 40 remaining papers and further exclusion based on detailed

criteria reject 25 papers out of this 40. The remaining 15 papers are read and finally 8 of them are selected.

In this electronic database search, the coverage of the search string was estimated by conducting a pilot search, which showed an initial coverage of 88%, and after a refinement of the search string, a recall of 100%. Relevancy, on the other hand, expresses how good the search identifies only relevant items, and is defined as the ratio of relevant items retrieved and all retrieved items. The precision value is low about 4% (considering 991 documents after the removal of duplicates and 39 selected primary studies). This is however expected since recall and precision are adversary goals, i.e. the optimization to retrieve more relevant items (increase coverage) implies usually a retrieval of more irrelevant items too (decrease relevancy) [78].

After electronic database search process, we found 39 relevant papers. In order to find more papers, we scanned the reference list of these papers. The results of paper reference scan give us 11 more relevant papers. These papers mainly come from conference or published source which are indexed neither in Scopus, IEEE Explore nor ACM Digital Library. This number suggested us to search for more papers in relevant conference or journal. The conference, journal scan process and random search by Google engine resulted in 7 more papers. Therefore, in the gray literature search, we found a fairy large portion of relevant studies: 18 papers out of total 59 selected papers, which consume 30.5% total of papers.

5.4.3 Quality assessment

According to Kitchenham, the study quality assessment can be used to guide the interpretation of the synthesis findings and to determine the strength of the elaborated inferences [9]. During the data extraction process, the quality assessment checklist is rated according to a “Yes”, “No” or “Partially” scales (detail of the checklist is given in Section 5.2.6). For each question, an answer “Yes” is given 1 point, “No” is given 0 point and “Partially” is given 0.5 point. Table 12 shows the summary of the assessed quality of the selected primary studies

5.4.4 Data synthesis

Data synthesis is a crucial step in conducting the review, which involves the collection of data, analyzing and summarizing the results from the systematic review [9]. The raw data from the data extraction forms were grouped together and analyzed accordingly for each research question. Chapter 6 is dedicated to present the details of the systematic review results focusing primarily on the elaborated data synthesis in answering each research question.

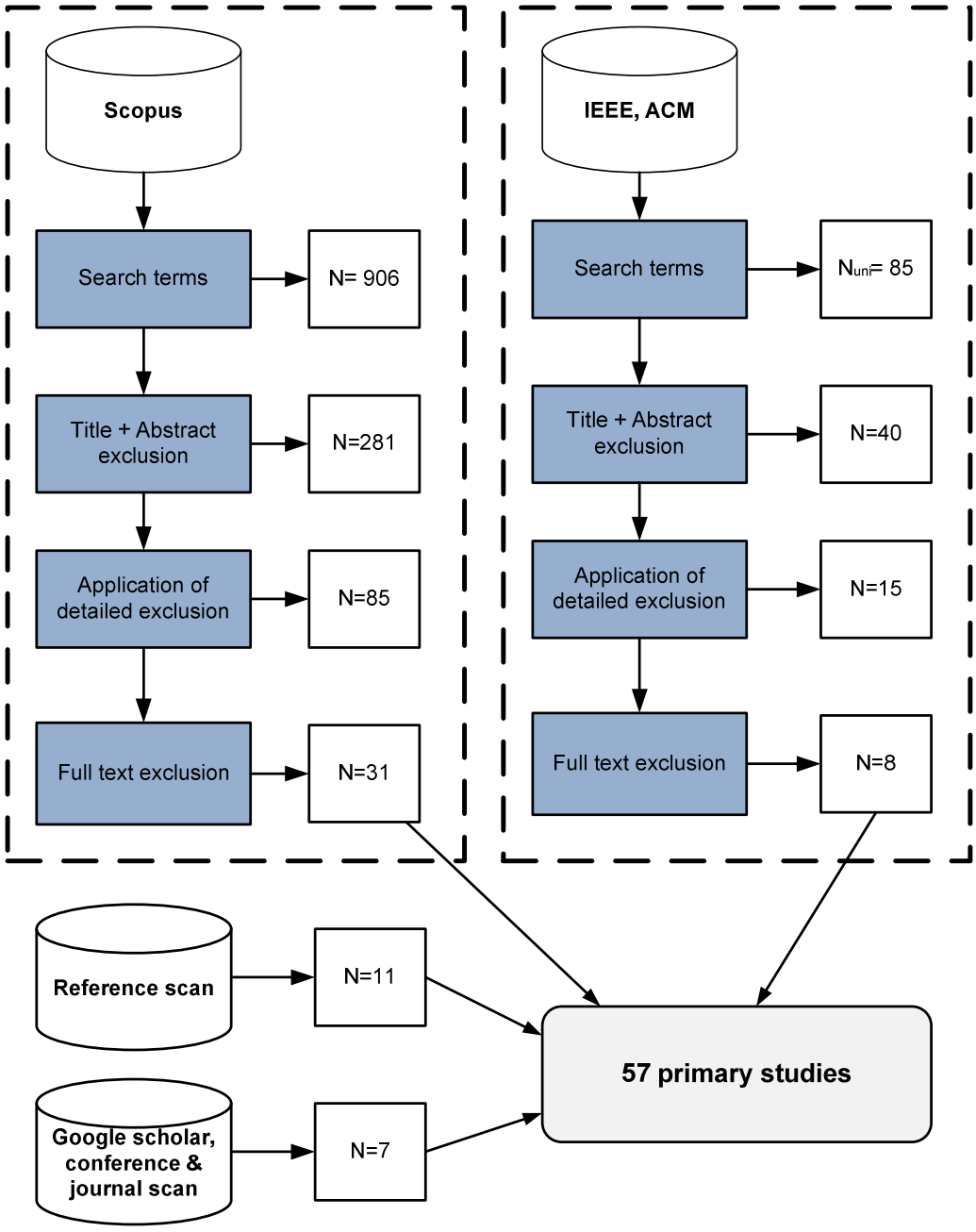

Figure 7: Systematic review selection result

Table 12: Quality assessment result

	Problem statement		Study design		Data collection				Data synthesis				Conclusion			Meta analysis			
Study	**Problem statement**	**Hypothesis**	**Study context**	**Sample representative**	**Control group**	**Desc. of var.**	**Desc. of measure**	**When & how collect**	**Data pre process**	**Outlier ana.**	**Descrp. statistic**	**Adequate statistics test**	**Control for comfounder**	**Ans. Ques.**	**Negative finding presented**	**Sample size**	**p value**	**Replicable?**	**Total**
ASK+09	1.0	1.0	1.0	0.0	0.5	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	14.0
SKM09	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	12.5
OEM+08	1.0	1.0	0.5	0.0	0.0	1.0	1.0	0.5	0.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	10.0
XHC08	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	9.0
GS08	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	9.0
SL08	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.5	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	10.5
PD07	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	0.5	0.0	7.5
OEG+07	1.0	1.0	0.5	0.5	0.0	1.0	1.0	0.5	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	11.5
ASK+07	1.0	1.0	1.0	0.0	0.5	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	13.0
ZL06	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	11.0
JSP+06	1.0	0.0	0.5	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	8.5
GFS05	1.0	1.0	0.5	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	11.5
SPS+03	1.0	0.0	0.5	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	8.5
BMW02	1.0	0.0	0.5	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	10.5
EBG+01	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	12.0
HCN98	1.0	0.0	0.0	0.0	0.0	1.0	0.5	1.0	0.0	0.0	1.0	0.5	0.0	1.0	0.0	1.0	0.0	0.0	7.0
BBM96	1.0	1.0	1.0	0.5	0.5	1.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	13.0
TLG05	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.5	0.0	1.0	0.0	0.5	0.0	0.0	5.0
SZP+06	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.5	0.0	1.0	1.0	1.0	0.0	0.0	6.5
BWL01	1.0	1.0	0.5	0.0	0.5	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	13.5
TKC09	1.0	0.0	0.5	0.0	0.0	1.0	0.5	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	8.0
BWD+00	1.0	1.0	1.0	0.0	0.5	1.0	1.0	1.0	0.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	14.0
HJB+08	1.0	1.0	0.5	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	0.0	0.0	8.0
Total	**23.0**	**13.0**	**8.5**	**1.0**	**2.5**	**22.0**	**22.0**	**14.0**	**3.0**	**4.0**	**19.0**	**18.5**	**3.0**	**21.5**	**20.0**	**22.5**	**16.5**	**5.0**	**10.6**

Chapter 6
Systematic review result

This chapter is used to present the findings from systematic review and meta analysis processes. In Section 6.1 and 6.2, characteristics of primary studies and their datasets are discussed. Section 6.3 presents the main contribution of this thesis as answers for proposed research questions.

6.1 Characteristic of primary studies

This chapter devotes for describing characteristics of selected studies from the systematic search. Figure 8 shows the distribution of primary studies based on publication year. There are 57 studies published in 15 years, from 1996 to 2010. It is revealed that no relevant papers are found prior to 1996. It was also the time when the Empirical Software Engineering journal was firstly published. It is worth noticed that in the late 90's, there were not so much publications and the most probable reason for this is that design metric was a new research area at that time, i.e. Chidamber & Kemerer metric suite was firstly introduced in 1991 and refined in 1996.

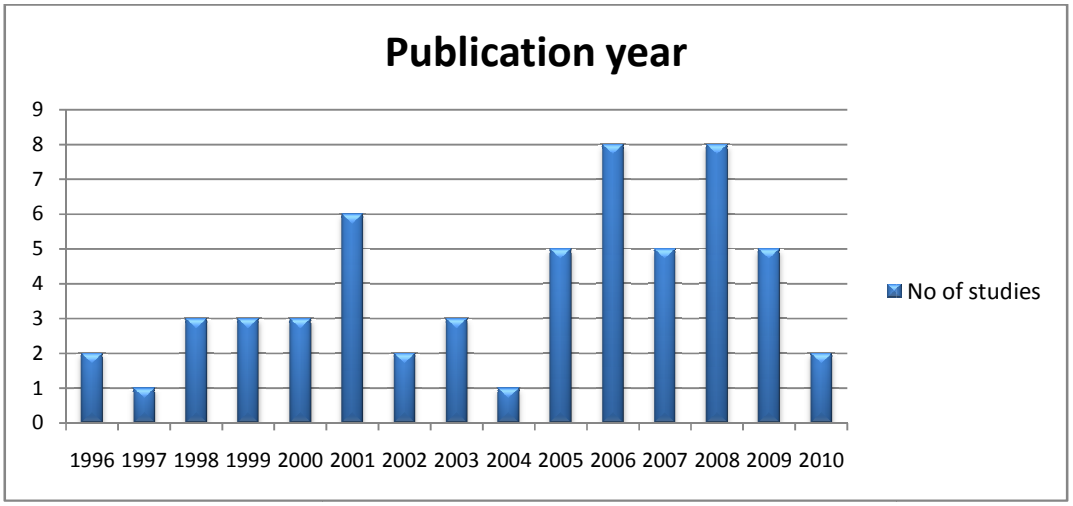

Figure 8: Studies by publication year

The figure also shows the increasing number of studies between 5-year periods. From 1995 to 1999, there are 9 relevant papers while from 2000 to 2004, the number of relevant papers are 15. This number is 33 for the period of 5 years from 2005 to 2009 which is more than double the number of previous period. We do not count 2010 since this year has not finished at the

time we conducted the research. This fact indicates the increasing interest of research community in this topic in recent years.

Table 13: Studies by publication channel

Publication channel	Type	No	%
IEEE Transactions on Software Engineering	Journal	12	21
International Conference on Software Engineering	Conference	6	10.3
Journal of Systems and Software	Journal	6	10.3
European Conference on Software Maintenance and Reengineering	Conference	3	5.3
Journal of Software Maintenance and Evolution	Journal	3	5.3
Software Quality Control	Book chapter	2	3.5
Journal of Information and Software Technology	Journal	2	3.5
Empirical Software Engineering	Journal	2	3.5
IEEE International Conference on Computer Systems and Applications	Conference	2	3.5
International Software Metric Symposium	Conference	2	3.5
International Symposium on Empirical Software Engineering	Conference	2	3.5
Technical Report	Technical report	2	3.5
Academic thesis	Academic thesis	2	3.5
Journal of Computer Science	Journal	1	1.8
Journal of Information Sciences	Journal	1	1.8
Journal of Object Technology	Journal	1	1.8
Lecture Notes in Computer Science	Book chapter	1	1.8
IEEE International Conference on Software Maintenance	Conference	1	1.8
International Conference on Signal Processing Systems	Conference	1	1.8
IEEE International Conference on Information Science and Engineering	Conference	1	1.8
Software Process Improvement and Practice	Journal	1	1.8
Studies in Computational Intelligence	Book chapter	1	1.8
Asia-Pacific Software Engineering Conference	Conference	1	1.8
Wuhan university journal of natural science	Journal	1	1.8
Total		**57**	**100**

Table 13**: Studies by publication channel** gives an overview of the studies according to publication channel. Publication channels describe the source where a study is published, e.g: journal, conference or technical report. It is shown that Transactions on Software Engineering, Conference on Software Engineering and Journal of System and software are most popular source of studies, which contributes 42% number of studies in total.

Regarding to the source type, 7% of selected studies comes from book chapter, 51% of those comes from International journals, 33% of those comes from International conferences. The remaining 9% of primary studies originates from local publication channels, such as university journals, academic theses or techinical reports.

6.2 Characteristic of data set

This section is used to describe characteristics of data set extracted from the primary studies. Since all empirical studies validate their hypothesis in some specific datasets, the characteristic of the data set is crucial to generalize the finding from these studies. It is also the case in studies of impact of design complexity on software quality. We would like to know this influence come from which project population. Besides, the information about the dataset is helpful to find a subgroup of the study population in which the impact of design metrics to software quality is consistent.

6.2.1 List of dataset

Table 14**: List of data set** shows a list of data set which is used to investigate the relationship between design metrics and external software quality. In the table, *Dataset name* column shows the conventional name taken from primary studies. Size, data type, language, domain and empirical type are characteristics of primary studies that are described in Section 5.3.2.

There are 59 dataset for correlation and regression analysis used in 57 primary studies. These datasets contain data for design metrics, measurement of fault proneness and maintainability. Among those, some datasets are investigated in many studies. NASA KC1 is the most common dataset for fault proneness studies, which are used by 5 different authors [83, 98, 99, 101, 103]. Besides, 3 versions of Eclipse are used in 3 primary studies [113, 128, 132] and 6 versions of Rhino are investigated in 2 studies but the same authors [82,105]. Different software versions are counted as different dataset because they are investigated separately in correlation and regression analysis. UIMS and QUES, firstly published by Li and Henry [49], are the most common datasets for maintainability study, which are used by 3 different authors [86,121,122]. A dataset from student project from an Indian university are used by the same author but with different investigated metric set in two studies ASK+09 [96] and ASK+07 [110].

Table 14: List of data set

Dataset Name	Original	Size	Data type	Language	Domain	Empirical type
NASA KC1	XHC08, PD07, SKM09, ZL06, GS08	145	Public	C++	Communication	Historical
Rhino 6 version	OEM+08, OEG+07	95	Public	Java	Internet	Historical
Eclipse 3 versions	ZXL10, SZP+06, SL08	5219	Public	Java	Software development	Historical
Succi 6 projects	JSP+06	93	Private industry	C++	Communication	Historical
Succi 2 projects	SPS+03	150	Private industry	C++	Communication	Historical
Piotr Erricson	TLG05	600	Private industry	C++	Communication	Historical
Sakura editor	WKK08	249	Public	C++	Document processing	Historical
JEdit	WKK08	538	Public	Java	Document processing	Historical
Collaborative Care	BS98	113	Private academic	C++	Information management system	Historical
iBatis	Mis002	264	Public	Java	Software development	Historical
Maryland experiment 2	BWD+00	113	Private	C++	Information management system	Observational
Indian University	ASK+09, ASK+07	85	Private	C++	Information management system	Observational
Mozilla 1.6	GFS05	3192	Public	C++	Internet	Historical
Briand Xpose	BMW02	144	Private	Java	Document processing	Historical
Emam framework	EBG+01	174	Private	C++	Communication	Historical
Maryland experiment 1	BBM96	180	Private	C++	Information management system	Observational
Briand LALO	BWL01	90	Private	C++	Software development	Historical

Tang 3 modules	TKC09	20, 27, 45	Private	C++	Document processing	Historical
Harrison 7 systems	HCN98	12	Private academic	C++	Control system	Observational
19 open source projects	ZX08	311 average	Public	Java	Unknown	Historical
	Ari06	unknown	Private industry	C++ and Java	Software development	Historical
	WK01	114	Private industry	C++	Communication	Historical
System-B	CKK+00	1420	Private industry	C++	Communication	Historical
ET++ framework	CKK+00	722	Private industry	C++	Software development	Historical
XForms,	CKK+00	221	Private industry	C++	Control system	Historical
	BJY01	227	Private industry	C++	Unknown	Historical
JFreeChart	WKK08	369	Public	Java	Graphic	Historical
UIMS & QUES	EE09, WHN+09, ZL07	39& 71	Public	Ada	Information management system	Historical
	WWS07	112	Private industry	C++	Information management system	Historical
	ABF04	136	Public	Java	Software development	Historical
50 open source projects	Mis05	96 average	Public	C++	Control system	Historical
Jmol	NAG05	169	Public	Java	Graphic	Historical
Jflex	NAG05	58	Public	Java	Document processing	Historical
E patient care	BS97, BS98	113	Private industry	C++	Information management system	Historical
LabPlot	Ali09	24	Public	C++	Graphic	Historical
Stellarium	Ali09	190	Public	C++	Graphic	Historical
	Ari01	> 1000	Private industry	SmallTalk	Software development	Historical

There are also some studies that investigate more than a single dataset, such as WKK08 [80], CKK+00 [94], NAG05 [89], Ali09 [131]. The main purpose of multiple datasets is to compare predictive performance across projects. Especially, there are two studies using large number of open source data sets, namely ZX08 [123] with 19 open source datasets and Mis05 [112] with 50 open source data set. However the detail information of those data sets is not reported. Therefore, we do not differentiate those datasets and treat them as single datasets.

6.2.2 Distribution of dataset by characteristics

In this section, datasets are shown in distribution of their characteristics. Figure 9 shows the distribution of datasets by programming language. The programming language means the language used in the software project in which the investigated data set is taken. It is shown that C++ and Java are two main investigated programming languages that are investigated. There are 38 datasets from C++ projects, which consumes 65% total of datasets and 18 datasets from Java projects, which consumes 30% of those. The remaining 3 datasets use object oriented programming language Ada and Smalltalk.

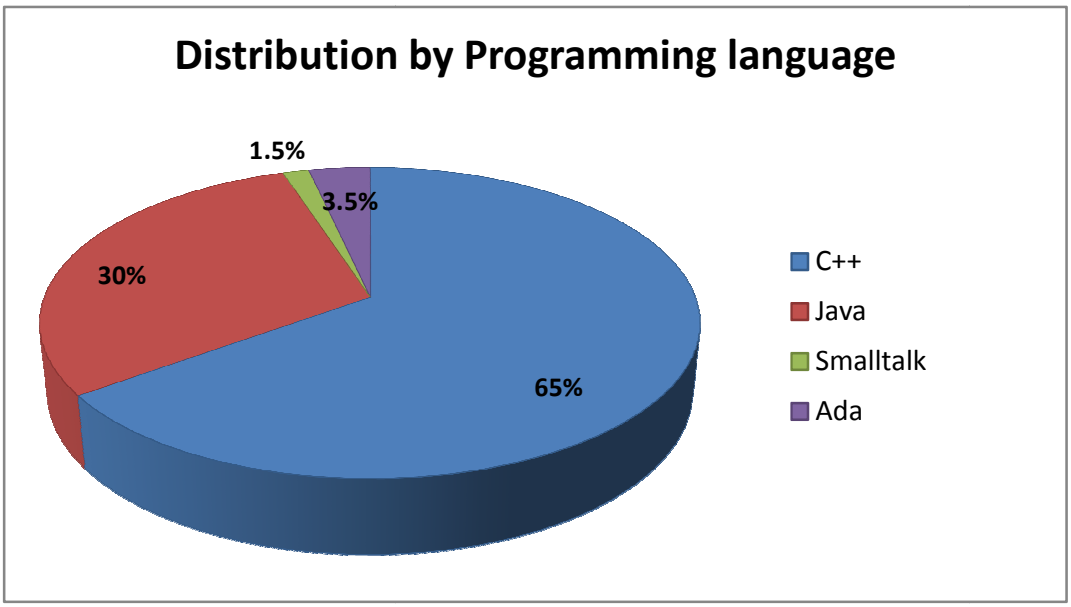

Figure 9: Distribution of datasets by programming language

Figure 10 shows the distribution of datasets by project business domain. Number of dataset in communication domain is the largest portion with 13 datasets (22% of total datasets). It is followed by number of datasets in software development domain and control system, which consumes 15% of total datasets for each. Information management systems has 8 datasets (14%), Internet has 7 datasets (12%) and Document processing has 7 datasets (12%). There are 4 datasets in Graphic domain, which consumes 7% of total datasets. There are two datasets whose domain information are not reported.

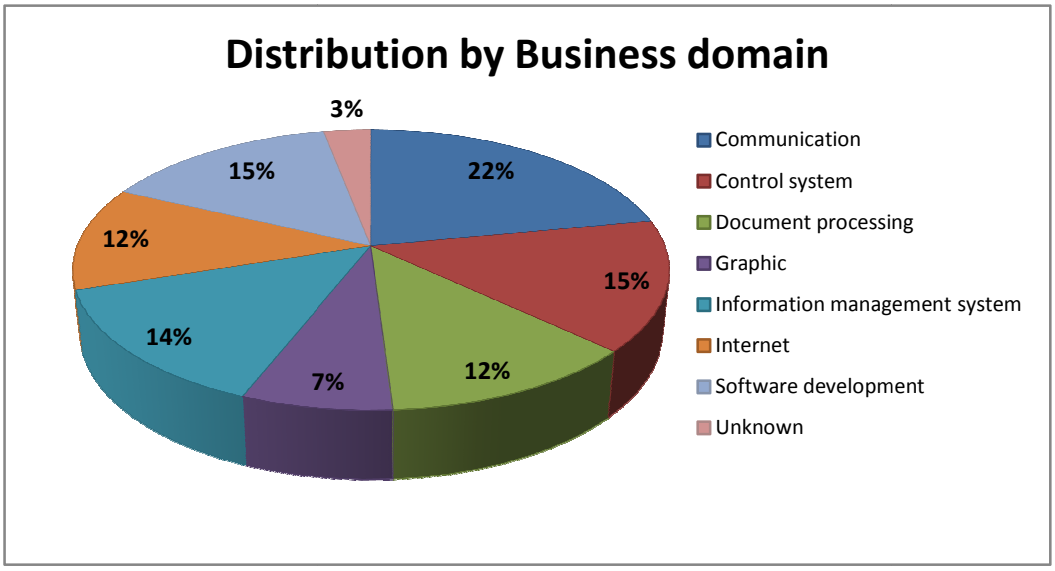

Figure 10: Distribution of datasets by domain

Figure 11 shows the distribution of datasets by their size. It is noticed that the size of dataset is the size of the software part which is taken as a sample for the investigation in the study. Dataset size is classified into 3 groups, namely small, medium and large group. Small size group contains datasets which have less than 100 data points (or classes). Large size datasets have more than 1000 classes. And in the between are medium size datasets (which have 100 to 1000 classes). The classification is ad hoc and refers from [4]. Figure 11 reveals the dominant of medium group with 32 datasets, which consumes 54% of total datasets. Small group has 20 datasets (34%) and large group has only 6 datasets (10%). There is a dataset whose size is not reported.

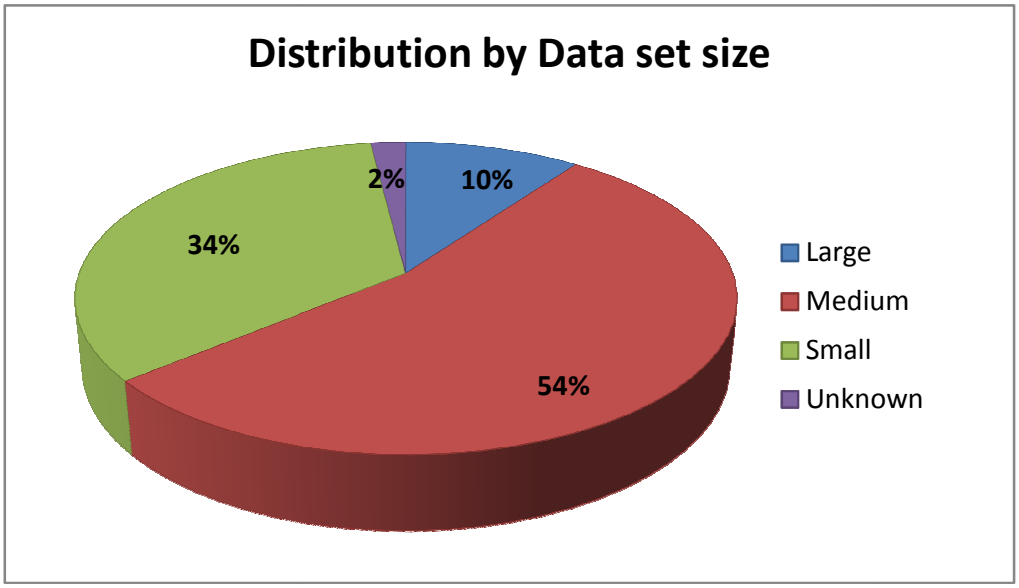

Figure 11: Distribution of datasets by size

Figure 12 shows the distribution of datasets by project type. There are 35 private datasets, which contributes 59% of total number of datasets. Among them, 10 datasets come from academic projects and 24 datasets come from industry project. The remaining 24 dataset are public, which contributes 41% of total number of datasets.

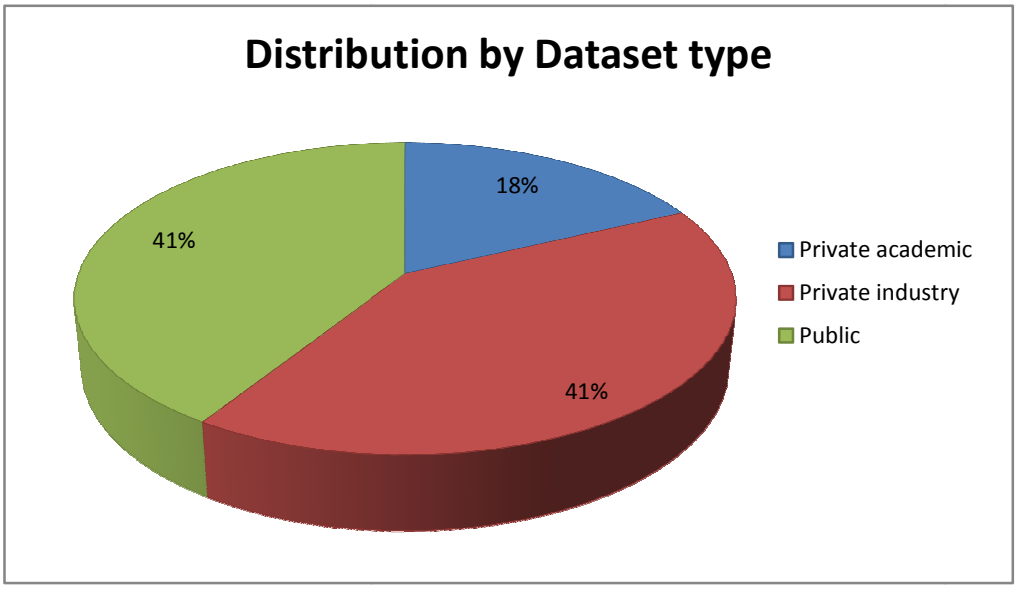

Figure 12: Distribution of datasets by type

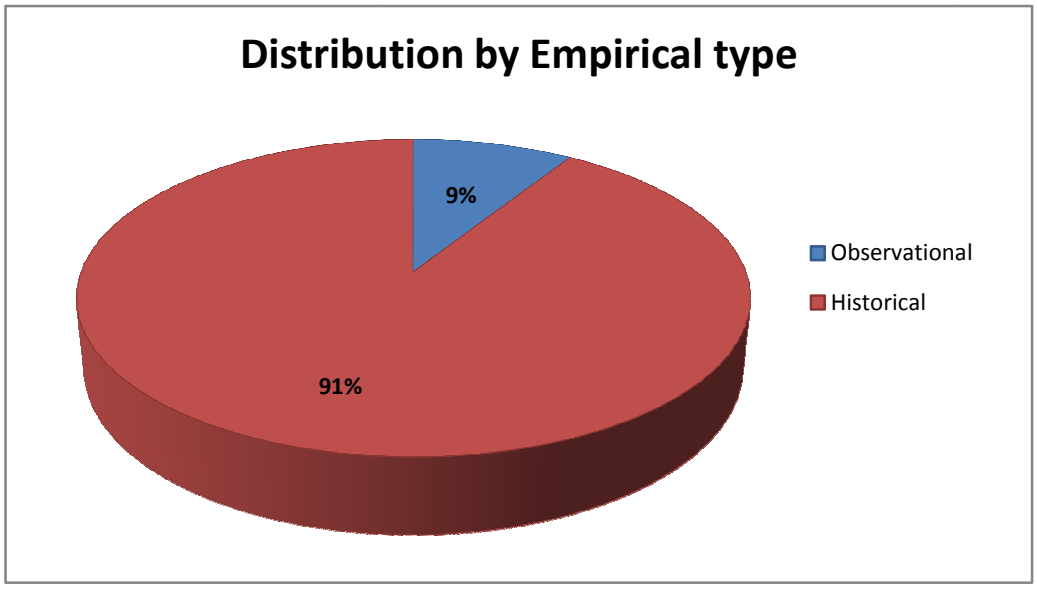

Figure 13: Distribution of datasets by empirical type

Figure 13 shows the distribution of studies by empirical design. Among 57 primary studies, there are only 5 observational studies, consumes 9% total number of studies. The 52 other studies use historical methods, which contribute 91% of total studies. It is noticed that all observational studies are from academic context.

6.2.3 Discussion

The figures shows the common characteristics of datasets used to investigate impact of design complexity on cost and quality. Overall, this influence is investigating datasets which mainly come from C++ or Java industry projects. The common research method is to collect the data from already complete projects. The number of investigated data points is various from few classes up to 5000 classes. Besides, there is a large portion of dataset that comes from an open source projects (almost 41% of total dataset, except for NASA, UIM and QUIS datasets).

With this information, it is interesting to know whether the influence of software design complexity on cost and quality is different across these characteristics. In particular, the number of questions such as:

- Is the impact of design complexity on quality in C++ project different from those in Java project?
- Is the impact of design complexity on quality in small-size dataset different from those in medium/ large-size dataset?
- Is the impact of design complexity on quality in Open source projects different from those in Closed source project?
- Is the impact of design complexity on fault proneness in pre defect collection phase different from those in post defect collection phase?

These questions will be answered in the process of answering Sub question 7 as a part of a search for moderator variable that can explain the disagreement among studies.

6.3 Research questions

This section presents the preliminary results for research questions.

6.3.1 SQ1: Which quality attributes are predicted using design complexity metrics?

6.3.1.1 Result

To identify the available external quality attributes in correlation and regression studies, we investigated the received studies after selection based on detail inclusion, exclusion criteria (step 3 in Section 5.4.2). Figure 13 shows the external quality attributes that are investigated for the relationship with design complexity in 100 received papers. There are four main investigated costs and quality attributes, namely reliability, maintainability, reusability and development effort that are refined in 9 sub categories as below:

- Defect density: includes studies that investigate relation between total known defects divided by the size (defect density) and design complexity. The studies relate to the prediction of residual defects.
- Fault proneness: includes studies that investigate the relation between probabilities of finding a defect in a class (fault proneness) with design complexity. The studies often relate to a classification of a class to faulty or non faulty group by value of design complexity metric.
- Vulnerability, security: includes studies that investigate the relation between numbers of faults that lead to security failure and design complexity
- Testability: includes studies that predict effort required for software testing due to level of design complexity [59]
- Maintainability: includes studies that predict time, efforts required to correct bugs, improve performance or adapt software to a change environment [60].
- Volatility: studies that investigate the relation between the numbers of enhancement in a class with design complexity.
- Debugging cost: includes studies that investigate the relation between effort and time for fixing a class with its design complexity. The studies often conduct correlation analysis or regression model of fixing time of a class and design complexity of the class.
- Development effort: include studies that investigate the relation between cost and time for development with its design complexity. The studies often conduct correlation analysis or regression model of development effort of a class and design complexity of the class.
- Refactoring effort: include studies that investigate the relation between efforts for refactoring a class with its design complexity. The studies often conduct correlation analysis or regression model of refactoring effort of a class and design complexity of the class

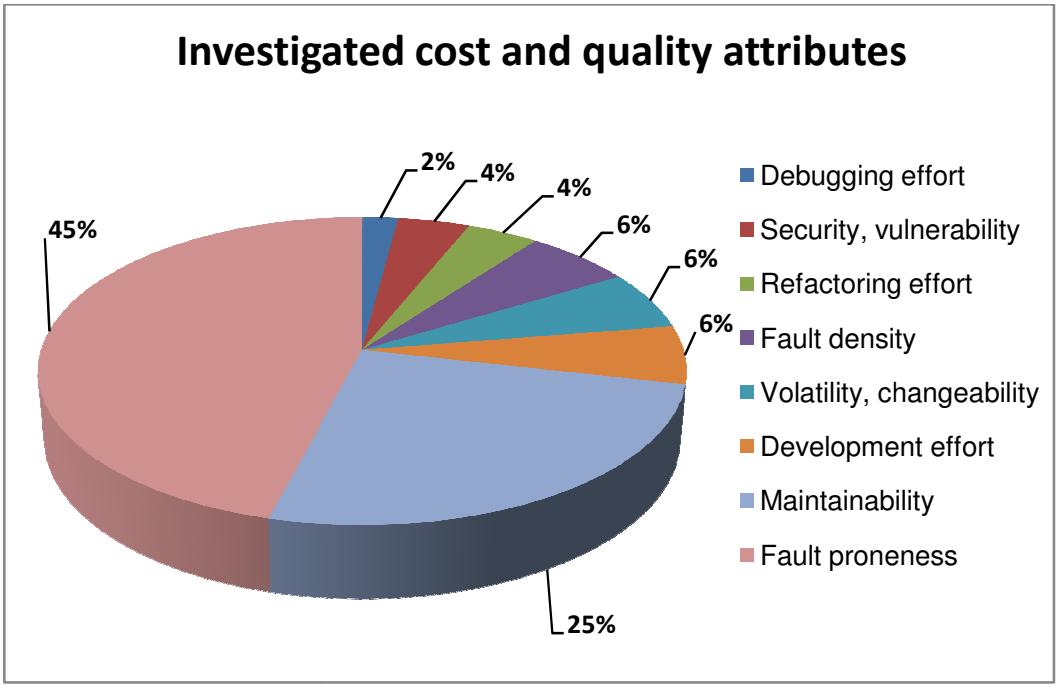

Figure 14: Investigated cost and quality attributes

Figure 14 reveals the domination of studies that focuses on fault proneness (45% of total studies) and maintainability (25% of the studies). The other quality attributes, such as fault density, testability or development effort contributes from 2% to 6% of total studies (2 to 6 studies per each attribute). Not mentioned about variability in reported information can reduce the number of usable papers, these number is already too small for conducting a quantitative aggregation. Therefore, we decide to skip those quality attributes and focus on fault proneness and maintainability for answering upcoming research questions.

6.3.1.2 Discussion

Fault proneness is the most frequently investigated dependent variable. First reason for this is that fault proneness is an indirect way to measure reliability, which is always an important quality, attributes to consider [67]. Secondly, fault proneness has an interesting application in practical since it helps to focus testing effort on high risk software modules [37]. Thirdly, the collection of fault data (including the assignment of faults to classes) is less difficult than collecting other data related to classes (such as effort). Besides, it is not necessary to collect all reported defects to classify a class as faulty or not.

Among selected studies, there are some different sources of software fault. Some papers use number of found defects during testing phase [79, 96, 110]. Other choices are number of class revision [108, 116], number of change in repository [113, 123, 138] and number of customer reported failure [108, 116]. This variation is mainly reflected in reported moderator variable

(such as defect collection phase). Therefore, we assume that it does not significantly affect the result of aggregation.

Maintainability is the second most frequently investigated dependent variable. There are four common choices for maintainability measures, namely maintainability index [89, 112, 123], maintenance time [87] and change volume [121, 131]. The maintainability index, which is used in 50% of the total maintainability papers, is calculated with certain formulae from lines-of-code measures, McCabe measures and Halstead complexity measures. Maintenance time, which is used in 35% of the total papers, is amount of time needed to understand, test and maintain a software unit. Change volume, which is used in 15% of the total papers, measures the amount of changes by number of changed items or number of changed LOC. Within maintainability papers, two choices of dependent variables are change proneness (the probability that a software unit is likely to change) and maintenance effort. The variety in dependent variables and measures indicates the difficulty in quantitatively aggregating maintainability papers.

Initially, Software cost is one of research target that we want to find its relationship with design complexity. However, the literature review reveals that there is not enough number of studies about software cost for further synthesis. Therefore, this attribute is left out from further research questions, though it is interesting to know.

6.3.2 SQ2: What kind of design complexity metrics is most frequently used in literature?

Result from SQ1 navigates our research focuses on software fault proneness and maintainability. In this question, we investigate the complexity dimension that is most frequently used in literature.

6.3.2.1 Result

To investigate the amount of research interest in each complexity dimension, we count the number of studies for each dimension.

Figure 15 shows investigated complexity dimensions that are reported in fault proneness studies. In 38 studies that investigate fault proneness, the common design complexity dimensions are: coupling, cohesion, inheritance, polymorphism and scale. The most concerned complexity dimention is coupling with 31 studies, which are used in 87% total of studies. It is followed by scale with 30 studies (84%), inheritance with 29 metrics (81%), cohesion with 24 metrics (68%) and polymorphism with three studies (9%).

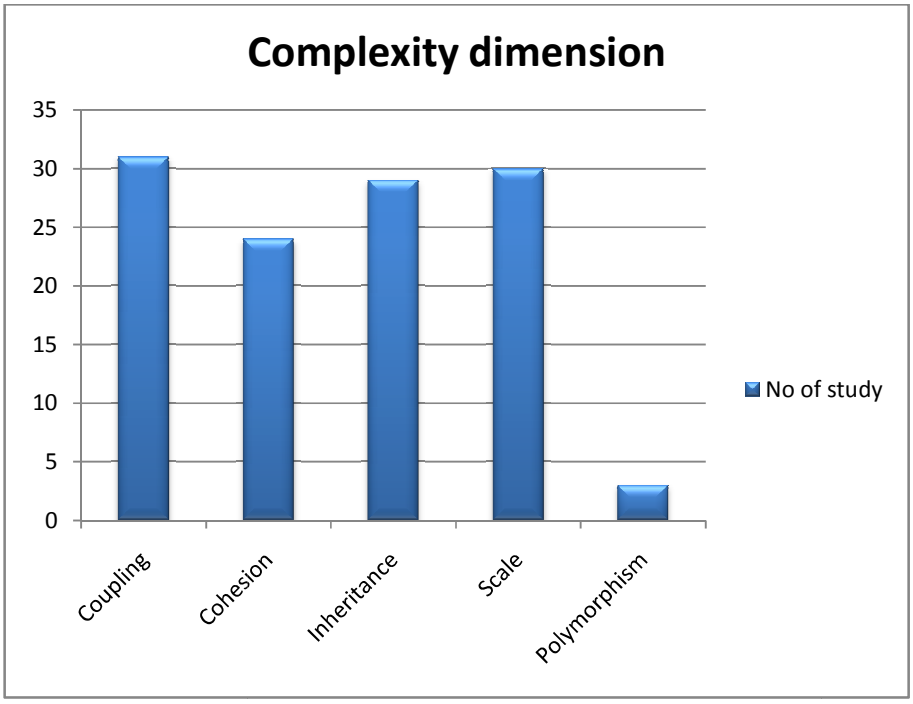

Figure 15: Number of studies in complexity dimensions – fault proneness

Figure 16 shows the investigated complexity dimensions that are reported in maintainability studies. In 19 studies that investigate maintainability, the common design complexity dimensions are: coupling, cohesion, inheritance, polymorphism and scale.The most concerned complexity dimention is coupling with 17 studies. It is followed by scale and inheritance with 13 investigated studies, cohesion with 9 studies and no study investigate polymorphism.

Among studies, most of the metrics are proposed and firstly investigated in the time period from 1990 to 1999. It is the time when object-oriented programming was gaining its popularity in practice (in 1990 standard reference manual which introduce basic for fully object oriented C++ was published [61]; in 1995 Java is firstly introduced). The increasingly large number of application written in C++ and Java has motivated proposals of vast variety of object oriented design metrics. They are the Chidamber& Kemerer metric suite [143, 144], MOOD metric set by Abreu [147], QMOOD metric set by Bansiya [151] and polymorphism metrics by Benlarbi [146] and frameworks for measuring coupling and cohesion by Briand [27, 29]. The complete list of complexity metrics are shown in Appendix B. In the Appendix, *Definition* column give the brief description of the metric. *Original* is a reference to the paper, which firstly introduced the metric. The *Type* is one of the complexity dimensions and *Predictor for* is the quality attributes that are predicted by using the metric.

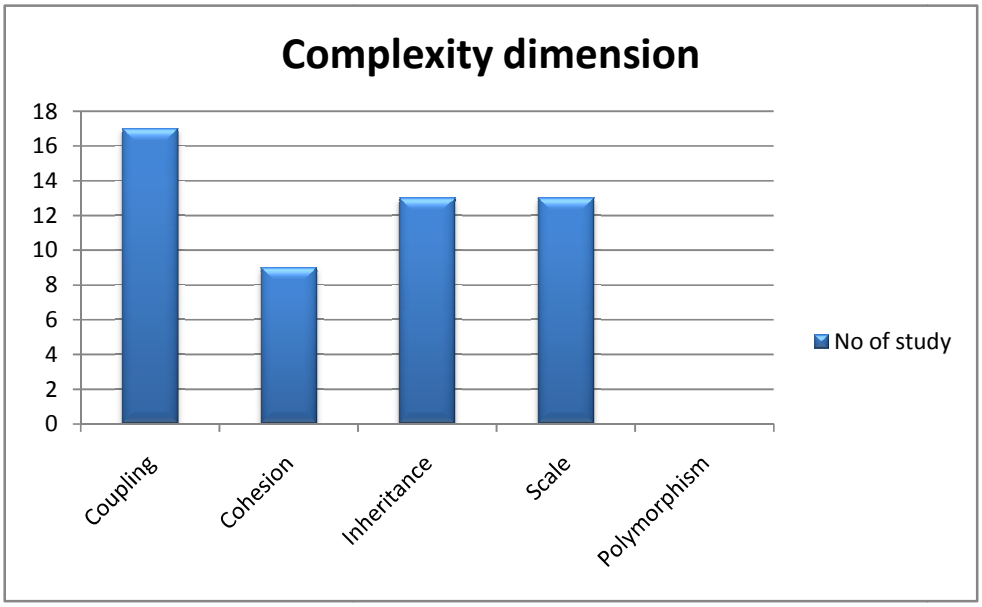

Figure 16: Number of studies in complexity dimensions – maintainability

6.3.2.2 Discussion

Among five design complexity dimensions, the order of number of investigated dimensions is consistent for both fault proneness and maintainability studies. Coupling is the dimensions that have largest number of studies, and it is followed by scale, inheritance and cohesion respectively. The number of studies could indicate the interest of researchers on the design complexity dimension.

Scale is conventional dimension of any measurement framework and has been used from the beginning of software program. Therefore, we cannot find the original or the first use of some scale metrics in correlation and prediction study. In object oriented environment, the normal scale metrics are the number of classes, methods or attributes. Inheritance is a feature of OO paradigm. Inheritance is intuitively easy to understand by counting the inheritance hierarchy level or inherited items. Polymorphism is also a new feature in OOP. The impact of polymorphism on complexity normally is hidden in scale our coupling metrics. Therefore it is not many metrics that measure polymorphism explicitly.

6.3.3 SQ3: Which design complexity metrics is most frequently used in literature?

After investigating the most common complexity dimension, we identify the most common complexity metrics in literature with the same method.

6.3.3.1 Result

To identify the most concerned design metrics in selected studies, the number of investigation are counted for each design metrics. The metric is counted if it is investigated in correlation analysis (Spearman, Pearson, etc) or regression analysis (univariate or multivariate models). The result of SQ2 give us 116 different design metrics in 38 fault proneness studies and 59 identical design metrics in 19 maintainability studies.

Figure 17 shows the distribution in metrics usage among fault proneness studies. It is noticed that there are few metrics are investigated in many studies (from 21 to 28 studies) and the remaining metrics are investigated in few studies (from 1 to 12 studies). This gap reveals some design metrics are highly concerned in investigating fault proneness.

Table 15 shows the list of most frequent used design metrics to evaluate or predict fault proneness. It is shown that all highly concerned metrics come from Chidamber & Kemerer metric set. The two inheritance-based metrics, namely NOC and DIT are most investigated design metrics with 28 studies (74% total of studies) and (71% total of studies), respectively. It is followed by CBO, LCOM and WMC (unified weight version) in Chidamber & Kemerer refined metric set [143] with 22 studies per each (58% total of studies). There are 21 studies that investigate RFC [143], which consumes 55% total of fault proneness studies.

Table 15: Most frequently used design metrics in fault proneness studies

Metric	Definition	Original	Type	# of studies
NOC	number of classes that directly inherit from a given class	Chidamber & Kemerer 1991[144]	inheritance	28
DIT	length of the longest path from the class to the root of inheritance hierarchy	Chidamber & Kemerer 1991 [144]	inheritance	27
CBO	number of other classes to which it is coupled, included inheritance-based coupling	Chidamber & Kemerer 1994 [143]	coupling	22
LCOM2	LCOM1 minus count number of pairs of methods that do (average percentage of the method in a class using an attribute and subtract from 100%)	Chidamber & Kemerer 1994 [143]	cohesion	22
WMC	number of all methods of a class	Chidamber & Kemerer 1991[144]	size	22
RFC1	RFCinf + not count methods indirectly invoked by methods in M	Chidamber & Kemerer 1994 [143]	coupling	21

Maintainability studies show a different picture from fault proneness studies. There are no gaps in the distribution of metrics usages in maintainability studies. The number investigations for design metrics are from 1 to 9. To illustrate the high concerned metrics, we pick the first four most frequent investigated design metrics (as shown in Table 16). Not surprisingly, all highly concerned design metrics come from Chidamber & Kemerer metric set. On top of the list is WMC with 9 studies, which consumes 47% total of studies. It is followed by RFC with 8 studies (42%), DIT with 7 studies (37%) and NOC with 6 studies (32%).

Table 16: Most frequent used design metrics in maintainability studies

Metric	Definition	Original	Type	# of studies
WMC	number of all methods of a class	Chidamber & Kemerer 1991 [144]	size	9
RFC1	RFCinf + not count methods indirectly invoked by methods in M	Chidamber & Kemerer 1994 [143]	coupling	8
DIT	length of the longest path from the class to the root of inheritance hierarchy	Chidamber & Kemerer 1991 [144]	inheritance	7
NOC	number of classes that directly inherit from a given class	Chidamber & Kemerer 1991 [144]	inheritance	6

6.3.3.2 Discussion

Figure 17 shows the distribution of usage of metrics among correlation and prediction studies. Interestingly, the distribution of usage of design metrics are followed Pareto-like distribution 20/60: the 20% most common metrics are investigated in 60% of the studies.

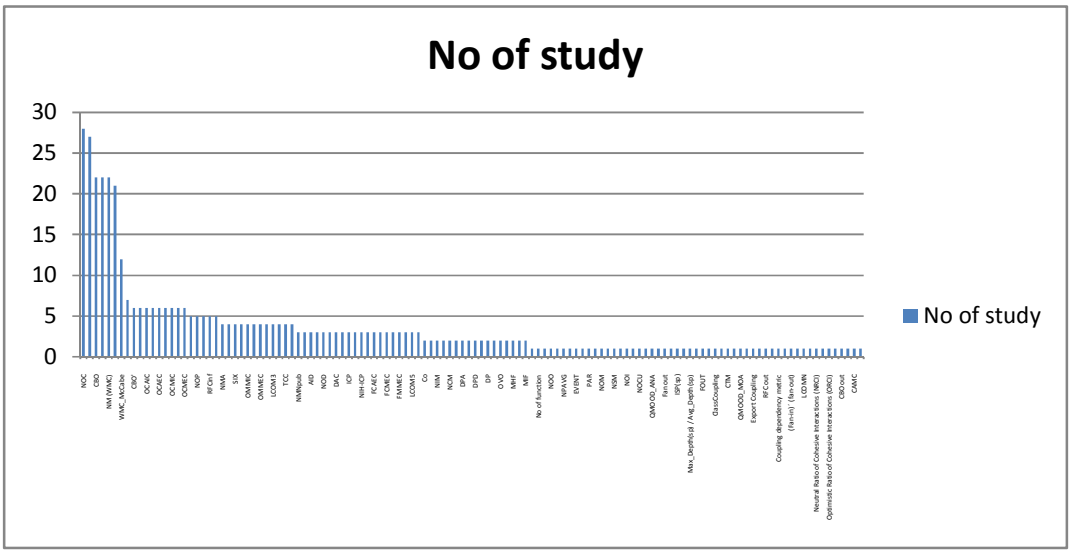

Figure 17: Distribution of usage of design metrics

The domination of Chidamber & Kemerer metric set in correlation and prediction studies could comes from the fact that they are one of the first complete design metrics with measure of all OO features, namely coupling, cohesion, inheritance and scale [144]. The metric set has strong foundation since they were developed based on the ontology of Bunge and validated against Weyuker's measurement theory principles [144]. More importantly, Chidamber et al. claimed that these measures can aid users in understanding design complexity, in detecting design flaws and in predicting certain project outcomes and external software qualities such as software defects, testing, and maintenance effort and provided an empirical sample of these metrics from two commercial systems [25]. Besides, the metric set is simple, well-understood [108] and straightforward to collect. These reasons seem to invoke great enthusiasm among researchers and software engineers, and a great amount of empirical studies have been conducted to evaluate those metrics. Over time, this metric suite is gradually growing in industry acceptance. In particular, they are being incorporated into development tools such as Rational Rose [97] and JBuilder [98]. Many others tool also auto collection feature for C&K metric set make it very easy to collect such as WebMetric [116] and JMetric [88].

Among fault proneness studies, the largest amount of studies for NOC, DIT but not WMC, RFC, CBO or LCOM can be explained by the variation in the use of those metrics. WMC has two versions, namely WMC unified weight and WMC weighted by Mc Cabe (34 studies if counted both investigations for both versions). RFC has 4 different versions which are investigated in 28 studies in total. CBO has 4 adjusted versions which are validated in 30 studies. LCOM has 6 different versions and they are investigated in 39 studies in total. Among 6 Chidamber and Kemmerer metrics, only NOC and DIT are used consistently among studies (the difference is only in granularity which we decided to skip).

6.3.4 SQ4: Which design complexity metrics are potentially predictors of quality attribute?

To find whether a design metric is a potential predictor of external attributes, we test each design metric with the following hypothesis:

H0: There is no impact of metric X on quality attribute Y.

Due to the variation of the use of design metrics in primary studies, we use vote counting to test the hypothesis H0 (detail of method is in Section 4.4). Assumed that each study report a significant value (p value) of correlation coefficient or regression coefficient between design metric X and quality attribute Y. It is worth noticed that some studies report the actual value of the significance test, but some others only report the significant level, which is represented by + if the correlation test is significant at 0.05 and by ++ if the test is significant at 0.01. To maximize number of included studies, we select the significance level at 0.05.

Among 116 design metrics investigated in fault proneness studies and 59 design metrics investigated in maintainability studies, we select the metrics whose significance level are reported in at least 6 different datasets. It is noted that these datasets are not necessary to come from different studies.

We apply the vote counting procedure on two outcomes and differentiate between positive impact and negative impact of X on Y. In the first case, the study is counted as a success if the reported p value is less than or equal to 0.05 and coefficient is positive (positive impact). The studies with p value are greater than 0.05 (non-significant) or the studies significant is counted as failure. In the second case, the study is counted as a success if the report coefficient is significant and negative (negative impact). The studies that report non-significant or significant but negative value of coefficient is counted as a failure. The null hypothesis is rejected if the portion of success in total studies is greater than 0.5. Otherwise, we accepted the hypothesis.

6.3.4.1 Result for fault proneness

In fault proneness studies, the most common statistical methods are Spearman correlation and univariate logistic regression. Therefore, we conduct vote counting separately for Spearman coefficient and Logistics regression coefficient.

Table 17: Hypothesis test for significantly positive Spearman coefficients – fault proneness

Metric	No of studies	No of data set	No of +	No of -	No of non significant	% of +	Positive hyp. test
NOC	8	19	6	1	12	32%	accept
DIT	7	14	2	0	12	14%	accept
CBO	8	17	10	0	7	59%	reject
LCOM	7	14	6	0	8	43%	accept
WMC	9	26	18	0	8	69%	reject
RFC	7	15	9	0	6	60%	reject
WMC Cabe	3	16	11	0	5	69%	reject
SDMC	1	6	6	0	0	100%	reject
AMC	1	6	6	0	0	100%	reject
NIM	1	6	6	0	0	100%	reject
NCM	1	6	6	0	0	100%	reject
NTM	1	6	6	0	0	100%	reject

a. Case 1: test for positive impact

Table 17 describes the null hypothesis test for significant positive impact of several design metrics on fault proneness in Spearman correlation result. There are 12 design metrics that are investigated in more than 6 data sets. Out of these 12 metrics, 9 metrics has portion of success more than 50%, namely CBO, WMC, RFC, WMC Mc Cabe, SDMC, AMC, NIM, NCM and NTM and therefore null hypothesis is rejected for these metrics. The 3 other metrics that have less than 50% of success portion are NOC, DIT, LCOM. Hence, we accept the null hypothesis: with spearman test, there is no positive impact of NOC, DIT or LCOM on fault proneness.

Table 18 describes the null hypothesis test by using vote counting for univariate logistic regression coefficient in fault proneness studies. There are 26 design metrics that are investigated in more than 6 data sets. Out of these 26 metrics, 14 metrics has portion of success more than 50 percent and therefore we can reject null hypothesis in these metrics. The other 12 metrics that have less than 50% of success portion are NOC, DIT, OCAEC, OCMIC, OCMEC, AHF, AIF, MIF, MHF, DAM, DCC and MFA. Therefore, we accept the null hypothesis: with univariate logistic regression, there is no positive impact of these metrics on fault proneness.

b. Case 2: test for negative impact

Similarly, the vote counting procedure is applied to test null hypothesis for negative impact of design metrics on fault proneness. It is easy to see that none of the listed design metric can reject the null hypothesis. In other words, with Spearman and univariate logistics regression, there is no negative impact of design metrics on fault proneness.

6.3.4.2 Result for maintainability

Among maintainability studies, there is a variety in usage of statistical method, such as Pearson, Spearman, Kendall, linear regression, logistic regression, etc. We only found the fairly large amount of significant value reported for Spearman coefficient. It is a reason why vote counting is only performed for Spearman.

a. Case 1: test for positive impact

Table 19 describes the null hypothesis test in Spearman correlation in maintainability studies. There are 6 design metrics that are investigated in more than 6 data sets. Out of these metrics, 4 metrics has portion of success more than 50 percent and therefore we can reject null hypothesis in these metrics. The other 2 metrics that have less than 50% of success portion are NOC and DIT. Therefore, we accept the null hypothesis: with spearman test, there is no positive impact of NOC, DIT or LCOM on fault proneness.

Table 18: Hypothesis test for significantly positive Odds ratios – fault proneness

Metric	No of studies	No of data set	No of +	No of -	No of non significant	% of +	Positive hyp. test
NOC	12	20	2	5	13	10%	accept
DIT	13	21	6	0	15	29%	accept
CBO	12	20	17	0	2	85%	reject
LCOM	9	15	9	0	6	60%	reject
WMC	11	24	21	0	3	88%	reject
RFC	12	22	20	0	2	91%	reject
WMC McCabe	9	21	21	0	0	100%	reject
OCAIC	6	6	5	0	1	83%	reject
OCAEC	6	6	3	0	3	50%	accept
OCMIC	6	6	3	0	3	50%	accept
OCMEC	6	6	3	0	3	50%	accept
SDMC	2	9	9	0	0	100%	reject
AMC	2	9	8	0	1	89%	reject
NIM	2	9	9	0	0	100%	reject
NCM	2	9	9	0	0	100%	reject
NTM	2	9	8	0	1	89%	reject
NOA	4	6	4	0	2	67%	reject
NMO	4	6	6	0	0	100%	reject
AHF	1	6	2	0	4	33%	accept
AIF	1	6	2	0	4	33%	accept
MHF	1	6	0	2	4	0%	accept
MIF	1	6	2	0	4	33%	accept
DAM	1	6	0	2	4	0%	accept
DCC	1	6	1	0	5	17%	accept
MFA	1	6	2	0	5	33%	accept
CIS	1	6	5	0	1	83%	reject

Table 19: Hypothesis test for significantly positive Spearman coefficients - maintainability

Metric	No of studies	No of data set	No of +	No of -	No of non significant	% of +	Positive hyp. test
WMC	5	7	6	0	1	86	reject
RFC	6	8	8	0	0	100	reject
DIT	3	5	2	0	3	40	accept
NOC	4	6	2	0	4	33	accept
MPC	3	6	6	0	0	100	reject
DAC	5	6	4	0	2	67	reject

b. Case 2: test for negative impact

Similarly, the vote counting procedure is applied to test null hypothesis for negative impact of design metrics on fault proneness. It is easy to see that none of the listed design metric can reject the null hypothesis.

6.3.4.3 Discussion

Fault proneness:

The Spearman correlation coefficient assesses the monotonic relationship between a metric and fault proneness logistic regression indicates the ability of the metric to predict fault proneness. In both cases, we cannot reject the null hypothesis for NOC and DIT. It indicates that vote counting shows no relationship between these inheritance metrics with fault proneness. In the other hand, null hypothesis is rejected for CBO, WMC, RFC, WMC Mac Cabe, SDMC, AMC, NIM, NCM and NTM in Spearman correlation as well as univariate logistic regression. This result confirms the relation of these metrics to predict fault proneness. Consequently, these metrics should be considered when one attempts to build fault proneness prediction models.

In case of LCOM, it isn't correlated to fault proneness in Spearman correlation but it is significant in logistic regression. The other metrics are rejected for null hypothesis in logistic regression model but are not investigated in Spearman test, namely OCAIC, NOA, NMO and CIS. All of these metrics should be considered also in attempt to construct fault proneness logistic regression model.

Maintainability:

We also cannot reject the null hypothesis for NOC and DIT due to the portion of either positive or negative success is smaller than 50% (see section 4.4). The other four metrics are rejected for null hypothesis, in which RFC and MPC are positive significant to maintainability in 100%

cases. Therefore, RFC, MPC and the two others (WMC, DAC) should be considered when constructing a maintainability prediction model.

Both quality attributes

In fault proneness and maintainability, vote counting shows that there is no evidence about relationship of NOC, DIT to external quality attributes. WMC, RFC is shown to be potential good predictors for external quality attributes in all the cases.

6.3.5 SQ5: Which design complexity metrics are helpful in constructing prediction model?

6.3.5.1 Result for fault proneness models

In order to find the meaningful metrics for fault prediction model, we investigated 22 multivariate regression models reported in 38 fault proneness studies. Table 19 summaries these models with the design metrics used, modeling technique, model validation method and accuracy value.

There are 5 statistical **modeling techniques** used to construct models, namely logistic regression (logistic), linear regression (linear), weighted least square (WLS), zero-inflated negative binomial regression (ZINBRM) and Multivariate Adaptive Regression Splines (MARS). Logistic regression is the dominating technique that is used in 17 models out of 22, consumes 77% total of models.

Model validation is method used to test the predictive performance of the model. There are three main model validation methods, namely local dataset, k fold cross validation (k cross validation) and version across validation. Models that are validated by the same local dataset are evaluated by a goodness-of-fit analysis to assess how well the model is at explaining the data that were used to build the model. Another choice for validating a prediction model is k-fold cross validation and version cross validation. In k-fold cross validation, a local data set is divided into n subset. Then, k models are built where each of the n subsets is successively used as a test set, while the other *k-1* subsets form the training set. In version-cross validation, the training set consist of one or several releases of a software system, while the testing set consists of another version of the same system or even releases of another system. Table 20 shows the most common validation method is k-cross validation with 10 models, which consumes 45% of the total models. There are 4 models used version-across-validation (18%), 2 models using whole local data set (9%) and 6 models without validation (28%).

Table 20: Multivariate regression model for fault proneness prediction

Studies	Best metric suite	Modeling technique	Model validation	Accuracy
OEM+08	AMC, NTM	Logistic	version-across validation	Average concordant 67.6%, discordant 15.8%, ties 16.6%
BWL01	CBO', ICP_L, ICH, NIH_ICP, OCAIC, OCMIC	Logistic	9-cross validation	Completeness 92%, accuracy 90%,
SKM09	CBO, SLOC, LCOM, NOC, RFC	Logistic	10-cross validation	Completeness 81.1%, precision 71.7%
XHC08	CBO, WMC, RFC, SLOC	Logistic	no	
GS08	CBO,DIT,LCOM, RFC, WMPC, SLOC, NMC	Logistic	local data set	Precision 76.8%, accuracy 79.3%, recall 71.6%
YSM02	CBOout, RFCin, NMC, NOC, DIT, LCOM	Logistic	no	
BBM96	DIT, RFC, NOC, CBO	Logistic	local data set	
BM99	DPA, SP, NIP	Logistic	no	
ASK+09	OMMIC, NMI, NM, RFC, PM, MPC, ICH	Logistic	9-cross validation	Sensitivity 86.4%, specificity 98%, completeness 86%
ASK+07	OMMIC, WMC, RFC	Logistic	9-cross validation	Sensitivity 81.9%, specificity 93.8%,
BWD+00	RFCinf, NOP, RFC1_L, FMMEC, NIH_ICP_L, CLD	Logistic	9-cross validation	Completeness 92%, accuracy 78%
ZL06	SLOC, RFC, NOC, CBO, LCOM	Logistic	10 cross validation	Completeness 76.14%, precision 69.7%, accuracy 62.1%
PD07	CBO, RFC, WMC, SLOC, LCOM	Logistic	10-cross validation	Sensitivity 59.7%, specificity 85.2%, precision 75.2%
GFS05	CBO, DIT, WMC, SLOC	Logistic	version-across validation	Completeness 55.9%, precision 58%, accuracy 84%
SL08	CTA, CTM, NOA	Logistic	version-across validation	Average ROC 0.63
OEG+07	CBO, DIT, LCOM, NOC, WMC Mc Cabe	Logistic	version-across validation	Average concordant 80%

SK03	SLOC, WMC, DIT, CBO, CBO*DIT	WLS	no	
SPS+03	RFC, DIT	ZINBRM	no	
BMW02	NIP, OCMIC, OCMEC	Linear	10-cross validation	Completeness 62%, accuracy 73.6%
BMW02	NIP, OCMIC, OCMEC,DIT	MARS	10-cross validation	Completeness 73%, accuracy 68%
CS00	EVENT, INHRTS	Linear	no	Above 70%
ZXL10	LOC + CCMax/ NLM	Logistic	10 cross validation	Concordant above 70%
SKM09	SLOC	Logistic	10 cross validation	Sensitivity 72.9%, specificity 68.6%, precision 70.34%, completeness 82.4%
ZL06	SLOC	Logistic	10 cross validation	Precision 71.03%, correctness 63.79%, completeness 77.05%
GFS05	SLOC	Logistic	10 cross validation	Precision 66.85%, correctness 72.98%, completeness 54.58%

The last column in Table 21 shows accuracy measure and its value for each model. It is shown a different **accuracy measures** which is classified into 3 main groups, namely confusion matrix-based measure, ROC curve based measure and concordant, discordant based measure. Confusion matrix-based measures, such as sensitivity, specificity, precision, completeness are derived from the confusion matrix in Table 20.

Table 21: Confusion matrix

		Actual	
		Positive	Negative
Predicted	Positive	True positive (TP)	False positive (FP)
	Negative	False negative (FN)	True negative (TN)

In Table 21, the prediction of a model has four possible outcomes: if a class is faulty and predicted accordingly, it is counted as true positive (TP); if it is wrongly predicted as non faulty, it is counted as false negative (FN). Conversely, a non faulty class is counted as true negative (TN) if it is predicted correctly or as false positive (FP) otherwise [El Eman 1999]. The metrics are defined as bellows:

- Sensitivity (recall): the percentage of classes correctly predicted to be fault prone: TP/ FN + TP
- Specificity: the percentage of non-occurrences correctly predicted, i.e. classes predicted not to be fault prone: FP/ TN + FP
- Completeness: number of faults in classes classified fault-prone, divided by the total number of faults in the system:
- Precision: the number of classes that are predicted correctly, divided by the total number of classes: TP / TP + FP
- Accuracy: the percentage of classes that are predicted correctly, either faulty or non-faulty: TP + TN / N

The other accuracy measure is Receiver operating characteristic (ROC) curve. ROC is a plot of sensitivity on y-coordinate versus 1 – specificity on the x-coordinate. We select many cutoff points between 0 and 1, and calculated sensitivity and specificity at each cut off point. The optimal choice of cutoff point is where value of sensitivity and specificity is maximized.

Concordant, discordant and ties are used by Olague et al. 2008 to evaluate predictive effectiveness of a univariate logistic regression model [82]. Concordant, discordant and ties are calculated by pairing the classes that have different actual values of the dependent variable. A pair is concordant if a faulty class has a larger predicted probability of being faulty. A pair is discordant if a faulty class has a smaller predicted probability of being faulty. A pair is tied if the predicted probabilities are equal. The predictive power of a model is assessed by Pc and Pd given by the formula:

$$P_c = \frac{n_c}{O_p \times O_n} \times 100\% \qquad P_d = \frac{n_d}{O_p \times O_n} \times 100\%$$

With nc and nd are the number of the concordant, discordant pairs in the data set, respectively. Op and On are the sets of positive and negative classes in the data set.

Among 16 model that has validation part, confusion matrix based method is used in 12 models, consumes, concordant, discordant is used in 3 models and ROC is used in 1 model.

Each model is also reported with best suit metric for prediction. The selection of the metric suit can be ad hoc by the researcher or step-wise selection from the larger set of metrics. Among 22 multivariate models, 18 models contains coupling metrics (82%), 15 models contain scale metrics (68%), 12 ones contain inheritance metrics (55%). Only 8 out of 22 models use cohesion metric (36%) and 3 models has polymorphism metrics (14%).

Table 22 shows a list of most common metrics used in 22 multivariate models.

Table 22: Most frequently used metrics in multivariate models

Metric	Original	Type	No of usage
RFC	[144]	coupling	9
CBO	[144]	coupling	9
DIT	[143]	inheritance	8
LCOM	[144]	cohesion	6
WMC	[143]	size	6
NOC	[143]	inheritance	5
OCMIC	[27]	coupling	5

6.3.5.2 Result for maintainability models

Table 23 summaries these models with the design metrics used, modeling technique, model validation method and accuracy value.

Table 23: Multivariate models for maintainability prediction

Studies	Best metric suite	Modeling technique	Model validation	Accuracy
ZX08	OSAVG, SNOC, CSO, CSA	Logistics	LOO cross validation	Adjusted R square: 0.486
Ari06	OMAEC, CS	Linear	Local dataset	R square: more than 50 percent of the variation in change effort is explained by the model.
BMP05	No of methods, No of association	Linear	Local dataset	R square: explain around 28% of the variation in maintenance effort
Mar247	Size1, size2, DIT, NOC, MPC, RFC, LCOM, DAT, WMC, NOM	Linear	Local dataset	R square: explain for 90% variation in the sample in UIMS and 87% in QUES
Mar037	WMC, CBO, LCOM, BDM	Linear	Local dataset	R square: 64 percent of the variance of the data set
EE09	WMC, RFC, LCOM, MPC, DAC, NOM, SIZE1, SIZE2	TreeNet	LOO cross validation	MMRE
Mis05	AMLOC, N, CDENS, MHF, DIT	Non linear	Cross program validation	Not discussed

6.3.5.3 Discussion

In general the validation with the same local dataset gives the most accuracy performance result, since the model parameter is constructed to fit to the local dataset. The version-across validation give the least predictive result among 3 validation methods due to the appearance of heterogeneity between software versions. However, a prediction model should be evaluated on unseen data to obtain more sensible measures of the predictive power of a particular model. There are still a large number of models without validation or local validation (37% of total models). K fold cross validation should be used when only single dataset is available. Version or software across validation should be used more frequent to find a generic prediction model.

It is shown that some design metrics that are found significantly in correlation analysis and univariate regression model are actually helpful in multivariate model. It is the cases of CBO, WMC and RFC which are frequently included in multivariate models. Interestingly, DIT and NOC, which are found not significantly in Spearman correlation and univariate regression analysis, but both, appear frequently in multivariate models. It indicates that DIT and NOC is not good as single indicator of fault proneness by it can complement other metrics to achieve better predictive performance. The evidences for that are interaction between DIT and CBO in SK03 [97], high correlation with RFC in SKM09 [103], XHC08 [83], ZL06 [98] and GFS05 [104]. Besides, there is no clear explanation for including NOC in multivariate models. OCMIC is accepted for null hypothesis with 50% portion of success. But OCMIC is popular among multivariate models. Therefore, OCMIC could be considered as one alternative for coupling metric (rather than C&K metrics).

6.3.6 SQ6: Is there an overall influence of these metrics on external quality attributes? What are the impacts of those metrics on those attributes?

In this systematic review, a main step is quantitative aggregation of data from primary studies. The overall influence of design metrics on fault proneness or maintainability can be quantified by effect sizes of the metrics on these external qualities. A meta analysis is conducted to find the global effect sizes. For the meta analysis procedure to be applicable and reliable, it requires the effect sizes from the same statistical methods and the consistency in the use of measure of independent variable and dependent variable. In maintainability studies, the variety in experiment variables and measures make it hard to aggregate. Hence, it is only feasible to find global effect size in fault proneness studies.

Weighted estimator method is applied for Spearman correlation coefficient and estimated odds ratio (detail given in Section 5.3.3.3) by fixed effect and random effect models. The results of these two analyses are compared to find the inference.

6.3.6.1 Result for Global Spearman correlation coefficient

Table 24 shows the list of effect size for design metrics over different data sets. For each dataset the following information are extracted:

- N: size of data set, measured by the number of class investigated. The size of dataset is used to weight for the effect size. Therefore the effect size from the larger dataset plays the more importance roles in the result of aggregation.
- Data type: project type
- Language: programming language used in the project
- Domain: business domain of the project
- Phase: the development life cycle phase when the defect is collected
- Empirical type: studies design type.

The detail description is given in Section 5.3.2. We collect the Spearman correlation coefficient for 6 design metrics: NOC, DIT, CBO, LCOM, WMC, RFC and code size metric: LOC (line of code) for purpose of comparison. Because not every study investigates the same metric set, the number of effect sizes for each metric is different. Particularly, there are 18 reported effect sizes for NOC, 22 reported effect sizes for DIT, 24 ones for CBO, 18 ones for LCOM, 33 ones for WMC, 17 ones for RFC and 18 ones for LOC.

Among studies, there is one that does not report information about programming language and defect collection phase (TLG05 [109]). Because of the business domain is telecommunication, the large size of the product, we assume the programming language used is C++.

Among studies, there are two dataset that are investigated in 2 studies (NASA KC1 by XHC08 and PD07, and Eclipse version 3.0 by SZP+06 and ZXL10). In NASA KC1, the correlation results are slightly different between two studies, which could come from the error in measurement and analysis. We decided to include both of these studies for analysis to test the sub group analysis in the later phase. Since the two effect sizes come from the same dataset, they should be categorized in the same subgroup. In Eclipse version 3.0, the variation between two reports (SZP+06 and ZXL10) is significant, especially in case of post release defects. The difference could come from the error in measurement procedure (though these two studies use the same theoretical metric but using different metric collectors), data collection procedure (i.e. outlier analysis, defect selection, etc). Because it is hard to say which of these two studies is the correct one, we decide to include both of these cases.

Table 24: Spearman coefficients with moderator variables in fault proneness studies

Original	N	Data type	Language	Domain	Phase	Empirical type	NOC	DIT	CBO	LCOM	WMC	RFC	LOC
XHC08	145	Public	C++	Communication	Pre	Historical	-0.1743	0.0411	0.5472	-0.0178	0.3383	0.2475	0.5763
PD07	145	Public	C++	Communication	Pre	Historical	-0.156	0.036	0.52	-0.006	0.352	0.245	0.56
OEM+08	95	Public	Java	Internet	Pre + post	Historical					0.419		0.504
OEM+08	126	Public	Java	Internet	Pre + post	Historical					0.395		0.386
OEM+08	179	Public	Java	Internet	Pre + post	Historical					0.297		0.31
OEM+08, OEG+07	178	Public	Java	Internet	Pre + post	Historical	0.203	0.122	0.216	0.331	0.381	0.373	0.407
OEM+08, OEG+07	198	Public	Java	Internet	Pre + post	Historical	0.346	0.384	0.47	0.503	0.529	0.551	0.519
OEM+08, OEG+07	201	Public	Java	Internet	Pre + post	Historical	0.198	-0.037	0.178	0.218	0.236	0.25	0.322
ZXL10	5219	Public	Java	Soft. development	Pre	Historical					0.321		
ZXL10	5219	Public	Java	Soft. development	Post	Historical					0.208		
ZXL10	6201	Public	Java	Soft. development	Pre	Historical					0.296		
ZXL10	6201	Public	Java	Soft. development	Post	Historical					0.26		
SZP+06	6614	Public	Java	Soft. development	Pre	Historical	0.04	0.11	0.4	0.23	0.31	0.38	
SZP+06	6614	Public	Java	Soft. development	Post	Historical	0.02	0.01	0.12	0.07	0.11	0.11	
ZXL10	8392	Public	Java	Soft. development	Pre	Historical					0.343		
ZXL10	8392	Public	Java	Soft. development	Post	Historical					0.284		
JSP+06	93	Private industry	C++	Communication	Post	Historical	-0.2	-0.07	0	0.11	0.14	0.11	0.08
JSP+06	119	Private industry	C++	Communication	Post	Historical	-0.13	0.59	0.36	0.24	0.22	0.31	0.24
JSP+06	101	Private industry	C++	Communication	Post	Historical	0.03	0.24	0.43	0.46	0.41	0.43	0.46
JSP+06	44	Private industry	C++	Communication	Post	Historical	-0.17	0.05	0.19	0.16	0.18	0.25	0.24
JSP+06	38	Private industry	C++	Communication	Post	Historical	-0.21	0.06	0.45	0.32	0.31	0.45	0.52
SPS+03	150	Private industry	C++	Communication	Post	Historical	-0.12	0.12	0.18	0.16	0.14	0.17	0.12
SPS+03	144	Private industry	C++	Communication	Post	Historical	-0.03	-0.05	0.2	0.22	0.35	0.34	0.37
HCN98	12	Private academic	C++	Controlled system	Pre	Historical			-0.18	0.53	0.46		
TLG05	600	Private industry	C++	Telecommunication	NP	Historical	0.02	-0.01	0.43	0.15	0.47	0.41	0.52
WKK08	249	Public	C++	Document editor	Pre + post	Historical	-0.025	-0.259	0.252	0.303	0.223	0.174	0.234
WKK08	538	Public	Java	Document editor	Pre + post	Historical	-0.011	0.037	0.268	0.21	0.12	0.107	0.157
BS98	113	Private academic	C++	MIS	Pre	Historical	0.01				0.414	0.417	
Mis002	264	Public	Java	MIS	Pre + post	Historical	-0.07	-0.03					

The Table 25 summaries the result of a meta analysis with two models: fixed effect and dynamic effect model. The "Point est." shows the global effect size of each metrics that are summarized by either fixed or random model. 95% CI shows the confidence interval 95% of the estimation. P is the value of significance tests; Q is the value of heterogeneity test. I square is the percentage presentation of heterogeneity (detail given in Section 5.3.4). It is noted that the Q and I square test is only available for Fixed effect model.

Table 25: Meta analysis with fixed and random effect model

	CBO	DIT	NOC	LCOM2	RFC1	WMC
Fixed model						
Point est.	0.28	0.06	0.03	0.17	0.26	0.28
95% CI	[0.27;0.30]	[0.04;0.07]	[0.01;0.04]	[0.15;0.18]	[0.25;0.28]	[0.27;0.29]
P	0.000	0.000	0.001	0.000	0.000	0.000
Q	420.548	98.430	68.335	50.003	54.343	431.602
I square	**94.531**	**82.729**	**75.122**	**74.001**	**77.918**	**92.586**
Random model						
Point est.	0.31	0.13	0.005	0.23	0.31	0.31
95% CI	[0.22;0.39]	[0.06;0.20]	[-0.04;0.05]	[0.15;0.29]	[0.20;0.39]	[0.27;0.35]
P	0.000	0.004	0.845	0.000	0.000	0.000

The result of Q tests shows high level heterogeneity for all design metrics (from 74% to 95%), especially for WMC (92.586%) and CBO (94.531%) and all calculated effect sizes are significant at 0.001. The high heterogeneity indicates that effect sizes come from a random population. Therefore, fixed model is not appropriate in this case. We use the random model to evaluate the global effect size.

The random models show that within 95% confidence interval, CBO, RFC, WMC has similar impact on fault proneness. The impact of LCOM2 on fault proneness is slightly smaller than these 3 metrics. Also, the global effect size shows the slightly stronger impact of LOC on fault proneness than these 3 metrics. Besides, CBO, RFC, WMC and LCOM2 have clearly larger impacts on fault proneness than DIT and NOC do. The impact of NOC on fault proneness is inconsistent due to its value fluctuate between 2 sides of zero.

Figure 18 to Figure 24 show forest plots of Spearman correlation coefficient for design metrics. The effect column shows point estimator calculated by the random effect model. The weight column shows the weight according to the random effect model. The square indicates the effect size estimate for each study. The size of each square is proportional to the relative weight of the study according to the fixed model. The relative weights of a random effects model are

generally more uniform than those of fixed-effect models, due to the incorporation of between study variance into all the studies weight. The horizontal lines indicate the 95% confidence interval for each study's effect size estimate according to the random effect model. The smaller the study is weighted, the larger confidence interval it obtains.

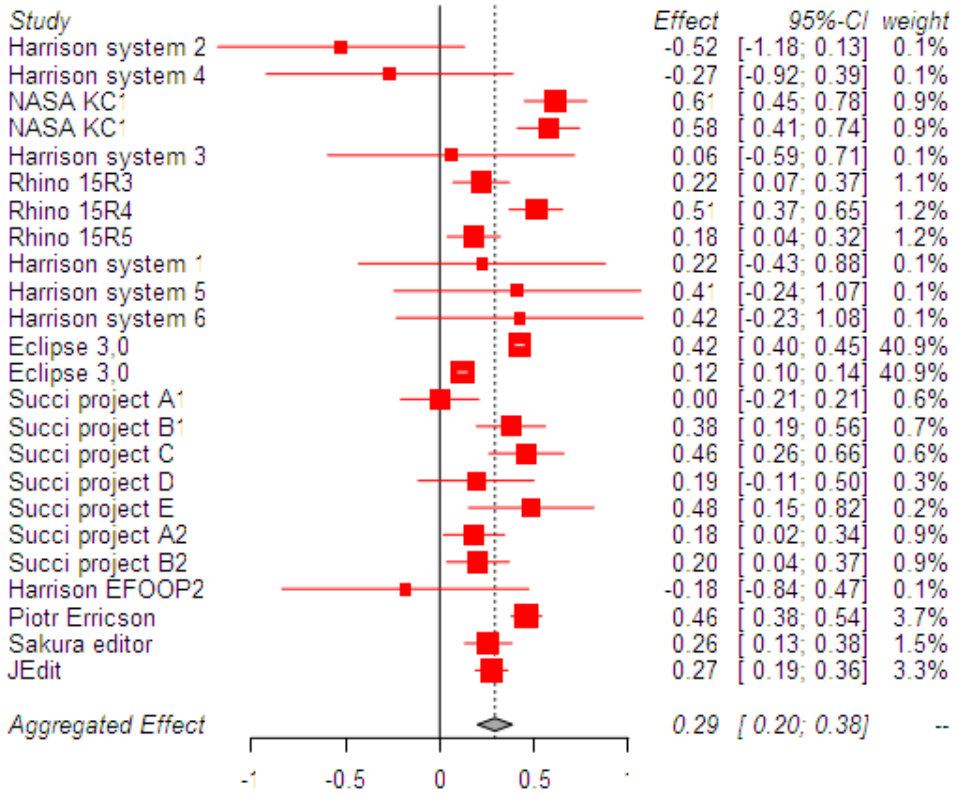

Figure 18: Forest plot for meta analysis of CBO

Figure 18 shows that expect for effect sizes from 3 Harrion systems (HCN98), all reported Spearman coefficient of CBO are larger than zero.

In Figure 19, 6 reported effect sizes of DIT are negative and the global Spearman coefficient is very small (0.06). Similar to DIT, global Spearman coefficient of NOC is trivial (0.03) and large portion of reported effect size are lower than zero (11 out of 18 effect sizes).

Figure 21 shows a forest plot of Spearman coefficient for LCOM. In this diagram, only effect sizes from NASA KC1 have negative value. In Figure 22, all reported effect sizes for RFC are clearly larger than zero. Figure 23 shows the fairly large Spearman coefficient of WMC. Interestingly, only one reported effect size from Harrison data set (HCN98) has a negative value.

Figure 24 shows a forest plot of Spearman coefficient for LOC. None of the reported effect sizes are negative and the mean global effect size is largest (0.39) among investigated metrics.

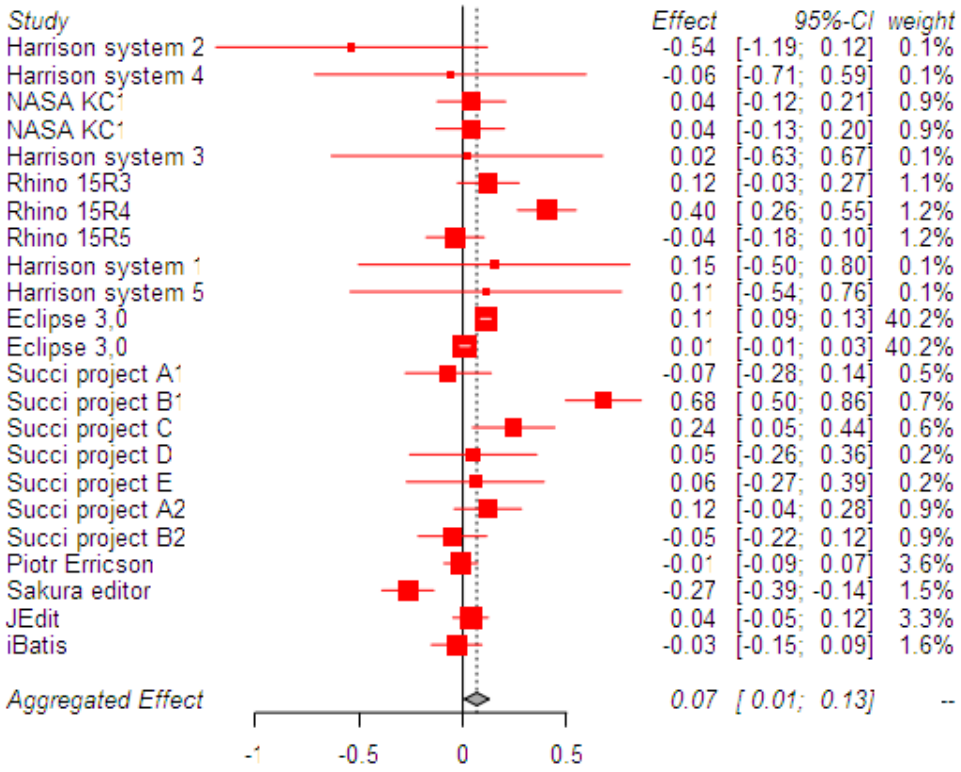

Figure 19: Forest plot for meta analysis of DIT

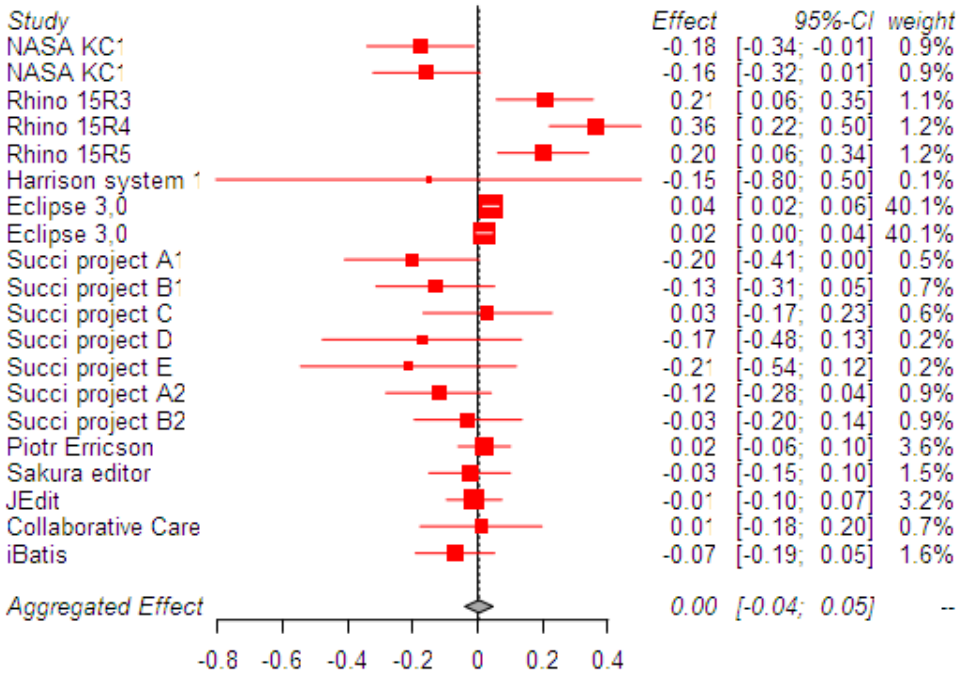

Figure 20: Forest plot for meta analysis of NOC

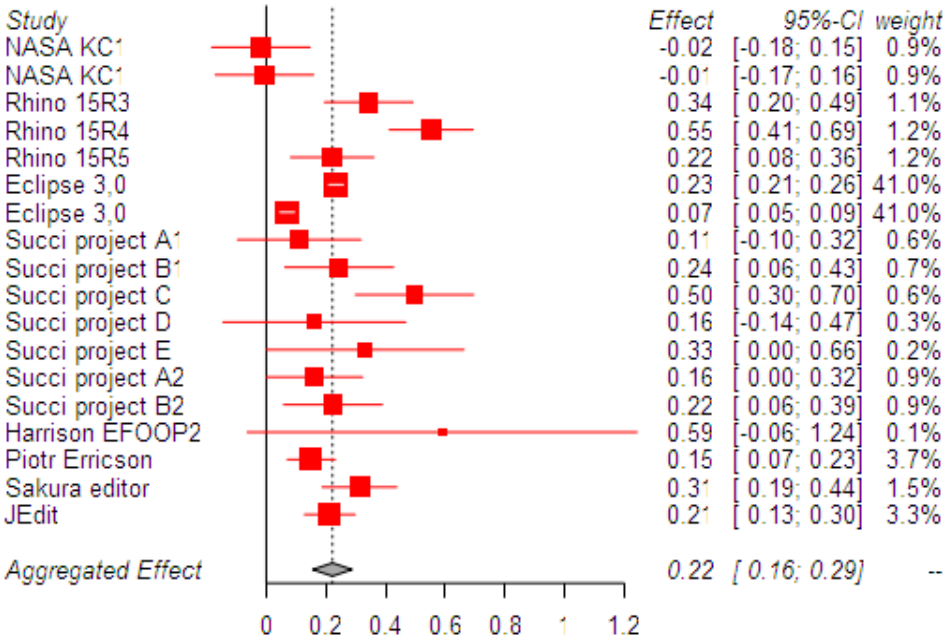

Figure 21: Forest plot for meta analysis of LCOM2

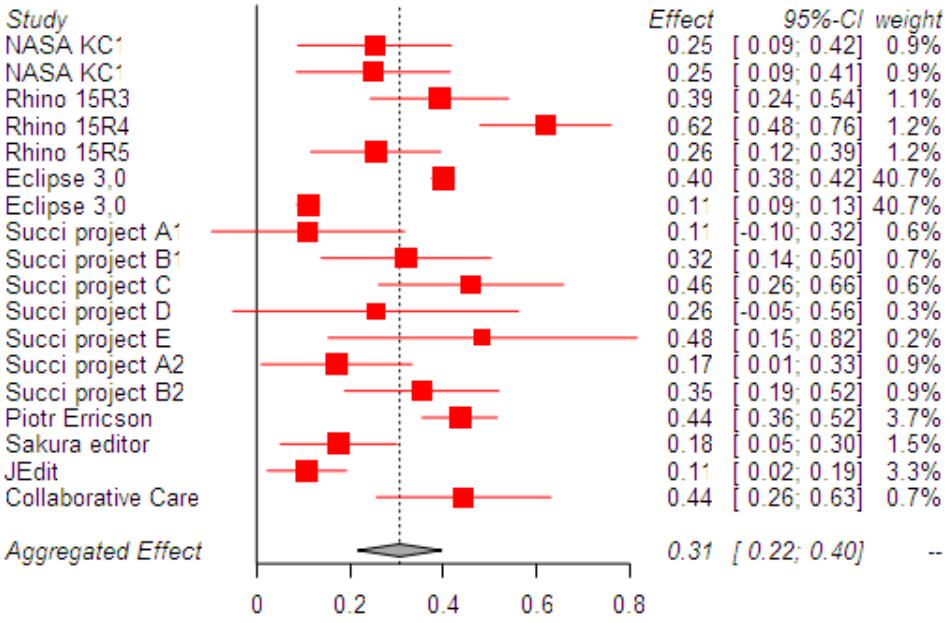

Figure 22: Forest plot for meta analysis of RFC1

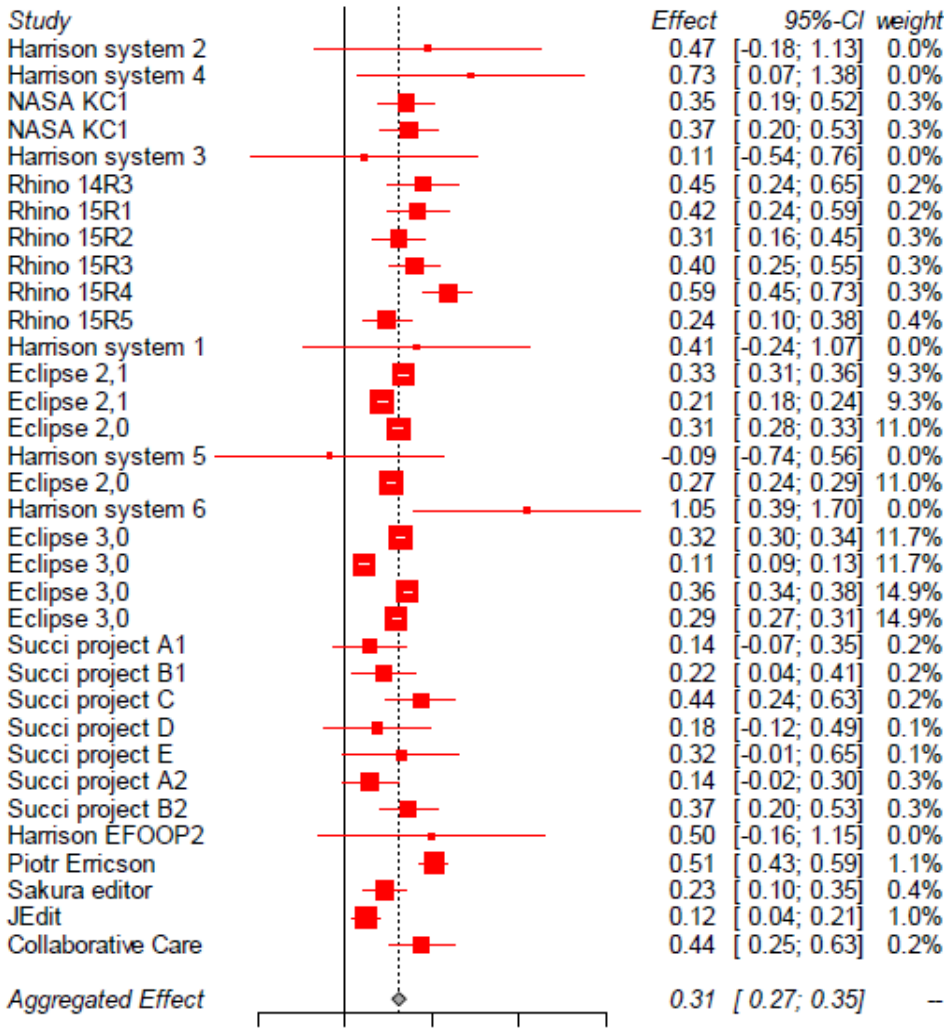

Figure 23: Forest plot for meta analysis of WMC

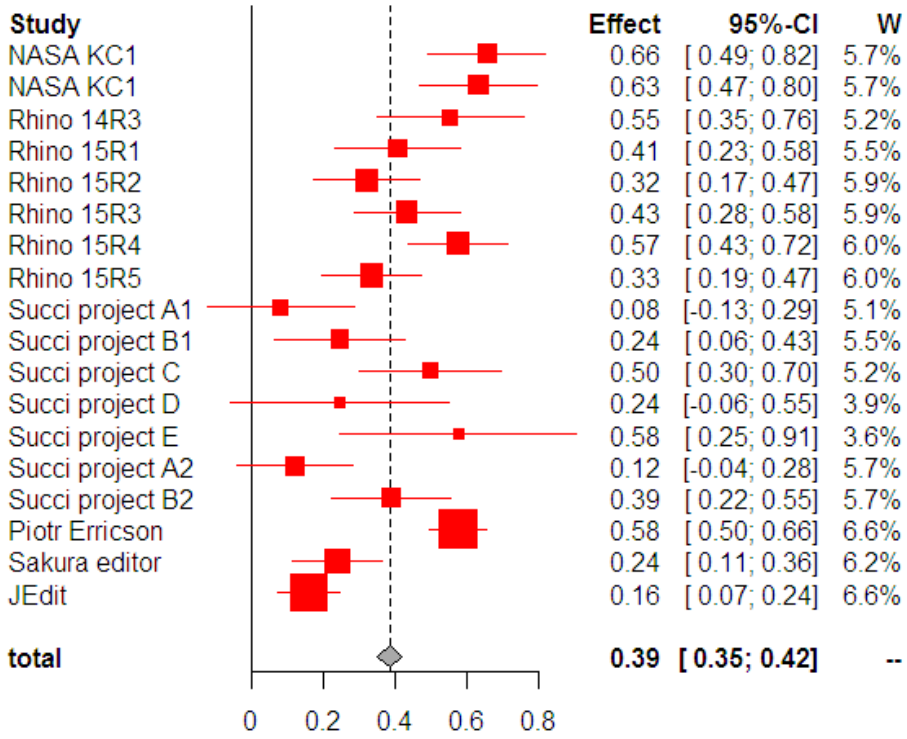

Figure 24: Forest plot for Spearman of LOC

6.3.6.2 Publication bias

Publication bias captures the idea that studies that report relatively large treatment effects are the ones that were most likely to be accepted for publications. The effect size estimated from a biased collection of studies will tend to overestimate the true effect. We use the funnel plots to assess the likely extent of publication bias, and its potential impact on the conclusions. Without the publication bias, studies should be distributed symmetrically on either side of the estimated effect size. In case of bias, the bottom part of the plot should show a higher concentration of studies on one side of the estimated effect size than the other. The evidence of publication bias can be shown by funnel plot and the adjustment of global effect size calculation can be done by trim and fill method (detail given in Section 5.3.7).

Figure 25 to Figure 29 show the funnel plot for five design metrics, namely CBO, NOC, LCOM2, RFC1 and WMC. Each white point represents for an effect sizes from primary study. We observe whether these white points distributed equally in two sides of the global effect size (which is represented by a vertical line). The black point represents for the filled effect sizes to achieve the symmetry of the plot. These funnel plots reveal the appearance of publication bias for all of these metrics.

Table 26 shows the result of "trill and fill" estimation. The column "Trimmed studies" shows the number of inserted points to achieve symmetric funnel plot for each metric. The column "Global effect size" shows the old estimated mean effect size by fixed effect model. The column "Adjusted effect size" shows the new estimated mean effect size after the insertion. The column "95% CI" gives the confidence interval of new calculated mean effect size.

It is shown that only mean effect size of LCOM2 is changed after the recalculation. It indicates that publication bias does not impact on our calculation significantly.

Table 26: Trim and fill result

Metric	Studies trimmed	Global effect size	Adjusted effect size	95% CI
CBO	1	0.28	0.28	[0.27;0.30]
DIT	0	0.06	X	X
NOC	5	0.03	0.03	[0.02;0.05]
LCOM2	5	0.17	0.15	[0.13;0.17]
RFC1	2	0.26	0.26	[0.24;0.28]
WMC	6	0.28	0.28	[0.27;0.29]

6.3.6.3 Result for estimated odds ratio

Table 27 shows the list of odds ratio as effect size for design metrics over different data sets. We collect the univariate logistic correlation coefficient for 6 design metrics: NOC, DIT, CBO, LCOM, WMC, and RFC. The odds ratio is estimated by exponential of these coefficients. Because not every study investigates the same metric set, the number of reported effect size for each metric is different. Particularly, there are 17 reported effect sizes for NOC, 19 reported effect sizes for DIT, 18 ones for CBO, 14 ones for LCOM, 28 ones for WMC and 21 ones for RFC.

Among studies, there are four dataset that are investigated in multiple studies (NASA KC1 by SKM09, GS08 and ZL06, and 3 versions of Eclipse SL08 and ZXL10). The reported result is significantly different among studies, which could come from the variation in measurement procedure (though these two studies use the same theoretical metric), data collection procedure (i.e. outlier analysis, defect selection, etc). Because it is hard to say which of these two studies is the correct one, we decided to include both of these cases.

The Table 29 summaries the result of a meta analysis with fixed effect model. The row "Point est." shows the global effect size of each metrics. The row 95% CI shows the confidence interval 95% of the estimation. N is number of dataset for each design metric.

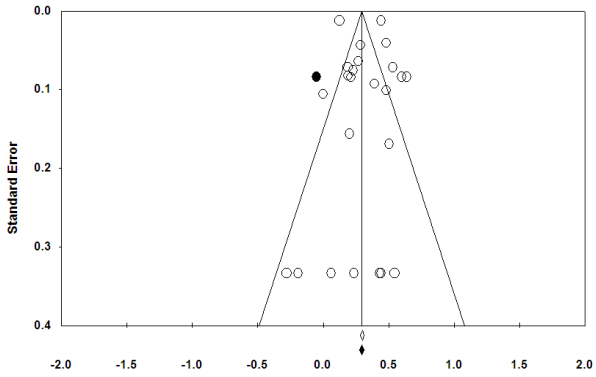

Figure 25: Funnel plot for CBO

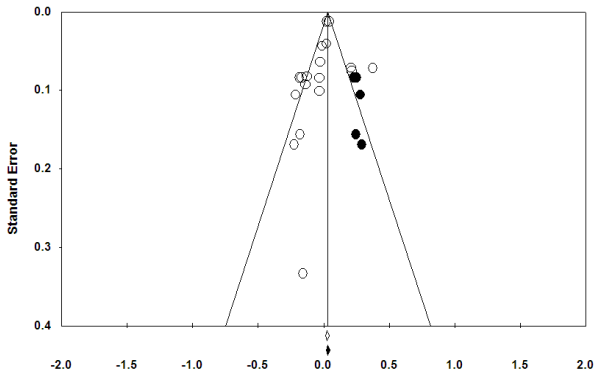

Figure 26: Funnel plot for NOC

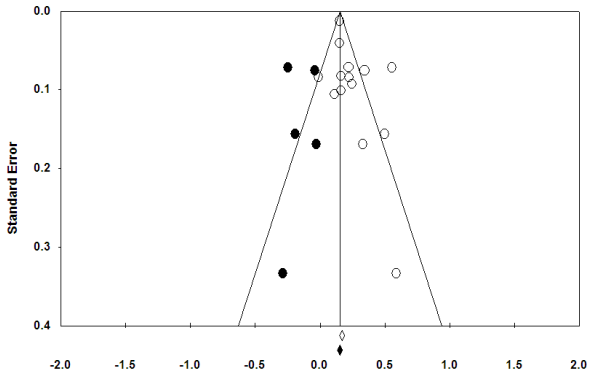

Figure 27: Funnel plot for LCOM

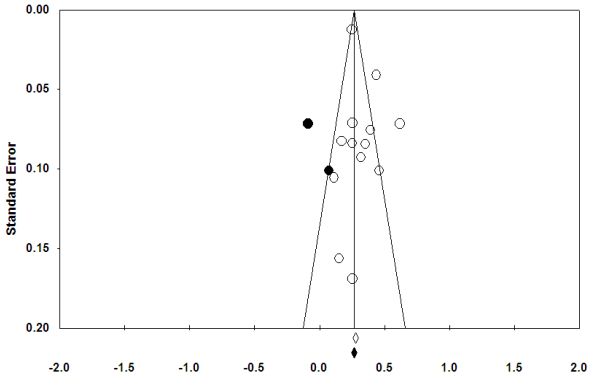

Figure 28: Funnel plot for RFC

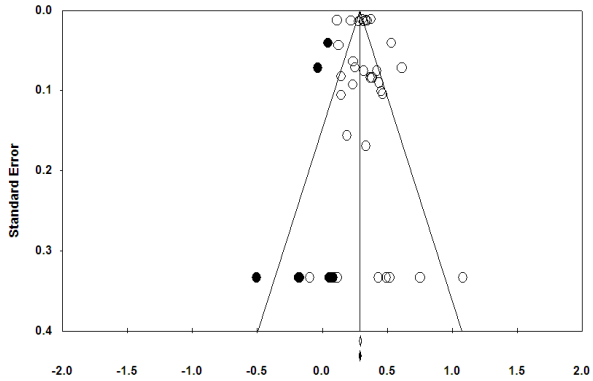

Figure 29: Funnel plot for WMC

Table 27: Odds ratios with moderator variables in fault proneness studies

Dataset Name	Original	N	D. type	Lang.	Domain	Phase	Empr. type	NOC	DIT	CBO	LCOM2	WMC	RFC1
Maryland experiment 2	BWD+00	113	Private	C++	MIS	Pre	Observational	0.230	2.013	1.383	1.000	1.079	1.089
Indian University	ASK+09,ASK+07	85	Private	C++	MIS	Pre	Observational	1.370	0.472	2.325	1.083	1.307	1.199
NASA KC1	SKM09	145	Private	C++	satellite	Pre	Historical	0.510	1.089	1.315	1.008	1.065	1.024
NASA KC1	ZL06	145	Private	C++	satellite	Pre	Historical	0.112	1.099	1.328	1.010	1.071	1.026
NASA KC1	GS08	145	Private	C++	satellite	Pre	Historical	0.649	1.055	1.230	1.000	1.025	1.010
Mozilla 1.6	GFS05	3192	Public	C++	web browser	Pre + post	Historical	0.960	1.818	2.951	8.331	2.912	2.382
Briand Xpose	BMW02	144	Private	Java	Document editor	Post	Historical	2.096	1.200			1.091	
Emam framework	EBG+01	174	Private	C++	Telecommunication	Post	Historical		0.885	1.324	1.007	1.028	1.042
Maryland experiment 1	BBM96	180	Private	C++	MIS	Pre	Observational	0.034	1.624	1.153		1.022	1.089
Briand LALO	BWL01	90	Private	C++	Program editor	Pre	Historical		0.519	1.292	1.001	1.084	1.014
Tang system 1	TKC09	20	Private	C++	Document editor	Pre + post	Historical					1.258	1.056
Tang system 2	TKC09	45	Private	C++	Document editor	Pre + post	Historical					1.452	1.065
Tang system 3	TKC09	27	Private	C++	Document editor	Pre + post	Historical					1.259	1.060
Rhino 14R3	OEM+08, OEG+07	95	Public	Java	web browser	Pre + post	Historical	1.720	1.710	1.190	1.200	1.050	1.060
Rhino 15R1	OEM+08, OEG+07	126	Public	Java	web browser	Pre + post	Historical	1.220	2.530	1.160	1.170	1.060	1.050
Rhino 15R2	OEM+08, OEG+07	179	Public	Java	web browser	Pre + post	Historical	1.030	1.290	1.080	1.260	1.040	1.040
Rhino 15R3	OEM+08, OEG+07	178	Public	Java	web browser	Pre + post	Historical	1.360	1.210	1.210	1.440	1.080	1.080
Rhino 15R4	OEM+08, OEG+07	198	Public	Java	web browser	Pre + post	Historical	1.370	2.020	1.260	1.770	1.090	1.080
Rhino 15R5	OEM+08, OEG+07	201	Public	Java	web browser	Pre + post	Historical	1.560	0.850	1.180	1.310	1.060	1.050
Eclipse 2.0	SL08	4454	Public	Java	Program editor	Pre	Historical	0.974	1.052	1.024		1.006	1.004
Eclipse 2.0	ZXL10	5219	Public	Java	Program editor	Pre	Historical					1.041	
Eclipse 2.0	ZXL10	5219	Public	Java	Program editor	Post	Historical					1.043	
Eclipse 2.1	SL08	5259	Public	Java	Program editor	Pre	Historical	0.990	1.051	1.026		1.007	1.004
Eclipse 2.1	ZXL10	6201	Public	Java	Program editor	Pre	Historical					1.049	
Eclipse 2.1	ZXL10	6201	Public	Java	Program editor	Post	Historical					1.034	
Eclipse 3.0	SL08	7066	Public	Java	Program editor	Pre	Historical	0.958	1.112	1.023		1.005	1.003
Eclipse 3.0	ZXL10	8392	Public	Java	Program editor	Pre	Historical					1.053	
Eclipse 3.0	ZXL10	8392	Public	Java	Program editor	Post	Historical					1.047	

Table 28 shows a mixed picture. Heterogeneity is absent for LCOM, RFC, WMC and at low level (13.133%) for CBO. However heterogeneity level is medium for DIT (72.465%) and high for NOC (97.115%). This observation indicates that fixed effect model is suitable to aggregate effect sizes for CBO, LCOM, RFC and WMC but not DIT and NOC.

Table 28: Meta analysis with fixed model

	CBO	DIT	NOC	LCOM2	RFC1	WMC
Point est.	1.047	1.093	0.931	1.189	1.010	1.036
95% CI	[1.00;1.08]	[1.05;1.13]	[0.90;0.97]	[1.05;1.32]	[0.97;1.05]	[1.02;1.05]
P	0.013	<0.001	<0.001	0.004	0.612	0.001
Q	18.419	61.741	519.859	9.739	1.183	4.916
I square	13.133	72.465	97.115	0	0	0
N	17	18	16	13	20	27

6.3.6.4 Discussion

The high heterogeneity between studies indicates the use of random effect model or unconditional inference model [53]. The model shows that effect size of DIT and NOC is trivial and fluctuate around zero. Effect size of LCOM is small. Effect size of CBO, RFC and WMC and LOC is at medium level. Among 7 metrics, LOC has the largest Spearman correlation coefficient with fault proneness. The results indicates that LOC should be a better than design metrics as a single indicator of fault proneness.

Overall, there is an agreement between the aggregated results from meta analysis using Spearman coefficient and meta analysis using Odd ratios. It is noticed that the estimated odds ratio is not comparable between design metrics due to the difference in measure scale of each metrics [113]. The only possible way to compare them is to convert all metrics to standard measurement scale. However the approach is not feasible due to the lack report of standard error.

6.3.7 SQ7: Do studies agree on these influences? If no, what explains this inconsistency? Is this explanation inconsistency across different metrics?

To identify the source of heterogeneity, subgroup analysis and cluster analysis are performed (detail in Section 5.3.5). This section presents the result of these analyses.

6.3.7.1 Result of subgroup analysis

The previous section shows an appearance of high heterogeneity for all design metric. In order to find the explanation of variation, we conduct the sub group analysis for 5 moderator

variable: language, project type, dataset type, defect collection phase and business domain. The results are given in Table 29 for Spearman coefficient and in Table 28 for odds ratio.

Table 29: Subgroup analysis for Spearman coefficients

Metric	Language	Project type	Dataset type	Phase	Domain
CBO	5.522	1.116	4.292	**83.614**	10.164
DIT	2.393	0.47	0.173	19.985	5.704
NOC	34.362	0.457	24.033	15.435	**76.333**
LCOM	0.956	1.033	0.011	**60.011**	23.738
RFC	5.287	0	2.882	**77.733**	9.346
WMC	31.733	0.48	3.925	**60.457**	7.479
LOC	7.236	0	2.176	**51.173**	49.563

In table 30, the reported number in table cell is variation explanation (ve) (detail in section 5.3.5.1), i.e. the value of first cell 5.522 means the moderator language explains for 5.522% of observed between group variation. The ve values that are greater than 50% are highlighted. It is shown that Phase (or Defect collection phase) can explain for more than 50% observed variation between group in CBO, LCOM, RFC, WMC and LOC. Domain can explain for 76.333% of observed variation between groups in NOC. There are no moderator can explain large number of variation in DIT.

The subgroup analysis shows that phase and domain reduce the heterogeneity between groups. The result indicates that impact of CBO, LCOM, RFC, WMC and LOC on pre-release fault proneness could be different from those on post release fault proneness. Besides, relation between LCOM and defects could be different in different business domains. However, the with-in study heterogeneity is reduced but not satisfactorily (most of groups are still at medium or high heterogeneity level). This fact infers the inefficiency in variation explanation of single variable moderators.

Table 30: 95% confidence interval of Spearman coefficients by moderator variables

	Language		Phase	
Hypothesis	C++	Java	Pre release	Post release
CBO	[0.20;0.41]	[0.13;0.45]	[0,40; 0,45]	[0,11; 0,16]
RFC	[0,20; 0,31]	[0,15;0,47]	[0,37; 0,42]	[0,15;0,38]
DIT	[-0,07; 0,18]	[0,01;0,15]	[0,08;0,13]	[-0,03;0,29]
NOC	[-0,11; -0,01]	[0,02;0,13]	[-0,18;0,06]	[-0,14;0,01]
LCOM	[0,11; 0,28]	[0,15;0,37]	[-0,07;0,31]	[0,10;0,32]
WMC	[0,23; 0,34]	[0,25;0,34]	[0,32; 0,34]	[0,17;0,30]
LOC	[0,26; 0,52]	[0,26; 0,52]	[0,53;0,76]	[0,20;0.36]

Table 31 shows that domain and language can explain for more than 50% of observed variation in NOC metrics. Again, there are no moderator can explain large number of variation in DIT

Table 31: Subgroup analysis for odds ratios

Metric	Language	Project type	Data set type	Phase	Domain
DIT	0.332	3.271	0.119	19.251	24.819
NOC	**54.504**	44.543	37.981	6.941	**64.907**

Table 32: Result of cluster analysis

Dataset Name	Original	NOC	DIT	CBO	LCOM2	WMC	RFC1	LOC
NASA KC1	XHC08	B	G	P	B	G	G	P
NASA KC1	PD07	B	G	P	B	G	G	P
Rhino 14R3	OEM+08					G		P
Rhino 15R1	OEM+08					G		G
Rhino 15R2	OEM+08					G		G
Rhino 15R3	OEM+08, OEG+07	P	G	G	G	G	P	G
Rhino 15R4	OEM+08, OEG+07	P	P	P	P	G	P	P
Rhino 15R5	OEM+08, OEG+07	P	G	G	G	B	G	G
Eclipse 2.1	ZXL10					G		
Eclipse 2.1	ZXL10					B		
Eclipse 2.0	ZXL10					G		
Eclipse 2.0	ZXL10					B		
Eclipse 3.0	SZP+06	G	G	P	G	G	P	
Eclipse 3.0	SZP+06	G	G	G	B	B	B	
Eclipse 3.0	ZXL10					G		
Eclipse 3.0	ZXL10					G		
Succi project A1	JSP+06	B	G	G	B	B	B	B
Succi project B1	JSP+06	B	P	P	G	B	G	B
Succi project C	JSP+06	G	G	P	P	G	P	P
Succi project D	JSP+06	B	G	G	G	B	G	B
Succi project E	JSP+06	B	G	P	G	G	P	P
Succi project A2	SPS+03	B	G	G	G	B	B	B
Succi project B2	SPS+03	G	G	G	G	G	G	G
Harrison EFOOP2	HCN98			B	P	G		
Piotr Erricson	TLG05	G	G	P	G	G	P	P
Sakura editor	WKK08	G	B	G	G	B	B	B
JEdit	WKK08	G	G	G	G	B	B	B
Collaborative Care	BS98	G				G	P	
iBatis	Mis002	G	G					

The observation from subgroup analysis motivates an execution of cluster analysis to identify unknown moderator variables that could explain heterogeneity satisfactorily.

6.3.7.2 Result of cluster analysis

Although the predefined moderator variables can help to explain the between group variation and therefore reduce the within group variation, this value is still at medium or high level, which indicate high heterogeneity. We aims at finding any unknown moderator variables that can reduce the heterogeneity within a group to zero or small level. The cluster analysis procedure helps to achieve this objective.

Table 33: Description of clusters

Metric	Cluster	TE	95% CI	p	I2	N	ve
NOC	Cluster 1	0.027	[0,01; 0,04]	0,001	0	7	88.419
NOC	Cluster 2	0.257	[0,18; 0,34]	< 0,001	37.326	3	NA
NOC	Cluster 3	-0.158	[-0,23; -0,09]	< 0,001	0	8	NA
DIT	Cluster 1	-0.275	[-0,40; -0,15]	< 0,001	0	2	62.641
DIT	Cluster 2	0.056	[0,04; 0,07]	< 0,001	62.664	18	NA
DIT	Cluster 3	0.507	[0,40; 0,62]	< 0,001	81.542	2	NA
CBO	Cluster 1	0.435	[0,41; 0,46]	< 0,001	12.369	10	92.166
CBO	Cluster 2	0.14	[0,12; 0,16]	< 0,001	49.574	11	NA
CBO	Cluster 3	-0.324	[-0,70; 0,05]	0,093	0	3	NA
LCOM	Cluster 1	0.536	[0,42; 0,65]	< 0,001	17.598	3	92.708
LCOM	Cluster 2	0.23	[0,21; 0,25]	< 0,001	0	11	NA
LCOM	Cluster 3	0.067	[0,04; 0,09]	< 0,001	0	4	NA
RFC	Cluster 1	0.114	[0,09; 0,14]	< 0,001	0	5	96.275
RFC	Cluster 2	0.281	[0,21; 0,35]	< 0,001	0	6	NA
RFC	Cluster 3	0.409	[0,39; 0,43]	< 0,001	41.482	6	NA
WMC	Cluster 1	0.885	[0,42; 1,35]	< 0,001	0	2	62.477
WMC	Cluster 2	0.328	[0,32; 0,34]	< 0,001	69.239	19	NA
WMC	Cluster 3	0.191	[0,18; 0,21]	< 0,001	87.045	12	NA
LOC	Cluster 1	0.373	[0,30; 0,44]	< 0,001	0	5	93.897
LOC	Cluster 2	0.176	[0,12; 0,23]	< 0,001	0	6	NA
LOC	Cluster 3	0.584	[0,53; 0,64]	< 0,001	0	7	NA

Table 32 shows the result of cluster analysis. We classified dataset in three clusters in order to minimize the within-cluster variation and maximize the between-cluster variation. The clusters are marked with color blue (B), green (G) and pink (P) according to the order of cluster mean

effect size. Table 33 describes the detail information of each cluster. In the table "TE" is the point estimator of effect size for each cluster.

The result of cluster analysis shows that heterogeneity of Spearman coefficient of DIT and WMC is unexplainable high. The subgroup and cluster analysis shows that effect size of DIT and WMC on fault proneness comes from random population.

In case of NOC and LCOM, the clusters have heterogeneity at zero or low level. Especially, effect sizes for LOC can be classified in three homogeneous clusters. In these cases we attempt to find the combination of moderator variables to account for each cluster. However with the available information is not possible to find such an explanation. Besides, the classification of studies in clusters is inconsistent across design metrics.

6.3.7.3 Discussion

Subgroup analysis

The results of subgroup analysis reveals that the impact of design complexity on fault proneness is no different between C++ and Java projects, Open source and Close source projects and among small, medium and large size data set. We also could not find the combination of these context factors that can explain for significant part of observed variation in the population. In some subgroups that the heterogeneity is absence or at low level, we can compare the impact of design metrics on fault proneness as representative of that group. The results are shown in Table 34:

Table 34: 95% confidence interval of C&K metrics within subgroup

Metric	Domain	TE	95% CI	I square
LOC	*	0.386	[0,35; 0,42]	84.47%
NOC	C++	-0.101	[-0,16; -0,04]	0%
RFC	C++	0.258	[0,20; 0,31]	26.71%
WMC	C++	0.285	[0,23; 0,34]	23%
NOC	Close source	-0.101	[-0,18; -0,03]	0%
CBO	Close source	0.233	[0,16; 0,31]	23%
LCOM	Close source	0.245	[0,17; 0,32]	33.8%
RFC	Close source	0.292	[0,22; 0,37]	39%
WMC	Close source	0.275	[0,20; 0,35]	28.7%
DIT	Observational	-0.063	[-0,36; 0,23]	0%
CBO	Observational	0.055	[-0,21; 0,32]	23.7%
DIT	Pre	0.106	[0,08; 0,13]	0%
WMC	Pre	0.332	[0,32; 0,34]	38.5%

We identified the subgroups for each metric that remains heterogeneity at low level (less than 40%). The global effect sizes within each subgroup can be calculated by fixed effect model due to low level of heterogeneity. The impact order of those metrics within some subgroups is observed as followings:

- In C++ projects, the impact order of metrics to fault proneness is: WMC > RFC >NOC. NOC has negative impact on fault proneness.
- In closed source projects, the impact order of metrics to fault proneness is: RFC > WMC > LCOM >CBO >NOC. NOC has negative impact on fault proneness.
- In observational study: CBO >DIT
- In relation with pre-release defects: WMC > DIT

The global effect size of LOC for whole population are calculated by random model effect and reported here for comparison. As shown in the Table 34, 12 out of 13 confidence interval of metrics effect sizes are bellows LOC confidence range (only with RFC, there is a small portion overlap: [0.22;0.37] and [0.35;0.42]). This observation infers that LOC is stronger correlated with fault proneness within these subgroups.

Cluster analysis

Cluster analysis classifies effect sizes of CBO, NOC, RFC, LCOM and LOC into clusters with absence of heterogeneity or with low level of heterogeneity. However we could not find a moderator variable that is common within each cluster. This is because the high relation among dataset characteristics, i.e. datasets from Succi studies will be from C++ projects, Communication domain and Historical design method. It suggests that the correlation and regression studies should be reported with more detail information about project and process factors. Cluster analysis cannot classify effect sizes of DIT and WMC into cluster with low heterogeneity level. This infers the depth of inheritance tree per class and number of method per class is very various among projects.

Chapter 7
Research Discussion

Meta analysis provides a general view about impact of design complexity on software quality. The result from a meta analysis allows us to answer frequently asked questions from literature in a larger scale of studies. The summary of results from single studies brings answers for these questions in general. Table 35 shows list of nine common hypotheses from literature that can be answered by our meta analysis.

Table 35: Hypothesis from literature

Common research questions	Investigated in
CBO has positive impact on software fault proneness	SKM09, SL08, OEG+07, ASK+07, ZL06, GFS05, SK03, BBM96, BWL01
RFC has positive impact on software fault proneness	SKM09, SL08, OEG+07, ASK+07, ZL06, GFS05, BBM96, BWL01
DIT has positive impact on software fault proneness	ASK+09, SKM09, SL08, OEG+07, ASK+07, ZL06, GFS05, SK03, BBM96, BWL01
NOC has negative impact on software fault proneness	SKM09, SL08, OEG+07, ASK+07, ZL06, GFS05, BBM96, BWL01
LCOM has positive impact on software fault proneness	ASK+09, SKM09, SL08, OEG+07, ASK+07, ZL06, GFS05, BBM96, BWL01
WMC has positive impact on software fault proneness	ASK+09, SKM09, OEM+08, SL08, OEG+07, ASK+07, ZL06, GFS05, SK03, BBM96, BWL01
CK metrics do not demonstrate as powerful impact as SLOC	XHC08, OEM+08, GS08, ZXL10
Do the effects of CK metrics differ across different programming languages?	ZL06, SK03, WKK08
Do the effects of CK metrics differ between pre release and post release defects?	SL08, SZP+06

7.1 Direction of relationship between C&K metrics and fault proneness

The conclusion about impact of C&K metrics are summarized in table 36. Our findings from results of vote counting and meta analysis is compared with the intuitive impression from literature towards these questions. The results are summarized as following:

CBO: vote counting for Spearman test and univariate logistic regression model shows that there is an evidence for significantly statistical positive relationship between CBO and fault proneness. However the portion of significantly positive value is fairly low, 10 out of 17 studies, consumes 59%. Among 24 reported effect size for CBO there is one zero value and one negative value. Meta analysis with a random effect model for Spearman coefficient shows this relationship is at moderate level: 0.31. The confidence interval is from 0.22 to 0.39. In logistic regression model, the estimated odds ratio for CBO is 1.047 with 95% confidence interval from 1.00 to 1.08. The confidence interval includes 1.00 where the relation is zero.

Table 36: Summary result of C&K metric set

Hypothesis	From literature	Our finding
CBO has positive impact on software fault proneness	Mostly yes	Yes
RFC has positive impact on software fault proneness	Mostly yes	Yes
DIT has positive impact on software fault proneness	Inconsistent	No
NOC has negative impact on software fault proneness	Inconsistent	No
LCOM has positive impact on software fault proneness	Mostly yes	No
WMC has positive impact on software fault proneness	Mostly yes	Yes
LOC has positive impact on software fault proneness	Mostly yes	Yes

RFC: vote counting for Spearman test and univariate logistic regression model shows that there is an evidence for significantly statistical positive relationship between CBO and fault proneness. The portion of significantly positive value in univariate logistic regression model is very high, fairly low, 20 out of 22 studies, consumes 91%. Meta analysis with a random effect model for Spearman shows the global effect size is at moderate level: 0.31. The confidence interval is from 0.20 to 0.39. Besides, all of the 17 reported Spearman coefficients for RFC are positive. However, the aggregated odd ratios of RFC fluctuate around neutral value since the interval confidence ranges from 0.97 to 1.05.

In one hand, the Spearman result reveals that CBO and RFC have similar correlation level with fault proneness. In another hand, the result of odd ratios shows that the possibility of observing no relation between CBO, RFC with fault proneness. Besides, the effect size interval of CBO and RFC can be generalized for the whole population due to zero value of heterogeneity.

DIT: the result of vote counting applied for both Spearman test and univariate logistic regression model shows that there is no evidence for significantly statistical relationship between DIT and fault proneness. Random effect based aggregation of Spearman coefficient show the correlation level is low: 0.13. In logistic regression model, the estimated odds ratio for CBO is 1.093 with 95% confidence interval from 1.05 to 1.13. The fairly large number of odds ratio (only smaller than ones of LCOM) shows it is useful for logistic regression. The heterogeneity of odds ratio of DIT is at medium level.

NOC: the result of vote counting applied for both Spearman test and univariate logistic regression model shows that there is no evidence for significantly statistical relationship between NOC and fault proneness. The result from meta analysis shows NOC's Spearman is trivial: 0.005 with 95% confidence interval from -0.04 to 0.05. The odds ratio confidence interval is from 0.90 to 0.97, which all is below 1. This result suggests that impact of NOC to fault proneness has a negative direction.

LCOM: the analysis results of LCOM show a mixed picture. In one hand, vote counting for Spearman test of LCOM accept the null hypothesis. Besides, the result from meta analysis shows effect size at low level: 0.23 and confidence interval is from 0.15 to 0.29. In another hand, vote counting for odd ratios estimated from a logistic regression rejects the null hypothesis. Aggregated odds ratios are 1.189, highest among 6 metrics which infers its usefulness in logistic regression model. The value is applicable for whole population due to absence of heterogeneity.

WMC: vote counting for Spearman test and univariate logistic regression model shows that there is an evidence for significantly statistical positive relationship between WMC and fault proneness. Meta analysis with random effect model for Spearman coefficient shows this relationship is at moderate level: 0.31 with confidence interval from 0.27 to 0.35. All reported Spearman effect sizes are clearly positive and all estimated odds ratios are larger than one. This result confirm about the positive impact of WMC on fault proneness.

In short, our findings statistically confirm the perception of literature about relationship between Chidamber and Kemerer metrics and fault proneness, expect for DIT and LCOM. The disagreement between studies and the trivial aggregated effect sizes for these metrics shows their ineffectiveness in indicating fault proneness.

7.2 Explanation for heterogeneity

Defect collection phase: subgroup analysis shows the evidence of difference Spearman effect size for pre-release defect group and post-release defect group. The defect collection phase moderator also helps to explain more than 50% variation of observed heterogeneity in 5 out of 7 investigated metrics. Besides, the 95% confidence interval of pre release and post release are

separated in case of CBO, RFC, WMC and LOC as shown in Table 37 (in the highlight data cells). For all of these metrics, the impacts on post release fault proneness are stronger than impacts on pre release fault proneness.

This difference can be explained by the nature of bugs [SL08]. Firstly, post-release system has been through a testing phase, in which the pre-release defects were found. In this phase, the high risk module, which is indicated by the high value of design complexity will be tested more. Post release defect are hence, either undetected in the quality assurance procedures (inspections, walkthroughs, and testing) during the development or newly introduced because of the changes made after the system was released. The design complexity is also less effective in detecting the faulty class. Therefore, the correlation of design complexity and post release defects is likely less than those of pre release defects.

Table 37: Explanation power of programming language and defect collection phase

	Language		Phase	
Hypothesis	**C++**	**Java**	**Pre release**	**Post release**
CBO	[0.20;0.41]	[0.13;0.45]	[0,40; 0,45]	[0,11; 0,16]
RFC	[0,20; 0,31]	[0,15;0,47]	[0,37; 0,42]	[0,15;0,38]
DIT	[-0,07; 0,18]	[0,01;0,15]	[0,08;0,13]	[-0,03;0,29]
NOC	[-0,11; -0,01]	[0,02;0,13]	[-0,18;0,06]	[-0,14;0,01]
LCOM	[0,11; 0,28]	[0,15;0,37]	[-0,07;0,31]	[0,10;0,32]
WMC	[0,23; 0,34]	[0,25;0,34]	[0,32; 0,34]	[0,17;0,30]
LOC	[0,26; 0,52]	[0,26; 0,52]	[0,53;0,76]	[0,20;0.36]

Programming language: sub group analysis does not support the appearance of programming language as a moderator variable since it can explain at most 30% of variations among design metrics. Also this moderator variable can only separate the confidence interval of correlation in C++ and Java in case of NOC. In the remaining six metrics, it cannot distinguish the correlation between C++ and Java projects.

In short, defect collection phase can be considered as a good moderator variable and the correlation of design complexity on pre release defects and post release defects are different. What is surprising to us is that programming language could not explain for the difference in the impact of design complexity on software fault proneness.

Chapter 8
Threats to validity

In this work, threats to validity could come from the systematic review and meta analysis procedures. We discuss possible these threats in four conventional categories: conclusion, internal, external and construct validity [37]. The threats to internal, external and construct validity are mainly from the systematic review procedure while threats to conclusion validity come from meta analysis process.

8.1 Internal validity

Internal validity refers to a casual relationship between treatment and outcome [37]. The possible internal validity in a systematic review could be in systematic review design, conduct and analysis:

- Bias in the selection of publications.
- Error in data collection.
- Quality of primary studies.
- Bias towards datasets from same study.
- Publication bias

Bias in the selection of publications can come from special interest of reviewer in any individual authors, studies or publication source. It can lead to the missing relevant papers or overestimate quality of individual studies. In this work, well-defined review protocol is an attempt to deal with this bias. Firstly, the pre selected data source and consistent usage of search term overall processes provides an objective view to all studies. Secondly, a clearly documented inclusion and exclusion criteria increases the consistency in selection of primary studies The protocol is also reviewed by supervisor who has experience in systematic review to reduce the bias of single reviewer. Last but not least, the results of test-retest (overlap portion between two pools is 87%) shows the fairly consistent in decision of review in selecting and ranking studies. Therefore, bias in selection of publications is not considered as a significant threat to our research.

Errors in data collections are from data miscalculation or mistyping. This threat could have a serious impact on the aggregated result of the meta analysis. To ensure the correctness of data collection procedure, we performed test-retest strategy for extracted data. The test is performed

by the reviewer and retest is performed by another researcher. Among 262 calculated data items, there are only one found wrong entered data items. The accuracy of data item is 99.61%, which is lower than our expectation (totally 100% correct in data entering). The test-retest result indicates that data extraction is satisfactorily accurate.

Our initially goal is to search for all reported effect size between design complexity and external quality. The drawback is inclusion of different quality studies. We try to address this problem by defining inclusion criteria, assessing quality checklist for primary studies and including only studies that met a minimum quality and rigor.

In observational study, one author can conduct an analysis in more than one datasets. The similarity of empirical design, conduct, data collection and analysis can have impact on reported effect sizes from the same author and from different authors. An example is the inclusion of 6 datasets from 2 papers of Succi et al. [103,116]. The bias of effect sizes from same author can be detected in cluster analysis since dataset original can be checked as one hidden moderator variable. The result of cluster analysis shows that even from the same author, the reported effect sizes of a metric are very different.

Publication bias refers to the general problem that positive research outcomes are more likely to be published than negative ones [9]. This threat is unavoidable since the result of a systematic review cannot be better than the primary studies. The funnel plots also detect the evidence of publication bias in most of all design metrics (Section 6.3.6.2). An attempt to reduce the publication bias is to search for studies in gray literature or unpublished sources. To the best of our knowledge, we found and selected 3 more studies from this channel (Section 5.4). Besides, trim and plot analysis shows that publication bias does not affect much on the estimation of global effect sizes.

8.2 External validity

External validity is concerned with the generalization of results outside the scope of the study [37]. External validity is closely connected with the generalizability or applicability of a study's findings. The possible external validity could be:

- The primary studies may not represent the overall objective of the review.
- The limitation of study's empirical design to observational and historical method.

Our research questions are derived from the research object by GQM approach. From the research questions, we clearly document the reason for each questions and its contribution to achieving the final research goal. The search string, inclusion/ exclusion criteria and data extraction strategy is formed to answer the research questions. This process ensures a proper focus of selected primary studies and extracted data on the research goal.

As the original purpose of meta analysis, the procedure should apply for controlled experiments with high control level of experiment variables. However due to the nature of research problem, historical and observational studies are investigated population. In historical and observation studies, it is a lack of well-controlled environment variables and warranty for randomization. Therefore, the validity of the findings in term of reliability, generalization and repeatability is less than those from controlled experiments.

8.3 Construct validity

Construct validity is concerned with the relationship between the theory and application [37]. The possible construct validity could be:

- The impact of software design complexity can be reflected by the relationship between coupling, cohesion, etc to fault proneness and maintainability.

Since design complexity is structural feature of design artifacts, the structural features of design artifacts will reflect its complexity. Prior to the systematic, a literature review reveals several aspects of design artifacts that are believed to contribute to the complexity of the design artifacts as well as final system. Besides, our systematic review also reveals the cost and external quality attributes that are investigated in design complexity studies. The selection and filter of studies ensure that the selected quality attributes are the ones reflect external software quality.

8.4 Conclusion validity

Conclusion validity refers to the statistically significant relationship between the treatment and the outcome [37]. The possible threats to conclusion validity are:

- Combining results from different experiments poses a high risk of comparing apples and oranges [38].
- Lack of reported moderator variables [64].
- The randomization of the data sets, the prerequisite for correct meta analysis is sometimes not checked [38].
- Studies are not always provided with the raw data [38].
- Effect sizes are not always checked for significance [38].
- Homogeneity test are not always performed [38].

The first problem can come from the usage of difference variables and measures among studies. This problem is extremely limited by selection of standard independent and dependent variables.

The second problem reflects fact that primary studies report only few context factors, which are the usage of programming language, business domain, defect collection phase. None of these moderator variables explains the existing heterogeneity satisfactorily. Many of process factors and personal factors that can contribute to find moderator variables, such as project development life cycle, development team structure, personnel capability and so on are missing. Lack of this information leads to an incomplete picture in subgroup analysis. Therefore, it suggests the better report in correlation and observational study for the purpose of aggregation.

As for randomization of the studies, we have effect sizes come from various publications and provide number of classes for each dataset. Nevertheless, we need to remark that all available data may not be a perfect representation of all data set. This problem cannot be checked if it is not mentioned in the primary studies.

Lacking of raw data is one of our main threats since it cause the problem in summary odds ratio from univariate logistic regression models. The calculation of standard error for logistic regression coefficient requires the information about 2x2 tables which is typical for logistic regression. However, this information is missing in all of the studies. There are only few of the studies reported this standard errors. This problem makes us unable to directly summary global odds ratio. Our alternative approach is to convert odds ratio to Pearson correlation coefficient and using dataset size as standard error. However this estimation may introduce a relative error.

The problem with heterogeneity and significance test doesn't arise since it is one of our main test and basic for subgroup and cluster analysis. Table 38 summarizes significant threats to the validity in this work.

Table 38: Summary of significant threats to validity

Internal validity	• Bias in selection of publications • Variety in quality of primary studies
External validity	• Limitation of study to observational and historical method
Construct validity	
Conclusion validity	• Lack of reported moderator variables • Lacking of raw data • Randomization of the data sets

Chapter 9
Conclusion

This section devotes to summarize the work done in the thesis, listing the major contributions of the thesis and naming some possible future work. In overall the main contributions of this thesis are:

1. Provides a state-of-art and comprehensive overview about the impact of design complexity on software cost and quality.
2. Provides a list of design metrics for constructing design complexity based models of software quality.
3. Presents a benchmark for Spearman correlation coefficients and Logistic regression models of Chidamber & Kemerer metric suite.

The contributions of the thesis are visualized by answering proposed research questions. The review of each research questions are described below.

9.1 Research questions revisited:

SQ1: Which quality attributes are predicted using design complexity metrics?

Firstly, quality attributes that are investigated in correlation and prediction studies were identified to answer SQ1. It was found that several external quality are investigated for correlation with design complexity, which are classified into 4 main group: maintainability (testability, maintenance effort, debugging effort, change proneness), reliability (fault proneness, fault density, vulnerability), development effort and reusability (volatility, refactoring cost). Among the quality sub-characteristics, software fault proneness and maintainability attract most concern of research interest with 75% of total studies.

SQ2: What kind of design complexity metrics is most frequently used in literature?

Thereafter, the frequency of using design complexity dimensions was identified to answer SQ2. The popularity of complexity dimensions are ranked by the number of studies in each complexity dimensions that are investigated for correlation with fault proneness and maintainability. It was revealed that for both fault proneness and maintainability studies, coupling are most concerned dimensions, and followed by scale and inheritance. There is fairly

low number of proposed cohesions metrics and polymorphism due to limitation of knowledge to these features.

SQ3: Which design complexity metrics is most frequently used in literature?

Thirdly, all the metrics that are investigated in the selected are accumulated and sorted by the number of studies investigated. The survey of 57 primary studies shows that Chidamber & Kemerer metric set was most used design metrics for correlation analysis and prediction external quality.

SQ4: Which design complexity metrics are potentially predictors of quality attribute?

To answer SQ4, the specific complexity metrics that are significantly related to quality attributes in correlation and regression analysis were identified to figure out the potential predictors of fault proneness and maintainability. Result of vote counting shows that there is an evidence of non-zero Spearman correlation coefficient and odds ratios between RFC, WMC, CBO, WMC McCabe, SDMC, AMC, NIM, NCM, NTM and fault proneness. In relation with maintainability, vote counting also reject null hypothesis for RFC, MPC, WMC and DAC. DIT and NOC are the only two metrics that are not found significant in either correlation or regression analysis with both fault proneness and maintainability. We also found the Pareto-like distribution applying for the choice of investigated design metric: 20% of the available metrics attracts 60% of investigation from researcher.

SQ5: Which design complexity metrics are helpful in constructing prediction model?

SQ5 attempts to find out the most useful complexity metrics in multivariate models by counting the number of times they are selected to construct the model. In term of complexity dimension, it was found that coupling; scale and inheritance are most frequently used for constructing multivariate prediction models, respectively. The three most used metrics are RFC, CBO and DIT. The comparison of predictive performance of design metrics in multivariate models suffers from variation in validation method, prediction technique and the initial set of investigated metrics. However, there are evidences that design-metric based prediction models perform not better than LOC-based prediction models do. Besides, we are not able to quantitatively summarize and generalize from multivariate models due to the variety in the use of metric set, modeling technique, validation method and accuracy measure.

SQ6: Is there an overall influence of these metrics on external quality attributes? What are the impacts of those metrics on those attributes?

To answer SQ6, a meta analysis is performed to aggregated Spearman correlation coefficient and estimated odds ratio for 6 design metrics in Chidamber & Kemerer metric set. It was found that it is not possible to find a global Spearman correlation coefficient due to high level of

heterogeneity. The result confirms the positive impact of WMC and RFC on software quality. With estimated odds ratio, global odds ratio is applicable for CBO, LCOM, RFC and WMC due to low or zero level of heterogeneity. DIT and NOC's odd ratios are attached with high level of heterogeneity. With in low-heterogeneity subgroup, it is found an impact order of design metrics on fault proneness. Among the group, WMC and RFC has more impact on fault prone than LCOM, CBO and DIT. NOC has a negative impact on fault proneness.

SQ7: Do studies agree on these influences? If no, what explains this inconsistency? Is this explanation inconsistency across different metrics?

From the question SQ6, we cannot conclude about the global impact of design metrics on fault proneness and maintainability due to appearance of high heterogeneity. SQ7 attempts to find the source of heterogeneity and its consistency over design metrics. The results show an inconsistency of source of heterogeneity among design metrics. Subgroup analysis showed that defect collection phase can explain more than 50% of observed variation between group for Spearman coefficient of CBO, LCOM, RFC, WMC and LOC. Domain can explain 75% of variation in sub group for NOC. In sub group analysis for odds ratio applied for DIT and NOC, Language and Domain are two moderators that explain more than 50% of variation between groups for effect size of NOC. In both case there is no subgroup moderator can explain efficient amount of heterogeneity for DIT. Cluster analysis confirms that no known moderator variables can explain for variation between studies satisfactorily.

Table 39 shows the summary of answers for proposed research questions.

Table 39: Summary of findings

Question	Answers
SQ1: Which quality attributes are predicted using design complexity metrics?	Reliability, maintainability, development effort and reusability
SQ2: What kind of design complexity metrics is most frequently used in literature?	In order: Coupling, scale, inheritance and cohesion.
SQ3: Which design complexity metrics is most frequently used in literature?	Fault proneness: RFC, WMC, CBO, WMC Mc Cabe, SDMC, AMC, NIM, NCM and NTM Maintainability: RFC, MPC, WMC and DAC
SQ4: Which design complexity metrics are potentially predictors of quality attribute?	C&K metric set: CBO, DIT, NOC, RFC, LCOM and WMC
SQ5: Which design complexity metrics are helpful in constructing prediction model?	C&K metric set: CBO, DIT, NOC, RFC, LCOM and WMC
SQ6: Is there an overall influence of these metrics on external quality attributes? What are the impacts of those metrics on those attributes?	No due to high heterogeneity. In few subgroups, WMC, RFC, CBO, LCOM shows stronger impact to fault proneness than DIT and NOC
SQ7: Do studies agree on these influences? If no,	No. Defect collection phase for few

what explains this inconsistency? Is this explanation inconsistency across different metrics?	design metrics.

9.2 Interpretation

Our findings can contribute to both research community and software practitioner in many ways. To researchers who are interested in quality prediction, we confirm the important of software coupling and scale dimension. To quality model builders, we suggest some design metrics that are potential predictors of fault proneness and maintainability. We also confirm that, regarding to the impact of design complexity on quality, correlation studies is too heterogeneous to generalize from them. In particular, there is no overall impact of design complexity on fault proneness. Besides, the fault prediction model based on design metrics should be built differently for pre release defect fault and post release fault.

To software practitioners who seek for early indicator of software quality, there are several suggestions. If the available software artifact is source code, practitioners should look for Line of Code (LOC) if they want to predict fault proneness. If the available software artifact is design documents, practitioners should look for WMC, CBO or RFC as the single indicator of fault proneness. Besides, the metrics that is similar with these can be used in any software systems, without restrict to Object oriented software. To construct a measurement framework that can reflect the software quality, ones can take design metric list in Appendix B as a reference.

9.3 Future work

The results of this thesis can be considered as a cornerstone for further study in the area of correlation and regression analysis of design complexity. After completion of this thesis, several tasks are identified as a future work to follow up the results of this thesis. Future work needed with respect to this thesis includes:

Aggregation results from multivariate prediction model:

> The main focus of this work is correlation analysis and univariate logistic regression models. While these methods can only assess the relationship between single metric and software quality, most of prediction models use combination of multiple metrics to increase the predictive power. An approach to aggregate the predictive ability of combination of design metrics is meaningful for the research area of cost and quality prediction.

Quality benchmarking:

One important application of design metrics is to build quality benchmarks for software systems. Since some design metrics has a moderate impact on software quality, values of those metrics can be used as benchmarking for complexity as well as quality of final software system. The complexity level or fault proneness level of different software modules or of same module but before and after rework can be compared by using the combination of metric set.

Empirical investigation of impact of context factors on software quality:

The meta analysis indicates the possible impact of context factors on fault proneness and maintainability. Therefore, these factors should be further studied by empirical studies. This study could be repeated with new moderator variables to identify the clusters in software projects.

Construction of a generic model prediction:

The idea is to build a quality prediction model, which is applicable across environment, project and company. Such a model requires the repository of prediction models with detail context information. Given a design complexity level, the conditional probability of a class to be faulty with is calculated for each context setting. The probability of a class to be faulty is the result of combination of conditional probability of each design metrics. Therefore, when there is a need to predict quality of developing software product, the model builder only need to collect information about context factors. The model parameters can be estimated based on parameters of similar models. Bayesian Network (BN) can implement this strategy or Case based reasoning (CBR). The generic prediction could make model builder free from collecting metrics data or defects.

Bibliography

[1] R.D. Banker, S.M. Datar, C.F. Kemerer, and D. Zweig, "Software complexity and maintenance costs," *Commun. ACM*, vol. 36, 1993, pp. 81-94.

[2] C. Bellini, R. Pereira, and J. Becker, "Measurement in software engineering: From the roadmap to the crossroads," *International Journal of Software Engineering and Knowledge Engineering*, vol. 18, 2008, pp. 37-64.

[3] L. Briand, J. Wüst, J. Daly, and D. Victor Porter, "Exploring the relationships between design measures and software quality in object-oriented systems," *Journal of Systems and Software*, vol. 51, 2000, pp. 245-273.

[4] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems with Applications*, vol. 36, 2009, pp. 7346-7354.

[5] T. DeMarco, "A metric of estimation quality," *Proceedings of the May 16-19, 1983, national computer conference*, Anaheim, California: ACM, 1983, pp. 753-756.

[6] N.E. Fenton, *Software Metrics: A Rigorous Approach*, Chapman Hamp; Hall, Ltd., 1991.

[7] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using Metrics to Predict OO Information Systems Maintainability," *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, Springer-Verlag, 2001, pp. 388-401.

[8] O. Gomez, H. Oktaba, M. Piattini, and F. Garcia, "A systematic review measurement in software engineering: State-of-the-art in measures," *ICSOFT 2006 - 1st International Conference on Software and Data Technologies, Proceedings*, 2006, pp. 224-231.

[9] B. A. Kitchenham, "Guidelines for performing Systematic Literature Reviews in Software Engineering", Ver 2.3, Keele University, EBSE Technical Report, 2007

[10] B. Kitchenham, "What's up with software metrics? - A preliminary mapping study," *J. Syst. Softw.*, vol. 83, 2010, pp. 37-51.

[11] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, 2009, pp. 367-377.

[12] M. Shepperd, Ed., *Software engineering metrics I: measures and validations*, McGraw-Hill, Inc., 1993.

[13] E. Arisholm, L. Briand, and E. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *Journal of Systems and Software*, vol. 83, 2010, pp. 2-17.

[14] M. Jørgensen, "Forecasting of software development work effort: Evidence on expert judgement and formal models," *International Journal of Forecasting*, vol. 23, 2007, pp. 449-462.

[15] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, 2004, pp. 37-60.

[16] M. Jørgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, vol. 33, 2007, pp. 33-53

[17] B.A. Kitchenham, E. Mendes, and G.H. Travassos, "Cross versus Within-Company Cost Estimation Studies: A Systematic Review," *IEEE Trans. Softw. Eng.*, vol. 33, 2007, pp. 316-329.

[18] L. C. Briand, J. Wüst, "Empirical Studies of Quality Models in Object-Oriented Systems", Advances in Computers, vol 56, 2002, pp. 98-167

[19] O. Dieste, A. Grimán, and N. Juristo, "Developing search strategies for detecting relevant experiments," *Empirical Software Engineering*, vol. 14, 2009, pp. 513-539.

[20] V.B. Kampenes, T. Dyba, J.E. Hannay, and D.I.K. Sjøberg, "Systematic review: A systematic review of effect size in software engineering experiments," *Inf. Softw. Technol.*, vol. 49, 2007, pp. 1073-1086.

[21] A.P. Field, "Meta-Analysis of Correlation Coefficients: A Monte Carlo Comparison of Fixed- and Random-Effects Methods," *Psychological Methods*, vol. 6, Jun. 2001, pp. 161-180.

[22] J.A. Lewis, S.M. Henry, D.G. Kafura, and R.S. Schulman, "An empirical study of the object-oriented paradigm and software reuse," *SIGPLAN Not.*, vol. 26, 1991, pp. 184-196.

[23] C. Ebert and R. Dumke, *Software Measurement: Establish - Extract - Evaluate - Execute*, Springer-Verlag New York, Inc., 2007.

[24] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object- Oriented Design," IEEE Trans. Software Eng., vol. 20, pp. 476-493, 1994

[25] L. Briand, J. Wuest, S. Ikonomovski, and H. Lounis, "A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study" Technical Report ISERN-98-29, Int'l Software Eng. Research Network, 1998

[26] W. Stevens, G. Myers, and L. Constantine, "Structured design," *Classics in software engineering*, Yourdon Press, 1979, pp. 205-232.

[27] L.C. Briand, J.W. Daly, and J.K. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Trans. Softw. Eng.*, vol. 25, 1999, pp. 91-121.

[28] G. Myers, *Composite/Structured Design*. Van Nostrand Reinhold, 1978

[29] L.C. Briand, J.W. Daly, and J. Wust, "A Unified Framework for Cohesion Measurement in Object-OrientedSystems," *Empirical Softw. Engg.*, vol. 3, 1998, pp. 65-117.

[30] A. Abubakar, J. AlGhamdi, and M. Ahmed, "Can Cohesion Predict Fault Density?," *Computer Systems and Applications, 2006. IEEE International Conference on.*, 2006, pp. 890-893.

[31] Meyer, Bertrand (1997). *Object-Oriented Software Construction, second edition*. Prentice Hall. ISBN 0-13-629155-4

[32] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Trans. Softw. Eng.*, vol. 20, 1994, pp. 476-493

[33] Simon, F., Beyer, D. "Considering Inheritance, Overriding, Overloading and Polymorphism for Measuring C++ Sources". Technical Report 04/00, Computer Science Reports, Technical University Cottbus, May (2000)

[34] N. Wilde, P. Matthews, and R. Huitt, "Maintaining Object-Oriented Software," *IEEE Softw.*, vol. 10, 1993, pp. 75-80.

[35] E. S. L. Mikhajlov. "The fragile base class problem and its impact on component systems". Proceedings of the Second International Workshop on Component-Oriented Programming (WCOP'97), number 5 in TUCS General Publications, pages 59-68. Turku Centre for Computer Science, 1997.

[36] Wikipedia, the free encyclopedia [Online]. Available: http://en.wikipedia.org/wiki/Polymorphism_in_object-oriented_programming

[37] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in software engineering: an introduction, Kluwer Academic Publishers, 2000.

[38] Succi, G. , Pedrycz, W., Djokic, S., Zuliani, P., Russo, B., "An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite", *Empirical Software Engineering* 10(1): 81-104 , 2005

[39] Endres, A., Rombach, D., *A Handbook of Software and System Engineering*, Harlow: Pearson Addison Wesley, 2003

[40] B. Boehm, H.D. Rombach, and M.V. Zelkowitz, *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili*, Springer-Verlag New York, Inc., 2005.

[41] C. Wohlin, M. Höst, and K. Henningsson, "Empirical Research Methods in Software Engineering," *Empirical Methods and Studies in Software Engineering*, Springer Berlin / Heidelberg, 2003, pp. 7-23.

[42] M.V. Zelkowitz and D.R. Wallace, "Experimental Models for Validating Technology," *Computer*, vol. 31, 1998, pp. 23-31

[43] A. Höfer and W. Tichy, "Status of empirical research in software engineering," *Lecture Notes in Computer Science*, vol. 4336/2007, pp. 10-19, 2007.

[44] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*, 2nd ed. Routledge Academic, January 1988.

[45] R.B. Grady, *Practica Software Metrics for Project Management and Process Improvement,* Hewlett-Packard Professional Books, Prentice Hall, *New Jersey*,1992.

[46] J. Devore, *Probability and Statistics for Engineering and the Sciences,* 7th Ed. Thomson Brooks, 2008.

[47] J. Aldrich, "Correlations Genuine and Spurious in Pearson and Yule," *Statistical Science*, vol. 10, Nov. 1995, pp. 364-376.

[48] J. Pearl, "Why there is no statistical test for confounding, why many think there is, and why they are almost right," UCLA Computer Science Department, Technical Report R-256, January 1998.

[49] W. Li and S. Henry, Object-oriented metrics that predict maintainability, *Journal of Systems and Software* **23** (1993), pp. 111–122.

[50] B. Kitchenham, H. Al-Khilidar, M.A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu, "Evaluating guidelines for empirical software engineering studies," *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, Rio de Janeiro, Brazil: ACM, 2006, pp. 38-47.

[51] J. Aldrich (1998) Doing Least Squares: Perspectives from Gauss and Yule, *International Statistical Review*, Vol.66, (1), pp. 61-81

[52] J.P. Guilford, *Fundamental statistics in psychology and education / J.P. Guilford, Benjamin Fruchter*, New York, McGraw-Hill, 1973.

[53] H. Cooper, L. V. Hedges, J. C. Valentine, *The handbook of research synthesis and meta-analysis* (2nd ed.), New York, Russell Sage Foundation, 2009.

[54] J. W. Creswell, *Research design : qualitative, quantitative, and mixed method approaches*, 2nd ed. Sage Publications, July 2003.

[55] J. E. Hannay, T. Dybå, E. Arisholm, D. I. K. Sjøberg, *The Effectiveness of Pair-Programming: A Meta-Analysis*, Information and Software Technology 55(7):1110-1122, 2009.

[56] M.E. Manso, J.A. Cruz-Lemus, M. Genero, and M. Piattini, "Empirical Validation of Measures for UML Class Diagrams: A Meta-Analysis Study," *Models in Software Engineering: Workshops and Symposia at MODELS 2008, Toulouse, France, September 28 - October 3, 2008. Reports and Revised Selected Papers*, Springer-Verlag, 2009, pp. 303-313.

[57] D. W. Hosmer and S. Lemeshow, *Applied logistic regression (Wiley Series in probability and statistics)*. Wiley-Interscience Publication, September 2000

[58] "IEEE Standard for a Software Quality Metrics Methodology," *IEEE Std 1061-1998*, 1998.

[59] ISO, "International standard ISO/IEC 9126. Information technology: Software product evaluation: Quality characteristics and guidelines for their use." 1991

[60] IEEE, IEEE Standard Glossary of Software Engineering Terminology, report IEEE Std 610.12- 1990, IEEE, 1990.

[61] M.A. Ellis and B. Stroustrup, *The annotated C++ reference manual*, Addison-Wesley Longman Publishing Co., Inc., 1990.

[62] L.M. Pickard, B.A. Kitchenham, and P.W. Jones, "Combining empirical results in software engineering," *Information and Software Technology*, vol. 40, Dec. 1998, pp 811-821

[63] R.J. Light and P.V. Smith, Accumulating evidence: Procedures for resolving contradictions among different research studies. *Harvard Educational Review* **41** (1971), pp. 429–471

[64] M. Ciolkowski, "Aggregation of Empirical Evidence," *Empirical Software Engineering Issues. Critical Assessment and Future Directions*, Springer Berlin / Heidelberg, 2007, p. 20

[65] LV. Hedges, I. Olkin, *Statistical Methods for Meta-analysis. Orlando*, FL: Academic Press, 1995.

[66] H. Cooper, L. Hedges, "Research synthesis as a scientific enterprise", Handbook of research synthesis (pp. 3-14). New York: Russell Sage, 1994.

[67] A. G. Koru, H. Liu, "Building effective defect- prediction models in practice", IEEE Software, 2005, pp.23-29

[68] J.P. Higgins, S.G. Thompson, J.J. Deeks, D.G. Altman, "Measuring inconsistency in meta-analyses", Brit. Med. J. 327 (4) (2003) 557–560

[69] J. Lau, J. P. Ioannidis, C. H. Schmid, "Quantitative synthesis in systematic reviews", Ann Intern Med; 127:820-6, 1997.

[70] M. Ciolkowski, "What do we know about perspective-based reading? An approach for quantitative aggregation in software engineering," *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, IEEE Computer Society, 2009, pp. 133-144.

[71] A. D. Oxman, G. H. Guyatt, "A consumer's guide to subgroup analyses", Annals of Internal Medicine; 116:78-84, 1992.

[72] J. Copas, JQ. Shi, **"**Meta-analysis, funnel plots and sensitivity analysis", *Biostatistics* 2000, 1(3)**:**247-262

[73] Duval, S. and Tweedie, R. (2000), "Trim and Fill: A Simple Funnel-Plot–Based Method of Testing and Adjusting for Publication Bias in Meta-Analysis". Biometrics, 56: 455–463

[74] Matthias Egger, G. Davey Smith, M. Schneider & C. Minder (September 1997). "Bias in meta-analysis detected by a simple, graphical test" *BMJ* **315** (7109): 629–624

[75] The R project for statistical computing [Online]. Available: http://www.r-project.org

[76] Comprehensive Meta analysis by Biostat [Online].Available: http://www.meta-analysis.com

[77] "Zotero | Home" [Online]. Available: http://www.zotero.org/

[78] R. Shatnawi, "A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems," *Software Engineering, IEEE Transactions on*, vol. 36, 2010, pp. 216-225.

[79] V. Basili, L. Briand, and W. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, 1996, pp. 751-761.

[80] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter language reuse," *Proceedings - International Conference on Software Engineering*, 2008, pp. 19-24.

[81] M. Cartwright and M. Shepperd, "An empirical investigation of an object-oriented software system," *IEEE Transactions on Software Engineering*, vol. 26, 2000, pp. 786-796.

[82] H. Olague, L. Etzkorn, S. Messimer, and H. Delugach, "An empirical validation of object-oriented class complexity metrics and their ability to predict error-prone classes in highly iterative, or agile, software: A case study," *Journal of Software Maintenance and Evolution*, vol. 20, 2008, pp. 171-197.

[83] J. Xu, D. Ho, and L. Capretz, "An empirical validation of object-oriented design metrics for

fault prediction," *Journal of Computer Science*, vol. 4, 2008, pp. 583-589.

[84] R. Harrison, S. Counsell, and R. Nithi, "An investigation into the applicability and validity of object-oriented design metrics," *Empirical Software Engineering*, vol. 3, 1998, pp. 255-273.

[85] F. Wilkie and B. Kitchenham, "An investigation of coupling, reuse and maintenance in a commercial C++ application," *Information and Software Technology*, vol. 43, 2001, pp. 801-812.

[86] M. Elish and K. Elish, "Application of TreeNet in predicting object-oriented software maintainability: A comparative study," *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 2009, pp. 69-78.

[87] M. Genero, J. Olivas, M. Piattini, and F. Romero, *Assessing object-oriented conceptual models maintainability*, 2003.

[88] L. Briand, W. Melo, and J. Wurst, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Transactions on Software Engineering*, vol. 28, 2002, pp. 706-720.

[89] N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Predicting the Probability of Change in Object-Oriented Systems," *IEEE Trans. Softw. Eng.*, vol. 31, 2005, pp. 601-614.

[90] M. Bocco, D. Moody, and M. Piattini, "Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation," *Journal of Software Maintenance and Evolution*, vol. 17, 2005, pp. 225-246.

[91] A. Han, S. Jeon, D. Bae, and J. Hong, "Behavioral dependency measurement for change-proneness prediction in UML 2.0 design models," *Proceedings - International Computer Software and Applications Conference*, 2008, pp. 76-83.

[92] A. Abubakar, J. AlGhamdi, and M. Ahmed, "Can Cohesion Predict Fault Density?," *Computer Systems and Applications, 2006. IEEE International Conference on.*, 2006, pp. 890-893.

[93] L. Briand, S. Morasca, and V.R. Basili, *Defining and validating high-level design metrics*, University of Maryland at College Park, 1994.

[94] M. Chaumun, H. Kabaili, R.K. Keller, F. Lustman, and G. Saint-Denis, "Design properties and object-oriented software changeability," *Proceedings of the Euromicro Conference on Software Maintenance and Reengineering, CSMR*, 2000, pp. 45-54.

[95] E. Arisholm, L. Briand, and A. Fayen, "Dynamic coupling measurement for object-oriented software," *IEEE Transactions on Software Engineering*, vol. 30, 2004, pp. 491-506.

[96] K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study," *Software Process Improvement and Practice*, vol. 14, 2009, pp. 39-62.

[97] R. Subramanyam and M. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects," *IEEE Transactions on Software Engineering*, vol. 29, 2003, pp. 297-310.

[98] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," *IEEE Transactions on Software Engineering*, vol. 32, 2006, pp.

771-789.

[99] G. Pai and J. Dugan, "Empirical analysis of software fault content and fault proneness using Bayesian methods", *IEEE Transactions on Software Engineering*, vol. 33, 2007, pp. 675-686.

[100] E. Arisholm, "Empirical assessment of the impact of structural properties on the changeability of object-oriented software", *Information and Software Technology*, vol. 48, 2006, pp. 1046-1055.

[101] B. Goel and Y. Singh, *Empirical investigation of metrics for fault prediction on object-oriented software*, 2008.

[102] M. Tang, M. Kao, and M. Chen, "Empirical study on object-oriented metrics", *International Software Metrics Symposium, Proceedings*, 1999, pp. 242-249.

[103] Y. Singh, A. Kaur, and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models", *Software Quality Journal*, vol. 18, 2009, pp. 3-35.

[104] T. Gymthony, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction", *IEEE Transactions on Software Engineering*, vol. 31, 2005, pp. 897-910.

[105] H. Olague, L. Etzkorn, S. Gholston, and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly Iterative or agile software development processes," *IEEE Transactions on Software Engineering*, vol. 33, 2007, pp. 402-419.

[106] F. Brito e Abreu and W. Melo, "Evaluating the impact of object-oriented design on software quality," *International Software Metrics Symposium, Proceedings*, 1996, pp. 90-99.

[107] L.C. Briand, J. Wurst, J.W. Daly, and D.V. Porter, "Exploring the relationship between design measures and software quality in object-oriented systems," *J. Syst. Softw.*, vol. 51, 2000, pp. 245-273.

[108] A. Janes, M. Scotto, W. Pedrycz, B. Russo, M. Stefanovic, and G. Succi, "Identification of defect-prone classes in telecommunication software systems using design metrics," *Information Sciences*, vol. 176, 2006, pp. 3711-3734.

[109] P. Tomaszewski, L. Lundberg, and H. Grahn, "Increasing the efficiency of fault detection in modified code," *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, 2005, pp. 421-430.

[110] K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Investigating effect of design metrics on fault proneness in object-oriented systems," *Journal of Object Technology*, vol. 6, 2007, pp. 127-141.

[111] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," *Proceedings - International Conference on Software Engineering*, 2006, pp. 452-461.

[112] S. Misra, "Modeling design/Coding factors that drive maintainability of software systems," *Software Quality Journal*, vol. 13, 2005, pp. 297-320.

[113] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," *Journal of Systems and Software*, vol. 83, 2010, pp. 660-

674.

[114] J. Bieman, D. Jain, and H. Yang, "OO design patterns, design structure, and program changes: An industrial case study," *Conference on Software Maintenance*, 2001, pp. 580-591.

[115] S. Benlarbi and W.L. Melo, "Polymorphism measures for early risk prediction," *Proceedings - International Conference on Software Engineering*, 1999, pp. 334-344.

[116] G. Succi, W. Pedrycz, M. Stefanovic, and J. Miller, "Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics," *Journal of Systems and Software*, vol. 65, 2003, pp. 1-12.

[117] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," *Proceedings - ICSE 2007 Workshops: Third International Workshop on Predictor Models in Software Engineering, PROMISE'07*, 2007.

[118] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," *Proceedings - International Conference on Software Engineering*, 2008, pp. 531-540.

[119] E. Arisholm and L. Briand, "Predicting fault-prone components in a Java legacy system," *ISESE'06 - Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering*, 2006, pp. 8-17.

[120] Ping Yu, T. Systa, and H. Muller, "Predicting fault-proneness using OO metrics. An industrial case study," *Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on*, 2002, pp. 99-107.

[121] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," *Journal of Systems and Software*, vol. 80, 2007, pp. 1349-1361.

[122] L. Wang, X. Hu, Z. Ning, and W. Ke, "Predicting object-oriented software maintainability using projection pursuit regression," *2009 1st International Conference on Information Science and Engineering, ICISE 2009*, 2009, pp. 3827-3830.

[123] Y. Zhou and B. Xu, "Predicting the maintainability of open source software using design metrics," *Wuhan University Journal of Natural Sciences*, vol. 13, 2008, pp. 14-20.

[124] A. Binkley and S. Schach, "Prediction of run-time failures using static product quality metrics," *Software Quality Journal*, vol. 7, 1998, pp. 141-147.

[125] L. Briand, J. Wust, and H. Lounis, "Replicated case studies for investigating quality factors in object-oriented designs," *Empirical Software Engineering*, vol. 6, 2001, pp. 11-58.

[126] M. Ware, F. Wilkie, and M. Shapcott, "The application of product measures in directing software maintenance activity," *Journal of Software Maintenance and Evolution*, vol. 19, 2007, pp. 133-154.

[127] K. El Emam, S. Benlarbi, N. Goel, and S. Rai, "The confounding effect of class size on the validity of object-oriented metrics," *IEEE Transactions on Software Engineering*, vol. 27, 2001, pp. 630-650.

[128] R. Shatnawi and W. Li, "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process," *Journal of Systems and Software*, vol. 81, 2008, pp. 1868-1882.

[129] K. El Emam, W. Melo, and J. Machado, "The prediction of faulty classes using object-oriented design metrics," *Journal of Systems and Software*, vol. 56, 2001, pp. 63-75.

[130] A.B. Binkley and S.R. Schach, "Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures," *Proceedings - International Conference on Software Engineering*, 1998, pp. 452-455.

[131] A. A. A. Zouri, "Emprical validation of class coupling metric as changability indicators in software evolution", *Master thesis,* King Fahd University of Petroleum and Minerals, 2009.

[132] A. Schroter, T. Zimmermann, R. Premraj, A. Zeller, "If Your Bug Database Could Talk…", *International Symposium on Empirical Software Engineering,* 2006.

[133] E. Arisholm, "Empirical assessment of changeability in object-oriented software", *PhD thesis*, University of Oslo, ISSN 1501-7710.

[134] A. B. Binkley , S. R. Schach , "Inheritance-Based Metrics for Predicting Maintenance Effort: An Empirical Study", *Technical Report TR97-05.*

[135] M. Lorenz and J. Kidd, *Object-oriented software metrics: a practical guide*, Prentice-Hall, Inc., 1994.

[136] S. Easterbrook, "Empirical research methods for software engineering," *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, Atlanta, Georgia, USA: ACM, 2007, pp. 574-574.

[137] W. Li and S. Henry, "Maintenance metrics for the object oriented paradigm," *Software Metrics Symposium, 1993. Proceedings, First International*, 1993, pp. 52-60.

[138] W. Li, "Another metric suite for object-oriented programming," *J. Syst. Softw.*, vol. 44, 1998, pp. 155-162.

[139] A. Lake and C.R. Cook, *Use of Factor Analysis to Develop OOP Software Complexity Metrics*, Oregon State University, 1994.

[140] M. Hitz and B. Montazeri, "Measuring Product Attributes of Object-Oriented Systems," *Proceedings of the 5th European Software Engineering Conference*, Springer-Verlag, 1995, pp. 124-136.

[141] S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," *IEEE Trans. Softw. Eng.*, vol. 7, 1981, pp. 510-518.

[142] B. Henderson-Sellers, *Object-oriented metrics: measures of complexity*, Prentice-Hall, Inc., 1996.

[143] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Trans. Softw. Eng.*, vol. 20, 1994, pp. 476-493.

[144] S.R. Chidamber and C.F. Kemerer, "Towards a metrics suite for object oriented design," *SIGPLAN Not.*, vol. 26, 1991, pp. 197-211.

[145] J.M. Bieman and B. Kang, "Cohesion and reuse in an object-oriented system," *SIGSOFT Softw. Eng. Notes*, vol. 20, 1995, pp. 259-262.

[146] S. Benlarbi and W.L. Melo, "Polymorphism measures for early risk prediction," *Proceedings*

of the 21st international conference on Software engineering, Los Angeles, California, United States: ACM, 1999, pp. 334-344.

[147] F.B.E. Abreu and W. Melo, "Evaluating the Impact of Object-Oriented Design on Software Quality," *Proceedings of the 3rd International Symposium on Software Metrics: From Measurement to Empirical Results*, IEEE Computer Society, 1996, p. 90.

[148] M. Christerson, P. Jonsson, and G. Overgaard, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Reading, MA: Addison Wesley, 1992, p. 524.

[149] D.P. Tegarden S.D. Sheetz and D.E. Monarchi, "Effectiveness of Traditional Software Metrics for Object-Oriented Systems," *Proc. Conf. Systems Science*,Hawaii, 1992.

[150] Y.-S. Lee, B.-S. Liang, S.-F. Wu, F.-J. Wang, "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow", in *Proc. International Conference on Software Quality*, Maribor, Slovenia, 1995.

[151] J. Bansiya, L. Etzkorn, C. Davis, and W. Li, A class cohesion metric for object-oriented designs, *Journal of Object-Oriented Program*, Vol. 11, No. 8, pp. 47-52. 1999.

[152] PROMISE Software Engineering Repository [Online], Available: http://promise.site.uottawa.ca/SERepository/index.html

[153] D. G. Bonett, *Transforming odds ratios into correlations for meta-analytic research*, American Psychologist 62, 254-255, 2007

[154] V. Rousson, T. Gasser, and B. Seifert, "Assessing intrarater, interrater and test–retest reliability of continuous measurements," *Statistics in Medicine*, vol. 21, 2002, pp. 3431-3446.

Appendix A

Primary studies selected for systematic review

No	Code	Title	Author
1	Sha10	A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems	R. Shatnawi
2	BBM96	A validation of object-oriented design metrics as quality indicators	V. Basili, L. Briand, and W. Melo
3	WKK08	Adapting a fault prediction model to allow inter language reuse	S. Watanabe, H. Kaiya, and K. Kaijiri
4	CS00	An empirical investigation of an object-oriented software system	M. Cartwright and M. Shepperd
5	OEM+08	An empirical validation of object-oriented class complexity metrics and their ability to predict error-prone classes in highly iterative, or agile, software: A case study	H. Olague, L. Etzkorn, S. Messimer, and H. Delugach
6	XHC08	An empirical validation of object-oriented design metrics for fault prediction	J. Xu, D. Ho, and L. Capretz
7	HCN98	An investigation into the applicability and validity of object-oriented design metrics	R. Harrison, S. Counsell, and R. Nithi
8	WK01	An investigation of coupling, reuse and maintenance in a commercial C++ application	F. Wilkie and B. Kitchenham
9	EE09	Application of TreeNet in predicting object-oriented software maintainability: A comparative study	M. Elish and K. Elish
10	GOP+03	Assessing object-oriented conceptual models maintainability	M. Genero, J. Olivas, M. Piattini, and F. Romero
11	BMW02	Assessing the applicability of fault-proneness models across object-oriented software projects	L. Briand, W. Melo, and J. Wurst
12	NAG05	Predicting the Probability of change in Object oriented systems	Nikolaos T., Alexander C., George S.
13	BMP05	Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation	M. Bocco, D. Moody, and M. Piattini

14	HJB+08	Behavioral dependency measurement for change-proneness prediction in UML 2.0 design models	A. Han, S. Jeon, D. Bae, and J. Hong
15	AAA06	Can Cohesion Predict Fault Density?	A. Abubakar, J. AlGhamdi, and M. Ahmed
16	BMB94	Defining and validating high-level design metrics	L. Briand, S. Morasca, and V.R. Basili
17	CKK+00	Design properties and object-oriented software changeability	M. Chaumun, H. Kabaili, R.K. Keller, F. Lustman, and G. Saint-Denis
18	ABF04	Dynamic coupling measurement for object-oriented software	E. Arisholm, L. Briand, and A. FÃ¸yen
19	ASK+09	Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study	K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra
20	SK03	Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects	R. Subramanyam and M. Krishnan
21	ZL06	Empirical analysis of object-oriented design metrics for predicting high and low severity faults	Y. Zhou and H. Leung
22	PD07	Empirical analysis of software fault content and fault proneness using Bayesian methods	G. Pai and J. Dugan
23	Ari06	Empirical assessment of the impact of structural properties on the changeability of object-oriented software	E. Arisholm
24	GS08	Empirical investigation of metrics for fault prediction on object-oriented software	B. Goel and Y. Singh
25	TKC09	Empirical study on object-oriented metrics	M. Tang, M. Kao, and M. Chen
26	SKM09	Empirical validation of object-oriented metrics for predicting fault proneness models	Y. Singh, A. Kaur, and R. Malhotra
27	GFS05	Empirical validation of object-oriented metrics on open source software for fault prediction.	T. Gymothy, R. Ferenc, and I. Siket
28	OEG+07	Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly Iterative or agile software development processes	H. Olague, L. Etzkorn, S. Gholston, and S. Quattlebaum
29	BM96	Evaluating the impact of object-oriented design on software quality	F. Brito e Abreu and W. Melo
30	BWD+00	Exploring the relationship between design measures and software quality in object-oriented systems	L.C. Briand, J. Wurst, J.W. Daly, and D.V. Porter,]

31	JSP+06	Identification of defect-prone classes in telecommunication software systems using design metrics	A. Janes, M. Scotto, W. Pedrycz, B. Russo, M. Stefanovic, and G. Succi
32	TLG05	Increasing the efficiency of fault detection in modified code	P. Tomaszewski, L. Lundberg, and H. Grahn
33	ASK+07	Investigating effect of design metrics on fault proneness in object-oriented systems	K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra,
34	NBZ06	Mining metrics to predict component failures	N. Nagappan, T. Ball, and A. Zeller
35	Mis05	Modeling design/Coding factors that drive maintainability of software systems	S. Misra
36	ZXL10	On the ability of complexity metrics to predict fault-prone classes in object-oriented systems	Y. Zhou, B. Xu, and H. Leung
37	BJY01	OO design patterns, design structure, and program changes: An industrial case study	J. Bieman, D. Jain, and H. Yang
38	BM99	Polymorphism measures for early risk prediction	S. Benlarbi and W.L. Melo
39	SPS+03	Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics	G. Succi, W. Pedrycz, M. Stefanovic, and J. Miller
40	ZPZ07	Predicting defects for eclipse	T. Zimmermann, R. Premraj, and A. Zeller
41	ZN08	Predicting defects using network analysis on dependency graphs	T. Zimmermann and N. Nagappan
42	AB06	Predicting fault-prone components in a Java legacy system	E. Arisholm and L. Briand
43	YSM02	Predicting fault-proneness using OO metrics. An industrial case study	Ping Yu, T. Systa, and H. Muller
44	ZL07	Predicting object-oriented software maintainability using multivariate adaptive regression splines	Y. Zhou and H. Leung
45	WHN+09	Predicting object-oriented software maintainability using projection pursuit regression	L. Wang, X. Hu, Z. Ning, and W. Ke
46	ZX08	Predicting the maintainability of open source software using design metrics	Y. Zhou and B. Xu
47	BS98	Prediction of run-time failures using static product quality metrics	A. Binkley and S. Schach
48	BWL01	Replicated case studies for investigating quality factors in	L. Briand, J. Wurst,

		object-oriented designs	and H. Lounis
49	WWS07	The application of product measures in directing software maintenance activity	M. Ware, F. Wilkie, and M. Shapcott
50	EBG+01	The confounding effect of class size on the validity of object-oriented metrics	K. El Emam, S. Benlarbi, N. Goel, and S. Rai
51	SL08	The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process	R. Shatnawi and W. Li
52	EMM01	The prediction of faulty classes using object-oriented design metrics	K. El Emam, W. Melo, and J. Machado
53	BS98	Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures	A.B. Binkley and S.R. Schach
54	Ali09	Empirical validation of class coupling metric as changeability indicators in software evolution	Ali Abdallah Al Zouri
55	SZP+06	If Your Bug Database Could Talk. . .	A. Schroter, T. Zimmermann, R. Premraj, A. Zeller
56	Ari01	Empirical assessment of changeability in object-oriented software	E. Arisholm,
57	BS97	Inheritance-Based Metrics for Predicting Maintenance Effort: An Empirical Study	Aaron B. Binkley , Stephen R. Schach

Appendix B

List of structural complexity metrics in the selected studies

Metric name	Definition	Original	Type	Predictor for
NOC	Number of classes that directly inherit from a given class	[144]	inheritance	F
DIT	Length of the longest path from the class to the root of inheritance hierarchy	[144]	inheritance	F
CBO	Number of other classes to which it is coupled, included inheritance-based coupling	[143]	coupling	F+M
CBO'	CBO + not count inheritance based coupling	[144]	coupling	F+M
CBOback	CBO that count only the back coupling	[85]	Coupling	M
CBOforward	CBO that count only the forward coupling	[85]	Coupling	M
LCOM1	number of pairs of methods in the class using no attribute in common	[144]	cohesion	F
LCOM2	LCOM1 minus count number of pairs of methods that do (average percentage of the method in a class using an attribute and subtract from 100%)	[143]	cohesion	F+M
LCOM3	number of connected components of undirected Graph where each node is a method and edge is the sharing attribute in common	[140]	cohesion	F
LCOM4	LCOM3 + G has an edge between vertices representing methods m and n if m invokes n or vice versa	[140]	cohesion	F
LCOM5	sum of number of methods witch reference attribute Ai	[140]	cohesion	F

LCOMN	LCOM2 but accept negative value	[143]	cohesion	F
WMC	Count number of all methods in a class with unified weight	[144]	size	F
WMC McCabe	WMC with weight as Mc Cabe	[144]	size	F+M
RFCinf	number of methods in the response set of the class (set that can potentially be executed in response to a message received by an object of that class)	[144]	coupling	F
RFC1	RFCinf + not count methods indirectly invoked by methods in M	[144]	coupling	F
NAI	number of attributes in a class	[107]	size	F
RFC in	RFC but count couplings between two classes that are in a subclass – super class relationships	[120]	coupling	F
RFC out	RFC that count couplings between two classes that are not in inheritance hierarchy	[120]	coupling	F
CBO in	CBO but count couplings between two classes that are in a subclass – super class relationships	[120]	cohesion	F
CBO out	CBO that count couplings between two classes that are not in inheritance hierarchy	[120]	cohesion	F
IFCAIC	interaction between classes: first part (A: coupling to ancestor, D: descendents, F: friend classes, IF: Inverse friends, O: Others), second part: (CA: class-attribute, CM: class-method, MM: method-method), last part: (IC: import, EC: export coupling)	[107]	coupling	F
FCAEC		[107]	coupling	F
IFCMIC		[107]	coupling	F
FCMEC		[107]	coupling	F
IFMMIC		[107]	coupling	F
FMMEC		[107]	coupling	F
ACAIC		[107]	coupling	F
OCAIC		[107]	coupling	F+M
DCAEC		[107]	coupling	F
OCAEC		[107]	coupling	F+M

ACMIC		[107]	coupling	F
OCMIC		[107]	coupling	F+M
DCMEC		[107]	coupling	F
OCMEC		[107]	coupling	F+M
OMMIC		[107]	coupling	F
AMMIC		[107]	coupling	F
OMMEC		[107]	coupling	F
DMMEC		[107]	coupling	F
NMpub	number of public methods	[107]	size	F
NOP	number of classes that a given class directly inherits from	[139]	inheritance	F
NMO	number of methods in a class that override a method inherits from its ancestors and does not override	[139]	inheritance	F
NMA	number of new methods in a class, not inherited, not overriding	[139]	size	F
NOA	number of classes that a given class directly or indirectly inherits from	[149]	inheritance	F
SIX	NMO * DIT/ (NMO +NMA + NMI)	[135]	inheritance	F
MPC	number of method invocations in a class	[137]	coupling	F+M
TCC	percentage of pairs of public methods of the class which are connected	[145]	cohesion	F
LCC	TCC but consider pairs of indirectly connected as well	[145]	cohesion	F
NMNpub	number of non public methods	[107]	size	F
NumPara	sum of the number of parameters of the methods implemented in a class	[107]	size	F
AID	average inheritance depth of its parent classes + 1	[142]	inheritance	F
CLD	maximum number of levels in the hierarchy that are below the class	[149]	inheritance	F
NOD	number of classes that indirectly or directly inherit from a class	[139]	inheritance	F

NMI	number of methods in a class that the class inherits from its ancestors and does not override	[135]	inheritance	F
DAC	number of attributes in a class that have as their type another class	[137]	coupling	F+M
DAC'	number of different classes that are used as types of attributes in a class	[137]	coupling	F+M
ICP	number of method invocations in a class, weighted by the number of parameters of the invoked	[150]	coupling	F+M
IH-ICP	ICP + counts invocations of methods of ancestors of classes	[150]	coupling	F+M
NIH-ICP	ICP + counts invocations to classes not related through inheritance	[150]	coupling	F+M
ICH	number of invocations of other methods of the same class, weighted by the number of parameters of the invoked method	[150]	cohesion	F
Co	E-V+1/ (V-1)(V-2) (in graph G)	[140]	cohesion	F
Coh	Variation of LCOM5	[107]	cohesion	F
NIM	number of methods in an instance object of a class	[135]	size	F
NCM	Number of class methods. The number of class methods implemented in a class, w here a class method denote a method that only access the data belonging to the class itself	[135]	size	F
SPA	static polymorphism in ancestor	[146]	polymorphism	F
DPA	dynamic polymorphism in ancestor	[146]	polymorphism	F
SPD	static polymorphism in descendant	[146]	polymorphism	F
DPD	dynamic polymorphism in descendant	[146]	polymorphism	F
SP	SPA +SPD	[146]	polymorphism	F
DP	DPA + DPD	[146]	polymorphism	F
NIP	polymorphism in non-inheritance relation	[146]	polymorphism	F
OVO	overloading in standalone class	[146]	polymorphism	F
AHF	attribute hiding factor: (1 - no of visible attribs)/ no of attribs	[147]	polymorphism	F
MHF	method hiding factor: (1 - no of visible methods)/ no of methods	[147]	polymorphism	F+M

AIF	attribute inheritance factor: no of inherited attribs/ no of attribs	[147]	inheritance	F
MIF	method inheritance factor: no of inherited methods/ no of methods	[147]	inheritance	F
POF	Polymorphism factor	[147]	polymorphism	F
COF	Coupling factor	[147]	coupling	F+M
ANA	average DIT of all classes in a system	[151]	inheritance	F
CTA	counts the number of reference types used in the attribute declarations	[138]	coupling	F
CTM	counts the number of method call expressions made into body of the measured method	[138]	coupling	F
NOO	counts number of operations	[138]	size	F
NPAVG	average number of parameters per method in a class, not include inherited	[135]	size	F
STATE	number of state per class in state model	[81]	size	F
EVENT	number of event per class in state model	[81]	size	F
NC	number of class	Classic	size	F+M
	number of global attributes	Classic	size	F
	number of private attributes	Classic	size	F
Base	number of immediate base classes	Classic	size	F
PAR	number of parameters	Classic	size	F
NF	number of attributes	Classic	size	F+M
NM	number of methods	Classic	size	F+M
SIZE2	NF + NM	Classic	Size	F+M
NSF	number of static fields	Classic	size	F
NSM	number of static methods	Classic	size	F
MO	number of methods that are overridden	[114]	Size	M
ACD	number of anonymous type declaration	[117]	size	M
NOI	number of interfaces	[117]	size	M

NOCU	number of files	[117]	size	M
DCC	direct coupling class	[151]	coupling	M
MOA	measure of aggregation: percentage of data declarations in the system with user-defined type to system-defined type	[151]	coupling	M
DAM	number of private and protected attribs/ no of attribs	[151]	size	M
Friend	number of friend method (only in C/C++)	[114]	Size	M
Desc	number of all classes that are derived from the class either directly or indirectly.	[114]	Inheritance	M
NAINH	number of inherited attributes in a class	[114]	inheritance	M
Fan out	number of its immediately subordinate modules	[141]	dependency	M
Fan in	number of its immediately super ordinate (i.e., parent or boss) modules	[141]	dependency	M
NBD	nested block depth		dependency	M
Class Coupling	No of classes coupled with C (attribute / parameter / return types)		coupling	M
CCM	Class Connection Metric	Jarallah 2001	cohesion	M
CAMC	Enhanced Class Connection Metric	Jarallah 2001	cohesion	M
CDM			coupling	M
CFF	Coupling complexity in the forward direction of a class linkage	[85]	coupling	M
CBB	Coupling complexity in the backward direction of a class linkage	[85]	coupling	M
MaxHAgg	The total number of aggregation hierarchies (whole-part structures) within a class diagram.	[87]	coupling	M
Nagg	The total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship).	[87]	coupling	M
NAggH	The total number of aggregation hierarchies (whole-part structures) within a class diagram.	[87]	coupling	M
NGenH	The total number of generalization hierarchies within a class	[87]	coupling	M

	diagram.			
CDM	coupling dependency metric	I. Jacobson 92	coupling	F+M
No of clients	number of class that use a given class	Binkley 98	coupling	M
CHNL	class hierarchy nesting level	M. Lorenz 93	Inheritance	M
NCIM	number of classes inheriting a method	I. Jacobson 92	Inheritance	M

Appendix C

Extraction form

Metadata	
Extraction date	
Code	
Title	
Author	
Year	
Source	
Extraction form	
Research type	Choose an item.
Dependent variable	
Dependent variable	Fault proneness
Metric collected (+unit)	
Collected by	
When to collect?	Choose an item.
Independent variable	
Metric collected (+unit)	
Collected by	
Complexity dimension	Choose an item.
Metric granularity	Choose an item.
When to collect?	Choose an item.
Context	
Type of data set	Choose an item.
Nature of software	Choose an item.
Industry domain	
Programming Language	Choose an item.
Development tool	
Development time	
Amount, skill of software staff	
Study design	
System size	
Development process	
Correlation analysis	
Statistic descriptive	
Sensitivity analysis	
Data pre processing method	
Statistic test	Choose an item.

Metrics significant at 0.01	
Correlation coefficient	
Metrics significant at 0.05	
Correlation coefficient	
No significant metrics	
Univariate regression model?	Choose an item.
Metrics significant at 0.01	
Regression coefficient	
Metrics significant at 0.05	
Regression coefficient	
No significant	
Coefficient estimation method	
Factor identification method	
Factor identification method	
Number of dimension	
Dimension and category	
Prediction model	
Prediction method	Choose an item.
Metric suites used	
How to select metrics	
Other tests	
Most effective metric suite	
Validation method	
Log likelihood	
R2	
Accuracy measure	
Accuracy value	
Compounding factors	
Compounding factors mentioned	
Correlated evaluation + threshold	Choose an item.
Correlate to independent variable	
Contribute to prediction model?	
Extra finding	
Threat	