SIZE AND EFFORT ESTIMATION BASED ON CORRELATIONS BETWEEN PROBLEM AND SOLUTION DOMAIN MEASURES FOR OBJECT ORIENTED SOFTWARE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TÜLİN ERÇELEBİ AYYILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

AUGUST 2015

SIZE AND EFFORT ESTIMATION BASED ON CORRELATIONS BETWEEN PROBLEM AND SOLUTION DOMAIN MEASURES FOR OBJECT ORIENTED SOFTWARE

Submitted by **Tülin ERÇELEBİ AYYILDIZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Information Systems, Middle East Technical University** by,

Prof. Dr. Nazife BAYKAL _____
Director, **Informatics Institute**

Prof. Dr. Yasemin YARDIMCI ÇETİN _____
Head of Department, **Information Systems**

Assoc. Prof. Dr. Altan KOÇYİĞİT _____
Supervisor, **Information Systems, METU**

Examining Committee Members:

Prof. Dr. A. Ziya AKTAŞ _____
Computer Engineering Dept., Başkent University

Assoc. Prof. Dr. Altan KOÇYİĞİT _____
Information Systems Dept., METU

Assoc. Prof. Dr. Aysu BETİN CAN _____
Information Systems Dept., METU

Prof. Dr. Onur DEMİRÖRS _____
Information Systems Dept., METU

Prof. Dr. Ali YAZICI _____
Software Engineering Dept., Atılım University

Date: 24.08.2015

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Tülin, Erçelebi Ayyıldız

Signature : _____

ABSTRACT

SIZE AND EFFORT ESTIMATION BASED ON CORRELATIONS BETWEEN PROBLEM AND SOLUTION DOMAIN MEASURES FOR OBJECT ORIENTED SOFTWARE

Erçelebi Ayyıldız, Tülin

Ph. D., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Altan Koçyiğit

August 2015, 92 pages

Software size measurement and effort estimation methodologies in use today usually take the detailed requirements of software to be developed as the primary input and a certain amount of time and expertise is needed for size measurement. This thesis analyzes the correlations between the problem domain measures such as the number of distinct nouns and distinct verbs in the requirements artifacts and the solution domain measures such as the number of software classes and methods in the corresponding object oriented software to develop an early and cost-effective software size and effort estimation methodology. For this purpose, five case studies have been conducted. In the first case study, 37 open source software projects are analyzed and a strong correlation between the problem and solution domain measures is observed. In order to validate the proposed methodology, the second and third case studies are conducted on commercial software projects. Therefore, a methodology based on linear regression analysis is proposed to estimate the solution domain measures of object oriented software projects. Moreover, significant correlations are also observed between the problem domain measures, the Use Case Points (UCP) and the Common Software Measurement International Consortium (COSMIC) Function Point (CFP) size measures and the effort required to develop software. Again, the linear regression analysis is carried out for size and effort estimations and prediction performances are evaluated via the fourth and fifth case studies. The results show that the proposed methodology provides more accurate results compared to the UCP and CFP methodologies in effort estimations.

Keywords: Software Size Measurement, Software Effort Estimation, Problem Domain Measures, Solution Domain Measures, Linear Regression

ÖZ

NESNE TABANLI YAZILIMLAR İÇİN PROBLEM VE ÇÖZÜM ALANI ÖLÇÜLERİ ARASINDAKİ İLİŞKİYE DAYALI BÜYÜKLÜK VE EFOR TAHMİNİ

Erçelebi Ayyıldız, Tülin

Doktora, Bilişim Sistemleri

Tez Yöneticisi: Doç. Dr. Altan Koçyiğit

Ağustos 2015, 92 sayfa

Halen kullanılmakta olan yazılım büyüklüğü ölçümü ve efor kestirimleri genellikle geliştirilecek olan yazılımın detaylı gereksinimlerini temel girdi olarak kullanırlar ve büyüklük ölçümü için bir miktar zamana ve uzmanlığa ihtiyaç duyarlar. Bu tez, nesne yönelimli yazılımlarda, farklı isim ve fiil sayıları gibi problem alanı ölçüleri ile yazılım sınıfları ve metotları gibi çözüm alanı ölçüleri arasındaki ilintiyi, erken ve maliyet-etkin yazılım büyüklük ve efor kestirimi paradigması geliştirmek için analiz etmektedir. Bu amaçla, beş örnek olay incelemesi gerçekleştirilmiştir. İlk örnek olay incelemesinde 37 açık kaynak yazılım projesi değerlendirilmiş ve problem alanı ve çözüm alanı ölçüleri arasında yüksek korelasyon olduğu gözlemlenmiştir. Önerilen paradigmayı doğrulamak için, ticari yazılım projeleri üzerine ikinci ve üçüncü örnek olay incelemesi gerçekleştirilmiştir. Böylece, nesne tabanlı yazılımlar için doğrusal regresyon analizine dayalı çözüm alanı ölçülerini tahmin etmek için bir paradigma önerilmiştir. Üstelik, problem alanı ölçüleri, UCP ve CFP büyüklük ölçüleri ve yazılımı geliştirmek için gerekli olan efor arasında da önemli bir korelasyon gözlemlenmiştir. Yine büyüklük ve efor kestirimleri için doğrusal regresyon analizi gerçekleştirilmiş ve kestirim performansları dördüncü ve beşinci örnek olay incelemeleri aracılığıyla değerlendirilmiştir. Sonuçlar önerilen paradigmanın efor belirlemede UCP ve CFP paradigmalarına göre daha doğru sonuçlar verdiğini göstermektedir.

Anahtar Kelimeler: Yazılım Büyüklük Ölçümü, Yazılım Efor Kestirimi, Problem Alanı Ölçüleri , Çözüm Alanı Ölçüleri, Doğrusal Regresyon

dedicated to my lovely daughter TÜRKÜ

&

to my beloved Dr. Nizam AYYILDIZ

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Altan Koçyiğit for his great support, enlightening ideas, criticism and insight. He was always patient and kind to me and I learned a lot from him throughout the thesis study.

I express my sincere thanks to my committee member Prof. Dr. A. Ziya Aktaş for his guidance, support, encouragement, endless patience and stimulating suggestions during this study. He read my numerous revisions and gave his assistance. Thank you for your trust in me.

I would like to thank my committee member Prof. Dr. Onur Demirörs for his ideas and support throughout the thesis study.

I would also like to thank my examining committee members Prof. Dr. Ali Yazıcı and Assoc. Prof. Dr. Aysu Betin Can for their contribution and support.

I would like to thank Başkent University for letting me involve in this thesis study and for the facilities provided for the completion of this thesis. And I am grateful to Assoc. Prof. Dr. Hasan Oğul for his ideas and valuable support.

I would like to thank the organizations in which I used their data and their personnel for their contributions who preferred to stay anonymous.

I am grateful to my best friends Nihal Uğur, Didem Ölçer, Serian Doma, Duygu Dede Şener and Deniz Demirkol. Meeting you is one of the greatest fortunes in my life. We had wonderful times together that I will never forget. And I am also grateful to Dr. Mehmet Dikmen, Dr. Pelin Toktaş and Dr. Özden Özcan Top for their support.

A very special gratitude goes to my mother Ferah Erçelebi and my father Dr. Hasan Erçelebi for being there for me whenever I needed them and for their constant love. I learned a lot from them throughout my life.

The most special thanks go to my husband Dr. Nizam Ayyıldız. He has been one of the greatest supporters of my life. He was with me for the hardest times. Thank you for loving me endlessly and believing in me.

AND my 2.5 years old daughter Türkü, who changed my life entirely; this thesis study would not have been possible without her love. She made me live the miracle of motherhood. We have grown up together. Thank you for being with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AE	: Actual Effort
AF	: Adjustment Factor
AUCP	: Adjusted Use Case Points
CASE	: Computer Aided Software Engineering
CMMI	: Capability Maturity Model Integrated
CFP	: COSMIC Function Point
CP	: Class Point
COCOMO	: Constructive Cost Model
COSMIC	: Common Software International Consortium
EER	: Extended Entity-Relationship
EF	: Environmental Factor
EFPA	: Early Function Point Analysis
ERD	: Entity Relationship Diagram
EQFP	: Early & Quick Function Points
FP	: Function Point
FPA	: Function Point Analysis
FFP	: Full Function Point
FSM	: Functional Size Measurement
FUR	: Functional User Requirements
IEC	: International Electrotechnical Commission
IFPUG	: International Function Point Users Group
ISO	: The International Organization for Standardization
LOC	: Line of Code
LOOCV	: Leave One Out Cross Validation
MdMRE	: Median Magnitude of Relative Error
MIS	: Management Information System
MMRE	: Mean Magnitude of Relative Error
MRE	: Magnitude of Relative Error
MSE	: Mean Square Error
NESMA	: Netherlands Software Users Metrics Association
NLP	: Natural Language Processing
NL-OOPS	: Natural Language – Object-Oriented Production System
NLU	: Natural Language Understanding
NLTK	: Natural Language Toolkit
OO	: Object Oriented
OLS	: Ordinary Least Squares
OOFP	: Object Oriented Function Points
PPMC	: Pearson Product Moment Correlation
Pred	: Prediction Quality
PRICE-S	: Programmed Review of Information for Costing and Evaluation System
SLIM	: Software Life Cycle Management
SLOC	: Source Lines of Code
SPSS	: Statistical Package for the Social Sciences
SRS	: Software Requirements Specification
SUD	: System Under Development
RUP	: Rational Unified Process
TCF	: Technical Complexity Factors

UAW	: Unadjusted Actor Weight
UCP	: Use Case Points
UML	: Unified Modeling Language
UUCP	: Unadjusted Use Case Points
UUCW	: Unadjusted Use Case Weight
VB	: Verb, Base Form
VBD	: Verb, Past Tense
VBG	: Verb, Gerund
VBN	: Verb, Past Participle
VBP	: Verb, non-3rd person Singular Present
VBZ	: Verb, 3rd person Singular Present

CHAPTER 1

INTRODUCTION

1.1. General

Software size is used for several purposes, such as cost/effort estimation, scheduling, quality assessment, benchmarking, risk assessment, productivity measurement, performance management and outsourcing contracts. Therefore it is very important to quantify the software size in a short amount of time as early as possible and with a little effort to make critical management decisions timely and in a cost-effective manner.

There is a multitude of size estimation methodologies proposed in the literature e.g. (Živković et al., 2005), (Laird and Brennan, 2006), (Azzeh and Nassif, 2013), (Ren and Dai, 2013). A common property of most of these methodologies is that they use functional user requirements as the primary input. Usually the measurements are based on identification of the sub-processes and/or data movements in the software. Hence detailed software requirements are needed and accurate measurements require spending a certain amount of time and effort.

The object oriented analysis and design paradigm strives for similarity between the problem and the solution domains to create understandable, and hence extensible and maintainable software (Rumbaugh et al., 1990), (Booch, 1986), (Jacobson et al., 1999) and (Larman, 2002). For this purpose, the problem domain is used as a source of inspiration in object design to assign domain familiar names and responsibilities to software objects. Consequently, a correlation and causality can be expected between some attributes of the problem domain and some attributes of the software created.

Object oriented analysis largely utilizes the problem domain descriptions and the stakeholders' domain knowledge as the input. Linguistic analysis is one of the most widely used methods to identify noteworthy concepts and transactions in the problem domain descriptions. Typically, noteworthy concepts and transactions are used while naming software objects and defining methods. In order to minimize the time and effort spent to carry out linguistic analysis on problem domain descriptions, Natural Language Processing (NLP) tools can also be utilized.

Application of linguistic techniques to object oriented software development was first initiated by Abbott (Abbott, 1983) and it is called noun-verb analysis in the literature. Abbott suggested that nouns are good candidates for software classes and verbs are

good candidates for software methods. This methodology was further developed by Booch (1986). Booch described an object-oriented design methodology where verbs suggest software methods and nouns in the problem description suggest objects and classes of objects. Saeki et al. also stated that "Nouns are considered as classes and their corresponding verbs as methods" (Saeki et al., 1987).

Some researchers also make use of the similarity of problem and solution domain to facilitate software design activities. For instance, problem domain descriptions and requirements are used to form initial Unified Modeling Language (UML) class diagrams for the software being developed. Vidhu Bhala and Abirami (2014), proposed a mechanism for generating a conceptual model from functional specifications automatically. Denis et al. (2009) state that nouns in the use case scenarios suggest possible software classes whereas the verbs suggest possible software methods. Elbandak et al. (2011) have developed a tool that can identify candidate software classes from requirement specifications semi-automatically. They identify nouns and verbs in the use cases to form a preliminary UML class diagram in which all nouns and verbs are identified as software classes and their methods, respectively.

Since software classes and methods are the basic building blocks of object oriented software, the number of software classes and the number of methods in those classes can serve as very useful measures that influence the other software measures such as line of code (LOC) and the effort required to develop the software. Accordingly, there are some measures specifically proposed for object oriented software. The most commonly referenced object oriented measures are proposed by Chidamber and Kemerer (1994), Lorenz and Kidd (1994) and Li and Henry (1993). Although these measures are the widely used object oriented measures, they have some drawbacks and they are criticized by some researchers as given later in Section 2.6.

In object oriented software engineering, use cases are the principal tools to capture functional requirements. Hence, they can serve as an input for predicting the size of the software and hence the effort required to develop it at an early phase of software development life cycle. In this context, Use Case Point (UCP) methodology is proposed by Karner (1993) for estimating size and effort for object oriented projects using use cases. In order to measure the software size and estimate the required effort, unadjusted use case weight, unadjusted actor weight, technical complexity factors (TCF), environmental factors (EF) and productivity are taken into account. However, TCF and EF can be evaluated differently by different measurers. This makes UCP neither strictly repeatable nor reproducible. UCP is also stated as fundamentally structurally defective since it uses weights and constants without criteria or a guide to interpretation (Abran, 2010).

Costagliola, et al. (2000) also proposed a new concept of class points methodology which consists of three main steps for object oriented software. These steps are class identification and classification, complexity level evaluation of each class and lastly total unadjusted class point estimation. This methodology estimates the size of object oriented software according to design documentation.

So far many size estimation researches have been carried out specifically for object oriented software. Some of these are based on applying the existing traditional size measures to object oriented software whereas the others are new ones just designed for object oriented software. However it is difficult to find completely rational and satisfactory model in order to measure the size of object oriented software and predict

the effort. Hence, reliable, accurate, faster and cheaper software size and effort estimation methodologies are still needed.

1.2. Research Methodology and Case Study Design

A novel size/effort estimation methodology for object oriented software is proposed in this thesis. The proposed methodology basically exploits similarities between the problem domain and the solution domain for object oriented software. The number of distinct nouns and the number of distinct verbs in the problem domain descriptions such as feature lists, use cases and other requirements artifacts are defined as the problem domain measures. The number of classes and the number of methods in the resulting software are considered as the solution domain measures. In the rest of the thesis, the terms "Problem domain measures" and "the number of distinct nouns and the number of distinct verbs", and also "solution domain measures" and "the number of software classes and the number of software methods" will be used interchangeably.

The research strategy followed through this thesis study is given step by step in Figure 1. First of all, correlations between the problem domain and the solution domain measures are analyzed to get an insight about which problem domain measures are useful for predicting solution domain measures. Moreover, the applicability of regression models to relate problem domain measures to UCP and CFP size measures and effort required to develop software is investigated. The study is performed in the nature of the "quantitative research" defined by Creswell (2013), as we use the correlational statistic to describe and measure the relationship between two or more variables.

Mainly, the following research questions are answered by the help of five case studies:

RQ1: Are there any correlations between the problem domain measures and the solution domain measures for object oriented software?

RQ2: Can these correlations be utilized to estimate the software size and development effort?

In order to address the first research question, open source projects are analyzed in the first case study. Open source software projects have been used increasingly, since project artifacts, such as source codes, user manuals, revision control histories, and developer communications, are freely available to researchers. Therefore, 37 open-source object oriented software projects are selected in three different domains and each domain is evaluated individually. The reason behind domain based evaluation is that coding styles and user manual documentations have similarities in the same domain. Since project selection bias is also one of the internal validity threats that should be considered, while selecting projects we paid attention to the constraints which are listed in Section 4.1. The projects which are disproportionate to any those of constraints are not considered.

In order to automate identification of the problem and solution domain measures we used well-known tools. We have used a mature commercial tool, Understand 2.0, to identify and count software classes and software methods. We use NLTK to collect the number of distinct nouns and number of distinct verbs in user manuals and low level requirements. NLTK has also been used in other studies for natural language processing tasks (Bird et al., 2009), (Lobur et al., 2011) and (Vidhu Bhala et al., 2014).

Lobur et al. (2011) stated that NLTK is an acceptable tool and it is widely used all over the world for scientific research.

Based on the correlation analysis results, a methodology based on linear regression is proposed to predict the solution domain measures of object oriented software in terms of the measures collected by using the measurements made on problem domain descriptions.

In order to validate our proposed methodology, we prefer to use software projects coded and documented with common professional standards. Since reliability of data collection is another important validity threat that should be considered, two different CMMI Level-3 certified defense industry companies are selected for Case Study #2 (12 projects) and Case Study #3 (14 projects). The companies that employ a systematic data collection process implement the software in object oriented programming languages and use English language in problem domain descriptions.

The company also provided UCP and CFP measures and actual effort for the projects analyzed in Case Study #3. So, with in the light of the measures they provided we conducted Case Study #4 in order to see whether we can estimate UCP and CFP through our proposed methodology. Since both UCP and CFP measurements are done by the company professionals, UCP and CFP measurements' can be considered reliable.

The correlation between the problem domain measures and UCP and CFP size measures and effort values are also exploited to create a linear regression analysis based size and effort estimation methodologies. The effort estimation with proposed methodology is also compared to the effort estimations utilizing UCP and CFP size measurements.

In this thesis study all of the statistical analyses are done by Minitab statistical tool. The results are also validated by using SPSS (Statistical Package for the Social Sciences) statistical tool. We obtained the same results with both of these two tools. Therefore, all statistical analyses that we observed were reliable.

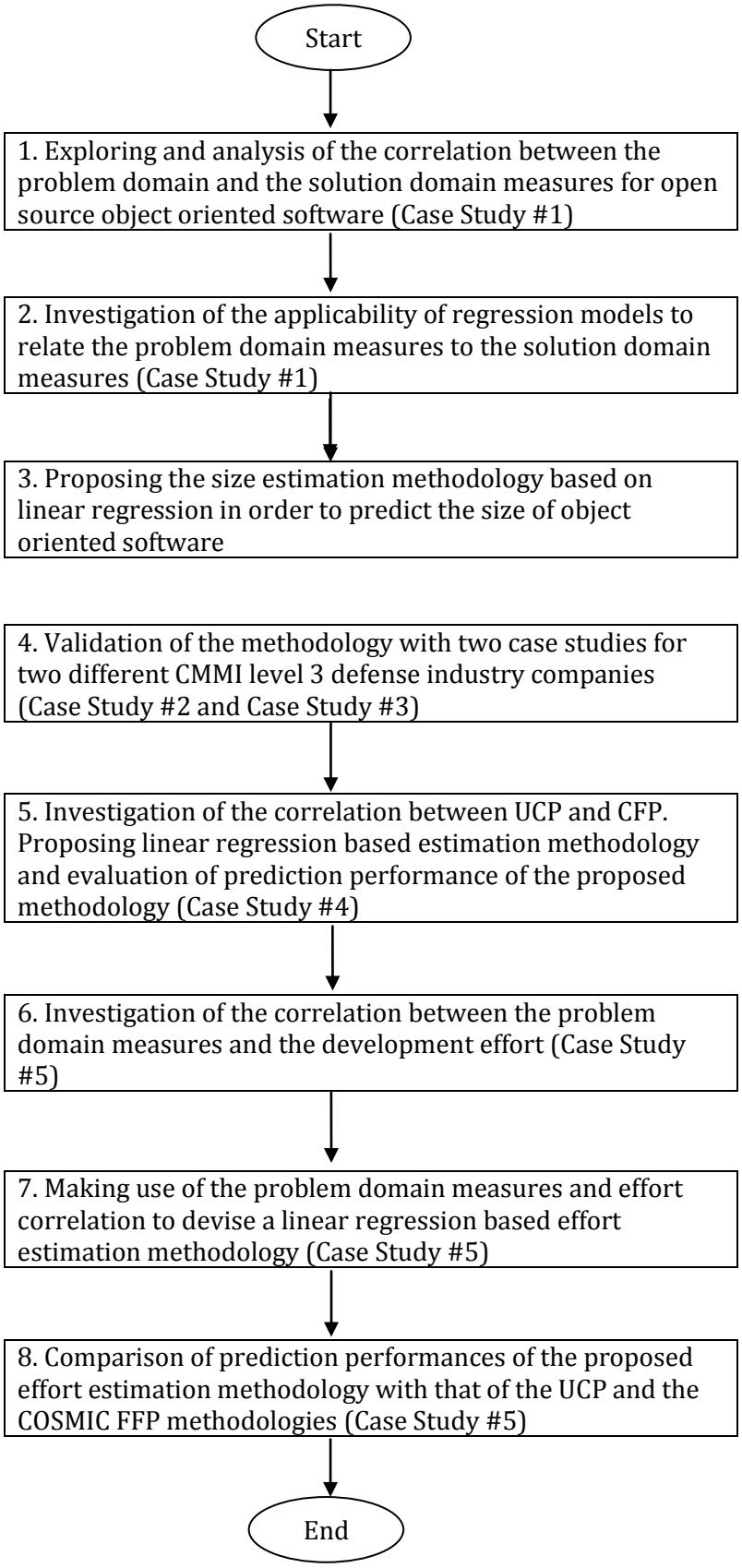

Figure 1: Research Steps

1.3. Organization of the Thesis

Following the Chapter 1, Introduction, the rest of this thesis study is organized as follows:

Chapter two presents a review of the software size and effort measurement and estimation methodologies.

Chapter three provides necessary background information and the descriptions of the techniques and notation used in the thesis study.

Chapter four focuses on the analysis of the correlation between the problem domain and the solution domain measures for object oriented open source software as the first group of case studies.

Chapter five proposes a solution domain measure estimation methodology and validates the methodology via other two groups of case studies, Case Study #2 and Case Study #3.

Chapter six proposes size prediction methodology using problem domain measures via Case Study #4.

Chapter seven proposes effort prediction methodology based on regression analysis via Case Study #5.

Chapter eight summarizes the overall findings, achievements, validity threats and possible directions for future work.

CHAPTER 2

RELATED RESEARCH

This chapter reviews the literature on software size and effort measurement and estimation methodologies. The problem and solution domain measures used for object oriented software are also presented and their weaknesses and strengths are discussed.

Section 2.1 presents brief history of existing Size Measurement methodologies. In Section 2.2 and Section 2.3, Use Case Point and COSMIC FFP methodologies are explained in detail. In Section 2.4, Size Estimation methodologies are described. In Section 2.5 frequently used effort estimation methodologies are described. In Section 2.6 Object Oriented Size Measures are discussed. Lastly, in Section 2.7 Mapping Problem Domain to Solution Domain issues are focused.

2.1. Software Size Measurement/Estimation Methodologies

Poor estimations are one of the main reasons for software failures (Tucker et al., 2002) and several attributes of a software project frequently is a function of the software size. Resource allocation, scheduling activities, quality and productivity management are performed based on the size of the software.

The Lines of Code (LOC) is the oldest and most widely used size measure that measures the size of software in terms of the lines of code in the source code. There are two LOC types: Physical LOC and Logical LOC. Physical LOC counts the text lines in the source code and Logical LOC counts the basic language constructs. Counting LOC is appropriate when the program is finished. Counting helps for measuring software size but when it is aimed to predict the effort, one cannot wait until the software is completed. Thus, estimating the LOC of the software is necessary before it is finished.

The idea of measuring the size of software in terms of its "functionality" is first proposed by Alan Albrecht in 1979. His methodology is known as Function Point Analysis (FPA) (Albrecht, 1979). This methodology has obtained a remarkable interest since it focuses on size measurement from user's viewpoint independent of the application itself.

Taking inspiration from Albrecht's methodology, several other measurement methodologies have been developed. Today, IFPUG FPA (ISO/IEC 20926: Software engineering – IFPUG 4.2 unadjusted functional size measurement method – counting practices manual, 2004), Mark II FPA (ISO/IEC 20968: Software engineering – Mk II function point analysis – counting practices manual, 2002), NESMA FPA (ISO/IEC 24570: Software engineering – NESMA functional size measurement methodology version 2.1 – definitions and counting guidelines for the application of function points analysis, 2005) and COSMIC Full Function Point (FFP) methodologies (The COSMIC

Functional Size Measurement Method Version 3.0.1 Measurement Manual, 2009) are well-known models that are accepted as international standards by ISO/IEC for functional size measurement. All these methodologies measure the functionality but they differ from each other with respect to the metrics and rules applied in measurement (Demirörs and Gencel, 2009).

In IFPUG FPA, constituent functions of a software application are divided into five categories for size measurement. First two categories, Internal Logical Files (ILF) and External Interface Files (EIF), are data function types. The rest of the three categories External Inputs (EI), External Outputs (EO) and External Inquiries (EQ) are transactional function types. After the elements have been tilled into these categories, complexity level of each function is determined according to the defined rules. Then, contribution of that element to the unadjusted function point count is determined by the weight assigned to the corresponding complexity level. The resulting unadjusted function point is adjusted by considering 14 general system characteristics. The degree of influence is assigned to each characteristic and the adjusted function point count is calculated using a specific formula (ISO/IEC 20926: Software engineering – IFPUG 4.2 unadjusted functional size measurement method – counting practices manual, 2004).

MARK II is a FSM methodology based on FPA like IFPUG, but its counting rules are different from the IFPUG. In MARK II, function points are calculated in several steps. The first step is to categorize the functional user requirements of the software into three types, which are inputs, exits and objects. Then each of these data types is counted.

NESMA FPA, a variant of IFPUG FPA, was proposed in 1990. NESMA FPA aims to simplify some of the IFPUG FPA sizing rules. Since NESMA's purpose was to use FPA for budgeting, they adapted several number of counting guidelines. This led to a several differences between IFPUG FPA and NESMA FPA in the early days. After 1994, except a few minor differences, the counting guidelines between two have been getting very similar.

Independence from the programming language used and coding styles of the developers is the main advantage of FPA methodologies and such methodologies are much more appropriate for early size measurement. However, many of these methodologies are mainly applicable for information system development projects and estimation of effort according to these measures doesn't usually consider the software development methodology used (Özkan et al., 2008). Moreover, measurements often take long times and effort or require more information regarding the software than that is available when the effort estimations are done. To overcome these problems, some simplifications have been proposed (Lavazza and Liu, 2012).

In order to simplify the FP counting process, the Early & Quick Function Points (EQFP) methodology is proposed in 1997. This methodology was originally proposed for IFPUG FPA to reduce the time and effort needed for measurement and to use non-detailed information about the project; however, the result is a less precise (Santillo et al., 2005), (Early & Quick, 2012) and (Lavazza and Lui, 2012). For the same context, Early Function Point Analysis (EFPA) technique was developed by the same research group (Meli, 1997a), (Meli, 1997b).

Antoniol et al. (1999) had proposed Object Oriented Function Points (OOFP) that utilizes design phase artifacts of a software development project. In this methodology,

class diagrams are used as the input and the number of associations and attributes of the classes are used in order to identify internal logical file complexity. Transactional functions are defined in terms of the software methods.

Object Points is another functionality-related measure which is alternative to function points. Object Points counts the reports, third generation programming language modules and screens developed in the application. Each count is weighted as simple, medium and difficult complexity factor (Banker et al., 1994) (Boehm et al., 2000).

Feature Points were used to identify main features of the software. It extends the Function Point methodology to add algorithms as a new class. An algorithm is described as the set of rules that must be fully stated to solve a major computational problem (Jones, 1987). Since Feature Points methodology is a variant of Function Point methodology and some basic requirements are needed for applying feature points methodology, it is not the earliest way of predicting the software size.

Class Point (CP) (Costagliola et al., 2000) another methodology that measures the size of object oriented software. The methodology consists of three main steps: identification and classification of software classes, evaluation of each class's complexity level and computation of the total unadjusted class point. The main drawback of this methodology is that it requires too much effort and knowledge to predict software size.

The effort prediction methodology proposed in this thesis is compared to the UCP and COSMIC FFP based effort prediction methodologies in Chapter 7. For this reason, UCP and COSMIC FFP methodologies are explained in detail in the following sections.

2.2. Use Case Points (UCP) Methodology

The methodology was proposed by G. Karner in 1993 (Karner, 1993) for estimating effort based on Use Cases. UCP methodology measures the functional size of a software system for which use cases are used to capture requirements (Abran, et al., 2009).

The methodology assigns weighting factors to actors according to actor classification as simple, average and complex. Actor types and their weighting factors are given in Table 1. Unadjusted Actor Weight (UAW) is calculated as the sum of all the weights assigned to the actors of the system.

Table 1: UCP Actor Types and Complexity Weight (Karner, 1993)

Actor Type	Weighting Factor
Simple	1
Average	2
Complex	3

Similarly, use cases are classified according to their complexity and they are assigned to weighting factors of 1, 2, and 3. Use case types and their complexity weights are given in Table 2. Unadjusted Use Case Weight (UUCW) is the sum of all the weights assigned to use cases of the system.

Table 2: UCP Use Case Types and Complexity Weight (Karner, 1993)

Use Case Type	Number of Transactions	Weighting Factor
Simple	<=3	1
Average	4 to 7	2
Complex	>=7	3

The sum of the UAW and UUCW gives the Unadjusted Use Case Points (UUCP) as in Equation 1.

$$UUCP = UAW + UUCW \quad \text{(Equation 1)}$$

In order to incorporate technical properties of the project, 13 Technical Complexity Factors given in Table 3 are considered. Then, Technical Complexity Factor (TCF) is computed as a function of TFactor (Equation 2), which is the weighted sum of value of the convenience assigned to each complexity factor. Each property is evaluated on a scale from 0 to 5 (where 0 means 'not applicable' and 5 means 'essential').

$$TCF=0.6 + (0.01*TFactor) \quad \text{(Equation 2)}$$

Table 3: Technical Complexity Factors (Karner, 1993)

Technical Factor	Weight
Distributed System	2
Response Objective	1
End User Efficiency	1
Complex Processing	1
Reusable Code	1
Easy to Install	0.5
Easy to Use	0.5
Portable	2
Easy to Change	1
Concurrent	1
Security Features	1
Access for Third Parties	1
Special Training Required	1

Project and team related features are taken into account by considering environment factors which are also referred to as development resources (Caroll, 2005) and measures the development team's effectiveness. The UCP methodology defines eight such factors given in Table 4. Environmental Factor (EF) is a function of EFactor (Equation 3), which is equal to the weighted sum of level of importance assigned to each factor in the range 0 to 5 (0 for "very weak", 5 for "very strong") (Ouwerkek and Abran, 2006).

$$EF=1.4 + (-0.03*EFactor) \quad \text{(Equation 3)}$$

Table 4: Environmental Factors (Karner, 1993)

Environmental Factor	Weight
Familiar with RUP	1.5
Application Experience	0.5
Object Oriented Experience	1
Lead Analyst Capability	0.5
Motivation	1
Stable Requirements	2
Part Time Workers	-1
Difficult Programming Language	-1

Finally, the Adjusted Use Case Points (AUCP), which is the size of the project, is computed as given in Equations 2 through 4.

$$AUCP = UUCP * TCF * EF \quad \text{(Equation 4)}$$

Several approaches can be used to convert the size obtained from the use case point evaluation to the required effort. For example; Karner's methodology assumes the productivity of 20 person hours per AUCP. In a study performed by K. Ribu in 2001 (Ribu, 2001), it is stated that each AUCP may require between 15 and 30 person hours. According to Schneider and Winter's study in 1998 (Schneider and Winter, 1998) the environmental factors should also be taken into account as follows;

- If EF <= 2 then 1 AUCP takes 20 person hours,
- If EF =3 or EF=4 then each AUCP takes 28 person hours,
- If EF>4 then this means there are many environmental factors and the project should be postponed until the EFs are rearranged.

According to M. Cohn (2005) the best solution is to calculate the organization's own historical records' with regards to the projects realized in the past of that organization. For example; if an organization realized 5 projects in the past and these projects took 44.000 person hours for a total of 2000 AUCP's, the average of this organization is 22 person hours per AUCP.

There are also some approaches to simplify the UCP methodology (Mohagheghi et al., 2005), (Ochodek et al., 2011) and (Ayyıldız et al., 2012). All these authors claimed that original UCP methodology has some drawbacks about TCF and EF. They state that these factors lack standardization and they are subjective.

2.3. COSMIC Full Function Point Methodology

COSMIC FFP was proposed by the Common Software Measurement International Consortium led by Abran and Symons in the late 1990s, to overcome some limitations of traditional function point analysis methodologies such as IFPUG and Mark II (Abran, et al., 2001). Now, it is one of the well-known models accepted as the international standards for functional size measurement by ISO/IEC (The COSMIC Functional SizeMeasurement Method Version 3.0.1 Measurement Manual, 2009) and it is

applicable for both real time software and Management Information System (MIS) development projects.

COSMIC FFP is independent of any development lifecycle model and it can be applied at any phase of the software development project. Indeed it can be derived without reference to methods used and physical or technical components.

COSMIC FFP methodology is composed of three main parts: measurement strategy, mapping and measurement. In the measurement phase, four types of data movements are identified which are Entry (E), Exit (X), Read (R) and Write (W) (Abran et al., 2001) and given in Figure 2.

- **Entry** moves data from the user to the functional process;
- **Exit** moves data from a functional process to the user;
- **Read** moves data from the persistent storage to a functional process;
- **Write** moves data from a functional process to the persistent storage.

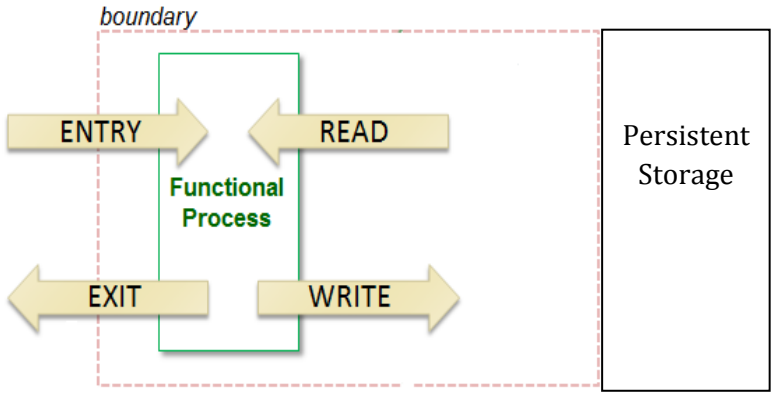

Figure 2: Identifying Data Movement Types

The COSMIC FFP methodology defines a standardized measure of software functional size expressed in the COSMIC Function Points (CFP) unit.

Finally the total size of the software is calculated as:

$$\begin{aligned} \text{SizeCFP (functional process}_i) &= \Sigma \text{ size(Entries}_i) \\ &+ \Sigma \text{ size(Exits}_i) \\ &+ \Sigma \text{ size(Reads}_i) \\ &+ \Sigma \text{ size(Writes}_i) \end{aligned} \qquad \text{(Equation 5)}$$

2.4. Size Estimation Methodologies

Apart from above size measurement methodologies, there are some approaches that predict the software size in terms of the solution domain measures. The earliest study is the work of Misic and Tesic (1998). They used Ordinary Least Squares (OLS) regression to predict the Source Lines of Code (SLOC) in terms of the number of software classes and the number of software methods from the class model. They concluded that the final source code size in SLOC could be estimated from its class model which is constructed at the design phase. That is, the total number of software

classes and the total number of software methods, both of which are known at the end of the design phase, correlate well with the software effort.

A similar study was conducted by Del Bianco and Lavazza (2005). They stated that the number of software classes had a moderate correlation to the final code size and hence it could be a useful size predictor.

Ronchetti et al. (2006) also conducted a study to analyze two software packages which were developed by a CMMI level 3 software company. In both cases, the number of software methods well correlated with the size of the resulting system. They stated that more than 59% of the code size was explained and they found that correlation is statistically significant at the 0.05 significance level (p-value).

Živković, Rozman and Herićko proposed the unified mapping of UML models into function points. The mapping procedure is defined in order to automate the counting steps (Živković et al., 2005). Their methodology is called OOFP2.

A similar study was conducted Herićko and Živković (2008). They addressed the problem of size estimation in iterative development. They proposed a methodology that enables early size estimation using UML artifacts. They upgraded OOFP2 methodology but the proposal was not validated on industrial projects.

Zhou et al., (2014) investigated the accuracy of early SLOC estimation approaches using the UML class diagram. They concluded that class diagram measures (the total number of software classes, total number of attributes and total number of software methods) can be used to predict SLOC of object oriented systems. Their analyses are based only on Java systems. They didn't validate their findings with other object oriented programming languages.

Hussain et al., (2013) approximate COSMIC functional size measurement from informally written textual requirements by using a supervised text mining approach and they demonstrate its applicability in widely used agile processes. Such requirements are expressed as user stories. Their aim is to develop an automatic tool that performs a quicker approximation of COSMIC functional size measurement without requiring the formalization of the requirements. In fact, their approach extends the idea presented in the Early&Quick methodology. Since they intend to estimate the development effort from requirements documents, they first need to use NLP techniques to extract the functional size of the software. They devise a solution for estimating the effort using the functional size as the primary variable and different cost drivers as other variables in a machine learning environment to perform various regression analyses. Hence, their analyses are based only on estimating CFP size measurement. They didn't compare their findings with other CFP measurements in the literature. On the other hand our proposed methodology analyzes the correlations between the problem domain measures such as the number of distinct nouns and distinct verbs in the requirements artifacts and the solution domain measures such as the number of software classes and methods in the corresponding object oriented software to develop an early and cost-effective software size and effort estimation methodology. In our proposed methodology, we also use NLP techniques in order to extract the problem domain measures. Instead of user stories, the proposed methodology is applied on problem domain descriptions like low level requirements and use cases. However, the methodology is conceptually applicable to any other requirements artifacts or pre-requirements level artifacts. Apart from the Hussain et al.,

(2013) study, our proposed methodology is specifically proposed for object oriented software. Since the counting processes are automated in our proposed methodology, time and effort needed for estimation is reduced considerably. Moreover, we are able to estimate the UCP and CFP size measures and as the results indicate, we can predict UCP and CFP size measurements earlier with using problem domain measures.

Ungan (2013) investigate the problem and the solution domains for a software size measurement methodology. In the problem domain, he measured COSMIC FFP from the functional requirements. In the solution domain, he measured number of classes, number of operations, number of operation parameters, number of class attributes, number of inter class connections and LOC.

2.5. Effort Estimation Methodologies

Since accurate effort estimation is one of the most significant issues in software management; various effort estimation methodologies have been developed. Effort estimation methodologies can be classified considering various aspects. Boehm classified the effort estimation methods into seven categories which are namely: Algorithmic Models, Expert Judgment, Analogy, Parkinson, Price-to-Win, Top- Down, and Bottom-Up (Boehm, 1981). In his classification, "expert estimation and bottom-up approach" is taken into account as a different approach. However, since analogy techniques work by comparing the current projects with previous ones; expert estimation and bottom-up approach can be considered in the scope of analogy based effort estimation techniques (Jørgensen et al., 2003). Top down effort estimation approaches are suitable in the early phases of the software life cycle and bottom up estimation approaches are suitable when each software component is known in detail.

2.5.1. Expert Judgment

Wideband Delphi methodology can be considered in the scope of Expert Judgment approach. It is a consensus based effort estimation technique and effort is predicted based on the judgments of one or more expert(s) (Anderson et al., 1999). This approach is suitable when the consultants are familiar with the projects to be developed. Although it uses expertise of various consultants, the methodology may fail to reach a consensus, and judgment errors might occur.

2.5.2. Top-Down Effort Estimation

These methodologies are suitable at the early phases of the software life cycle (Anderson, et al., 1999). Based on the historical information in the organization, and comparing the project with previous similar ones, overall effort for the project is estimated. (Jørgensen, 2004). Later, the effort is distributed over the lower level components considering life-cycle phases. Although top-down approach is easy and fast to implement, it is less accurate when compared to bottom-up approach (Anderson, et al., 1999).

Curve Fitting Estimation Models such as Constructive Cost Model (COCOMO), Software Life Cycle Management (SLIM) and Programmed Review of Information for Costing and Evaluation System (PRICE-S), which are based on statistics and curve fitting, can be considered in the scope of the top-down approach.

COCOMO is one of the most widely used effort estimation models. There are three different levels of COCOMO which are Basic, Intermediate and Detailed. The effort is calculated according to three different difficulty modes of the projects, with Basic COCOMO. Organic mode is used to calculate effort for small size projects. The development team is familiar with application and language and constraints are not strict. Semi-Detached mode is used to calculate effort for the projects in which the constraints are greater than the organic mode. For Semi-Detached mode, the team is not very familiar with the application to be developed. Embedded mode is used to calculate effort for relatively large scale projects in which the constraints are strict. Based on these difficulty modes above, the following formula is used with three different variables given in Table 5:

$$Effort = a*Size^{b} \quad \text{(Equation 6)}$$

Table 5: Variables of Basic and Intermediate COCOMO Formulas

	Basic		Intermediate	
Mode	**a**	**b**	**a**	**b**
Organic	2.4	1.05	3.2	1.05
Semi-Detached	3.0	1.12	3.0	1.12
Embedded	3.6	1.20	2.8	1.20

2.5.3. Bottom-Up Effort Estimation

To be able to use bottom-up estimation, each task in the work breakdown structure of the project should be well known, and historical data that involves productivity should be reliable. Since detailed information about the requirements and tasks are required to use this methodology, it is not suitable in the early phases. When the detail level of the requirements is suitable to use the methodology, the size of each task or component is estimated, and the required effort is predicted using historical productivity of the organization or the team (Demirors and Gencel, 2009). The methodology is sufficiently reliable when the productivity of the team is consistent; however, it requires too much time to calculate (Anderson, et al., 1999) Therefore, it can be perceived as a time consuming process.

2.6. Object Oriented Size Measures

There are two opinions for the measurement of object oriented software. Some researchers claim that traditional measures are not suitable for object oriented software and new ones are needed (Bieman,1996). Others believe that traditional measures can be applied to object oriented software, may be with some modifications and additions (Shepperd and Cartwright, 1997), (Tegarden et al., 1992).

Hence, a significant number of object oriented measures have been proposed in the literature. The most commonly used measures are defined by Chidamber and Kemerer (CK) (1994), Lorenz and Kidd (1994), Li and Henry (1993).

Chidamber and Kemerer (1994)(CK) measures reflect the overall quality of object oriented software and CK measures are available at the class level (Sharma et al., 2012a). Class based three measures used for size measurement are:

- Weighted Methods per Class (WMC),
- Depth of Inheritance Tree of a class (DIT) and
- Number of Children of a class (NOC).

WMC is an average number of methods per class and each method has a complexity weight based on the method type used. Both the number and complexity of methods are indicators of how much time and effort is required for developing and maintaining the class.

The DIT is the maximum length from a node to the root of the tree where multiple inheritances involved (Chidamber and Kemerer, 1994). DIT measures reusability and maintainability. A class with a small DIT value is more likely to be reusable (Sharma et al., 2012b).

NOC measure is defined as the number of children of a class (Chidamber and Kemerer, 1994). A class which has many children is considered as a poorly designed class (Chatzigeorgiou, 2003). Lower value of NOC helps in complexity and maintainability.

Lorenz and Kidd (1994) measures were divided into three categories which are class size, class inheritance and class internal. In class size category, the number of the attributes and the number of the methods are the basic focus. Since many other measures were defined by Lorenz and Kidd, their six popular size measures are:

- Number of Methods (NM)
- Number of Public Methods (NPM)
- Number of Public Variables (NPV)
- Number of Variables (NV)
- Number of Class Methods (NCM)
- Number of Class Variables (NCV)

NM is the total number of the all public, private and protected methods in a class. NPM basically counts the number of public methods in a class. NPV measure is used to count the number of public variables in a class. Hereof Lorenz and Kidd stated that if the NPV is larger for one class, the class has more relationships with other objects. NV measure counts the total number of public, private and protected attributes in a class. They also stated that ratio of the total number of variables to private and protected variables points the effort required by that class. Moreover they also stated that the number of methods in the class (NCM) and the number of attributes in the class (NCV) reflect the size of a class (Lorenz and Kidd, 1994).

Li and Henry (1993) also proposed object oriented measures to measure size of the software. Their three size measures are:

- Number of Local Methods (NLM),
- Number of Ancestor Classes (NAC),
- Number of Descendent Classes (NDC)

NLM is defined as the number of the local methods which are defined in a class and accessible outside the class. NAC is similar to DIT as measures the number of ancestor of a class (Kandpal and Kandpal, 2012). NDC measure is defined as the total number of descendent classes (subclass) of a class. It is an alternative measure to NOC. Li and Henry (1993) stated that the NDC measure captures the attribute of classes better than NOC.

Although these measures are widely used object oriented measures, they have some drawbacks. For example; CK measures are just available at the post-design and the implementation phases of the software development life cycle. For instance, they can be applied on the source code (during the implementation phase) and on the class diagram (after the design phase) (Herr and Cunningham, 1999).

The Lorenz and Kidd measures are also criticized by the researchers (Harrison et al., 1997) for merely being counts of class properties. They stated that, quality factors are not evaluated by counting the number of public methods and variables in different ways (Baroni and Abreu, 2003).

All of the measures mentioned above are obtained at the end of the coding. Therefore, they are not available at an early phase of software development life cycle.

2.7. Mapping Problem Domain to Solution Domain

Application of linguistic techniques in object oriented software development was initiated by Abbott (1983) and it is known as noun-verb analysis in the literature. It is suggested that nouns are good candidates for classes and verbs are good candidates for operations/methods. Therefore, a textual analysis technique is proposed to analyze software requirements to obtain basic operations and data types (Abbott, 1983).

Booch (1986) further developed this approach and he described an object-oriented design methodology. Booch stated that "nouns in the problem description represent objects and classes of objects and verbs represent operations". Both Abbott (1983) and Booch (1986) have not produced practical working systems that reflect their findings.

In those years, Chen (1983) proposed basic rules for translation of English sentences to an Entity Relationship Diagram (ERD). Chen stated that "a common noun corresponds to an entity type in an ERD". Moreover, he claimed that "a transitive verb corresponds to a relationship type in ERD".

Saeki et al (1987) also stated that "Nouns are considered as classes and their corresponding verbs as methods". They tried to achieve formal specifications from the informal textual requirements. Nouns and verbs are identified from the informal requirements automatically. However, their system cannot identify which words are necessary for the construction of the formal specifications. Therefore, after each sentence is processed somebody is needed to analyze the system results manually.

Meziane and Vadera (2004) produced a workable system that generates ERD. But it needs user intervention. For example; accepting or rejecting noun phrases which are represented in the final model can be done sentence by sentence.

Gomez et al. (1999) also produced a rule based ER generator system which creates ER models from natural language specifications. They used specific and generic rules to

link the semantics of some words in the sentences and to identify entities and relationships. Natural Language Understanding (NLU) system uses a semantic interpretation approach and constitutes knowledge representation structures.

Mich (1996) and Mich and Garigliano (2002) described an NL-based system that is called NL-OOPS (Natural Language – Object-Oriented Production System). The purpose of the system is using NL specifications to generate object-oriented analysis models. NL-OOPS system expressed how a large scale Natural Language Processing system (which is called LOLITA) can be used to support the object oriented analysis stage.

Perez-Gonzalez and Kalita (2002) have proposed a semi natural language tool (4WL) to automatically generate object models from natural language text. Their tool (which is called GOOAL) exhibit object oriented static and dynamic model views of the problem.

CM-Builder, which is one of the Natural Language Processing based tool, by Harmain and Gaizauskas (2003) has used robust Natural Language Processing techniques to analyze requirements texts which are written in English. It constructed (either automatically or interactively with an analyst) an initial UML Class Model which represents the object classes and the relationships among them. The initial class model can be directly input to a graphical CASE (Computer Aided Software Engineering) tool by a human analyst for further refinement.

Zhou and Zhou (2004) had presented another conceptual modeling system based on linguistic patterns. Their framework generates class diagrams from unstructured system requirement documents. Their proposed conceptual modeling was not automated since they assume that system analysts take many decisions during the object oriented analysis and modeling stage.

Al-Safadi (2009) proposes a semi-automated methodology for designing databases in detailed ERD notation. This methodology used textual documents in order to generate semi-automated conceptual data model.

Bajwa et al. (2009) had proposed a NLP based automated system for converting natural language descriptions to object oriented models. Their system used the user requirements and generated code in multi-languages. In order to identify classes, objects, attributes, methods and associations the natural language text was semantically analyzed. Then, UML diagrams were generated according to formerly extracted information. Nevertheless, system details are not given and the system's utility cannot be determined.

Elbandak et al., (2011) have developed a tool that can identify candidate software classes from requirement specifications semi-automatically. They identified nouns and verbs in the use cases to form a preliminary UML class diagram in which all nouns and verbs are identified as software classes and their methods, respectively.

Vidhu Bhala and Abirami, (2014) proposed a mechanism for generating a conceptual model from functional specifications automatically. From the functional specifications, relationships and classes are automatically identified. This identification is based on the grammatical structures of sentences. The proposed mechanism integrates Extended Entity Relationship (EER) notations into the class relationships.

Tripathy et al., (2014) proposed an approach to automatic construction of UML diagrams from a parsed text of requirements.

Thakur and Gupta (2014) proposed a tool which generated the sequence diagrams from use case specifications automatically. In order to identify problem level objects and interactions between them, the methodology used natural language parser.

Abirami et al., (2015) presented a framework which identified the functional and non functional requirements from the requirements document automatically. Then, they transformed these requirements to the conceptual model.

CHAPTER 3

BACKGROUND

In this thesis, problem domain measures are related to different software size measures and development effort. Problem domain and solution domain terms are defined in Section 3.1. The problem domain measures and collection methods are given in Section 3.2. Section 3.3 presents the solution domain measures and their collection methods. In Section 3.4, correlation and regression analyses are explained in detail. In Section 3.5 outlier analysis is given and Section 3.6 presents the approaches to assess the accuracy of estimates.

3.1. Problem and Solution Domains

Problem domain is an engineering term, which involves the real life needs and problem descriptions. It represents the environment in which a solution will have to operate, as well as the problem itself. Understanding the boundaries and characteristics of a problem, requirements identification and requirement elicitation lie in the problem domain analysis. The problem domain descriptions include the user requirements, user stories, use cases, process models, laws, regulations, and so on.

Solution domain is composed of the developed software, its architecture and the execution environment. It may also contain the activities performed for building a system can also be considered as a part of the solution domain.

3.2. Problem Domain Measures

In the object oriented analysis and design paradigm, nouns and verbs in the documents that describe the problem suggest the names of software classes and names of the software methods. In this manner, the gap between the problem domain descriptions and solution domain descriptions is lowered. Therefore understandable, and hence maintainable software could be created (Rumbaugh et al., 1990), (Booch, 1993), (Jacobson et al., 1999) and (Larman, 2002). Problem domain analysis can be carried out by using problem domain descriptions and stakeholders' domain knowledge. Linguistic analysis is one of the most widely used methods to identify noteworthy concepts and transactions in the problem domain.

In this thesis, the number of distinct nouns and the number of distinct verbs in software descriptions such as feature lists, use cases, requirements, problem descriptions and other requirements artifacts constitute the problem domain measures. In order to measure problem domain measures the following five facts can be applied:

Fact 1: All improper nouns are candidate classes (Abbott, 1983), (Saeki et al., 1987), (Booch, 1993) and (Elbandak et al., 2011).

Fact 2: All verbs are candidate methods (Abbott, 1983), (Saeki, et al., 1987), (Booch, 1993) and (Elbandak et al., 2011).

Fact 3: Part of Speech tags (Table 7) are used for identifying nouns and verbs (Elbandak et al., 2011).

Fact 4: Nouns are always converted to their singular form (Elbandak et al., 2011).

Fact 5: Duplicate nouns and verbs are eliminated (Elbandak et al., 2011).

A tool can be used to minimize the time and effort spent for linguistic analysis and counting nouns and verbs in problem domain descriptions. There are several such tools available. Table 6 summarizes the strengths and weaknesses of some of these natural language processing tools (Giganto et al., 2008). In this thesis, Natural Language Toolkit (NLTK) is used for noun and verb identification. NLTK is a suite of programs and libraries for symbolic and statistical natural language processing for the Python programming language (Loper and Bird, 2002). It has been widely used to teach natural language processing to linguistics or computer science students (Bird et al., 2008).

Table 6: Natural Language Processing based Tools

Tool	Strength	Weakness
(Saeki et al., 1987)	Identifies nouns and verbs	Needs user intervention to refine results
NL-OOPS (Mitch and Garigliano, 2002)	Identifies attributes and classes	Unwanted classes, high user intervention
CM-Builder (Harmain and Gaizauskas, 2003)	Identifies attributes and classes	Produces synonymous classes
GOOAL (Perez-Gonzalez and Kalita, 2002)	Identifies classes and attributes, can generate sequence diagram	Needs user intervention to resolve ambiguity
(Li et al., 2005)	Identifies classes	Needs user intervention to resolve ambiguity
(Bryant, 2000) and (Lee, 2002)	Identifies classes	Needs user intervention to resolve ambiguity

We have developed a program (given in Appendix A) in Python programming language to use NLTK for problem domain measure collection. This program takes textual documents in .txt format as the input. Words in the input documents are classified into nouns and verbs by using the part of speech tags given in Table 7.

Table 7: Used Part of Speech Tag Prefixes in NLTK

Prefix	Actual Types	Examples
NN	Noun, singular	school
NNS	Noun, plural	schools
VB	Verb, base form	eat
VBD	Verb, past tense	ate
VBG[1]	Verb, gerund	eating
VBN	Verb, past participle	eaten
VBP	Verb, non-3rd person singular	eat
VBZ	Verb, 3rd person singular	eats

After identifying nouns and verbs by using NLTK automatically, the nouns and verbs are stemmed. Stemming is the process for converting derived words to their base form. For instance, plural terms are replaced by their singular counterparts, and words like eating, and eats are stemmed to eat. For stemming English words with NLTK, NLTK's WordNet Lemmatizer module is used. With the help of this module the following steps are done automatically:

- plural terms are made singular;
- duplicate terms are eliminated;
- synonyms are consolidated into a single term;
- nouns and verbs are listed in alphabetic order;
- irrelevant words (no meaning words, not noun/not verb) are removed.

Last step is done by NTLK's WordNet dictionary which is a large lexical database of English. If identified nouns and verbs are not found in the dictionary then they are removed from the list automatically.

Throughout this thesis study, the following versions of the libraries are used:

- NLTK version 3.0.1
- Python programming language version 3.4.1
- Wordnet lemmatizer module version 3.0

Most of the NLTK based methodologies given in Related Research Chapter (Chapter 2) need user intervention. For example; accepting or rejecting noun phrases which are represented in the final model are processed sentence by sentence. However, proposed methodology does not need user intervention for identifying problem domain measures. It is done automatically by the developed program.

[1] NLTK can not differentiate the gerund forms of nouns and gerund forms of verbs. Therefore, VBG is not considered in this study.

3.3. Solution Domain Measures

Since software classes are the basic building blocks of object oriented software, the number of software classes and the number of software methods in software are expected to influence the other software size measures such as LOC and the effort required to develop the software. Therefore, in this thesis, we consider the software classes and software methods in the source code as the solution domain measures.

In order to count software classes and software methods automatically, we have used Understand version 2.0, which is a commercial static code analysis software tool. Understand 2.0 is widely used to perform automatic documentation, reverse engineering and code metrics calculations for projects with large code-bases (Understand, 2008).

3.4. Correlation and Regression Analysis

The correlation between two random variables is a measure of how well the random variables are related. In statistics, one of the most commonly used measures of correlation is the Pearson's Correlation Coefficient (a.k.a. the Pearson Product Moment Correlation - PPMC). It shows the strength of linear relationship between two random variables. Pearson's correlation coefficient is a value between -1 and 1:

- 1 means that there is a perfect positive correlation,
- -1 means that there is a perfect negative correlation,
- 0 means that there is no linear relationship.

Correlation values between +0.5 and +1.0 have been accepted as high correlation (DeSanto et al., 2010).

It is possible to obtain a high correlation which is insignificant. Therefore, it is crucial to look at significance level together with correlation (Ahmed et al., 2013). In order to measure the significance of the analyses, for each correlation value the corresponding p-value is also calculated. P-value corresponds to the probability of finding a correlation by chance. The significance level (denoted as α) of 0.05 is traditionally considered acceptable for tests (Jain, 1991).

The null hypothesis (H_0) for this test is that the all of the slopes of the regression line (β_1, β_2 …) are equal to "0". The alternative hypothesis (H_a) is that none of the slopes are equal to "0".

- H_0: $\beta_1=0$, $\beta_2=0$, ... $\beta_k=0$
- H_a: $\beta_1\neq0$, $\beta_2\neq0$, ……$\beta_k\neq0$

When the probability associated with the criterion is smaller than a given α-level, the alternative hypothesis is accepted.

For all the correlation coefficients, the p-values less than significance level $\alpha=0.05$ is a strong evidence of significance (Brook, 2010), and hence these results can be

considered statistically significant and one can be confident that the relationship between variables is not due to chance.

Linear Regression is a classical statistical technique used to explain or predict the behavior of a dependent variable (DeSanto et al., 2010).

Generally, a linear regression equation takes the form of;

$$y = \beta_0 + \beta_1 x \qquad \text{(Equation 7)}$$

where "y" is the dependent variable to be predicted, "x" is the independent variable used to predict "y", "β_0" is the y-intercept of the line and "β_1" is the slope of the line (amount of increase or decrease in the mean of "y" for every 1-unit increase in "x").

If there are multiple independent variables a multiple linear regression equation this time becomes;

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_k x_k \qquad \text{(Equation 8)}$$

A strong correlation between two or more random variables can be taken as an indication of applicability of regression techniques to predict one of the variables in terms of the other variables. Nevertheless, the applicability of linear regression models for the purpose of the prediction should also justified. For this purpose, the following can be used:

a) Normality Analysis

b) Scatterplots

c) Residual Plots

In statistics, normality tests are used in order to determine if a data set is well-modeled by a normal distribution. The normality test is one of the widely used tools for judging normality, especially for small sample sizes.

In our study, normality is evaluated using Ryan-Joiner test, as implemented in Minitab tool (Brook, 2010). The null hypothesis (H_0) for this test is that the error is normally distributed. The alternative hypothesis (H_a) is that the error is not normally distributed. When the probability associated with the criterion is smaller than a given α-level, the alternative hypothesis (H_a) is accepted.

In this thesis, α-level is selected as 0.05 for Ryan-Joiner test. If p-value is less than 0.05 null hypothesis is rejected, otherwise null hypothesis is accepted (Ryan and Joiner, 1976). Since p-value<0.05 indicates non normality, logarithmic or root transformations should be applied for normalizing variable.

Scatterplots of the dependent (number of software classes or number of software methods) and independent variables (the number of distinct nouns or the number of distinct verbs) can be used to observe the linearity of the data points. In a scatterplot, the continuous line shows the regression line that represents the relationship between the dependent and independent variable. Data points correspond to dependent

variable versus independent variable of the projects. Note that when the data points are close to regression line, the prediction accuracy is high.

Residual is a graph that shows the difference between the actual and estimated values of the dependent variable. The linear regression analysis said to be appropriate if the data points in a residual plot are randomly scattered in the graph; otherwise, a non-linear model would be more suitable (Miles, 2014).

Please note that, all the statistical analyses in this thesis are performed by using the Minitab statistics tool version 17.

3.5. Outlier Analysis

An outlier is significantly different data points from other observations which fall outside the regression line. It can be very small or extremely large data points in data sets. Outliers can occur because of measurement errors including data entry errors, random errors, chance or unassignable causes.

Since outliers affect the accuracy of a regression, they should be identified. In order to identify outliers, several outlier detection techniques proposed in the literature can be used. One of the well-known outlier detection techniques is Cook's Distance (Cook's D) (Cook, 1977). According to this technique the data points with Cook's Distance greater than 4/n are treated as an outlier, where n is the number of projects that are considered for the analysis (Bollen and Jackman, 1990).

Cook's Distance is calculated as follows:

$$D_i = \frac{\sum_{j=1}^{n}(Y_{j-}Y_{j(i)})^2}{p\ MSE} \qquad \text{(Equation 9)}$$

where Y_j is the prediction from the regression model for observation *j*; $Y_{j(i)}$ is the prediction for observation *j* from a refitted regression model in which observation *i* has been omitted; p is the number of fitted parameters in the model and lastly the MSE is the Mean Square Error of the regression model which is described in Section 3.6.1.

In this study, the cause of the outlier occurrence can be due to insufficiently/extremely large written problem domain descriptions (independent variables) and/or source codes (dependent variables). There can be a very small project with extremely detailed descriptions or very a large project with insufficiently written problem domain descriptions.

Outlier detection for a small data set is very difficult. After the removal of the first outlier another outlier might appear. Since the process is repeated until no more data points are removed from the data set, very small number of projects might remain and the statistical analysis cannot be done accurately. Therefore, while analyzing the open source projects C~N and M~V are investigated together. If the data point's Cook's distance is larger than 4/n threshold for both C~N and M~V at the same time, then it is treated as outlier and removed from the data set.

3.6. Estimation Accuracy Evaluation

In this thesis, criteria listed below are used to assess the accuracy of method prediction model. All of these criteria are calculated with a program (given in Appendix A) written in R programming language which is a software platform for statistical computing (Ihaka and Gentleman, 1996).

- Magnitude of Relative Error (MRE),
- Mean of MRE (MMRE),
- Median of MRE (MdMRE),
- Prediction quality (Pred(e)),
- Mean Square Error (MSE),
- Coefficient of determination (R^2),
- Significance level (P-value).

3.6.1. MRE, MMRE, MdMRE, Pred(e) and MSE

The most common measures of the estimation accuracies are the Mean of MRE (MMRE) and median of MRE (MdMRE), where the MRE is defined as:

$$MRE = \frac{|AV - EV|}{AV} \qquad \text{(Equation 10)}$$

where AV is the actual value, and EV is the estimated value. MRE measure the difference between the actual and estimated values relative to the actual value for a given project (Conte et al., 1985). Hence, MMRE and MdMRE are calculated as follows:

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^{n} \text{MRE}_i \qquad \text{(Equation 11)}$$

$$\text{MdMRE} = \text{median}\,(\text{MRE}_i) \qquad \text{(Equation 12)}$$

The main difference between MMRE and MdMRE is that MMRE is more sensitive to predictions containing large MRE values. According to Conte, Dunsmore and Shen good predictions should give a MMRE and MdMRE smaller than 25% (Conte et al., 1986). On the other hand, Hastings and Sajeev state that a value of 0.20 can be considered as predictive, a value between 0.20 and 0.50 is considered acceptable, and a value greater than 0.50 is considered unacceptable (Hastings and Sajeev, 2001).

But, MRE-based accuracy measures have been criticized by several researchers in software engineering (Shepperd et al., 2000), (Foss et al., 2003) and (Jørgensen, 2007). Foss et al. (2003) performed a simulation study, in order to investigate whether MMRE is a reliable selection criterion or not. Their findings suggest that MMRE is an unreliable selection criterion in many cases. MMRE is sensitive to extremely large MREs, whereas MdMRE is less sensitive to extreme values. Instead of MMRE, Jørgensen (1995) suggested using MdMRE for avoiding the influence of outlier MRE values.

Prediction quality is calculated on a set of n projects as:

$$Pred(e) = k/n \qquad \text{(Equation 13)}$$

where k is the number of projects for which MRE is less than or equal to "e". That is, "e" is the selected threshold value for MRE. According to Conte, Dunsmore and Shen the value of Pred(0.25) should be greater than or equal to 0.75 (Conte et al., 1986). On the other hand, Tate and Verner suggested that a more realistic level of performance for the Pred(e) measure is Pred(0.30), and they conclude that for an acceptable estimation model the value of Pred(0.30) should exceed 0.70 (Tate and Verner, 1990). In this thesis, prediction quality for both e= 0.25 and e=0.30 are used for comparison.

For the pred(25) and pred(30) point of view, all of the projects are below the 0.70 threshold. But, Kitchenham et al., (2001) stated that pred(e) is insensitive to the degree of inaccuracy of estimates outside the specified threshold value. For instance, a pred(25) would not distinguish predictions deviate by 26% and predictions deviate by 260%. Jørgensen (2007) also criticized Conte, Dunshmore and Shen's (1985) 0.75 threshold for pred (25). He stated that there is no reference or argumentation in order to verify this threshold.

Finally, MSE is the statistical measure of the average of the squares of the errors. Two or more models can be compared by using their MSEs to assess how well they explain a given set of observations. The smaller MSE values are better. The MSE is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (AV_i - EV_i)^2 \qquad \text{(Equation 14)}$$

where AV_i is the actual value, EV_i is the estimated value of the i^{th} project and n is the number of projects.

3.6.2. Coefficient of Determination

The coefficient of determination (R^2) is an indicator of how well the model fits the data. The higher the R^2 value, the better the fitness of models. The R^2 values greater than 0.90 are considered predictive with high confidence, R^2 values between 0.70 and 0.90 are considered strong relationships that can be used with confidence, and if R^2 is less than 0.50 the model is not considered reliable (Hastings and Sajeev, 2001).

3.6.3. Cross Validation

In order to evaluate the prediction performance of a model on a given sample set, the Leave One Out Cross Validation (LOOCV) technique could be used. The LOOCV technique involves using a single observation from the original sample as the validation data, and the remaining observations as the training data (Stone, 1974) and (Picard and Cook, 1984). This is repeated until each observation in the sample is used once as the validation data. Then, MRE value could be computed for each sample, and overall prediction performance could be evaluated according to MMRE, MdMRE, MSE, etc.

CHAPTER 4

CORRELATION BETWEEN PROBLEM DOMAIN AND SOLUTION DOMAIN SIZE MEASURES FOR OPEN SOURCE PROJECTS

This chapter presents the investigation of correlations between the problem domain and the solution domain measures. For this purpuse, 37 open source object oriented software projects are used to conduct Case Study #1. These analyses serve as the foundation for the proposed estimation methodologies.

In software engineering community, the open source software projects have been used increasingly, since project artifacts, such as source codes, revision control histories, and developer communications, are freely available to researchers. Thousands of open source projects are available on the Internet, which makes open source software an ideal target for researchers with a desire to understand how software is built.

In this chapter, the correlations between the problem domain and the solution domain measures for open source software projects are analyzed and the applicability of regression analysis for prediction of the solution domain measures in terms of the problem domain measures is assessed. To the best of our knowledge, problem domain measures (the number of distinct nouns and the number of distinct verbs) have not been used for estimating software size and effort. Hence, this becomes the major contribution of this thesis study.

In this chapter, the first research question and partly the second research question raised in the introduction is addressed.

Section 4.1 describes analyzed projects which are used to conduct Case Study #1. Section 4.2 presents problem and solution domain measure correlations, proposed prediction methodology and accuracy evaluations on open source projects. In Section 4.3 Outlier analysis of the projects are presented. Lastly, in Section 4.4, a discussion of the findings is presented.

4.1. Analyzed Projects (Case Study #1)

In the analyses, 37 open source software projects in three different categories are considered;

- Open Source Games Projects (14 projects),

- Open Source Personal Organizer Projects (10 projects),

- Open Source Project Management Software Projects (13 projects).

Open source projects do not have any requirements artifact and problem domain descriptions. For this reason, in this thesis, user manuals (as the document that describes the program usage in English) are used as the closest approximation of the problem domain descriptions.

Projects are selected among open source projects listed in Wikipedia [2] and SourceForge[3], by paying attention to the following points:

- The project must be an open-source project. That is, source code must be available.

- The software must have been implemented in an object-oriented programming language.

- The user manual of the software is available in the official website of the project. The most important point that should be taken into consideration is that user manuals should be complete and consistent. That is, it should clearly and sufficiently describe the functionality and use of the software

The utilized open source games, personal organizers, and project management software are listed in Table 8, Table 9 and Table 10, respectively.

Table 8: Utilized Game Software Projects[4]

Project Name	Web Site
Adonthell	http://adonthell.nongnu.org/index.shtml
Exult	http://exult.sourceforge.net/
LinCity	http://lincity.sourceforge.net/
Enigma	http://www.nongnu.org/enigma/
Nuvie	http://nuvie.sourceforge.net/
BattleCity	http://www.battlecity.com.ua/
Rigs of Rods	http://www.rigsofrods.com/content/
BZFlag	http://bzflag.org/
FreeOrion	http://www.freeorion.org/
Wesnoth	http://www.wesnoth.org/
Planeshift	http://www.planeshift.it/
Lierox	http://www.openlierox.net/
CrackAttack	http://www.aluminumangel.org/attack/
Torcs	http://torcs.sourceforge.net/

[2] https://en.wikipedia.org/wiki/Comparison_of_project_management_software
https://en.wikipedia.org/wiki/List_of_open-source_video_games
https://en.wikipedia.org/wiki/List_of_personal_information_managers
[3] http://sourceforge.net/directory/business-enterprise/project-management/os:windows/
http://sourceforge.net/directory/business-enterprise/todo-lists/os:windows/
http://sourceforge.net/directory/games/os:windows/
[2,3]Last accessed in December 2014
[4] Last accessed in January 2015

Table 9: Utilized Personal Organizer Software Projects[5]

Project Name	Web Site
Xournal	http://xournal.sourceforge.net/
Taskwarrior	http://taskwarrior.org/
Chandler	http://chandlerproject.org/
Nevernote	http://nevernote.sourceforge.net/
GloboNote	http://globonote.info/
Rachota	http://rachota.sourceforge.net/
Iteraplan	https://www.iteraplan.de/en
Todomoo	http://todomoo.sourceforge.net/
OpenGroup WareCoils	http://www.opengroupware.us/
FreeMind	http://freemind.sourceforge.net/

Table 10: Utilized Project Management Software Projects[6]

Project Name	Web Site
LibrePlan	http://www.libreplan.com/
KForge	http://pythonhosted.org/kforge/
GanttProject	http://www.ganttproject.biz/
Tree.io	http://tree.io/
Plandora	http://www.plandora.org/
ProjectLibre	http://www.projectlibre.org/
Project.Net	http://www.project.net/
Scrinch	http://scrinch.sourceforge.net/
Onepoint Project	http://www.onepoint-project.com/home/overview
Task Juggler	http://www.taskjuggler.org/
Sonar Qube	http://www.sonarqube.org/
Freeplane	http://freeplane.sourceforge.net/
OFBiz	http://ofbiz.apache.org/

4.2. Problem and Solution Domain Measure Correlations for Case Study #1

Noun and verb identification in the problem domain descriptions are done by using the program given in Appendix A. The program uses NLTK and classifies words in the input documents into nouns and verbs by using the part of speech tags given in Table 7. The nouns and verbs identified are then stemmed by using NLTK's WordNet Lemmatizer module and plural terms are made singular, duplicate terms are eliminated, synonyms are consolidated into single term, nouns and verbs are listed in alphabetic order and irrelevant words (no meaning words, not noun/not verb) are removed automatically.

In this thesis study, it is claimed that the number of distinct nouns and number of distinct verbs in the problem domain descriptions can give an insight about the solution domain measures of object oriented software. Therefore, the software classes and methods in the source code which are downloaded from project web site are counted by using "Understand" (2008) static code analyzer tool.

Measurement results for the 37 open source projects are presented in Table 11 is for game projects, Table 12 is for personal organizer projects and Table 13 is for project

[5] Last accessed in February 2015
[6] Last accessed in December 2014

management projects. In these tables, **N** denotes the number of distinct nouns and **V** denotes the number of distinct verbs in the requirements. **C** denotes the number of classes in the software and **M** denotes the total number of methods in the software classes.

Table 11: Problem Domain and Solution Domain Measurement Results for Game Projects

Project	Problem Domain		Solution Domain	
	N	V	C	M
Adonthell	84	60	198	1887
Exult*	544	298	595	7432
LinCity	141	87	195	1458
Enigma	462	215	449	6499
Nuvie	229	108	285	5045
BattleCity	81	42	70	848
Rigs of Rods	166	99	257	5356
BZFlag	356	221	747	10531
FreeOrion	336	223	740	14805
Wesnoth	532	305	931	13678
Planeshift	212	106	224	5134
Lierox	288	142	804	14637
CrackAttack	121	88	50	585
Torcs*	722	320	209	4952

Table 12: Problem Domain and Solution Domain Measurement Results for Personal Organizer Projects

Project	Problem Domain		Solution Domain	
	N	V	C	M
Xournal	268	153	337	2461
Taskwarrior	253	141	226	1192
Chandler	127	94	197	1740
Nevernote	456	284	178	2168
GloboNote	182	122	328	2731
Rachota	196	116	320	1065
Iteraplan	1064	540	1479	6529
Todomoo	188	100	348	2589
OpenGroup WareCoils*	1400	712	976	5833
FreeMind	382	195	1113	7564

***** These projects have been removed as an outlier whose description is given in Section 4.3.

Table 13: Problem Domain and Solution Domain Measurement Results for Project Management Projects

Project	Problem Domain		Solution Domain	
	N	V	C	M
LibrePlan	506	265	3290	23266
KForge	81	48	412	1337
GanttProject	125	52	1300	6954
Tree.io	176	97	618	2474
Plandora	335	145	719	7691
ProjectLibre	537	286	2304	27261
Project.Net	628	352	4058	42953
Scrinch	134	95	286	1495
Onepoint Project	198	104	696	7991
Task Juggler	287	172	332	2323
Sonar Qube	525	236	2970	16643
Freeplane	282	177	2159	12221
OFBiz	355	147	2579	17265

After identifying and counting distinct nouns and distinct verbs in the users' manuals, and counting the software classes and software methods in the source code, Pearson's Correlation Coefficient is computed by the help of Minitab statistics tool. The Pearson's Correlation Coefficients, r_{XY}, between the problem domain measures, X, and the solution domain measures, Y, and the corresponding p-values for game software projects, personal organizer software projects and project management software projects are given in Table 14.

Table 14: Pearson's Correlation Coefficients and P-values for Open Source Projects

	X	Y	r_{XY}	P-value
GAME PROJECTS	N	C	0.834	0.001
	V	M	0.802	0.002
PERSONAL ORGANIZER PROJECTS	N	C	0.817	0.007
	V	M	0.607	0.083
PROJECT MANAGEMENT PROJECTS	N	C	0.859	0.000
	V	M	0.898	0.000

As it can be seen from Table 14, all r values are above 0.60. Since, it has been accepted that results between 0.5 and 1.0 has high correlation (DeSanto et al., 2010) it can be said that there are strong positive relationships between the problem and the solution domain measures. When P-values are considered, only M-V pair of personal organizer software has p value which is slightly greater than 0.05 threshold. Therefore we can be 91.7% confident that the strong correlation between variables is not due to chance. All other's p values are smaller than the 0.05 threshold. This is a strong evidence of significance and hence correlations can be considered statistically significant. That is, we can be 95% confident that the strong correlation between variables is not due to chance.

In order to assess the applicability of regression analysis, it has also been checked to see if the errors are well modeled by the normal distribution. Normality evaluation (Ryan and Joiner, 1976) results are given in Table 15.

Table 15: Ryan-Joiner Normality Test Results for Open Source Projects

	C ~ N	M ~ V
Game Projects	p-value>0.100	p-value=0.074
Personal Organizer Projects	p-value>0.042	p-value>0.014
Project Management Projects	p-value>0.100	p-value>0.100

According to results in Table 15 for normality, game and project management software projects have p-value>0.05. Since p-value >0.05 indicates normality, Ryan-Joiner test results approve the applicability of regression analysis to predict the solution domain measures in terms of the problem domain measures.

Personal organizer projects show non-normal distribution according to their p values. Therefore we should apply transformation. Since the logarithm transformation is one of the most popular transformation (Feng et al., 2013), logarithm transformation is applied in order to normalize the data. Transformed values are used for our statistical analysis. Although we have done the statistical analysis on transformed values, we should back transform our results. For the logarithmic transformation, we would back transform our results by raising 10 to the power of our number.

Transformed values for the personal organizer projects are given in Table 16.

Table 16: Transformed Values for Personal Organizer Projects

Project	Problem Domain		Solution Domain	
	logN	logV	logC	logM
Xournal	2.42813	2.18469	2.52763	3.39111
Taskwarrior	2.40312	2.14922	2.35411	3.07628
Chandler	2.10380	1.97313	2.29447	3.24055
Nevernote	2.65896	2.45332	2.25042	3.33606
GloboNote	2.26007	2.08636	2.51587	3.43632
Rachota	2.29226	2.06446	2.50515	3.02735
Iteraplan	3.02694	2.73239	3.16997	3.81485
Todomoo	2.27416	2.00000	2.54158	3.41313
FreeMind	2.58206	2.29003	3.04650	3.87875

After logarithmic transformation Ryan-Joiner test results are given in Table 17.

Table 17: Ryan-Joiner Normality Test Results for Personal Organizer Projects after Logarithmic Transformation

	C ~ N	M ~ V
Personal Organizer Projects	p-value>0.100	p-value>0.100

According to results in Table 17 for normality, Ryan-Joiner test results approve the applicability of regression analysis to predict the solution domain measures in terms of the problem domain measures.

In order to observe the differences between the actual and estimated values of the dependent variable (obtained by applying the regression equation), the scatterplots and corresponding residual plots for open source game, personal organizer and project management projects are given in
Figure 3 through Figure 14, respectively.

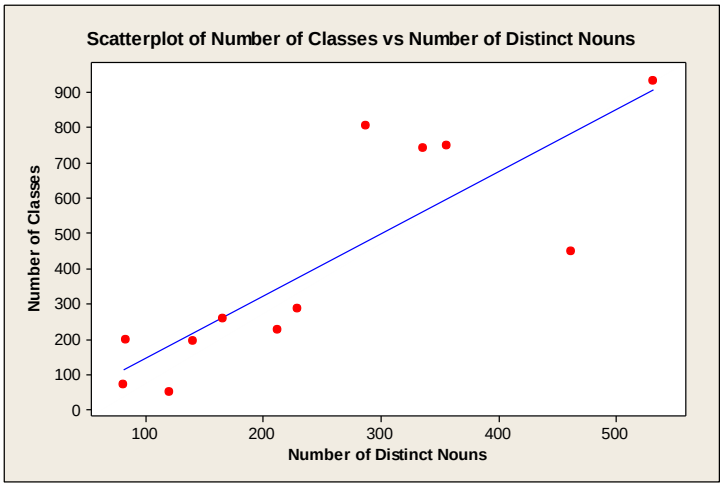

Figure 3: Scatterplot of Number of Classes vs. the Number of Distinct Nouns for Game Projects

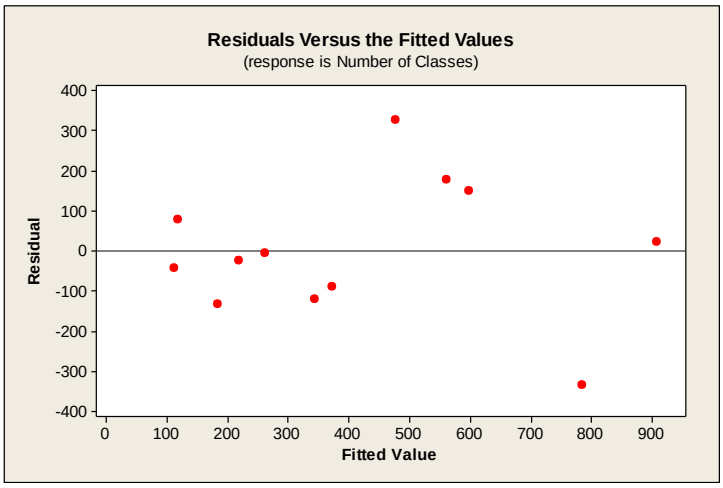

Figure 4: The Residuals vs. the Number of Distinct Nouns against the Number of Classes for Game Projects

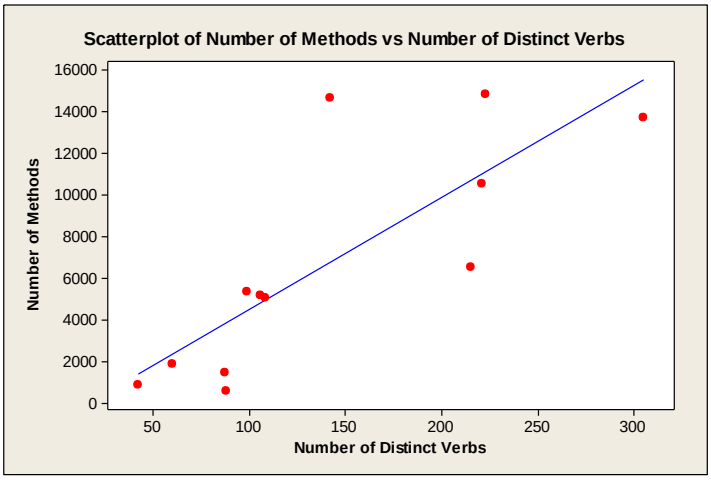

Figure 5: Scatterplot of Number of Methods vs. the Number of Distinct Verbs for Game Projects

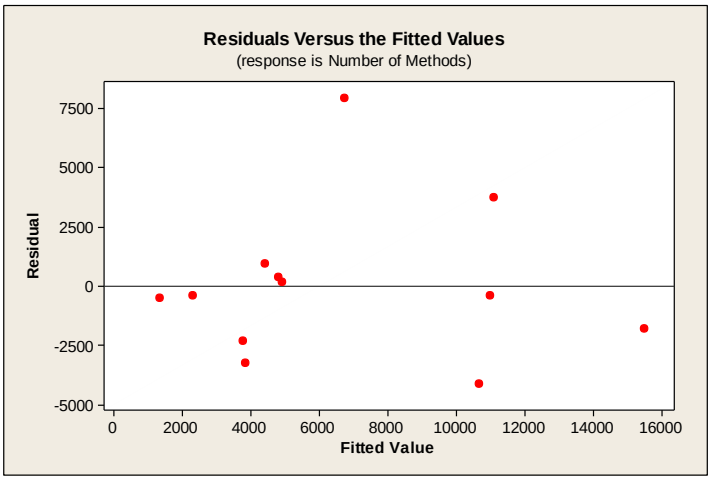

Figure 6: The Residuals vs. the Number of Distinct Verbs against the Number of Methods for Game Projects

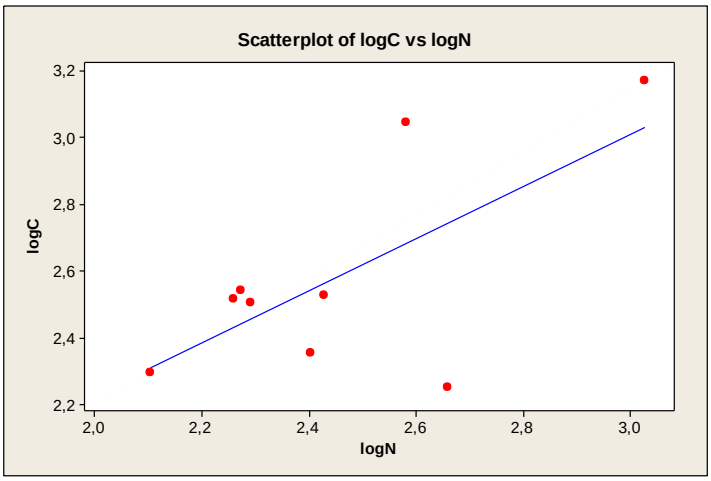

Figure 7: Scatterplot of Log C vs. the Log N for Personal Organizer Projects

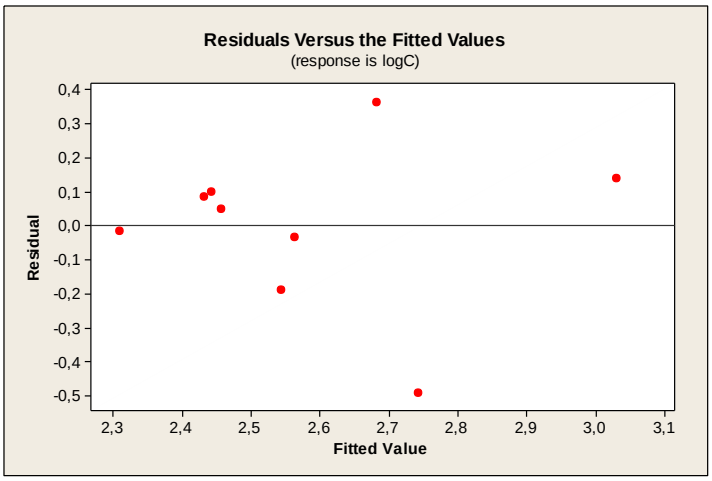

Figure 8: The Residuals vs. the Log N against the Log C for Personal Organizer Projects

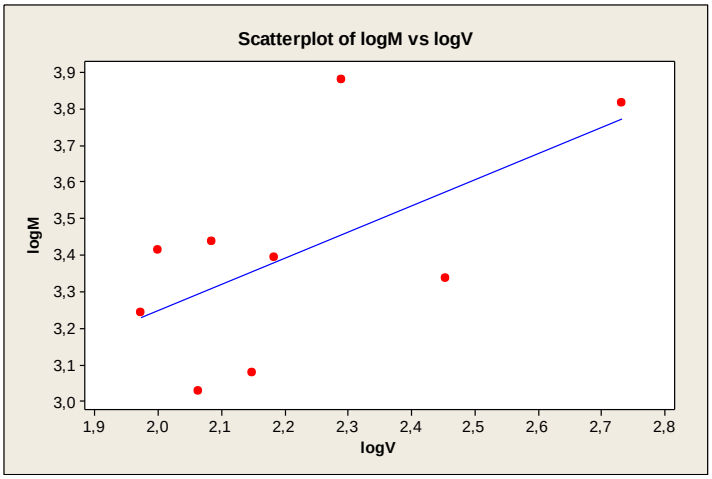

Figure 9: Scatterplot of Log M vs. the Log V for Personal Organizer Projects

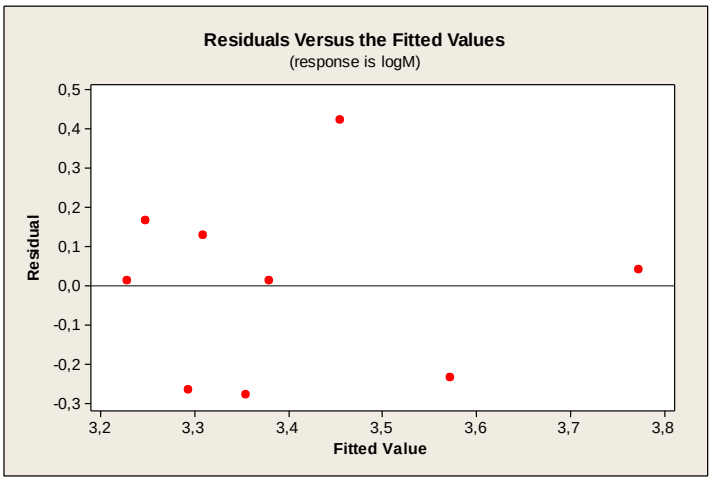

Figure 10: The Residuals vs. the Log V against the Log M for Personal Organizer Projects

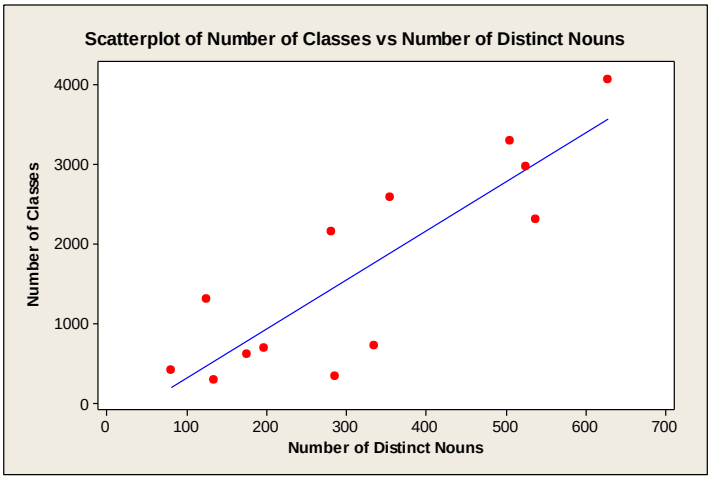

Figure 11: Scatterplot of Number of Classes vs. the Number of Distinct Nouns for Project Management Projects

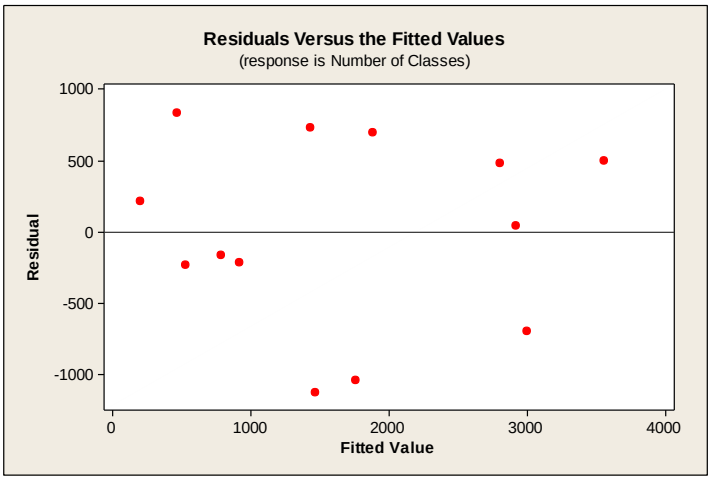

Figure 12: The Residuals vs. the Number of Distinct Nouns against the Number of Classes for Project Management Projects

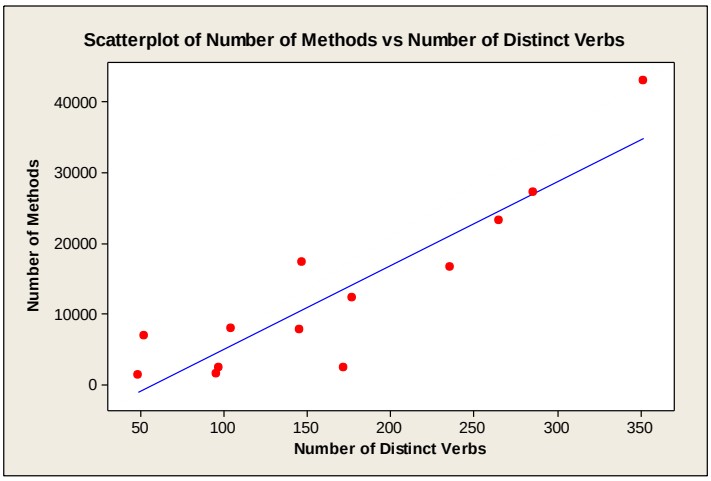

Figure 13: Scatterplot of Number of Methods vs. the Number of Distinct Verbs for Project Management Projects

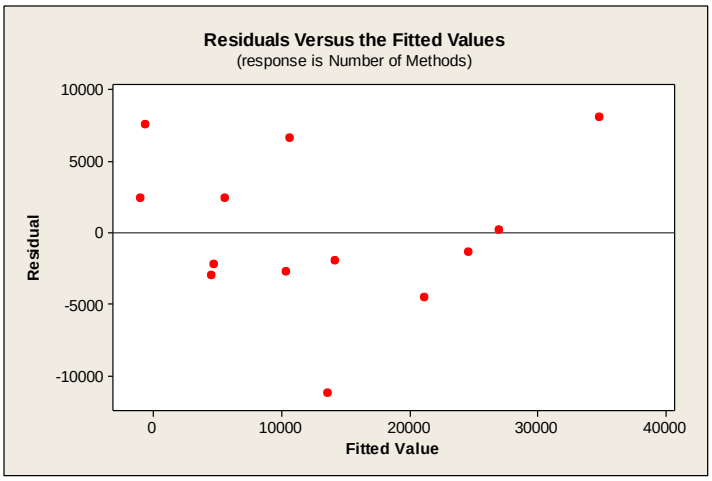

Figure 14: The Residuals vs. the Number of Distinct Verbs against the Number of Methods for Project Management Projects

If the points in a residual plot are randomly dispersed, a linear regression model is said to be appropriate for the data; otherwise, a non-linear model is more appropriate (Miles, 2014). As it can be seen from Figure 4, Figure 6, Figure 8, Figure 10, Figure 12 and Figure 14 there is no particular pattern and the variables are randomly scattered above and below the Residual=0 line. Therefore, linear regressions can be used for the analyzed open source projects.

The derived regression equation for the number of classes in the game projects is:

$$C = -28.912 + 1.761\ N \qquad \text{(Equation 15)}$$

For the Equation 15, R^2=0.700 and for predictor variable N p-value=0.00074. Since, R^2 value is above 0.50 the model is considered reliable and p value shows that the prediction models is statistically significant since it is all smaller than the 0.05 threshold.

The derived regression equation for the number of methods in the game projects is:

$$M = -889.4 + 53.7\ V \qquad \text{(Equation 16)}$$

For the Equation 16, R^2=0.644 and predictor variable V is significant as its p-value=0.0017.

The derived regression equation for the number of classes in project management projects is:

$$C = -300.35 + 6.15\ N \qquad \text{(Equation 17)}$$

For the Equation 17, R^2=0.738 and predictor variable N is significant as its p-value=0.00017.

The derived regression equation for the number of methods in project management projects is:

$$M = -6710.4 + 118.2\ V \qquad \text{(Equation 18)}$$

For the Equation 18, R^2=807 and predictor variable V is significant as its p-value=$3.1x10^{-5}$. Since, R^2 values for the regression equations for C and M are above 0.70 the models can be used with confidence and p values show that the prediction models are statistically significant since they are all smaller than the 0.05 threshold.

Please note that, for personal organizer projects, according to the normality test results, log transformation has been decided to be applied. Hence, the derived regression equation for the number of classes in personal organizer projects is:

$$\log C = 0.670 + 0.780\ \log N \qquad \text{(Equation 19)}$$

For the Equation 19, R^2=45.1 and predictor variable N is significant as its p-value=0.048. For this model, R^2 is below the 0.50 threshold.

The derived regression equation for the number of methods in personal organizer projects is:

$$\log M = 1.812 + 0.718\ \log V \qquad \text{(Equation 20)}$$

For the Equation 20, R^2=36.5 and predictor variable V is not significant as its p-value=0.08. For this model, p value is slightly above the 0.05 threshold but R^2 value is below the 0.50 threshold.

Table 18, Table 19 and Table 21 present the prediction accuracy evaluation of the open source projects by using LOOCV. In these tables, Y~X stands for the regression analysis where Y is the dependent variable to be predicted and X is the independent variable.

Table 18: Prediction Accuracy for Game Projects

Projects	**C ~ N**	**M ~ V**
	MRE	**MRE**
Adonthell	0.399	0.237
LinCity	0.125	1.596
Enigma	0.747	0.640
Nuvie	0.313	0.026
BattleCity	0.624	0.612
Rigs of	0.024	0.172
BZFlag	0.199	0.043
FreeOrion	0.239	0.250
Wesnoth	0.024	0.133
Planeshift	0.537	0.063
Lierox	0.405	0.539
CrackAttack	2.683	5.563
Prediction Accuracy	**pred(0.25)=0.417** **pred(0.30)=0.417** **MMRE =0.527** **MdMRE=0.356** **MSE =39712**	**pred(0.25)=0.50** **pred(0.30)=0.583** **MMRE =0.823** **MdMRE=0.244** **MSE = 12784055**

As the results for game projects, given in Table 18, indicate the MMRE result for C~N slightly over the 0.50 threshold. According to Hastings and Sajeev (2001), the MdMRE result for C~N is acceptable. According to Conte et al., (1986) MdMRE result for M~V is also acceptable. Since MMRE is more sensitive to predictions containing large MRE values, MMRE result for M~V is over the 0.50 threshold. Predicton quality values for both C~N and M~V are below the 0.70 threshold. However, there is one project whose MRE value is 0.313 which is slightly over the pred(0.30) threshold.

Correspondingly, the criticism of MRE based accuracy measures and prediction quality measures are given in Section 3.6.1.

Table 19: Prediction Accuracy for Project Management Projects

Projects	**C ~ N**	**M ~ V**
	MRE	**MRE**
LibrePlan	0.145	0.057
KForge	0.520	1.776
GanttProject	0.640	1.081
Tree.io	0.264	0.920
Plandora	1.446	0.355
ProjectLibre	0.302	0.006
Project.Net	1.122	0.187
Scrinch	0.830	2.019
Onepoint	0.317	0.301
Task Juggler	3.409	4.859
Sonar Qube	0.014	0.272
Freeplane	0.336	0.162
OFBiz	0.270	0.382
Prediction Accuracy	**pred(0.25)=0.231** **pred(0.30)=0.385** **MMRE =0.663** **MdMRE=0.317** **MSE = 529188**	**pred(0.25)=0.308** **pred(0.30)=0.385** **MMRE = 0.953** **MdMRE=0.355** **MSE = 42304784**

As the results for project management projects, given in Table 19 indicate, the MdMRE results for both C~N and M~V are acceptable according to Hastings and Sajeev (2001). Both MMRE and prediction quality values are above the thresholds for C~N and M~V. Actually, there are three projects whose MRE values are 0.302, 0.317 and 0.336 for C~N. They are very close to pred(30) threshold. Therefore, from the pred(0.30) point of view, the result can be considered as acceptable for C~N. There is also one project for M~V whose MRE value is 0.301 which is slightly over the pred(0.30) threshold.

The results for the regression models for the personal organizer projects together with back transformed values of the predicted values and the actual values are given in Table 20.

Table 20: Back Transformed Values for Personal Organizer Projects

	PREDICTED VALUES				ACTUAL VALUES	
Projects	**logC**	**10^{logC}**	**logM**	**10^{logM}**	**C**	**M**
Xournal	2.56	366	3.38	2399	337	2461
Taskwarrior	2.54	350	3.35	2262	226	1192
Chandler	2.31	204	3.23	1691	197	1740
Nevernote	2.74	554	3.57	3739	178	2168
GloboNote	2.43	270	3.31	2039	328	2731
Rachota	2.46	287	3.29	1966	320	1065
Iteraplan	3.03	1072	3.77	5931	1479	6529
Todomoo	2.44	277	3.25	1768	348	2589
FreeMind	2.68	482	3.46	2855	1113	7564

The prediction accuracy of the personal organizer projects are given according to the back transformed values in Table 21.

Table 21: Prediction Accuracy for Personal Organizer Projects

Projects	**Log C ~ Log N**	**Log M ~ Log V**
	MRE	**MRE**
Xournal	0.086	0.025
Taskwarrior	0.548	0.897
Chandler	0.035	0.028
Nevernote	2.112	0.724
GloboNote	0.176	0.253
Rachota	0.103	0.846
Iteraplan	0.275	0.091
Todomoo	0.204	0.317
FreeMind	0.566	0.622
Prediction Accuracy	**pred(0.25)=0.555 pred(0.30)=0.666 MMRE =0.456 MdMRE=0.204 MSE =81216**	**pred(0.25)=0.333 pred(0.30)=0.444 MMRE =0.422 MdMRE=0.317 MSE = 3124020**

As the results for personal organizer projects, given in Table 21, indicate the regression model gives acceptable MMRE and MdMRE results for both Log C~Log N and Log M~Log V according to Hastings and Sajeev (2001). From the pred(e) point of view, there are two projects which have 0.253 and 0.317 MRE results for Log M~Log V. They are very close to pred(0.25) and pred(0.30) thresholds. Therefore, from the pred(0.30) point of view, Log C~Log N can be considered as acceptable, since the prediction performance gets close to 0.70 threshold. On the other hand pred (0.25) results are below the 0.70 threshold for both Log C~Log N and Log M~Log V.

4.3. Outlier Analysis for Case Study #1

Outlier analysis results (Cook's Distances) for the game domain are given in Table 22. Cook's Distance threshold is 4/14=0.285. Torcs projects' Cook's Distances are greater

than the threshold value for both C~N and M~V. Therefore, Torcs project is treated as outlier and removed from the data set. For this project, the reason being identified as an outlier is having extremely detailed problem domain descriptions compared to its source code.

Table 22: Outlier Analysis for Game Projects

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Adonthell	0.00382	0.015326
Exult	0.00005	0.107238
LinCity	0.00991	0.034662
Enigma	0.00689	0.008631
Nuvie	0.00329	0.000095
BattleCity	0.05246	0.035142
Rigs of	0.00241	0.002175
BZFlag	0.05612	0.016229
FreeOrion	0.05628	0.141688
Wesnoth	0.21890	0.086400
Planeshift	0.01031	0.000353
Lierox	0.09706	0.190956
CrackAttack	0.06545	0.060061
Torcs	2.26641	0.591949

Table 23: Outlier Analysis for Game Projects after Removal of Torcs Project

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Adonthell	0.010261	0.007937
Exult	0.472944	0.623205
LinCity	0.003187	0.036026
Enigma	0.259962	0.042961
Nuvie	0.007911	0.000151
BattleCity	0.021666	0.018085
Rigs of	0.000251	0.003684
BZFlag	0.065612	0.005043
FreeOrion	0.072314	0.156361
Wesnoth	0.117216	0.007623
Planeshift	0.018507	0.000570
Lierox	0.158883	0.226815
CrackAttack	0.065841	0.065984

In Table 23, outlier analysis results after Torcs project is dropped are given. Note that the new threshold value is 4/13=0.307 and now a new outlier arises. Exult projects' Cook's Distances are greater than the threshold value for both C~N and M~V. Therefore, Exult project is treated as outlier and removed from the data set. For this project, the reason being identified as an outlier is having insufficient problem domain descriptions compared to its source code.

Table 24: Outlier Analysis for Game Projects after Removal of Exult Project

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Adonthell	0.030258	0.002263
LinCity	0.001646	0.038455
Enigma	0.890921	0.170442
Nuvie	0.012631	0.000091
BattleCity	0.009568	0.004294
Rigs of	0.000092	0.005100
BZFlag	0.059141	0.002269
FreeOrion	0.070723	0.156487
Wesnoth	0.010071	0.224970
Planeshift	0.024327	0.000584
Lierox	0.177044	0.270139
CrackAttack	0.060073	0.074011

In Table 24, outlier analysis results after Exult project is dropped are given. Note that the new threshold value is 4/12=0.333 and there is no more project whose Cook's Distances are greater than the new threshold value for both C~N and M~V. So, the outlier search is finished at this step.

The outlier analysis for the project management projects are given in Table 25.

Table 25: Outlier Analysis for Project Management Projects

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
LibrePlan	0.057738	0.00673
KForge	0.018184	0.03017
GanttProject	0.188824	0.28315
Tree.io	0.004893	0.01308
Plandora	0.103878	0.01128
ProjectLibre	0.157155	0.00017
Project.Net	0.181510	1.13628
Scrinch	0.014286	0.02365
Onepoint	0.007615	0.01336
Task Juggler	0.127530	0.18017
Sonar Qube	0.000544	0.05056
Freeplane	0.053165	0.00561
OFBiz	0.048449	0.06522

Outlier analysis results for project management domain are given in. Cook's Distance threshold is 4/13=0.307 and there is no project whose Cook's Distances are greater than the threshold value for both C~N and M~V. So, the outlier search is finished at this step.

The outlier analysis for the personal organizer projects are given in Table 26.

Table 26: Outlier Analysis for Personal Organizer Projects

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Xournal	0.00326	0.001537
Taskwarrior	0.02058	0.048082
Chandler	0.00980	0.009791
Nevernote	0.08028	0.039250
GloboNote	0.00006	0.001260
Rachota	0.00064	0.051594
Iteraplan	0.64460	0.100982
Todomoo	0.00004	0.001634
OpenGroup WareCoils	**2.43527**	**0.502303**
FreeMind	0.22501	0.375128

Outlier analysis results for personal organizer domain are given in Table 26. Cook's Distance threshold is 4/10=0.400 and OpenGroup WareCoils projects' Cook's Distances are greater than the threshold value for both C~N and M~V. Therefore, this project is treated as outlier and removed from the data set. For this project, the reason being identified as an outlier is having detailed problem domain descriptions compared to its source code.

Table 27: Outlier Analysis for Personal Organizer Projects after Removal of OpenGroup WareCoils Project

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Xournal	0.003819	0.001274
Taskwarrior	0.023722	0.042745
Chandler	0.000570	0.004918
Nevernote	0.229556	0.099603
GloboNote	0.002114	0.002656
Rachota	0.000242	0.043331
Iteraplan	0.495135	0.000078
Todomoo	0.003479	0.004912
FreeMind	0.279302	0.356973

In Table 27, outlier analysis results after OpenGroup WareCoils project is dropped are given. Note that the new threshold value is 4/9=0.444 and there is no project whose Cook's Distances are greater than the threshold value for both C~N and M~V. So, the outlier search is finished at this step.

4.4. Discussion

In this chapter, the following research questions are addressed.

- Are there any correlations between the problem domain measures and the solution domain measures for object oriented software?
- Can these correlations be utilized to estimate the software size?

According to the first group of case studies, 37 open source software development projects, high correlations between the problem domain measures and the solution domain measures are observed in Table 14. Hence, the first research question is answered.

For the second research question, these 37 open source software development projects' accuracy estimations have showed that, some of the projects give acceptable pred(25), pred(30), MMRE and MdMRE results. However, the Game projects' and the Project Management projects' MMRE results are over the 0.50 threshold.

MRE-based accuracy measures and prediction quality criticisms are given in Section 3.6.1.

As an overall assessment in order to determine the estimation accuracy, all of the criteria should be examined. Examination of only one criterion does not reflect the success level of the model.

Please also note that, in the regression equations derived, there may be a large negative constant (e.g. Equation 18) and the equations may produce irrelevant results when the number of nouns and/or verbs is small. Therefore, such equations must be used cautiously especially when the number of nouns/verbs is less than the smallest number of nouns/verbs in the corresponding project set.

Moreover, the open source projects have some drawbacks because of their nature. For example;

- Every project has documented and coded by different person. So, there is consistency problem between projects.

- There is no documentation standard. One project's documentation can be extremely large as compared to its source code; on the other hand the other projects' documentation can be insufficient as compared to its source code.

For most of the open source projects, use cases or other requirements artifacts are not available. Hence, in this thesis study, we utilized the user manuals in order to identify the number of distinct nouns and the number of distinct verbs. The most significant limitation of this case study is that the user manuals are just approximations to problem domain descriptions. Hence, we might not expect high prediction accuracies from the analyses. But the main point in this case study is to understand the nature of correlations in a specific context and to gain some insight about potential uses. In the next Chapter, we have used real life commercial software projects to demonstrate the potential use of correlation between the problem domain and the solution domain measures for size and effort prediction in the industry.

CHAPTER 5

SOLUTION DOMAIN MEASURE PREDICTION

This chapter answers our second research question via two case studies conducted on two sets of commercial software projects of two different companies. First, a solution domain measure prediction methodology that utilizes the findings and the approach presented in Chapter 4 is proposed. In order to validate the methodology, the correlations between the problem domain measures and the solution domain measures are evaluated, applicability of linear regression analysis is investigated and prediction performances of the derived models are evaluated on two sets of commercial software development projects.

In Section 5.1 the solution domain measure prediction methodology is proposed. Section 5.2 presents the second case study performed on twelve software projects of a CMMI Level-3 certified defense industry company. Finally, Section 5.3 presents the third case study conducted on fourteen software development projects of a different CMMI Level-3 certified defense industry company.

5.1. Solution Domain Measure Prediction Methodology

The analysis presented in Chapter 4 reveals a strong correlation between the problem domain measures and the solution domain measures for the analyzed open source projects. Chapter 4 also demonstrates the applicability of linear regression analysis for the prediction purposes. So, how can we exploit this strong correlation? The basic answer given to this question in the scope of this thesis is using correspondences between the problem and the solution domain measures to estimate the solution domain measures in terms of problem domain measures. Therefore, the following novel estimation methodology is proposed:

a) Take a set of already completed projects. Repeat steps b-c by using the requirements artifacts and source codes of each project;
b) Count the distinct nouns and distinct verbs in the requirements artifacts. This step can be automated by using a natural language processing tool;
c) Count the classes and methods in the source codes. This step can be automated by using a static code analysis tool;
d) Check suitability of the regression analysis. If the data is not suitable for regression analysis, detect and drop outliers and/or try logarithm (or root) transformation (these methods are explained and illustrated in Section 4.2 and Section 4.3);

e) Derive regression equations to predict the number of classes and the number of methods in terms of the number of distinct nouns and the number of distinct verbs;
f) Repeat steps b-d after each completed project to update the prediction model.

This methodology is applied on and validated by two sets of projects in the following sections.

5.2. Case Study #2

In order to investigate the correlation between the problem domain measures and the solution domain measures and to derive prediction models according to the methodology proposed in Section 5.1, twelve completed software development projects of a CMMI Level-3 certified defense industry company operating in Turkey (company X) have been analyzed as the second group of Case Studies . The projects are implemented in the C++ programming language on Eclipse CDT (C/C++ Development Tool). The software is developed according to DO-178B standard (RTCA DO-178-B, 1992). For managing product development the company uses the SCRUM agile software development methodology and pair programming technique is used in writing source codes. The revision controls of the documentations are handled by IBM Rational ClearCase tool. For bug tracking IBM Rational ClearQuest is utilized in the company. Each project has been developed by a team of 5 professional software engineers. The analysis is carried out for the subsystems of avionics mission control software. The low level requirements in the SRS and the final source code are used to collect the problem domain measures and the solution domain measures, respectively. Due to confidentiality reasons, further details of the projects cannot be given in this thesis; The projects are referred to as Project X_1, Project X_2, …, Project X_12 in the subsequent sections.

5.2.1. Measures and Correlation Analysis

In this section, the correlations between the problem domain measures and the solution domain measures of object oriented software for the second case study are analyzed and applicability of the regression analysis for prediction is investigated.

The problem domain measures considered in this section are the number of distinct nouns and distinct verbs in the problem descriptions. Problem domain measures are identified automatically by NLTK from the low level requirements specified in the SRS document of the projects. The nouns and verbs are identified according to the different POS tags given in Section 3.2.

The number of software classes and the total number of methods in the classes constitute our solution domain measures. In order to automate the counting process, Understand version 2.0 code analyzer tool has been used.

Measurement results for the twelve projects are presented in Table 28.

Table 28: Problem and Solution Domain Measurement Results for Case Study #2

Project	Problem Domain		Solution Domain	
	N	V	C	M
Project X_1	126	13	102	785
Project X_2	27	20	135	1292
Project X_3	31	8	52	417
Project X_4	13	7	43	355
Project X_5	52	25	82	1478
Project X_6	64	24	117	1357
Project X_7	29	19	78	729
Project X_8	66	12	65	547
Project X_9	26	8	67	546
Project X_10	182	35	732	3304
Project X_11	325	23	744	639
Project X_12	167	25	435	1520

In this table, **N** denotes the number of distinct nouns in the requirements, **V** denotes the number of distinct verbs in the requirements, **C** denotes the number of classes in the software and **M** denotes the total number of methods in the software classes.

The Pearson's correlation coefficients, r_{XY}, between the problem domain measures, X, and the solution domain measures, Y, are given in Table 29.

Table 29: Pearson's Correlation Coefficients and P-values for Case Study #2

X	Y	r_{XY}	P-value
N	C	0.900	0.000
V	M	0.866	0.000

As it can be seen from Table 29 all r values are above 0.850. Since, it has been accepted that results between 0.5 and 1.0 has high correlation (DeSanto et al., 2010) it means that there are very high positive relationships between the problem domain measures and the solution domain measures.

When P-values are considered, all values are less than the 0.05 threshold. So, it can be concluded that all findings are statistically significant with the 0.05 threshold.

5.2.2. Regression Analysis

Before applying linear regressions, it has also determined if the errors are well modeled by a normal distribution. Normality evaluation results of the projects are given in Table 30.

Table 30: Ryan-Joiner Normality Test Results for Case Study #2

C ~ N	M ~ V
p-value = 0.050	p-value > 0.100

According to results in Table 30 for normality, since p-value≥0.05 indicates normality, Ryan-Joiner test result shows the applicability of problem domain measures in order to predict the number of classes and number of methods.

After normality analysis, Equation 21 gives the number of classes in the software as a function of the number of distinct nouns and the number of distinct verbs in the problem domain descriptions.

$$C = -14.378 + 2.549\ N \qquad \text{(Equation 21)}$$

For the Equation 21, R^2=0.809 and predictor variable N is significant as pvalue=6.8×10^{-5}<0.05.

Equation 22 gives the number of methods in the software as a function of the number of distinct nouns and the number of distinct verbs in the problem domain descriptions.

$$M = -104.4 + 62.1\ V \qquad \text{(Equation 22)}$$

For the Equation 22, R^2=0.535 predictor variable V is significant as p-value= 0.006<0.05.

For the final regression equations derived above, all R^2 values are higher than 0.50. Therefore, the models can be considered reliable and the P-values are smaller than 0.05 threshold. Thus, prediction model is statistically significant.

In order to show the differences between the actual and estimated values of the dependent variable (obtained by applying the regression equation), scatterplots and residual plots are given in Figure 15 through Figure 18.

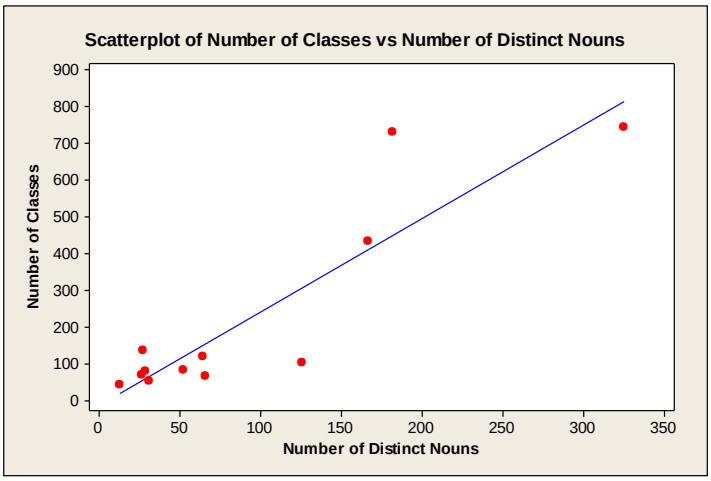

Figure 15: Scatterplot of Number of Classes vs. the Number of Distinct Nouns

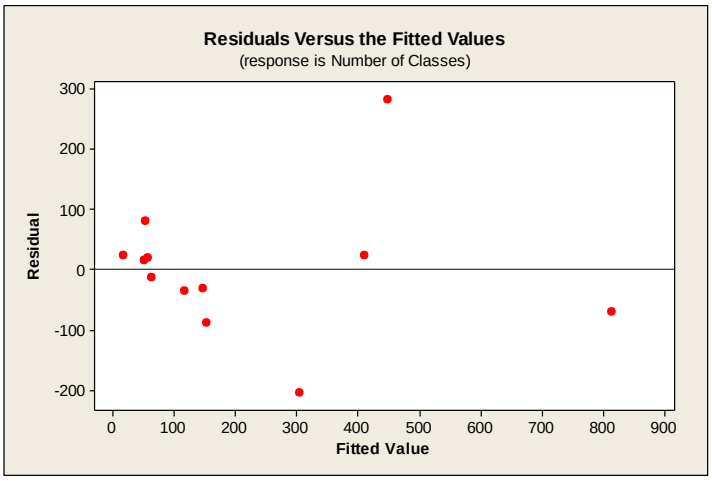

Figure 16: The Residuals vs. the Number of Distinct Nouns against the Number of Classes

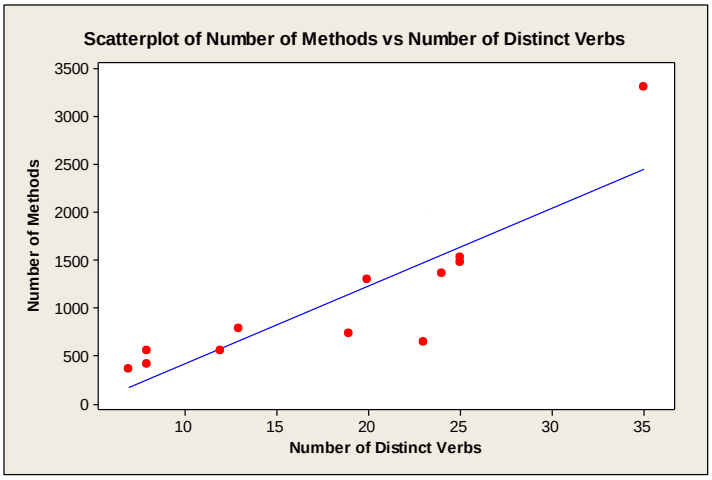

Figure 17: Scatterplot of Number of Methods vs. the Number of Distinct Verbs

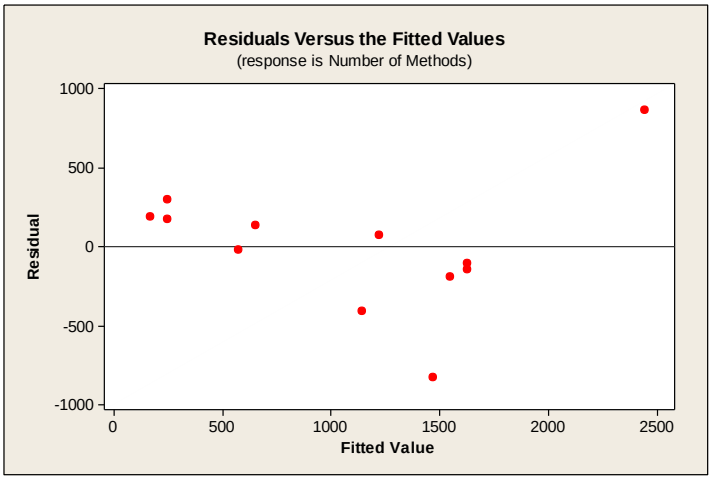

Figure 18: The Residuals vs. the Number of Distinct Verbs against the Number of Methods

As it can be seen from Figure 16 and Figure 18, there is no particular pattern and the variables are randomly scattered above and below the Residual=0 line. Therefore linear regression model is said to be appropriate for the data.

Outlier analysis results (Cook's Distances) for Case Study #2 are given in Table 31.

Table 31: Outlier Analysis for Case Study #2

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Project X_1	0,168190	0,00693
Project X_2	0,037755	0,00135
Project X_3	0,000881	0,02606
Project X_4	0,004191	0,03879
Project X_5	0,005599	0,01154
Project X_6	0,003869	0,01583
Project X_7	0,001927	0,04620
Project X_8	0,029813	0,00031
Project X_9	0,001345	0,08080
Project X_10	0,267639	2,50544
Project X_11	0,929744	0,25897
Project X_12	0,003724	0,00603

According to results given in Table 31, Cook's Distance threshold is 4/12=0.333 and there is no project whose Cook's Distances are greater than the threshold value for both C~N and M~V. So, the outlier search is finished at this step.

5.2.3. Prediction Performance

The accuracy of the linear regression based prediction approach is evaluated in terms of MRE, MMRE, MdMRE, Pred(0.25) and Pred(0.30) by LOOCV technique.

The results of the prediction accuracy evaluation are summarized in Table 32.

As the results indicate the regression model M~V give acceptable MMRE and predictive MdMRE results according to Hastings and Sajeev's evaluation. Since predicton quality value pred(0.30) is greater than 0.70 for M~V, the results can be considered as acceptable. Predicton quality value pred(0.25) is almost acceptable since prediction quality values are nearly 0.70. For C~N, MMRE and MdMRE are slightly greater than the 0.50 and 0.25 thresholds. Predicton quality values pred(0.25) and pred(0.30) are below the 0.70 threshold.

Table 32: Prediction Accuracy for Case Study #2

Projects	C ~ N	M ~ V
	MRE	MRE
Project X_1	2.008	0.104
Project X_2	0.596	0.119
Project X_3	0.243	0.058
Project X_4	0.563	0.069
Project X_5	0.441	0.020
Project X_6	0.271	0.021
Project X_7	0.236	0.475
Project X_8	1.367	0.171
Project X_9	0.225	0.281
Project X_10	0.385	0.373
Project X_11	0.094	2.043
Project X_12	0.054	0.047
Prediction Accuracy	**pred(0.25)=0.417 pred(0.30)=0.50 MMRE =0.541 MdMRE=0.329 MSE =19270**	**pred(0.25)=0.667 pred(0.30)=0.75 MMRE =0.316 MdMRE=0.112 MSE =573235**

5.3. Case Study #3

In this section, the prediction methodology presented in Section 5.1 is applied again in order to investigate the correlation between the problem domain measures and the solution domain measures and to develop the prediction model fourteen completed software development projects of another CMMI Level-3 certified defense industry company operating in Turkey (company Y). The missions of the company include developing national and international projects in areas such as Command Control Systems Software and Mission Support System Software, and performing research and new technology development. The projects analyzed are implemented in the Java programming language by using Eclipse Java development tools (JDT). As a requirement management tool the company uses Rational Dynamic Object Oriented Requirements System (DOORS). For UML modeling, Rational Rhapsody is used in the company. Each project has been developed by a team of 4-8 professional software engineers. The detailed fully dressed use cases have been used to capture functional requirements. These use cases and the resulting source codes are utilized to collect the problem domain measures and the solution domain measures, respectively. Due to confidentiality reasons, further details of the project cannot be given in this thesis and the projects are referred to as Project Y_1, Project Y_2, …, Project Y_14 in the subsequent sections.

5.3.1. Measures and Correlation Analysis

In this case study, in order to identify problem domain measures, the use cases written by the company are used and contrary to other case studies, manual noun/verb

identification is carried out to avoid errors[7] that may affect the accuracy of our analysis. The number of software classes and the total number of methods in the classes are the solution domain measures. In order to automate the counting process, the static code analysis software tool, Understand, has been used.

Measurement results for the fourteen projects are presented in Table 33. In this table, **N** denotes the number of distinct nouns in the use cases, **V** denotes the number of distinct verbs in the use cases, **C** denotes the number of classes in the software and **M** denotes the total number of methods in the software classes.

Table 33: Problem and Solution Domain Measurement Results

Projects	Problem Domain		Solution Domain	
	N	V	C	M
Project Y_1	517	248	341	2879
Project Y_2	715	344	484	4102
Project Y_3	243	136	189	1899
Project Y_4	383	195	302	2644
Project Y_5	80	53	62	661
Project Y_6	99	61	61	780
Project Y_7	195	97	157	985
Project Y_8	199	103	152	836
Project Y_9	343	187	292	1998
Project Y_10	209	118	174	1937
Project Y_11	132	69	99	599
Project Y_12	105	51	79	623
Project Y_13	680	287	513	3108
Project Y_14	121	57	78	775

Pearson's correlation coefficients, r_{XY}, between the problem domain measures, X, and the solution domain measures, Y, are given in Table 34.

Table 34: Pearson's Correlation Coefficients and P-values

X	Y	r_{XY}	p-value
N	C	0.99	1.26×10^{-11}
V	M	0.97	4.14×10^{-09}

As it can be seen from the table all r values are above 0.96 which means that there are very strong positive relationships between the problem domain measures and the

[7] Due to confidentiality reasons and in accordance with the non-disclosure agreement (NDA) signed between the company and us we are only allowed to work only on the hard copies of the use cases and the soft copies are not provided. We could convert the printed documents to electronic form by using an Optical Character Recognition (OCR) tool. Instead, we identified nouns and verbs manually to avoid potential OCR errors that may affect the accuracy of our analyses. Nevertheless, we encourage practitioners to automate the counting process by a natural language processing tool such as NLTK as described in Section 3.2.

solution domain measures. For the all the correlation coefficients the p-values are less than significance level α=0.05. Hence these results can be considered statistically significant and we can be confident that the relationship between variables is not due to chance.

5.3.2. Regression Analysis

In this section, by using the data given in Table 33, regression analysis is carried out and regression equations are derived for predicting the solution domain measures by using problem domain measures. Before applying linear regression, we have to check if the errors are well modeled by a normal distribution by using Ryan-Joiner Normality Test. Normality evaluation results of the projects are given in Table 35.

Table 35: Ryan-Joiner Normality Test Results

C ~ N	M~ V
p-value > 0.100	p-value > 0.100

According to results in Table 35 for normality, since p-value >0.05 indicates normality, Ryan-Joiner test result shows the applicability of using number of distinct nouns in order to predict the number of classes and it also shows the applicability of using number of distinct verbs in order to predict the number of methods.

The scatterplots and the regression lines are given in Figure 19, Figure 21 for Equation 22 and Equation 23, respectively. The corresponding residual plots are given in Figure 20, Figure 22. Data points in a residual plot give insight about the linearity of the model. As it can be seen from, the plots, there is no particular pattern and the variables are randomly scattered above and below the Residual=0 line.

By using the data given in Table 33, regression analysis is carried out and regression equations are derived for predicting the solution domain measures by using problem domain measures.

Equation 23 gives the number of classes in the software as a function of the number of distinct nouns and the number of distinct verbs in the problem domain descriptions.

$$C = 10.9468 + 0.7037N \qquad \text{(Equation 23)}$$

For the Equation 23, R^2=0.98 and predictor variable N is significant as p value=1.3×10^{-11}.

Equation 24 gives the number of methods in the software as a function of the number of distinct nouns and the number of distinct verbs in the problem domain descriptions.

$$M = 40.400 + 11.595V \qquad \text{(Equation 24)}$$

For the, Equation 24 R^2=0.949 and predictor variable V is significant as p-value= 4.1×10^{-9}.

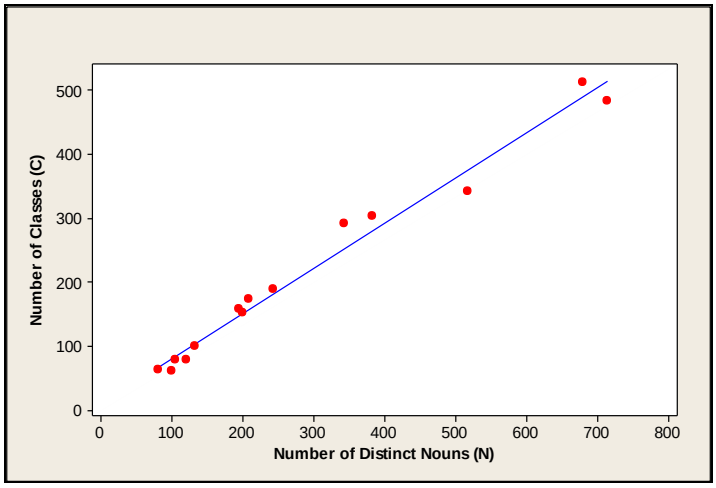

Figure 19: Scatterplot of Number of Classes vs. the Number of Distinct Nouns

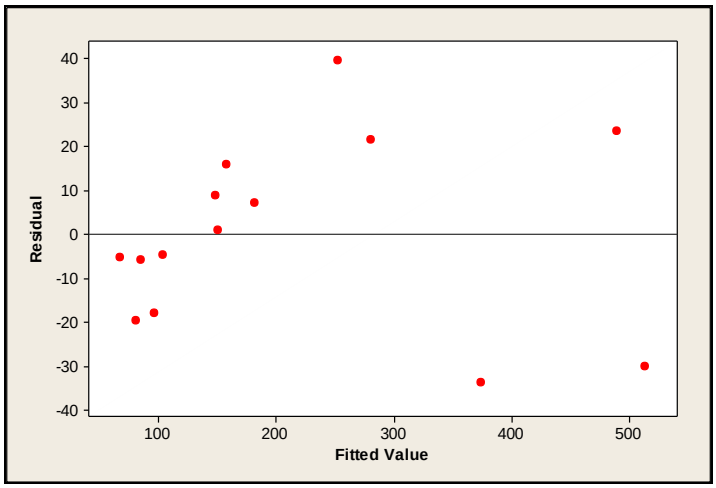

Figure 20: The Residuals vs. the Number of Distinct Nouns against the Number of Classes

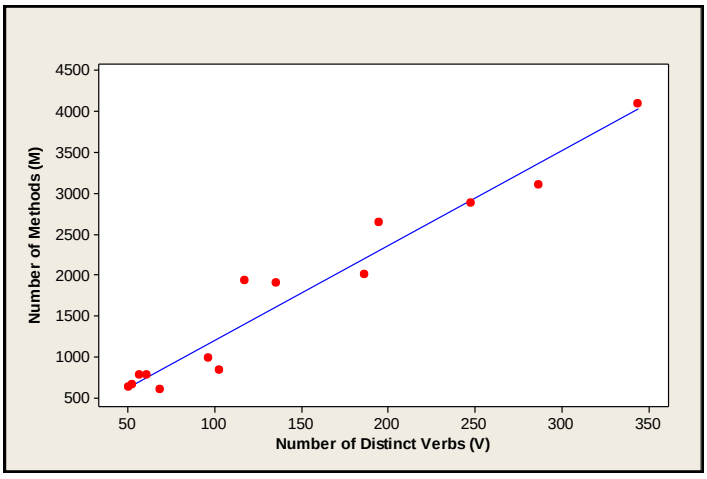

Figure 21: Scatterplot of Number of Methods vs. the Number of Distinct Verbs

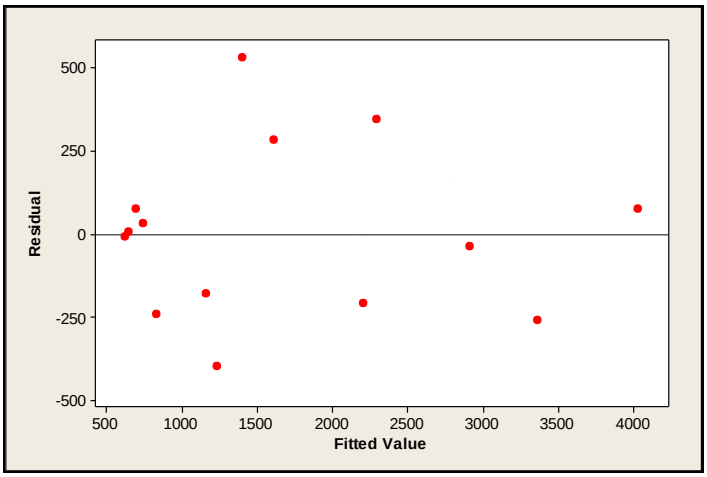

Figure 22: The Residuals vs. the Number of Distinct Verbs against the Number of Methods

Outlier analysis results (Cook's Distances) for Case Study #3 are given in Table 36.

Table 36: Outlier Analysis for Case Study #3

Projects	C ~ N	M ~ V
	Cook's Distance	Cook's Distance
Project Y_1	0,268941	0,002290
Project Y_2	0,933513	0,045217
Project Y_3	0,004467	0,046666
Project Y_4	0,049825	0,094985
Project Y_5	0,005587	0,000049
Project Y_6	0,069156	0,001251
Project Y_7	0,008246	0,024768
Project Y_8	0,000106	0,114223
Project Y_9	0,145999	0,033004
Project Y_10	0,025517	0,177569
Project Y_11	0,003442	0,062675
Project Y_12	0,005901	0,000106
Project Y_13	0,426326	0,208694
Project Y_14	0,051376	0,006895

According to results, the Cook's Distance threshold is 4/14=0.285 and there is no project with Cook's Distance greater than the threshold value for both C~N and M~V. So, the outlier search is finished at this step.

5.3.3. Prediction Performance

In this sub section, the accuracy of the prediction approach is evaluated according to MRE, MMRE, MdMRE, Pred(25), Pred(30), and MSE.

In order to evaluate the prediction performance, regression equations are derived and predictions are compared to the actual values by using the LOOCV technique. The results are summarized in Table 37.

Table 37: Prediction Accuracy for Number of Classes and Methods

Projects	**C ~ N**	**M ~ V**
	MRE	**MRE**
Project Y_1	0.118	0.015
Project Y_2	0.101	0.030
Project Y_3	0.040	0.160
Project Y_4	0.078	0.143
Project Y_5	0.099	0.011
Project Y_6	0.370	0.048
Project Y_7	0.061	0.201
Project Y_8	0.007	0.521
Project Y_9	0.147	0.116
Project Y_10	0.100	0.296
Project Y_11	0.055	0.457
Project Y_12	0.085	0.016
Project Y_13	0.069	0.111
Project Y_14	0.263	0.110
Prediction Accuracy	**pred(0.25)=0.86** **pred(0.30)=0.93** **MMRE = 0.114** **MdMRE=0.092** **MSE = 653**	**pred(0.25)=0.79** **pred(0.30)=0.86** **MMRE = 0.160** **MdMRE=0.113** **MSE = 76144**

As the results indicate the regression models give predictive MdMRE and acceptable MMRE according to Hastings and Sajeev's evaluation. Since predicton quality values pred(0.25) and pred(0.30) are both greater than 0.70, the results can be considered as acceptable.

CHAPTER 6

SIZE PREDICTION USING PROBLEM DOMAIN MEASURES

UCP and CFP are widely accepted software size measures. In Chapter 5, it has been shown that, the number of software classes and the number of software methods can be predicted by using the problem domain descriptions.

Consequently, it may be expected that the number of software classes and methods are well correlated with other software size measures. Hence, the fourth case study is performed on projects of company "Y" to derive linear regression based prediction models for UCP and COSMIC FFP by using the problem domain measures.

For this purpose, correlations between the UCP and COSMIC FFP sizes of the software and the problem domain measures are investigated. Then applicability and performance of the size prediction methodology that uses problem domain measures as the input are evaluated via Case Study 4.

Section 6.1 presents the size measures and correlations between the UCP and CFP sizes of the software and the problem domain measures. Section 6.2 presents the regression analysis of the size prediction model. Finally, Section 6.3 presents the prediction performance of the linear regression based size prediction model.

6.1. Measures and Correlation Analysis (Case Study #4)

In order to measure the size of the software projects, uses cases are utilized for the UCP measurements and Functional User Requirements (FUR) expressed in the SRS documents are used for CFP measurements.

The UCP measurements are made by a team that includes the author of this thesis. The CFP measurements are made by the company.

The data collected is presented in Table 38.

Table 38: UCP and CFP Size Measures

Projects	N	V	UCP	CFP
Project Y_1	517	248	579.75	372
Project Y_2	715	344	738.50	531
Project Y_3	243	136	308.85	188
Project Y_4	383	195	542.40	291
Project Y_5	80	53	69.40	144
Project Y_6	99	61	90.80	132
Project Y_7	195	97	259.00	195
Project Y_8	199	103	361.50	191
Project Y_9	343	187	416.75	312
Project Y_10	209	118	341.20	249
Project Y_11	132	69	316.00	268
Project Y_12	105	51	170.60	187
Project Y_13	680	287	674.00	345
Project Y_14	121	57	190.60	239

The Pearson's Correlation Coefficients, r_{XY} together with p-values for the problem domain measures X and the size measures Y are given in Table 39.

Table 39: Prediction Accuracy for UCP and CFP

X	Y	r_{XY}	p-value
N	UCP	0.953	0.000
V	UCP	0.954	0.000
N	CFP	0.886	0.000
V	CFP	0.897	0.000

As it can be noticed from the table, there are strong positive correlations between the UCP and CFP size measurements and the number of distinct nouns and the number of distinct verbs in the problem descriptions.

The correlations are statistically significant as the p-values are less than the significance level α=0.05. Therefore, one can be confident that the relationships between the effort and problem domain measures are not due to chance.

6.2. Regression Analysis

Before applying linear regression, normality tests are conducted. Ryan-Joiner Normality Test results are given in Table 40.

Table 40: Ryan-Joiner Normality Test Results

UCP~ N	UCP~ V	CFP~ N	CFP~ V
p-value > 0.100	p-value > 0.100	p-value > 0.100	p-value > 0.100

According to results in Table 40 for normality, it has been observed that all p-values are greater than 0.05 threshold. Since p-value>0.05 indicates normality, it can be concluded that errors follow the normal distribution.

Scatterplots and residual plots of the problem domain measures versus the UCP and CFP are given in Figure 23 through Figure 30.

In each scatterplot, the continuous line shows the regression line that represents the relationship between the dependent and independent variable and the data points correspond to dependent variable versus independent variable.

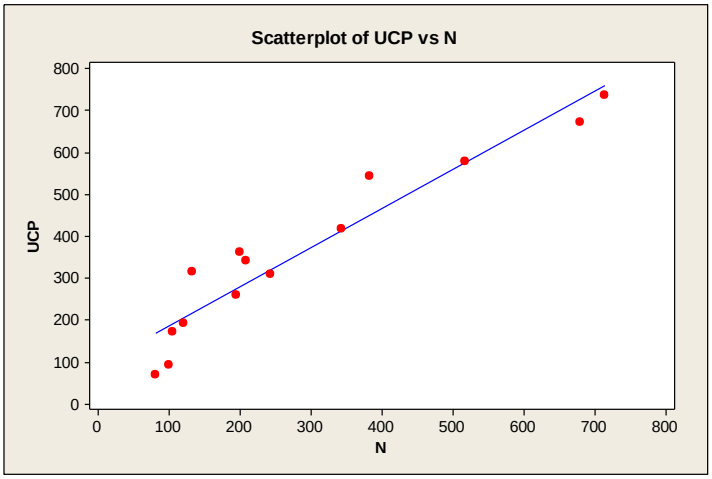

Figure 23: Scatterplot of UCP vs. the Number of Distinct Nouns

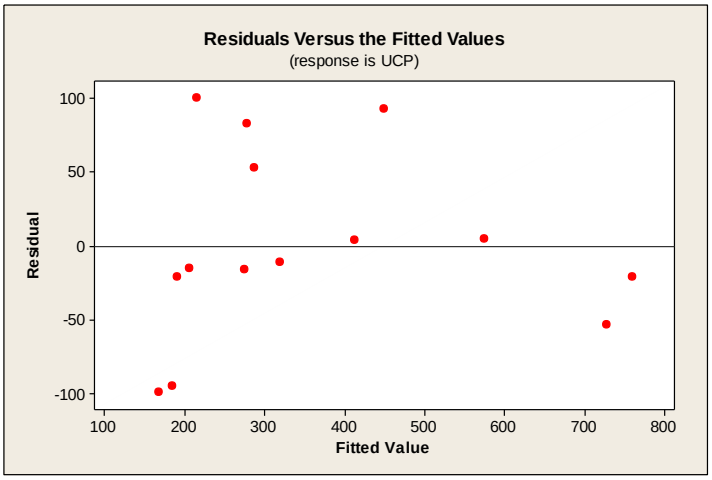

Figure 24: The Residuals vs. the Number of Distinct Nouns against the UCP

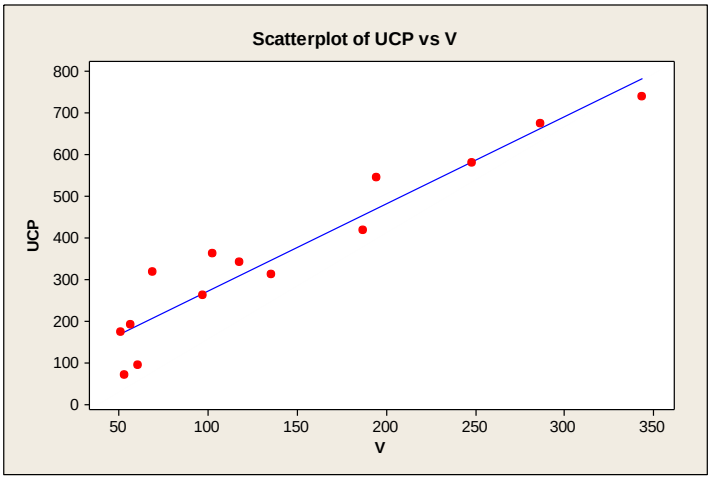

Figure 25: Scatterplot of UCP vs. the Number of Distinct Verbs

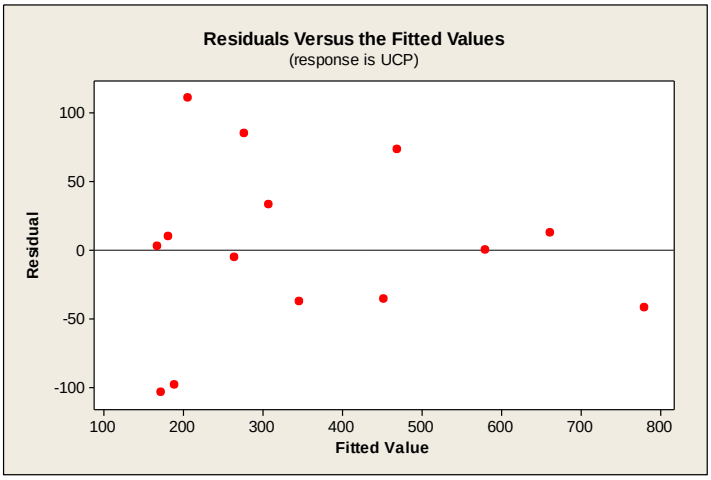

Figure 26: The Residuals vs. the Number of Distinct Verbs against the UCP

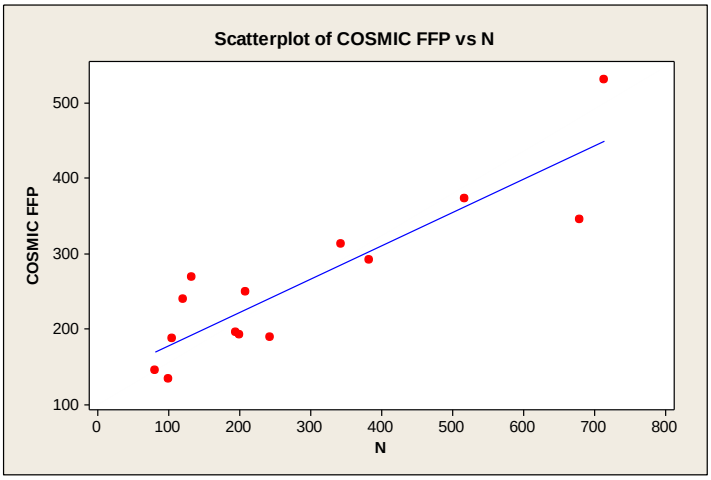

Figure 27: Scatterplot of CFP vs. the Number of Distinct Nouns

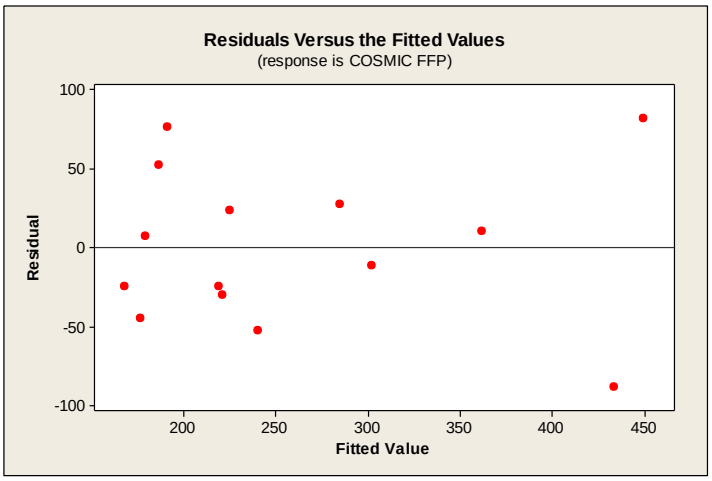

Figure 28: The Residuals vs. the Number of Distinct Nouns against the CFP

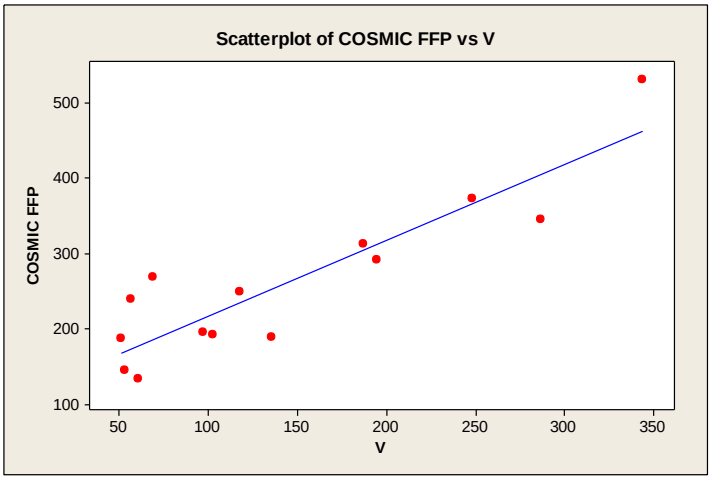

Figure 29: Scatterplot of CFP vs. the Number of Distinct Verbs

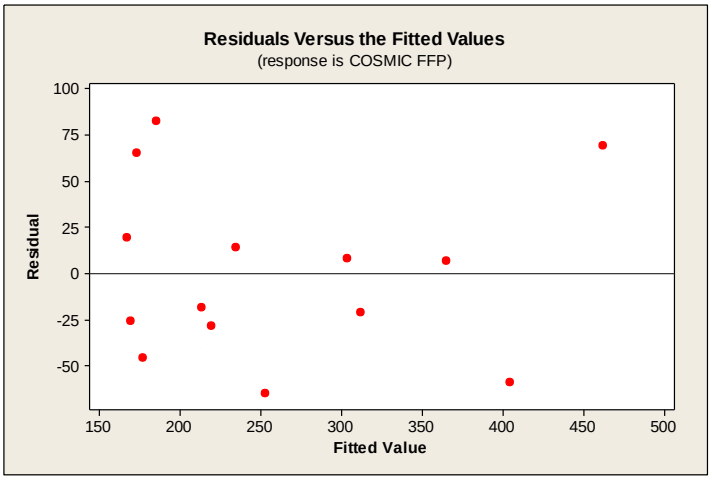

Figure 30: The Residuals vs. the Number of Distinct Verbs against the CFP

According to the residual plots in Figure 24, Figure 26, Figure 28 and Figure 30 there is no particular pattern and the variables are randomly scattered above and below the Residual=0 line.

The analysis results suggest that the problem domain measures are correlated with UCP and CFP size measurements and regression analysis is appropriate for the considered projects. Therefore, it is possible to apply the regression analysis that is similar to the approach presented in Section 5.1 for size prediction.

The regression equation to predict UCP in terms of the number of distinct nouns the UCP size prediction equation is:

$$UCP\_{estimated}=93.801+0.932\ N \qquad \text{(Equation 25)}$$

For the Equation 25, $R^2=0.909$ and predictor variable N is significant as its p-value=1.3×10^{-7}. Similarly, in terms of the number of distinct verbs, the UCP size prediction equation is:

$$UCP\_{estimated}=61.95+2.09\ V \qquad \text{(Equation 26)}$$

For this model, $R^2=0.910$ and predictor variable V is significant as its p-value=1.3×10^{-7}.

The regression equation to predict CFP in terms of the number of distinct nouns the CFP size prediction equation is:

$$CFP_{estimated}=133.3654+0.4419\ N \qquad \text{(Equation 27)}$$

For the Equation 27, $R^2=0.785$ and predictor variable N is significant as its p-value=2.5×10^{-5}. Similarly, in terms of the number of distinct verbs, the CFP size prediction equation is:

$$CFP_{estimated}=133.365+0.4419\ V \qquad \text{(Equation 28)}$$

For this model, $R^2=0.805$ and predictor variable V is significant as its p-value=1.37×10^{-5}.

6.3. Prediction Performance

The accuracy of the size prediction models are evaluated in terms of MRE, MMRE, MdMRE, Pred(0.25) and Pred(0.30) obtained by LOOCV technique.

The accuracy evaluation results for UCP and CFP prediction by using the number of distinct nouns and the size prediction by using the number of distinct verbs are presented in Table 41 and Table 42, respectively.

Table 41: Prediction Accuracy for UCP Size Prediction

Projects	UCP ~ N	UCP ~ V
	MRE	MRE
Project Y_1	0.007	0.000
Project Y_2	0.029	0.057
Project Y_3	0.036	0.120
Project Y_4	0.169	0.134
Project Y_5	1.425	1.488
Project Y_6	1.048	1.086
Project Y_7	0.063	0.021
Project Y_8	0.227	0.233
Project Y_9	0.008	0.086
Project Y_10	0.154	0.095
Project Y_11	0.313	0.347
Project Y_12	0.123	0.012
Project Y_13	0.079	0.018
Project Y_14	0.083	0.050
Prediction Accuracy	**pred(0.25)=0.786 pred(0.30)=0.786 MMRE =0.269 MdMRE=0.103 MSE =4884**	**pred(0.25)=0.786 pred(0.30)=0.786 MMRE = 0.268 MdMRE=0.091 MSE =4849**

Table 42: Prediction Accuracy for CFP Size Prediction

Projects	CFP ~ N	CFP ~ V
	MRE	MRE
Project Y_1	0.027	0.017
Project Y_2	0.153	0.130
Project Y_3	0.280	0.345
Project Y_4	0.039	0.072
Project Y_5	0.171	0.178
Project Y_6	0.341	0.346
Project Y_7	0.125	0.096
Project Y_8	0.158	0.151
Project Y_9	0.086	0.025
Project Y_10	0.093	0.056
Project Y_11	0.284	0.306
Project Y_12	0.038	0.103
Project Y_13	0.257	0.172
Project Y_14	0.218	0.273
Prediction Accuracy	**pred(0.25)=0.714 pred(0.30)=0.929 MMRE =0.163 MdMRE=0.156 MSE =4039**	**pred(0.25)=0.714 pred(0.30)=0.786 MMRE =0.163 MdMRE=0.141 MSE =3291**

According to the results, pred(0.25) and pred(0.30) for all UCP~N , UCP~V, CFP~N and CFP~V models are greater than 0.70 and they are acceptable.

The results in terms of MMRE and MdMRE for all UCP~N, UCP~V, CFP~N and CFP~V models are predictive.

According to MSE results UCP~V is slightly better than the UCP~N and CFP~V is slightly better than the CFP~N.

Since, both UCP and CFP methodologies need some effort and some experience in measurement, with our proposed methodology, one can save time for the measurement by using natural language processing tools. Therefore it can be concluded that the number of distinct nouns and the number of distinct verbs can be used in order to estimate UCP and COSMIC FFP size measurements earlier in the software development lifecycle.

CHAPTER 7

EFFORT PREDICTION

There are studies such as Misic and Tesic, (1998) that point out the correlation between the effort and the total number of classes/methods. Therefore, in this chapter investigation and utilization such correlations are identified in order to evaluate the applicability of a linear regression based effort prediction methodology that uses problem domain measures as the input. The accuracy of the proposed methodology is compared to efforts predicted by using the UCP and CFP size measurements of the corresponding software via Case Study 5.

Section 7.1 presents the measures and correlation analysis of the proposed effort prediction methodology. In Section 7.2 regression analysis of the methodology is discussed. Lastly, Section 7.3 evaluates the prediction accuracy of the methodology.

7.1. Measures and Correlation Analysis (Case Study #5)

The effort data collected by the company and size measurement results for the fourteen software development projects introduced in Section 5.2.1 are presented in Table 43.

Table 43: Actual Effort and Measured Size

Projects	AE (person hour)	UCP	CFP
Project Y_1	10561	579.75	372
Project Y_2	13105	738.50	531
Project Y_3	5819	308.85	188
Project Y_4	8342	542.40	291
Project Y_5	2165	69.40	144
Project Y_6	2354	90.80	132
Project Y_7	4667	259.00	195
Project Y_8	6439	361.50	191
Project Y_9	7210	416.75	312
Project Y_10	5336	341.20	249
Project Y_11	5597	316.00	268
Project Y_12	2989	170.60	187
Project Y_13	11286	674.00	345
Project Y_14	2678	190.60	239

Please note that, AE denotes the actual effort data in person-hours.

The first step of the analysis involves computing Pearson's Correlation Coefficient in order to check if there is a correlation between actual effort, Y, and problem domain measures, X. The computed Pearson's Correlation Coefficients, r_{XY}, together with p-values are given in Table 44. As it can be noticed from the table, there is a very high positive correlation between the actual effort and the number of distinct nouns and the number of distinct verbs in the problem descriptions. For the all the correlation coefficients the p-values are less than significance level α=0.05. Therefore, results can be considered statistically significant and one can be confident that the relationships between the effort and problem domain measures are not due to chance.

Table 44: Pearson's Correlation Coefficients and P-values for Effort

X	Y	r_{XY}	p-value
N	AE	0.965	2.309×10^{-8}
V	AE	0.969	1.09×10^{-8}

7.2. Regression Analysis

Scatterplots and residual plots of the problem domain measures versus the effort are given in Figure 31 through Figure 34. According to the residual plots in Figure 32 and Figure 34, there is no particular pattern and the variables are randomly scattered above and below the Residual=0 line.

Ryan-Joiner Normality Test Results are also given in Table 45. Since p-value >0.05 indicates normality, Ryan-Joiner test results approve the applicability of regression analysis to predict the effort in terms of the problem domain measures.

Table 45: Ryan-Joiner Normality Test Results

AE ~ N	AE ~ V
p-value > 0.100	p-value > 0.065

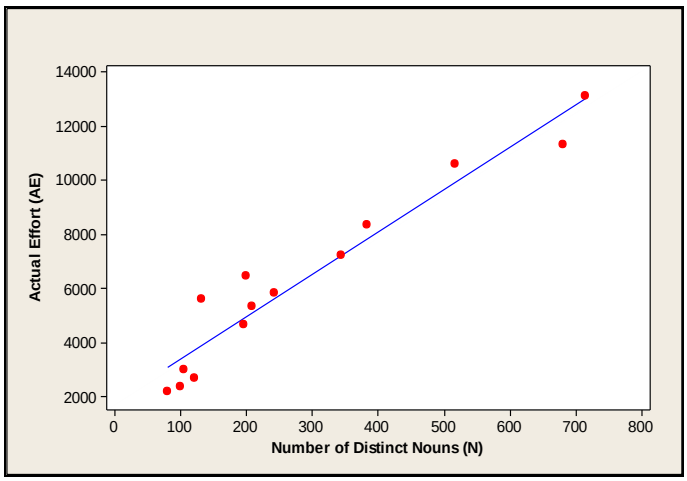

Figure 31: Scatterplot of Actual Effort vs. the Number of Distinct Nouns

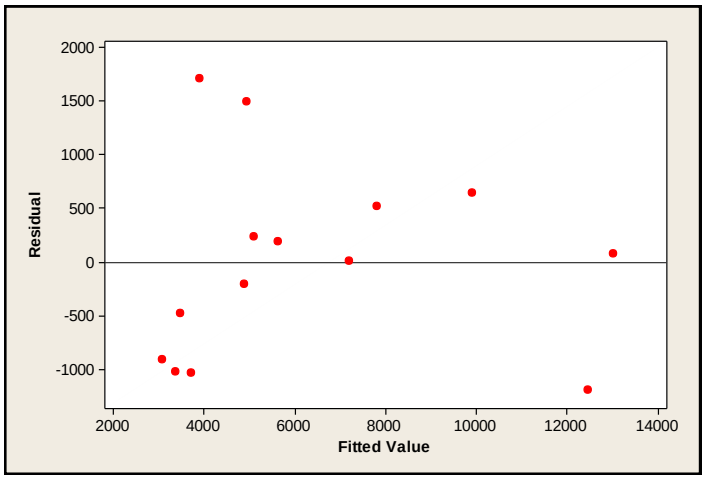

Figure 32: The Residuals vs. the Number of Distinct Nouns against the Actual Effort

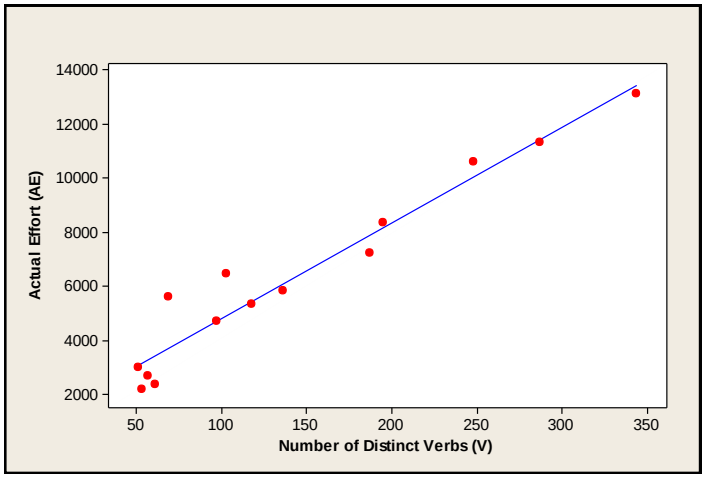

Figure 33: Scatterplot of Actual Effort vs. the Number of Distinct Verbs

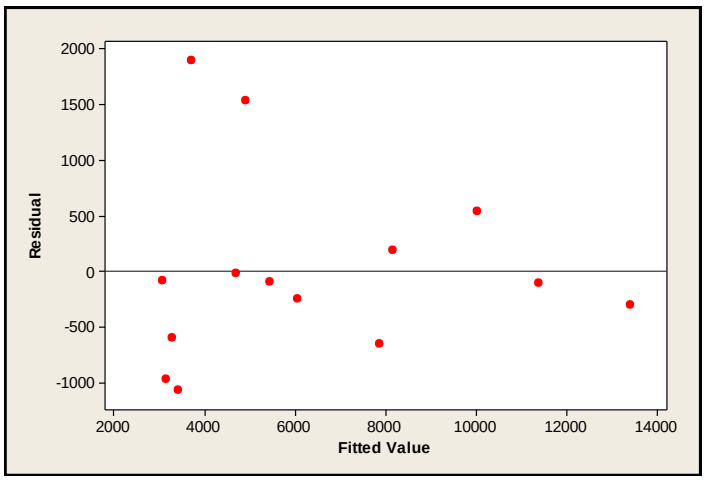

Figure 34: The Residuals vs. the Number of Distinct Verbs against the Actual Effort

By using the data given in Table 33 and Table 43, regression analysis is carried out for predicting the effort by using the number of distinct nouns and verbs in the problem domain descriptions. Equation 29 relates effort to the number of distinct nouns and Equation 30 relates to effort to the number of distinct verbs. In Equation 29 and in the rest of the thesis study, EE denotes the estimated effort.

$$EE = 1824.94 + 16.57\ N \qquad \text{(Equation 29)}$$

For the Equation 29, R^2=0.932 and the predictor variable N is significant as its p-value=2.31×10^{-8}. Similarly, in terms of the number of distinct verbs, the effort prediction equation is:

$$EE = 1269.67 + 35.28\ V \qquad \text{(Equation 30)}$$

For this model, R^2=0.94 and predictor variable V is significant as its p-value=1.1×10^{-8}.

7.3. Prediction Accuracy

In this section, the accuracy of the effort prediction models are evaluated in terms of MRE, MMRE, MdMRE, Pred(0.25) and Pred(0.30) obtained by LOOCV technique. Moreover, the prediction performances of the models are compared to the prediction performances of the effort prediction models based on UCP and CFP size measurements. The accuracy evaluation results for the effort prediction by using the number of distinct nouns (EE~N) and the effort prediction by using the number of distinct verbs (EE~V) are presented in Table 46.

Table 46: Prediction Accuracy for Effort

Projects	**EE ~ N**	**EE ~ V**
	MRE	**MRE**
Project Y_1	0.072	0.061
Project Y_2	0.010	0.039
Project Y_3	0.035	0.046
Project Y_4	0.068	0.025
Project Y_5	0.493	0.524
Project Y_6	0.500	0.521
Project Y_7	0.050	0.006
Project Y_8	0.254	0.261
Project Y_9	0.002	0.100
Project Y_10	0.048	0.020
Project Y_11	0.343	0.384
Project Y_12	0.184	0.031
Project Y_13	0.158	0.013
Project Y_14	0.442	0.260
Prediction Accuracy	**pred(0.25)=0.64 pred(0.30)=0.71 MMRE = 0.190 MdMRE=0.115 MSE = 1060809**	**pred(0.25)=0.64 pred(0.30)=0.79 MMRE = 0.164 MdMRE=0.054 MSE = 861927**

According to the results, from the MMRE and MdMRE point of view, EE~N and EE~V are found predictive. From the pred(e) point of view, EE~V is slightly better than EE~N. The Pred(e) values for effort estimation are not as good as the size estimation accuracy results but the prediction performances can still be found acceptable in many contexts.

In order to predict the effort by using UCP and CFP, the organization's historical records' regarding the projects completed in the past can be used. The company has recently considered UCP methodology for size measurement, and hence enough historical data is not available. Therefore, it could be meaningful to use 20 person hours per UCP productivity value as proposed by Karner (Karner, 1993). According to the company's historical data 27 person-hours per CFP is being used to predict the effort. However, to ensure fair comparison, instead of using the constant productivity values, the linear regression analysis is carried out to relate the effort to the UCP and CFP measurements and the LOOCV technique is applied to evaluate prediction accuracy. In Table 47, the prediction accuracy evaluations for the effort estimation based on CFP (EE~ CFP) and the effort estimation based on UCP (EE~UCP) are presented.

Table 47: Prediction Accuracy for Comparison

Projects	**EE ~ CFP**	**EE ~ UCP**
	MRE	**MRE**
Project Y_1	0.109	0.073
Project Y_2	0.202	0.067
Project Y_3	0.310	0.066
Project Y_4	0.146	0.131
Project Y_5	0.419	0.376
Project Y_6	0.114	0.250
Project Y_7	0.060	0.005
Project Y_8	0.371	0.019
Project Y_9	0.096	0.004
Project Y_10	0.133	0.133
Project Y_11	0.184	0.003
Project Y_12	0.447	0.079
Project Y_13	0.250	0.020
Project Y_14	1.220	0.359
Prediction Accuracy	**pred(0.25)=0.57** **pred(0.30)=0.64** **MMRE = 0.290** **MdMRE=0.193** **MSE = 3014125** **R^2=0.808**	**pred(0.25)=0.79** **pred(0.30)=0.86** **MMRE = 0.113** **MdMRE=0.070** **MSE = 376425** **R^2=0.976**

According to the results in terms of MMRE and MdMRE, EE~UCP is predictive and EE~ CFP is acceptable. EE~UCP gives better MMRE, pred(0.25) and pred(0.30) compared to EE~ CFP, EE~N and EE~V. However, MdMRE and MSE of EE~V are better than that of all others. Specifically, EE~N and EE~V are much better compared to EE~ CFP. Please note that, both UCP and CFP methodologies need some effort and some experience in measurement. However, our proposed methodology is based on counting nouns and verbs in problem domain descriptions and it is possible to save time for the measurement by using natural language processing tools.

CHAPTER 8

CONCLUSIONS

In this thesis, correlations between the problem domain measures and the solution domain measures are investigated. Based on the findings, linear regression analysis based size and effort estimation methodologies are proposed and prediction performances are evaluated.

In Section 8.1 summary of the thesis study and contributions achieved by the proposed methodologies are presented. Validity threats of the study are discussed in Section 8.2. The suggestions for future work are presented in Section 8.3.

8.1. Summary of the Thesis Study and Contributions

The key contributions of this thesis study are investigating the relation between the problem domain and solution domain measures for object oriented software to make predictions for size and effort.

Hence, the following research questions are answered during the research study:

RQ1: Are there any correlations between the problem domain measures and the solution domain measures for object oriented software?

RQ2: Can these correlations be utilized to estimate the software size and development effort?

In order to answer the first research question, 37 open source software projects have been analyzed. Problem domain descriptions are given as an input to the NLTK. Plural, duplicate and no meanings words are extracted with the help of NLTK's WordNet Lemmatizer module. Solution domain measures are collected by using a static code analyzer tool, Understand 2.0. Problem and solution domain measures correlations are investigated. The results revealed a high correlation between the problem and solution domain measures. In order to show the differences between the actual and estimated values of the dependent variable, scatterplots and residual plots are also given. Since the data points in a residual plot are randomly dispersed, a linear regression model is said to be appropriate for the models (Miles, 2014). Moreover, in order to check that the errors are well modeled by a normal distribution, normality analysis is done. Outlier detection is performed with using Cook's Distance technique. According to this technique three open source project, whose Cook's Distance is greater than 4/n, is treated as an outlier and removed from the data set.

To assess the prediction accuracy MRE, MMRE, MdMRE, Pred(25), Pred(30) and MSE are computed for the number of classes predicted and number of methods predicted. Acceptable results are observed according to utilized projects with respect to Hastings and Sajeev (2001) classification.

Therefore, a methodology based on linear regression is proposed to estimate size and effort required for object oriented software by using the measurements made on problem domain descriptions. In order to validate the proposed methodology to answer the second research question, twelve projects of a CMMI Level-3 certified defense industry company operating in Turkey and fourteen projects of another CMMI Level-3 certified defense industry company operating in Turkey have been analyzed. For the first company's project low level requirements and for the second company's project fully dressed use cases and resulting source codes are used for identifying problem and solution domain measures.

The analyses have revealed a high correlation between the problem and solution domain measures. Therefore, the number of software classes and the total number of methods in the software can be estimated by using the problem domain measures (part of speech tags) in the requirements artifacts.

The same approach is also applied in order to estimate the UCP and CFP size measures and the effort required for developing software. The prediction accuracy evaluation reveals that plausible predictions can be obtained by using the problem domain descriptions. As the results indicate, we can predict UCP and CFP size measurements earlier with using problem domain measures. Performance of the effort estimation methodology is also compared to that of the UCP and CFP based effort estimation methodologies. The results show that, for the projects evaluated, the proposed methodology provides accurate results compared to the UCP and CFP methodologies in effort estimation.

In this thesis, the proposed methodology is applied on problem domain descriptions like low level requirements and use cases. However, the methodology is conceptually applicable to any other requirements artifacts or pre-requirements level artifacts.

Since the counting processes are automated, time and effort needed for estimation is reduced considerably.

In this applicability of the methodology is also observed through open source projects and case studies, none of the methodologies in the literature include such applications.

Finally, the open source projects and case studies results showed that the number of software classes and the total number of methods in the software can be estimated by using the problem domain measures in the requirements artifacts.

8.2. Validity Threats

Due to the nature of the quantitative research, it is possible that some validity issues might arise. In the following limitations and validity threats of this thesis study are discussed.

Limited size of the datasets is one of the validity threats that should be considered. 14 completed projects of a company for Case Study #2 and 12 projects of a company for

Case Study #3 have been utilized for data regarding. The considered dataset in this study is still larger than the other published datasets (Ochodek et al., 2011). Nevertheless, the results may not be easily generalized without increasing the number of projects in the dataset.

Regression analysis in such a small dataset is another import validity threat. Before applying regression analysis, the correlation between the independent and dependent variables are checked and the statistical significance is confirmed via p-values. Then, goodness of fit of the models and statistical significance of the estimated parameters are confirmed. For this purpose, R^2 values, residuals analyses and Ryan-Joiner test are used.

Although the projects analyzed (Case Study #2, Case Study #3) are entirely distinct projects developed by different teams, they are in the same domain (i.e., defense industry) and developed by the same organization. Therefore, use case and requirements writing styles, architectures and coding styles for these projects are very similar to each other. Therefore, in addition to increasing the number of projects, various domains and development organizations is indispensable for generalization of the approach presented in this thesis.

Project selection bias for the open source projects are also one of the validity threats that should be considered. While selecting projects the constraints which are listed in Section 4.1 for open source software projects are considered attentively. The projects which are disproportionate to any those of constraints are not considered. In order to justify proposed methodology three different open source project domains are selected.

The other validity threat that must be considered is the errors and subjectivity involved in counting nouns and verbs. In order to minimize the risk of error and to ensure the consistency, the same person identified all nouns and verbs in the use cases manually for Case Study#3. For open source projects and Case Study#2 nouns and verbs are identified automatically by using NLTK which is well known and widely used natural language tool in studies (Bird et al., 2008). Reliability of the tools that are used in counting the number of classes and methods is another important issue that should be considered. A mature commercial tool, Understand 2.0, is used in this study to collect solution domain size measures like in similar studies such as (Zhou et al., 2014).

Reliability of effort data collection and CFP and UCP measurements are the other important issues. The company providing the projects is a CMMI Level-3 certified one and it has defined processes. There is a systematic effort data collection process defined in the company and the effort data presented in this thesis can be considered reliable. CFP measurements are also done by the company professionals. Company has been using COSMIC FFP methodology for several years and the professionals are highly experienced in measurements. The company has recently decided to consider UCP size measurement methodology for the new projects. However, the measurements have been performed by a person experienced in UCP size measurement.

Minitab tool is used throughout the study for correlation and regression analyses. As a crosscheck, SPSS (Statistical Package for the Social Sciences) tool is also used to compute some of the correlation coefficients and regression equations presented in the thesis and the same results are obtained.

8.3. Future Work

As a future work, the following improvement opportunities regarding to proposed methodology are identified:

- Investigation of the accuracy of proposed methodology by increasing the number of projects and extending idea to provide better estimations with using different languages (such as Turkish).

- Investigation of the architectural attributes in order to understand if they can be incorporated in order to improve the predictions performance of the study?

- Application of the proposed methodology on pre-requirements level artifacts in order to verify early size estimation capability.

- Consideration of the different domains and different companies for accuracy evaluations.

REFERENCES

Abbott, R.J., Program design by informal English descriptions. Communications of the ACM, 26, pp. 882–894, 1983.

Abirami, S., Shankari, G., Akshaya, S., Stihika, M., Conceptual Modeling of Non-Functional requirements from Natural Language Text, Smart Innovation, Systems and Technologies, Volume 33, pp. 1-11, 2015.

Abran, A., Desharnais, J. M., Oligny, S., St-Pierre, D., Symons, C., COSMIC-FFP Measurement Manual, Version 2.1, The Common Software Measurement International Consortium, 2001.

Abran, A., Cuadrado Gallego, J.J., Software estimation models & economies of scale, 21st International Conference on Software Engineering and Knowledge Engineering, SEKE'2009, pp. 625-630, 2009.

Abran, A., Software Metrics and Software Metrology, Wiley-IEEE Computer Society Press, ISBN: 978-0-470-59720-0, 2010.

Ahmed, M.A., Ahmad, I., AlGhamdi, J.S., Probabilistic size proxy for software effort prediction: A framework, Information and Software Technology, Volume:55, pp. 241-251, 2013.

Albrecht, A., Measuring Application Development Productivity, Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83-92, 1979.

Al-Safadi, L.A.E., Natural language processing for conceptual modeling. International Journal of Digital Content Technology and its Applications 3 (3), pp. 47–59, 2009.

Anderson, J., Branch, E., Luedtke, T., Carson, S., Falconi, J., Janda, R., Parametric Estimating Handbook: Reinvention Laboratory, DOD, NASA, 1999.

Antoniol, G., Lokan, C., Caldiera, G., Fiutem, R., A function point-like measure for object-oriented software, Empirical Software Engineering, Volume 4, pp. 263-287, 1999.

Antoniol, G., Fiutem, R., Lokan, C., Object-oriented function points: an empirical validation, Empirical Software Engineering, Volume 8, pp. 225-254, 2003.

Ayyıldız, T.E., Koçyiğit, A., Kara, A., Use Case Point (UCP) Methodology for Software Effort Estimation, 9th International Conference on Electronics, Computer and Computation, Ankara, Turkey, 2012.

Azzeh, M. and Nassif, A.B., Fuzzy Model Tree for Early Effort Estimation, 12th International Conference on Machine Learning and Applications (ICMLA), Volume 2, pp. 117-121, 2013.

Bajwa, I.S., Samad, A., Mumtaz, S., Object oriented software modeling using NLP based knowledge extraction. European Journal of Scientific Research 35 (1), pp. 22–33, 2009.

Banker, R., Kauffman, R.J., Wright, C., Zweig, D., Automating Output Size and Reuse Metrics in a Repository Based Computer Aided Software Engineering (CASE) Environment, IEEE Transactions on Software Engineering, Vol.20, No.3, pp. 169- 187, 1994.

Baroni, A. L. and e Abreu, F. B., A Formal Library for Aiding Metrics Extraction, 4th International Wokshop on OO Rengineering, 2003.

Bieman, J. M., Metric Development for Object Oriented Software, Software Measurement, pp. 75-92, 1996.

Bird, S., Klein, E., Loper, E., Baldridge, J., Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics, Columbus, Ohio, USA, June 2008.

Boehm, B., Software engineering economics: Prentice-Hall Englewood Cliffs, NJ, 1981.

Boehm, B., Abts, C., Horowitz, E., Madachy, R., COCOMO II Model Definition Manual: Center for Software Engineering, USC, 2000.

Bollen, A.K. and Jackman, R.W., Regression diagnostics: An expository treatment of outliers and influential cases, in Fox, John; and Long, J. Scott (eds.); Modern Methods of Data Analysis, Newbury Park, CA, Sage, pp. 257-291, 1990.

Booch, G., Object-oriented development,IEEE Transactions on Software Engineering, 12(2), pp. 211–221, 1986.

Booch, G., Object-Oriented Analysis and Design with Applications, 2nd Edition, Hardcover, Addison-Wesley Professional, ISBN-13: 978-0805353402, 1993.

Brook, Q., Lean six sigma minitab: The Complete Toolbox Guide for all Lean Six Sigma Practitioners, 3rd ed, OPEX Resources Ltd, 2010.

Bryant, B.R., Object-Oriented Natural Language Requirements Specification, In the Proceedings of ACSC 2000, The 23rd Australasian Computer Science Conference, Australia, 2000.

Burg, J., Linguistic Instruments in Requirements Engineering, IOS Press, 1997.

Caroll, E.R., Estimating Software based on Use Case Points, OOPSLA‟05, ACM, 2005.

Chatzigeorgiou, A., Mathematical Assessment of Object-Oriented Design Quality, IEEE Transactions on Software Engineering, Vol. 29, No. 11, 2003.

Chen, P.P., English sentence structure and entity-relationship diagrams, Information Science, Vol.1, No.1, Elsevier, pp. 127-149, 1983.

Chidamber, S. R. and Kemerer, C. F., A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering, Vol. 20, No. 6, 1994.

Cohn, M., Estimating with Use Case Points, Methods & Tools, Global knowledge source for software development professionals, Volume 13, No. 3, 2005.

Conte, S.D., Dunsmore, H.E., Shen, V.Y., Software Effort Estimation and Productivity, Advances in Computers, Vol. 24, pp. 1-60, 1985.

Conte, S.D., Dunsmore, H.E., Shen, V.Y., Software Engineering Metrics and Models, Benjamin-Cummings Publishing Co., Inc. Redwood City, CA, USA ©1986, ISBN:0-8053-2162-4, 1986.

Cook, R.D., Detection of Influential Observations in Linear Regression, Technometrics (American Statistical Association), Vol. 19, Issue 1, pp. 15–18, 1977.

Costagliola, G., Ferrucci, F., Tortora, G., Vitiello, G., A metric for the size estimation of object oriented graphical user interfaces, International Journal of Software Engineering and Knowledge Engineering, Vol.10, No. 5, pp. 581–603, 2000.

Creswell, J. W., Research design: Qualitative, quantitative, and mixed methods approaches: Sage Publications, Fourth Edition, Inc, 2013.

Del Bianco, V.D. and Lavazza, L., An Empirical assessment of function point-like object-oriented metrics, Proceedings of the 11th International Software Metrics Symposium, pp. 10-40, 2005.

Demirörs, O. and Gencel, C., Conceptual Association of Functional Size Measurement Methods, Software, IEEE, Vol. 26, Issue. 3, pp. 71-78, 2009.

Dennis, A., Wixom, B.H., Tegarden, F., Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, 2nd Edition, John Wiley and Sons, 2009.

DeSanto, C., Totoro, M., Moscartelli, R., Introduction to statistics 9th Edition, Pearson, ISBN: 055876830X, 2010.

Early & Quick, Early & Quick Function Points for IFPUG methods Version 3.1 Reference Manual 1.1, April 2012.

Elbendak, M., Vickers, P., Rossiter, N., Parsed use case descriptions as a basis for object-oriented class model generation, The Journal of Systems and Software, Vol. 84, pp. 1209-1223, 2011.

Feng, C., Wang, H., Lu, N., Tu, X. M., Log transformation: application and interpretation in biomedical research. Statist. Med., Vol. 32, Issue 2, pp. 230–239. doi: 10.1002/sim.5486, 2013.

Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, I., A simulation study of the model evaluation criterion MMRE, IEEE Transactions on Software Engineering, Vol. 29, Issue 11, pp. 985-995, 2003.

Giganto, R. and Smith, T., Derivation of Classes from Use Cases Automatically Generated by a Three Level Sentence Processing Algorithm, 3rd International Conference on Systems, IEEE, 2008.

Gomez, F., Segami, C., Delaune, C., A system for the semiautomatic generation of E-R models from natural language specifications, Data & Knowledge Engineering, Vol. 29, No. 1, pp. 57–81, 1999.

Harmain, H.M. and Gaizauskas, R., CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis, Automated Software Engineering, Vol. 10, pp. 157–181, 2003.

Harrison, R., Counsell, S., Nithi, R., An Overview of Object-Oriented Design Metrics, Proceedings of the 8th International Workshop on Software Technology and Engineering Practice (STEP'97), 1997.

Hastings, T.E. and Sajeev, A.S.M., A vector based approach to software size measurement and effort estimation, IEEE Transactions on Software Engineering, Vol, 24, Issue 4, pp. 337-350, 2001.

Herićko, M. and Živković, A., The size and effort estimates in iterative development, Information and Software Technology, Vol. 50, pp. 772-781, 2008.

Herr, N. and Cunningham, J.B., Hands-on Chemistry Activities with Real-life Applications, John Wiley & Sons Inc., New York, USA, 1999.

Hussain, I., Kosseim, L., Ormandjieva, O., Approximation of COSMIC functional size to support early effort estimation in Agile, Data&Knowledge Engineering, Vol. 85, pp. 2-14, 2013.

Ihaka, R. and Gentleman, R., R: A Language for Data Analysis and Graphics, Journal of Computational and Graphical Statistics, Vol. 5, Issue 3, pp. 299-314, 1996.

ISO/IEC 20968: Software engineering – Mk II function point analysis – counting practices manual, 2002.

ISO/IEC 20926: Software engineering – IFPUG 4.2 unadjusted functional size measurement method – counting practices manual, 2004.

ISO/IEC 24570: Software engineering – NESMA functional size measurement method version 2.1 – definitions and counting guidelines for the application of function points analysis, 2005.

Jacobson, I., Booch, G., Rumbaugh, J., The Unified Software Development Process, 1st edition, Addison-Wesley Professional, ISBN-13: 978-0201571691, 1999.

Jain, R., The art of computer systems performance analysis, in: Techniques for Experimental Design, Measurement, Simulation and Modeling, John Wiley and Sons, Inc, 1991.

Jones, T.C., A Short History of Function Points and Feature Points, Software Productivity Research Inc., USA, 1987.

Jørgensen, M., Experience with the Accuracy of Software Maintenance Task Effort Prediction Models, IEEE Transactions on Software Engineering, Vol. 21, Issue 8, pp. 674-681, 1995.

Jørgensen, M., Indahl, U., Sjoberg, D., Software effort estimation by analogy and regression toward the mean. The Journal of Systems & Software, Vol. 68, No. 3, pp. 253-262, 2003.

Jørgensen, M., Top-down and bottom-up expert estimation of software development effort, Information and Software Technology, Vol. 46, No. 1, pp. 3-16, 2004.

Jørgensen, M., A Critique of How We Measure and Interpret the Accuracy of Software Development Effort Estimation, 1st International Workshop on Software Productivity Analysis and Cost Estimation, pp. 15-22, 2007.

Kandpal, M. and Kandpal, A., Critical Analysis of Traditional Size Estimation Metrics for Object Oriented Programming, International Journal of Computer Applications, Vol. 58, No. 13, 2012.

Karner, G., Metrics for objectory. Diploma thesis, University of Linkoping, Sweden. No. LiTH-IDA-EX-9344, 21, 1993.

Kitchenham, B.A., Pickard, L.M., MacDonell S.G., Shepperd, M.J., What accuracy statistics really measure, IEEE Proceedings Software, Vol. 148, No. 3, pp. 81-85, 2001.

Laird, L.M. and Brennan, M.C., Software Measurement and Estimation: A Practical Approach, John Wiley and Sons Inc., Hoboken, New Jersey, ISBN: 978-0-471-67622-5, 2006.

Larman, C., Applying UML and Patterns: An Introduction to object-oriented analysis and design and the unified process, 2nd ed., Prentice Hall PTR, Upper Saddle River, NJ, 2002.

Lavazza, L.A. and Liu, G., A Report on Using Simplified Function Point Measurement Processes, The Seventh International Conference on Software Engineering Advances (ICSEA 2012), pp. 367-372, 2012.

Lee, B.S., Automated Conversion from Requirements Documentation to an Object- Oriented Formal Specification Language, In the Proceedings of SAC, Spain, 2002.

Li, W. and Henry, S., Object-Oriented Metrics that Predict Maintainability, Journal of Systems and Software, Vol. 23, 1993.

Li, K., Dewar, R.G., Pooley, R.J., Computer Assisted and Customer Oriented Requirements Elicitation, In the Proceedings of the 13th IEEE International Conference on Requirements Engineering, 2005.

Loper, E. and Bird, S., NLTK: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics-Vol. 1, pp. 63-70, 2002.

Lorenz, M. and Kidd, J., Object-Oriented Software Metrics, Prentice Hall, 1994.

Meli, R., Early and Extended Function Point: A New Method for Function Points Estimation, IFPUG-Fall Conference, Scottsdale, Arizona, USA, September 15-19, 1997a.

Meli, R., Early Function Points: A New Estimation Method for Software Projects, ESCOM 97, Berlin, 1997b.

Meli, R., Functional And Technical Software Measurement: Conflict or Integration?, Software Measurement Conference (FESMA'00), 2000.

Meziane, F. and Vadera, S., A Comparison of Computer Science and Software Engineering Programmes in English Universities, Proceedings of the 17th IEEE International Conference on Software Engineering Education and Training, (CSEE&T 2004), Norfolk, Virginia, USA. pp. 65-70, 2004.

Mich, L., NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA, Natural Language Engineering Vol. 2, No. 2, pp. 161–187, 1996.

Mich, L. and Garigliano, R., NL-OOPS: a requirements analysis tool based on natural language processing. In: Proceedings of Third International Conference on Data Mining Methods and Databases for Engineering , Southampton, UK, WIT Press, pp. 321–330, 2002.

Miles, J., Residual Plot, Wiley StatsRef: Statistics Reference Online, DOI: 10.1002/9781118445112.stat06619, John Wiley & Sons, Ltd., 2014.

Misic, V.B. and Tesic,, D.N., Estimation of effort and complexity:an object-oriented case study, The Journal of Systems and Software, Vol. 41, Issue 2, pp. 133-143, 1998.

Mohagheghi, P., Anda, B., Conradi, R., Effort Estimation of Use Cases for Incremental Large-Scale Software Development. ICSE 05 May 15-21, Copyright ACM 1-58113-963-2.05.0005, 2005.

Ochodek, M., Nawrocki, K., Kwarciak, K., Simplifying effort estimation based on use case points, Information and Software Technology, Vol. 53, pp. 200-213, 2011.

Ouwerkek, J. and Abran, A., An Evaluation of the Design of Use Case Points (UCP), Proceedings of the International Conference on Software Process and Product Measurement, Spain, 2006.

Özkan, B., Türetken, O., Demirörs, O., Software functional size: For cost estimation and more, Software Process Improvement Communications in Computer and Information Science, Vol. 16, pp. 59-69, 2008.

Perez-Gonzalez, H.G. and Kalita, J.K., Automatically generating object models from natural language analysis. In: OOPSLA '02: Companion of the 17th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages and Applications , ACM, pp. 86–87, 2002.

Picard, R.R. and Cook, R.D., Cross Validation of Regression Models, Journal of the American Statistical Association, Vol. 79, No. 387, pp. 575-583, 1984.

Poel, G., Towards a Size Measurement Framework for Object Oriented Specifications, Proc. Of the FESMA'98, Antwerp, Belgium, May 6-8, pp. 379-394, 1998.

Ren, X. and Dai, Y., A New Method to Estimate Software Size, International Asia Conference on Industrial Engineering and Management Innovation (IEMI2012) Proceedings, pp. 631-638, 2013.

Ribu, K., Estimating Object-Oriented Software Projects with Use Cases, Master of Science Thesis, University of Oslo, Department of Informatics, 2001.

Ronchetti , M., Succi , G., Pedrycz , W., Russo, B., Early estimation of software size in object-oriented environments a case study in a CMM level 3 software firm, Information Sciences Vol. 176, pp. 475–489, 2006.

RTCA DO-178-B, Software Consideratons in Airborne Systems and Equipment Certification, Technical Report RCTA Paper No. 548-92/SC167-177, RCTA, 1140 Connecticuit Avenue, Wasgington D.C., July 1992.

Rumbaugh, J., Blaha, M., Lorensen, W., Eddy, F., Premerlani, W., Object Oriented Modeling and Design, 1st edition, Prentice-Hall, ISBN-13: 978-0136298410, 1990.

Ryan, T.A. and Joiner, B.L., Normal Probability Plots and Tests for Normality, Technical Report, Statistics Department, The Pennsylvania State University, 1976.

Saeki, M., Horai, H., Toyama, K., Uematsu, N., Enomoto, H., Specification framework based on natural language. In Proceedings of the 4th international Workshop on Software Specification and Design, IEEE, pp. 87–94, 1987.

Santillo, L., Conte, M., Meli, R., Early & Quick function point: sizing more with less, Software Metrics, 11th IEEE International Symposium, Como, pp.19-22, 2005.

Schneider, G. and Winter, J.P., Applying Use Cases: A Practical Guide, Addison Wesley, 1998.

Sharma, A. K., Kalia, A., Singh, H., Taxonomy of Metrics for Assessing Software Quality, International Journal of Engineering Research and Technology (IJERT), Vol. 1, Issue 06, 2012a.

Sharma, A. K., Kalia, A., Singh, H., Metrics Identification for Measuring Object Oriented Software Quality, International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Vol. 2, Issue 5, 2012b.

Shepperd, M. and Cartwright, M., An Empirical Investigation of Object Oriented Software System, Technical Report No. TR 97/01, Dept. of Computing, Bournemouth University, UK, 1997.

Shepperd, M., Cartwright, M., Kadoda, G., On building prediction systems for software engineers, Empirical Software Engineering, Vol. 5, Issue 3, pp.175-782, 2000.

Stone, M., Cross Validatory Choice and Assesment of Statistical Predictions, Journal of the Royal Statistical Society, Series B, Vol. 36, Issue 2, pp. 111-147, 1974.

Tate, G. and Verner, J., Software costing in practice, The Economics of Information and Software, R. Veryard, Butterworth-Heinemann, pp. 101-126, 1990.

Tegarden, D. P., Sheetz, S.D., Monarchi, D. E., Effectiveness of Traditional Metrics for Object Oriented Systems, Proceedings 24th Hawaii International Conference on System Sciences, Vol. 4, pp. 359-368, 1992.

Teologlou, G., Measuring Object Oriented Software with Predictive Object Points, Shaker Publishing, ISBN 90-423-0075-2, 1999.

Thakur, S.J. and Gupta, A., Automatic generation of sequence diagram from use case specification, Proceedings of the 7th India Software Engineering Conference (ISEC'14), Article No. 20, 2014.

The COSMIC Functional Size Measurement Method Version 3.0.1 Measurement Manual, 2009.

Tripathy, A., Agrawal, A., Rath, S.K., Requirement Analysis using Natural Language Processing, Fifth International Conference on Advances in Computer Engineering (ACE 2014), Kochi, India, 2014.

Tucker, A. and Boehm, B., Point/Counterpoint: On the Balance between Theory and Practice/Software Engineering Is a Value-Based Contact Sport, IEEE Software, Vol. 19, pp. 94-97, 2002.

Understand user guide and reference manual, Scientific Toolworks, Inc., V 2.0, http://www.math.ntu.edu.tw/~wwang/cola_lab/knowledge/download/understand/understand_2p0.pdf, 2008.

Ungan, E., A functional software measurement approach bridging the gap between problem and solution domains, PhD thesis, Informatics Institute, Department of Information Systems, Middle East Technical University (METU), Turkey, 2013.

Vidhu Bhala, R.V. and Abirami, S., Conceptual modeling of natural language functional requirements, The Journal of Systems and Software, Vol. 88, pp. 25-41, 2014.

Živković, A., Rozman, I., Herićko, M., Automated software size estimation based on function points using UML models, Information and Software Technology, Vol. 47, Issue 13, pp. 881-890, 2005.

Zhou, N. and Zhou, X., Automatic Acquisition of Linguistic Patterns for Conceptual Modeling, Drexel University, 2004.

Zhou, Y., Yang, Y., Xu, B., Leung, H., Zhou, X., Source code size estimation approaches for object oriented systems from UML class diagrams: A comparative study, Information and Software Technology, Vol. 56, pp. 220- 237, 2014.

APPENDIX A

PYTHON CODE and R SCRIPTS

In this thesis, in order to identify the number of distinct nouns and the number of distinct verbs Python programming language is used. The used source code is given below.

```
import os
import nltk
import re
filelist=['Libre',
          'kforge','TeamLab','ganttproject','treeio','plandora',
          'ProjectLibre','ProjectNet','Scrinch','onepoint','TaskJuggler',
          'SonarQube','Freeplane','OFBiz','Adonthell','Exult','LinCity',
          'Enigma','Nuvie','BattleCity','Rigs','BZFlag','FreeOrion',
          'Wesnoth','Planeshift','Torcs','Lierox','CrackAttack',
          'Xournal','TaskWarrior','Chandler','nevernote','GloboNote',
          'Rachota','iteraplan','Todomoo','OpenGroupware','FreeMind']

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet as wn
wnl = WordNetLemmatizer()

def print_stats(list,fo,tagstr,text,v_n):
    i=0
    fo.write(tagstr+"\n")
    fo.write("-" * len(tagstr)+"\n")
    max_len=8;
    for item in list:
        lemmatized_item=wnl.lemmatize(item)
        if len(lemmatized_item)>max_len:
            max_len=len(lemmatized_item)
        if v_n=="NOUN":
            if len(wn.synsets(lemmatized_item, wn.NOUN))!=0:
                fo.write("%s" % lemmatized_item)
                item_num=text.count(item)
                item_freq=100*item_num/len(text.split())
                num_of_tabs=int(max_len/8)-int(len(lemmatized_item)/8)+1
                fo.write("\t"*num_of_tabs)
                fo.write(str(item_num)+"\t"+"%"+str(item_freq)+"\n")
                i=i+1
        elif v_n=="VERB":
            if len(wn.synsets(lemmatized_item, wn.VERB))!=0:
                fo.write("%s" % lemmatized_item)
                item_num=text.count(item)
                item_freq=100*item_num/len(text.split())
                num_of_tabs=int(max_len/8)-int(len(lemmatized_item)/8)+1
                fo.write("\t"*num_of_tabs)
                fo.write(str(item_num)+"\t"+"%"+str(item_freq)+"\n")
                i=i+1

    fo.write("." * 15+"\n")
    fo.write(tagstr+" stats:"+"\n")
    fo.write("-" * len(tagstr+" stats:")+"\n")
    fo.write("found:"+str(len(list))+"\n")
    fo.write("removed:"+str(len(list)-i)+"\n")
    fo.write("final:"+str(i)+"\n")
    fo.write("\n")
```

```
    print(tagstr)
    print ("-" * len(tagstr))
    print("found:"+str(len(list)))
    print("removed:"+str(len(list)-i))
    print("final:"+str(i))
    print()
    return i

def analyze_file(file):
    #noun tags
    nouns = set()#nn + nns
    #verb tags
    verbs = set() # vb + vbd + vbn + vbp + vbz
    fin=open('C:/tln/'+file+'.txt',encoding="iso-8859-1")
    a=fin.read()
    sentences = re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s', a)
    for stuff in sentences:
        tokens = nltk.word_tokenize(stuff)
        tags = nltk.pos_tag(tokens)
        for tag in tags:
            if tag[1]=='NN' or tag[1]=='NNS':
                nouns.add(wnl.lemmatize(tag[0].lower()))
            elif tag[1]=='VB' or tag[1]=='VBD' or tag[1]=='VBN' or
tag[1]=='VBP' or tag[1]=='VBZ':
                verbs.add(wnl.lemmatize(tag[0].lower(), 'v'))

    #write noun stats to file
    fo=open('C:/tln/results/'+file+'_wordnet_noun_stats'+'.txt','w')
    print(file+'(wordnet_stats)')

    #nouns stats
    list=nouns
    nouns_final=print_stats(sorted(list),fo,"nouns",a,"NOUN")

    fo.close()

    #write verb stats to file
    fo=open('C:/tln/results/'+file+'_wordnet_verb_stats'+'.txt','w')

    #verbs stats
    list=sorted(verbs)
    verbs_final=print_stats(list,fo,"verbs",a,"VERB")

    fo.close()

    return [nouns_final, verbs_final]

fo=open('C:/tln/results.txt','w')
fo.write("\t"+"nouns"+"\t"+"verbs"+"\n")
for file_item in filelist:
    listx=analyze_file(file_item)
    fo.write(file_item)
    for item in listx:
        fo.write("\t")
        fo.write("%s" % item)
    fo.write("\n")
fo.close()
```

For the accuracy evaluations R programming language is used. The used R scripts are given below:

```
library(DAAG)
setwd("C:/XXXX")
fn="XXX.txt"
  f <- file(fn)
  d <- read.table(f,header=TRUE)
  fitCN <- lm(C~V, data=d)
  cvCN <- cv.lm(df=d,fitCN,m=length(d$C))
  summary(fitCN)

result <- function(a,p){
  mre<-abs(a-p)/a
  mmre <- mean(mre)
  mdmre <- median(mre)
  pred30 <- sum(mre<=0.3)/length(mre)
  pred25 <- sum(mre<=0.25)/length(mre)
  list(mre=mre, mmre=mmre, mdmre=mdmre, pred30=pred30, pred25=pred25)
}
print("Class-Noun Results")
result(d$C,cvCN$Predicted)
```

```
library(DAAG)
setwd("C:/XXXX")
fn="XXX.txt"
  f <- file(fn)
  d <- read.table(f,header=TRUE)
  fitMV <- lm(M~V,data=d)
  cvMV <- cv.lm(df=d,fitMV,m=length(d$M))
  summary(fitMV)
result <- function(a,p){
  mre<-abs(a-p)/a
  mmre <- mean(mre)
  mdmre <- median(mre)
  pred30 <- sum(mre<=0.3)/length(mre)
  pred25 <- sum(mre<=0.25)/length(mre)
  list(mre=mre, mmre=mmre, mdmre=mdmre, pred30=pred30, pred25=pred25)
}
print("Method-Verb Results")
result(d$M,cvMV$Predicted)
```

CURRICULUM VITAE

PERSONAL INFORMATION

Tülin Erçelebi Ayyıldız was born in Denizli Turkey in 1981. She received her bachelor degree from Computer Engineering in Çankaya University in 2005. She received her M.Sc. degree from Computer Engineering in Graduate School of Natural and Applied Sciences of Hacettepe University in 2008.

Her research interests include size and effort estimation measures, software project management and software engineering. You can contact her at tulinercelebi@gmail.com

WORK EXPERIENCE

Company **:** Başkent University, Ankara
Department **:** Computer Engineering
Position **:** Research Assistant
Duration **:** September 2008, ……..

Company **:** Hacettepe University, Ankara
Department **:** Computer Engineering
Position **:** Research Assistant
Duration **:** December 2006 **–** July 2008

EDUCATION

PhD. Degree (2009-2015)

Information Systems Department /Middle East Technical University

MSc. Degree (2005-2008)

Computer Engineering Department /Hacettepe University

Bachelor's Degree (2000-2005)

Computer Engineering, Çankaya University

PUBLICATIONS

Ayyıldız, T.E. and Koçyiğit, A., Size&effort estimation based on correlations between problem&solution domain metrics for object oriented software, International Journal of Software Engineering and Knowledge Engineering (IJSEKE), 2015. (Submitted in May 2015, currently under second revision)

Ayyıldız, T.E. and Koçyiğit, A., Correlations Between Problem Domain and Solution Domain Size Measures for Open Source Software, 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2014), doi: 10.1109/SEAA.2014.11, pp: 81-84, IEEE, 2014.

Ayyıldız, T.E. and Koçyiğit, A., An Early Software Effort Estimation Method Based on Use Cases and Conceptual Classes, Journal of Software (JSW), Volume: 9, No: 8, doi:10.4304/jsw.9.8.2169-2173, ISSN 1796-217X, pp: 2169-2173, 2014.

Ayyıldız, T.E., Koçyiğit, A., Peker, D., Comparison of Three Software Effort Estimation Methodologies with Case Study, Global Journal on Technology, Volume 4, No: 2, pp 257-262, 2013.

Ayyıldız, T.E., Koçyiğit, A., Kara, A., Use Case Point (UCP) Methodology for Software Effort Estimation, 9th International Conference on Electronics, Computer and Computation, 01-03 November 2012, Ankara, TURKEY, 2012.

Ayyıldız, T.E. ve Koçyiğit, A., Yazılım Geliştirmede Kullanım Durumu Puanı ile Efor Tahmini, 28. Ulusal Bilişim Kurultayı, Bilişim 2011, 26-29 Ekim 2011, Ankara, TÜRKİYE, 2011.

Ayyıldız, T.E., Akıllı Ajan (Intelligent Agent) Benzetiminin Yüksek Düzeyli Mimari (HLA- High Level Architecture) ile Olan Uygunluğunun İncelenmesi, 31. Ulusal Yöneylem Araştırması ve Endüstri Mühendisliği Kongresi (YAEM) 2011, Sakarya Üniversitesi, Sakarya, TÜRKİYE, 2011.

Ayyıldız, T.E., Yüksek Düzeyli Mimarinin (HLA- High Level Architecture) Hava Tahmin Benzetimi için Uygunluğunun İncelenmesi, Engineering Sciences, eJournal of New World Sciences Academy, 2011, Volume: 6, Number: 1, Article Number: 1A0150, Series : 1A, ISSN : 1308-7231, pp: 286-295, 2010.

Oğul, H., Beyan, Ç., Eren Özsoy, Ö., Yıldız, K., **Ayyıldız, T.E.**, Sönmez, B., MicroRNA target recognition from compositional features of aligned microRNA mRNA dublexes, International Symposium on Innovations in Intelligent Systems and Applications, Kayseri, TURKEY, 2010.

INTERESTS

Music (folk song listener), travelling (abroad), sports (pilates and swimming), and cinema (science fiction).