OPTIMAL UTILIZATION OF HISTORICAL DATA SETS FOR THE CONSTRUCTION OF SOFTWARE COST PREDICTION MODELS

QIN LIU

A thesis submitted in partial fulfilment
of the requirements of the
University of Northumbria at Newcastle
for the degree of
Doctor of Philosophy

Research undertaken in the
School of Computing, Engineering and Information Sciences

December 2005

Abstract

The accurate prediction of software development cost at early stage of development life-cycle may have a vital economic impact and provide fundamental information for management decision making. However, it is not well understood in practice how to optimally utilize historical software project data for the construction of cost predictions. This is because the analysis of historical data sets for software cost estimation leads to many practical difficulties. In addition, there has been little research done to prove the benefits.

To overcome these limitations, this research proposes a preliminary data analysis framework, which is an extension of Maxwell's study. The proposed framework is based on a set of statistical analysis methods such as correlation analysis, stepwise ANOVA, univariate analysis, etc. and provides a formal basis for the erection of cost prediction models from historical data sets. The proposed framework is empirically evaluated against commonly used prediction methods, namely Ordinary Least-Square Regression (OLS), Robust Regression (RR), Classification and Regression Trees (CART), K-Nearest Neighbour (KNN), and is also applied to both heterogeneous and homogeneous data sets. Formal statistical significance testing was performed for the comparisons.

The results from the comparative evaluation suggest that the proposed preliminary data analysis framework is capable to construct more accurate prediction models for all selected prediction techniques. The framework processed predictor variables are statistic significant, at 95% confidence level for both parametric techniques (OLS and RR) and one non-parametric technique (CART). Both the heterogeneous data set and homogenous data set benefit from the application of the proposed framework for improving project effort prediction accuracy. The homogeneous data set is more effective after being processed by the framework. Overall, the evaluation results demonstrate that the proposed framework has an excellent applicability.

Further research could focus on two main purposes: First, improve the applicability by integrating missing data techniques such as listwise deletion (LD), mean imputation (MI), etc., for handling missing values in historical data sets. Second, apply benchmarking to enable comparisons, i.e. allowing companies to compare themselves with respect to their productivity or quality.

Contents

List of Figures

List of Tables

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

This research has been supported and funded by the Faculty of Technology at Southampton Solent University and the School of Computing, Engineering and Information Sciences at Northumbria University. I thank them all for their confidence in me and am grateful for them for providing me an excellent work environment during the past years.

I am deeply indebted to my supervisors Dr. Robert Mintram, Dr. William Henderson, and Professor Margaret Ross. They gave me the great opportunity to write this thesis and provided me with highly beneficial support in various aspects. They extraordinarily advised and motivated me with their enthusiasm, creativity, and patience during all the in-depth technical discussions we had.

I appreciate the support and the fruitful discussions with Dr. Barbara Kitchenham (Keele University, U.K.), Dr. Katrina Maxwell (DATAMAX Consultancy, France), Dr. John Moses (Sunderland University, U.K.) and Dr. David Giddings (Northumbria University, U.K.). I also received invaluable advice from Dr. Paul Vickers and Dr. Joe Faith at the mid-point examination of this thesis.

I have furthermore to thank Dr. Katrina Maxwell and Mr. Peter Hill (ISBSG Research, Australia) for providing invaluable data sets that were used for the evaluation of the research.

I am very grateful for the love and support of my parents Mr. KeCheng Liu and Mrs. Rong Yang, my brother HaiMing Liu and all my dear friends.

Especially, I would like to give my special thanks to my husband Tian Xu whose patient love enabled me to complete this work.

Chapter 1

Introduction

1.1 Introduction

The research described in this thesis provides a formal basis for building software project effort prediction models based on historical data sets. It defines a formal procedure to identify what project attributes and how much project data from a historical data set should be used to build project effort prediction models.

Most researchers [88, 26] define software cost prediction as the process of predicting the amount of effort required to build a (new) software system. Fenton [88] proposed that the term "cost estimation" and "effort estimation" are sometimes used interchangeably. This thesis follows the assumption that these two terms are synonymous.

This chapter introduces and briefly summarizes current practices regarding the construction of data driven[1] prediction models for software cost estimation. It describes the problems addressed in this thesis and their importance. It sketches the proposed solutions and research contributions, and outlines the structure of this thesis.

1.2 Cost Estimation

The growth in demand for high quality software and the increased investment in software projects indicates that software development is becoming one of the key markets worldwide. As reported by Sauer and Cuthbertson [270] in 2003, half of all capital investment today is in information and communication technologies. Accurate prediction of software development costs may have a vital economic impact. In contrast to their importance, software projects cost estimation have a poor reputation for performance. In fact, according to Mukhopadhyay [220] and Lederer [183], some 60% of large projects significantly overrun their estimates (with an error percentage that can vary from 100% to 200%) and 15% of the software projects are never completed due to the grossly inaccurate estimation of development effort. The US-based Standish Group's 1995 CHAOS Report [106] reinforces the opinion that project performance deficit is widespread, cost and schedule overruns high, and the delivered functionality is less than expected. The Standish Group research shows that 31.1% of projects will be canceled before they ever get completed. Further, on average, only 16.2% of 175,000 software projects are completed on time and within their budget. Further results indicate that 52.7% of

[1] "data-driven" models in this thesis refers to models derived from historical data sets.

projects will cost 189% of their original estimates. Sauer and Cuthbertson [270], in their survey - "The State of IT Project Management in the UK 2002-2003", based on data collected from 1,500 practising IT project managers in UK between 2002 and 2003, reported that average overrun on budget is 18%, average overrun on schedule is 23% and average underachievement on scope/functionality is 7%. Although these figures suggest that improvements are occurring, the number of projects achieving all their targets remains low at 16%. The BEST-Pro estimation survey [214] in Norway which was conducted in 2003 involving 18 companies and 52 projects also indicated a majority of software project (76%) had encountered effort overruns. The average magnitude of these overruns was 41%. With the variability of software characteristics and the continuing emergence of new technologies, it is becoming harder and harder to accurately estimate software development cost.

1.3 Historical Data Sets

One issue that has been the focus of research in software engineering is to construct cost prediction models based on historical data sets. Such data sets are collections of attributes, including cost, of completed projects. Figure 1.1 illustrates the general process of cost prediction from historical data sets. The steps are:

Step 1: Define a conceptual prediction model (specific model), for instance, define predictor and predicted variables. For example, predict *effort* by project *size*, where *effort* is the predicted variable, and the *size* is the predictor variable.

Step 2: Select a prediction technique for example, Ordinary Least Square Regression (OLS), Case Based Reasoning (CBR) or Artificial Neural Networks (ANN).

Step 3: Set up the prediction model from the historic data set. For example, ANN and CBR models are trained on a selection of records from the data set. This step can also include a validation phase where the established model is tested against a further selection of records from the data set.

Step 4: Using the established model, predict the new projects' *effort* from known predictor variables.

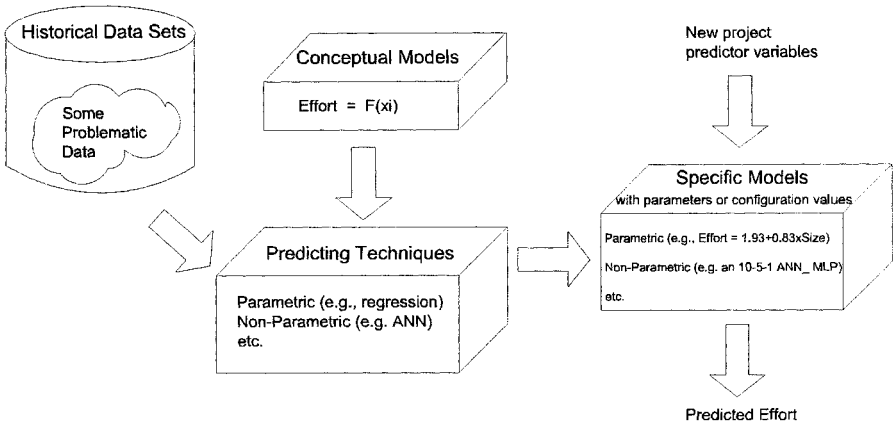

Figure 1.1: Constructing Effort Prediction Models

Although many prediction methods have been developed, e.g. mathematical, statistical, rule based, machine learning and hybrid methods, using such methods can often be complex and may not always lead to accurate estimates [174, 264, 160]. Almost all of these cost estimation techniques share a common attribute: require historical data of completed projects [173, 24, 14, 17, 286, 269, 289]. However, when building most of these models, researchers frequently encounter a common problem [313, 114, 262], i.e, the inconsistency and inadequacy of the historical data sets. The variety of the data sources, e.g. government, industry, research institution, simulated etc and the characteristics of the data from these sources, e.g. outliers, collinearity, number of factors, number of samples etc caused difficulties of replication of predictions and comparison of models.

Preliminary data analysis has not often been performed in previous researches, which means there is not a formal procedure to define what project predictors and how much project data from a historical data set should be used when building a cost model.

1.4 Preliminary Data Analysis

To overcome the limitations presented in section 1.3 for building cost models from historical data sets, this thesis proposes a preliminary data analysis framework, that is an extension of Maxwell's analysis of variance [203]. This proposed framework defines a set of statistical analysis techniques to remove extreme project samples, statistical not significant and inter-related project attributes. This provides a formal basis for constructing cost prediction models from historical data sets. The statistical analysis methods within the proposed framework include correlation analysis, stepwise regression, stepwise ANOVA, univariate analysis, etc. In addition, the proposed framework provides an insight into the analyzed data sets in a graphical view, e.g., histogram plot, two-dimension correlation plot, normality Q-Q plot, etc. Figure 1.2 demonstrates the role of the proposed framework in the process of constructing cost prediction models. Statistical methods and tools have been used in software engineering data analysis for

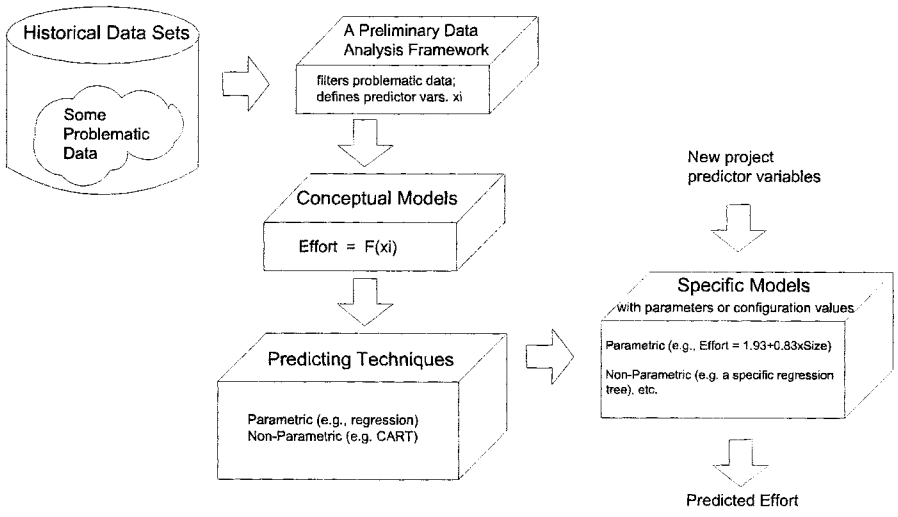

Figure 1.2: The Role of the Preliminary Data Analysis Framework

project cost estimation since the 1970's. From the application of basic scatter diagrams, and statistical trend lines [251], to Analysis of Variance (ANOVA) [202, 168, 203], researchers have endeavored to investigate techniques for analyzing software engineering data. Some systematic methods have been introduced, for example Kitchenham's "A procedure for analyzing unbalanced data sets" [168] based on Bailey and Basili's work

[10] and Maxwell's [203] "Applied statistics for software managers" based on previous work for European Space, Military and Industrial Applications [202]. These methods have not been evaluated by their application to a standard and extensive historical data sets, nor have they been applied in more than one cost estimation technique.

According to the above, replication of results and rationalisation of performance across different data sets is a scientifically desirable objective. This thesis provides a comprehensive, systematic, and repeatable evaluation of the proposed framework to achieve the objective. The framework processed data set is compared to other data sets, including raw unprocessed data, raw data with the most significant predictor variable, semi processed data i.e raw data without outliers, and the framework processed data set using only the most significant predictor variable. These comparisons are repeated for a comprehensive set of commonly used estimation methods, namely: Ordinary Least-Square Regression (OLS), Robust Regression (RR), K-Nearest Neighbour (KNN) and Classification and Regression Trees (CART). The proposed evaluation method is consistently applied to two representative cost data sets from different application domains (ISBSG R9 data set) and from the same application domain (Bank63 data set).

Homogeneous data, either company or domain specific are believed to provide a better basis for accurate estimates [133]. However, practitioners are often faced with the lack of explicit cost and resource data collected from past projects, because data collection is an expensive, time-consuming process for individual organizations [88, 285]. The collaboration of organizations to form heterogeneous data sets (multi-organizational) provides the possibility for reduced data collection costs, faster data accumulation and shared information benefits [120, 182]. The issue of heterogeneous initiatives providing means to better exploit data for cost estimation, especially for companies without locally collected or application domain specific data available, is considered within this thesis.

1.5 Research Issues

In the context of the previous discussion, the following sections elaborate in more detail the points outlined above. The issues addressed are:

- Selecting predictor variables in historical data sets.

- Empirical evaluation of prediction performance based on the selected predictor variables.
- Exploitation of heterogeneous and homogeneous data for cost estimation.

These problems, their proposed solutions and the chapters in this thesis where they are discussed are illustrated in Figure 1.3.

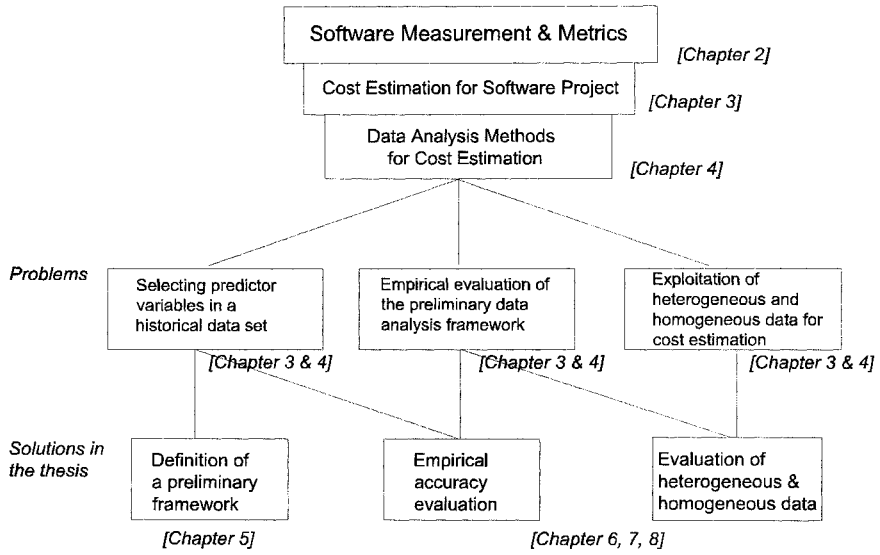

Figure 1.3: Problems and Solutions in the Thesis

1.5.1 Selecting Predictor Variables in a Historical Data Set

In the context of cost estimation methods there are issues that are not or only partly addressed by models derived from historical data sets. These issues are related to the constraints of software engineering data, discussed in further detail in section 4.2. Appropriately dealing with these constraints has an influence on the accuracy of the predictions obtained through the different methods and models. Several such constraints were recognized by Briand *et al* [41]:

- The problems inherent in making assumptions about the statistical distributions of predicted and predictor variables.

- The mixture of varied types of variables (e.g., continuous and discrete).
- The potential interdependencies of predictor variables.
- Problems associated with missing data values.

A paradigm within which these constraints can be addressed would represent a significant and positive contribution to the resolution of these difficulties.

This thesis proposes a preliminary data analysis framework, which is an extension of Maxwell's analysis of variance [203] procedure. The proposed framework extends Maxwell's analysis procedure in ways that:

- Formally define reduction rules that facilitate:
  - criteria for removing outliers;
  - rules for incorporating inter-correlated variables;
  - clarification of statistical methods that handle differently scaled variables e.g. ratio, ordinal, and nominal.
- Further analyze potential inter-correlated predictor variables, e.g. adopts univariate analysis to identify collinearity between categorical predictor variables.

1.5.2 Empirical Evaluation of Prediction Performance

There are a few software project data analysis procedures [168, 133, 48, 203] are proposed in the literature. These are described in Chapter 4. Although the details of these procedures are dissimilar, they have the same underlying principle. That is, they all use statistical methods, in particular the Analysis of Variance Approach (ANOVA), to identify the significant predictor variables. The significance of a predictor variable depends on how much the percentage variation in the predicted variable (i.e., project effort) is explained by the predictor variable.

Since there are few instances of systematic data analysis procedures there is even less evidence of empirical evaluation of these procedures. Kitchenham's [168] forward pass residual analysis method was evaluated only by application to Boehm's COCOMO 81 data set. Jeffery *et al.* [133]'s data analysis procedure is not formally defined. They did

not clarify the specific predictor variables, nor the method to select those variables in building their prediction model. Although they evaluated the accuracy of the estimates by comparing ordinary least squares regression and an analogy-based tool based on two data sets, they did not compared the accuracy of the estimates between the raw data sets and analyzed data sets. In addition, one of the data sets (ISBSG R5 data) was not the latest version when the research was conducted, and the other one (Megatec data) can not be accessed by other researchers. Above all, Jeffery *et al.*'s research provided an insufficient evaluation of their data analysis procedure. Similarly, Briand *et al.* [48]'s research also focused on comparing the accuracy of the estimates between data sets from different resources. Moreover, a few aspects of Briand *et al.*'s analysis procedure that should be further considered that the detection and removal of outliers is not addressed and the Principal Components Analysis is descriptive rather than analytic. Typically, Jeffery *et al.* and Briand *et al.*'s use preliminary data analysis on an informal and ad hoc basis rather than as an independent, intrinsic procedure.

Compared with the above researches, Maxwell's analysis procedure is formally prescribed and well-defined. Maxwell's book "Applied Statistics for Software Managers" was the first complete guide to using statistical methods in software project management and process improvement. As the name implies, the book presents an easy-to-use methodology that enables project managers to obtain results without deep mathematical expertise. However, some definitions (e.g., outliers) are quite informal. It is significant however, that throughout the book, model evaluation and validation is limited to testing residuals based on a single data set.

To overcome the above limitations, the thesis provides a comprehensive accuracy evaluation. The predictive accuracy of the framework processed data is empirically evaluated by defining a systematic and repeatable evaluation method. The framework processed data set is compared to other data sets:

- The raw data set.
- The data set only consists of the most significant predictor variable and the predicted variable from the raw data set.
- The raw data set without outliers.
- The data set only consists of the most significant predictor variable and the predicted variable from the framework processed data set.

These comparisons are repeated for a comprehensive set of commonly used estimation methods, namely: Ordinary Least-Square Regression (OLS), Robust Regression (RR), K-Nearest Neighbour (KNN) and Classification and Regression Trees (CART).

1.5.3 Heterogeneous and Homogeneous Data

Historical data based cost models require a substantial amount of data. Company-specific data are believed to provide a better basis for accurate cost estimates [10, 133]. This is because they allow for a number of advantages, such as a better control of the data collection process, better understanding of the measurement methods and attributes used to characterize the projects etc [24]. However, usually not enough or only a small amount of company specific data are available from past similar projects to develop cost estimation models with appropriate accuracy. Multi-organizational databases, where organizations share project data collected in a consistent manner, can address the above mentioned issues.

To address these questions, the defined evaluation method in the last section is consistently applied to two representative cost data sets from different application domains (ISBSG R9 data set) and from the same application domain (Bank63 data set) respectively. Both of the data sets are in the public domain (can be accessed free). The ISBSG R9 data set is the latest version of the data collection from ISBSG in 2005. The Bank63 data is the same data set released and deployed in Maxwell's research.

1.6 Research Approach

This thesis identifies problems of building cost estimation models from historical data sets, and proposes a systematic preliminary data analysis framework to tackle the problems. The proposed framework provides a formal basis for construction of data-driven cost estimation models, and consequently improves the prediction accuracy of commonly used cost estimation models. A repeatable empirical method for evaluating the accuracy of the proposed framework is defined in this thesis. This evaluation method is build upon the approaches proposed by Basili [15], Shepperd *et al.* [277] and Kitchenham *et al.* [172] .

The following sections summarize how the proposed solution is empirically evaluated

and presents the corresponding research hypotheses.

1.6.1 Research Aims and Hypotheses

The aim of this research is:

To provide a formal basis for building effort prediction models based on a historical data set. It defines a formal process that identifies what project attributes and how many samples from a historical data set should be used in building effort prediction models.

As a consequence of this programme we erect the following hypotheses, the demonstration of which supports the contention that the programme has been successful. In other words, if the hypotheses are proved or demonstrated satisfactorily then we argue that the research aims are fulfilled.

Hypothesis 1 The predictions based on the data set yielded by a preliminary analysis performs as well as or better than these based on both raw data set or raw data set with the most significant predictor (e.g., size).

Hypothesis 2 The predictor variables extracted by the proposed framework are statistically significant (at 95% confidence level).

Hypothesis 3 Both the heterogeneous data sets and homogenous data sets benefit from the application of the proposed preliminary data analysis framework for improving project effort prediction accuracy.

1.6.2 Empirical Evaluation of the Preliminary Data Analysis Framework

The goal of this research is to define an easy to interpret [2] preliminary data analysis framework which provides efficient data for accurate software effort estimation. As illustrated in Figure 1.4, a historical data set for effort prediction is defined as a project vector set (PVS). This data set contains j project vectors (PV_j) and each vector

[2] "Easy to interpret" in this context means the use of graphical data representations.

consists of i predictor variables (x_i) and one predicted variable (Y). First, the proposed framework eliminates outliers and obtains a sub set of PVS as PSS which contains $m(m \leq j)$ project vectors (PV_m) and each vector consists of i factors (x_i). PSS is a row subset of PVS, we also call it the raw data set without outliers. Second, the proposed framework ranks the predictor variables by their statistical explanatory power [3] on project effort, i.e. the more the variable explains the variation of project effort, the more significant the variable is. The outcome is a column subset of PSS. This subset is defined as EPSS, which contains $m(m \leq j)$ project vectors (PV_m) and each vector consists of $k(k \leq i)$ factors (x_k). We call EPSS the (final) framework processed data set containing extracted predictor variables (or cost factors) and project samples for construction of cost prediction models. The predictive accuracy of the framework processed data is

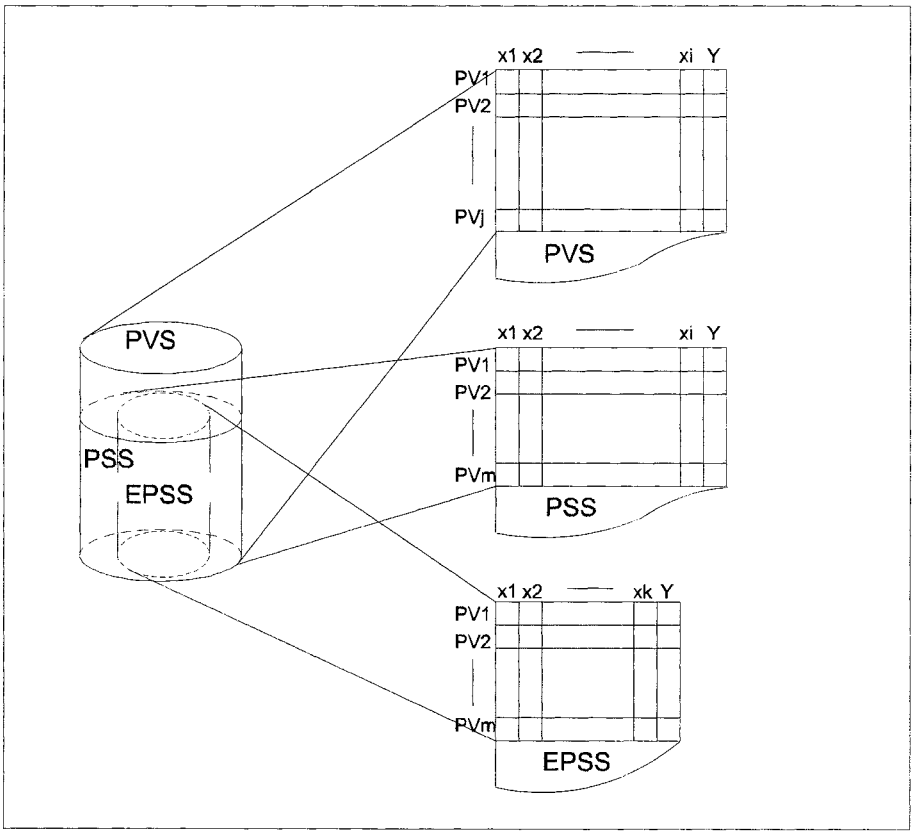

Figure 1.4: Extracting Optimal Subsets by The Proposed Framework

empirically evaluated by defining a systematic and repeatable evaluation method. The

[3] actually the variance

framework processed data set (EPSS) is compared to other data sets, including the raw data set (PVS), raw data set with the most significant predictor variable (PVS_{Size}), the raw data set without outliers (PSS) and the framework processed data set with the most significant predictor variable ($EPSS_{Size}$). These comparisons are repeated for a set of commonly used estimation methods.

Based on the high-level estimation model selection criteria in section 6.4, four cost estimation methods are selected for evaluating the preliminary analysis framework. The selected methods are Ordinary Least-Square Regression (OLS), Robust Regression (RR), K-Nearest Neighbour (KNN) and Classification and Regression Trees (CART). To define the best K of KNN, for each set of inputs 10 KNNs are built where $K = 1$ to 10, and the best result is selected for comparison. section 6.4 provides concrete configuration values for each selected method, that has the consequence of making all experiments in this study repeatable. The accuracy evaluation criteria are critically reviewed. The research adopts both classic criteria (i.e., BMRE, MMRE, MdMRE, Pred(l)) and more recently proposed criteria (i.e., MMER, MdMER). The detailed definition of these selected criteria are presented in section 6.5.

Two case studies are provided based on a multi-company data set (ISBSG R9 data) and a company-specific data set (Bank63 data). Both data sets are in the public-domain.

1.7 Research Contribution

This research establishes a preliminary data analysis framework for building effort prediction models based on historical data sets. It adopts and extends earlier work of other researchers, e.g. Kitchenham [168], and Maxwell [202, 203]. Moreover, the results obtained from experiments demonstrate that the preliminary data analysis proposed within this research improves the performance for selected prediction techniques in software cost estimation. Furthermore, it could be able to identify the significance of the software project metrics in estimating project effort.

The novelty of this research is that it provides a formal basis for building effort prediction models based on historical data sets for the first time and evaluates the predictive accuracy in an original way. The principal contributions made by this research are:

- Formally establish a statistically based preliminary data analysis framework for analyzing software project data.

- Provides a basis for the erection of cost prediction models at early stage of the project development by defining a preliminary analysis framework of historical data sets.

- Provides empirical evidence to support the accuracy and applicability of the above framework.

1.8 Organization

This document is divided into 4 major parts. The first provides the context in which the research is based and includes Chapters 1 to 4. The second addresses the major research aim relate to construction and evaluation of cost estimation models based on a historical data set. This includes Chapters 5 and 6. The third addresses the research aim directed toward empirically evaluation by two case studies, includes Chapter 7 and 8. The final part summarizes the conclusions and implications of this research and consists of Chapter 9 and 10. The following sections describe the contents of each chapter.

This thesis is organized as follows. Chapter 2 presents the needs and constraints in software measurement within the context of software engineering. Chapter 3 discusses common cost model construction approaches. Chapter 4 expounds the popular data analysis methods in software engineering. Chapters 3 and 4 also present the research motivations and gaps. Chapter 5 illustrates the proposed solutions and presents a systematic preliminary data analysis framework for building effort prediction models from a historical data set. Chapter 6 elaborates the detailed research hypotheses and empirical evaluation methods. Chapters 7 and 8 explain how this framework may ultimately be used based on both a heterogenous data set and a homogenous data set respectively. Experimental results of selected approaches are provided to evaluate the validation and applicability of the proposed framework in these two chapters. Chapter 9 discusses the research presented within this thesis and explains how the research aims have been met and draws the conclusions of the project. Chapter 10 describes future directions that could be explored using this research as a basis.

1.9 Tools

A free statistical software package R 2.1.0, which was developed at University of Minnesota [62], was used.

1.10 Summary

This chapter identified accurate software cost prediction as a critical concern for project success. Three primary issues have been highlighted:

1. The limitations of currently used data-driven estimation methods and the related difficulty selecting cost factors (predictor variables) for cost estimation.
2. Empirical evaluation of prediction performance based on selected variables.
3. Exploitation of heterogeneous and homogeneous data for cost estimation.

As a solution for the first issue, a Preliminary Analysis Framework for Cost Estimation has been proposed. A comprehensive approach to evaluate the proposed framework based on commonly used cost estimation methods has been illustrated, which addresses the second issue. This evaluation approach is also applied to both heterogeneous and homogeneous data sets, which addresses the third issue. Finally, the structure of the dissertation is outlined.

Part I

CONTEXT

Chapter 2

Measurement and Metrics

Not everything that counts can be counted.
Not everything that can be counted counts.

Albert Einstein

2.1 Introduction

This chapter presents the background of software measurement and metrics within the context of software engineering. It explains the definitions, a brief history, and current state of software metrics. This chapter highlights the role of *productivity* estimation in software measurement and quality control. The relationship between *Productivity* with *project effort* or *cost* is explained. *Productivity* estimation related metrics, both direct metrics (*project size* measures) and indirect metrics (*project effort*), are introduced in detail.

2.2 Software Measurement and Metrics

As argued by Fred S. Roberts [257] that "A major difference between a *well developed* science such as physics and some of the *less well developed* sciences such as psychology or sociology is the degree to which things are measured". It is an undisputed statement that measurement is crucial to the progress of all sciences. Scientific progress is made through observations and generalizations based on data and measurements, the derivation of theories as a result, and in turn the confirmation or refutation of theories via hypothesis testing based on further empirical data. Science and engineering can be neither effective nor practical without measurement.

Improving *quality* is the core of *Software Engineering* [81, 247]. It has also been well recognized in the quality community [76] that in order to improve software quality (products and processes), it is necessary to make meaningful measurement.

2.2.1 Definitions

Software Measurement provides approaches to quantification of quality aspects of software, related to the product, the process and the resources. There are many definitions about software measurement, such as [119], Hetzel defined *Software Measurement* as a "quantified observation on some attribute or aspect of the software product, process or project". As defined by Fenton [88] that "measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules". To further explain this

concept Fenton described measurement as capturing information about attributes of entities. Where an entity is an object or an event (e.g., a software development project) in the real world and an attribute is a feature or property of an entity (e.g., cost, size, duration of the project).

Software Metrics is a term that embraces many activities, all of which involve some degree of software measurement [88], such as cost and effort estimation, productivity measures and models, reliability models, structural and complexity metrics, capability-maturity assessment and so on. Grady and Caswell [101] defined a software metric as follows:

> A software metric is a standard way of measuring some attribute of the software process. Examples of these attributes are size, costs, defects, communications, difficulty and environment.

The terms software measurement and metrics are often used as synonyms [319]. In 1993, IEEE [284] defined a "metric" as synonymous with a "software quality metric" and a software quality metric as a function with input and output.

> Software quality metrics have software data as inputs and a single numerical values as output. the output is interpreted as the degree to which software possesses a given attribute that affects its quality.

It was stated by Horst Zuse in his book "A framework of software measurement" [319], that prediction of costs or effort is one of the major requirements of software measurement. Other important areas of software measurement are software size measurement [110, 3], complexity measurement[318], structure measurement and reliability measurement[88]. More recently, Object-oriented software measures were developed and defined. Zuse[319] reported in 1997 that over two hundred software measures for object oriented software were found.

2.2.2 History of Software Metrics

The invention of software measurement and metrics is strongly related to the evolution of software engineering. Motivated by the software crisis in 1960s, the revolutionary decade of software engineering was coming. This new discipline was not only to focus

on establishing theoretical foundations and tactical disciplines in developing software, but also initiating metrics of software.

Early Thoughts of Metrics: 1960s and earlier

In the 1960s software engineering literature, researchers like Dijkstra [78] made the observation that a piece of source code was a series of essentially mathematical statements that should be mathematically proved correct, or incorrect. Although, counting lines of code and errors were discussed on a very superficial level [119], it represented the earliest attempt at applying software measurement in management. By 1965, the "*unconditional jump*" was proposed as a "*major obstacle*" by many authors, such as Dijkstra [78], Knuth [176] and Yourdon [314]. In *"Structured Programming"* Dijkstra [78] not only coined the term "structured programming" but also emphasized the importance of error prevention, as opposed to error cure. Since then, considerable attention has been paid to the structure of computer programs. The cost estimation models Delphi [116] and Nelson's SDC [228] were introduced in 1966. In 1968, the earliest paper about software complexity was published by Rubey *et al.* [261]. In 1971, Akiyama [2] proposed a basic regression-based model for module defect density in terms of the module size measured in thousand Lines Of Code (KLOC). This is possibly the first attempt to use metrics for software productivity prediction.

There was some work on costs measurement [116, 228] and complexity measurement [261], but a little regarding any other aspects of software measurement, such as size, which could be sub-described as functionality, complexity, etc. [119], only a track of literature about the field and not a single prominent book on the subject. These crude software measurement theories had been scattered until 1976 the first book in the field was published, "Software Metrics" by Tom Gilb [97], followed a year later by Maurice Halstead's "Elements of Software Science" [110].

A Proliferation of Metrics: 1970s

The measurement of software size in terms of Lines Of Code (LOC)or thousand LOC (KLOC) has been frequently used since the early-1970's [2, 312]. By the mid-1970's, the obvious drawbacks of using the simple metrics of measure such as LOC as a surrogate measure for such different notions of program size such as functionality and complex-

ity, were recognized [88]. The magnitude of costs involved in software development and maintenance magnifies the need not only for software engineering methodologies, but also a scientific foundation to support programming standards and management decisions by measurement. As Curits [68] pointed out:

> Rigorous scientific procedures must be applied to studying the development of software systems if we are to transform programming into an engineering discipline. At the core of these procedures is the development of measurement techniques and the determination of cause effect relationships.

The need for more discriminating measures became especially urgent with the increasingly large software systems attempted and combined with the diversity of programming languages employed. Thus, the decade starting from the mid-1970's also saw an explosion of interest in software metrics application. For example, measures of software complexity, which were pioneered by the likes of Halstead [110] and McCabe [204]; and measure of size, such as *Function Point* pioneered by Albrecht [3] and later by Symons [298]. Software attributes such as design coupling and cohesion and computer system performance analysis and modelling also emerged and helped shape the literature of the period, but they had little effect on the practicing engineer or typical software process. Large number of publications, especially some dedicated books [110, 97, 27], were devoted to software metrics. Linda M. Ott's survey [233] gives detailed lists of metrics proposed during the period.

Criticisms and Refinement of Metrics: 1980s

During the 1980s the field matured significantly. Research continued on software complexity metrics including software science metrics. Prominent measures of the period included measures of extended function points and test coverage. As Ott [233] pointed out, although there was a flurry of activity in the field during 1970s to early-1980s, there was also much skepticism about the area of software metrics and much criticism of the methodologies used. Some of the criticisms were focused at specific metrics or metric systems, such as software science [70, 64, 181], and cyclomatic complexity metrics [234, 70, 82]; some of the criticisms were focused on the statistical techniques used [216, 112]; and some were of a general nature [67, 68, 69, 70, 77, 21, 75]. Typical

of the comments about the state of affairs were those of Curits' [67]:

> While fascinating work has been performed, research in the field often lacks the direction provided by sufficient definitions of the constructs studies. In particular, there are too many metrics and too little explanatory theory or empirical evaluations...

Researchers became aware of the demand for empirical work, which concerned with evaluating the effectiveness of specific software metrics and models. For example, Albrecht and Gaffney [4] examined the relationships between function points, software science metrics and KLOC. Data from several development sites were used to validate the hypotheses. The application of function points in 24 projects from a data-processing organization of a large financial institution were analyzed by Behrens [20]; Basili *et al.* [16] analyzed software science metrics, cyclomatic complexity, and source lines of code using data from several Fortran projects developed within a production environment.

In addition to work on specific metrics and models, much recent work has focused on systematical activities. Grady and Caswell [101, 100] published the first and extensive experience reports of a company wide software metrics programme. Basili, Rombach and colleagues [17] presented the Goal-Question Metric (GQM) programme. By borrowing some simple ideas from the Total Quality Management field, Basili and his colleagues [109] proposed a simple scheme for ensuring that metrics activities were always goal-driven and pointed out that a metrics programme established without clear and specific goals and objectives is almost certainly doomed to fail.

Because prediction provides software managers with a forecast of the quality and productivity (effort/size) of the operational software early in the development process so that corrective actions can be taken. The desire of predicting project cost in terms of effort stimulated the development of cost prediction models and automation tools. For instance, the Jensen cost model [137] was proposed in 1983. ESTIMACS, which was originally developed by Howard Rubin in the late 1970s as Quest (Quick Estimation System), was subsequently integrated into the Management and Computer Services (MACS) line of products [263]. Putnam [252, 251] developed a constraint model called Software LIfe Model (SLIM) to be applied to projects exceeding 70,000 lines of code. Putnam's model assumes that effort for software projects is distributed similarly to

a collection of Rayleigh curves [239]. In 1981 Barry Boehm published his pioneering book "Software Engineering Economics" [24], which helped popularize the emerging of project-estimating models and tools.

Other evidence of progress and interest is reflected in the series of fine books that appeared in the late 1980s including Grady and Caswell's "*Software Metrics: Establishing a Company-wide Program*" [101]. Musa *et al.*'s "*Software Reliability: Measurement, Prediction, Application*" [222] and Humphrey's "*Managing the Software Process*" [125].

2.2.3 Recent Work on Metrics

During 1990s, many companies have started to deploy object-oriented (OO) technology in their software development efforts. OO measures and metrics have been proposed and have been increased discussions in recent years. One special characteristic of the decade is the emphasis on process measurement and software capability evaluation. Another significant feature that began in the 1980s and continues unabated today is the proliferation of automated measurement tools and support that has become available. Measurement activities have evolved to total quality management and customer satisfaction programmes.

Object Oriented Metrics

A very early investigation of OO-Measures can be found in Rocacher's "*Metrics Definitions for Smalltalk*" [258]. Lorenz [194] proposed eleven metrics as OO design metrics in 1993. In early 1993, the IBM Object Oriented Technology Council (OOTC) [65] published a white paper on OO metrics with recommendations to the product division. The first book about object-oriented software metrics were published by Lorenz and Kidd [195], which expanded their metrics work by publishing a suite of recommended OO metrics with multiple metrics for each of the following categories: method size, method internals, class size, class inheritance, methods inheritance, class internals, and class externals. Almost all of the OOTC metrics were included in Lorenz and Kidd's comprehensive suite.

In 1994 Chidamber and Kemerer [58] proposed six OO design and complexity metrics, which later became the commonly referred to CK metrics suite. Rosenberg *et al.* [259] discussed metrics used for OO projects at the NASA Software Assurance

Technology Center(SATC). They recommend the six CK metrics plus three traditional metrics, namely, cyclomatic complexity, lines of code, and comment percentage based on SATC experience. In 1997 Chidamber *et al.* [58] applied the CK metrics suite to three financial application software programs and assessed the usefulness from a managerial perspective. The metrics values from these three systems showed significant differences compare with Chidamber and Kemerer 's results [58]. Therefore, it seems that more empirical studies need to be accumulated before reliable threshold values of the CK metrics can be determined.

Briand and Wüst [48] discussed the impact of OO metrics such as coupling, cohesion and complexity on development *Cost* in terms of *Effort*. They concluded that fairly accurate predictions of class effort can be made based on simple measures of the class interface size alone. However, more sophisticated coupling and cohesion measures do not help to improve predictions.

Considerable research and discussions on OO metrics have taken place in recent years, for example, Li and Henry [186], Henderson-Sellers [117], De Champeaux [57], Briand [38], Kemerer [161], Card and Scalzo [54], and Babsiya and Davis [8]. With regard to the direction of OO metrics research, there seems to be agreement that it is far more important to focus on empirical validation (or refutation) of the proposed metrics than to propose new ones.

Other Key Features

The 1990s opened with a flurry of object oriented measurement activity and interest. One special characteristic of the decade is the emphasis on process measurement and software capability evaluation. Triggered by Watts Humphrey's book [125] "*Managing the Software Process*" in 1989 and support by the Software Engineering Institute, process assessment and practices baselining and benchmarking have become hot topics. Total quality management and customer satisfaction programs have been mainly considered; the publications reflected that interest include Card and Galss's "*Measuring Software Design Quality*" [55], Capers Jones' "*Applied Software Measurement*" [143], Bob Grady's "*Practical Software Metrics for Project Management and Process Improvement*" [100], and Putnam and Myers' "*Measures for Excellence: Reliable Software on Time, Within budget*" [251].

Another characteristic of the decade is that consultants and software package vendors have found the measurement area to be fertile ground and start to offer a broad range of products and services to help companies launch or improve their measurement programs [119]. Most of the *cost estimation* models and automated tools were developed during this period. For instance, Analogy-Based Estimation and response tools called ANGEL (ANaloGy Estimation tool) [280, 279]. Evolved versions of COCOMO [26] and SLIM [250] continue contributing to the field. A knowledge-based software project estimating tool from Software Productivity Research (SPR), which is called Checkpoint, was developed from Jones' studies [143]. Wittig and Finnie [311] used SPQR/20 a software estimation tool to generate simulated software development project data. Details of these cost models will be presented in chapter 3.

2.2.4 Successes, Failures and New Directions of Software Metrics

Although hundreds of metrics have been proposed in the field, not all of them have survived. Much academic metrics research is inherently irrelevant to industrial needs, and much industrial metrics activity is poorly motivated [85]. Pfleeger *et al.* [237] observed that in many cases, highly theoretical software measurement results are never tested empirically, new metrics are defined but never used, and new theories are promulgated but never exercised and modified to fit reality. For example in the US the single biggest trigger for industrial metrics activity has been the Capability Maturity Model for Software (CMM) [125]. Fenton and Neil [85] summarized the successes and failures of software metrics and concluded that only a few basic metrics are qualified as candidates for success. They are the enduring LOC metrics [88], Metrics relating to defect counts [98], McCabe's cyclomatic number [204], and Function Points [3, 298]. While such metrics can be considered as successful given their popularity, their limitations are well known, and misapplication is still common.

Based on testing a number of popular hypotheses about basic metrics, Fenton and Ohlsson [87] drew several conclusions of software defect prediction, such as complexity and/or size measures alone cannot provide accurate predictions of software defect, and traditional statistical (regression-based) methods are inappropriate for defects prediction. Fenton and other researchers believe applying Bayesian Belief Networks is by far the best solution for solving the above problems, and published a series of pa-

pers [86, 225, 226, 227] to support these findings.

2.3 Categories of Software Measurement and Metrics

Fenton [84] defined measurement as "the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules". There are various aspects, which have different attributes of a project that can be measured. Because software projects vary in many ways, making comparison at a detailed level is rather difficult, information is often required at a summary level and in the form of ratios, e.g. productivity and delivery. Treble and Douglas summarized a list of some of the attributes of a project that are candidates for measurement, as Table 2.1 reproduced from [301] shows. The follow-

Attributes	Potential Measurements
Product	The size of the system being constructed; The complexity of the system; The defects discovered; The changes requested and carried out.
Work	The effort expended; The elapsed time over which the effort is expended.
Money	Cost and benefits
People	Skills, knowledge and commitment of project management; Skills, knowledge and commitment of the technical staff; Skills, knowledge and commitment of the user staff.
Techniques and methods	Management approaches used; Development approaches used.
Environmental factors	Machine characteristics; Surroundings; Imposed constraints.

Table 2.1: Project Attributes and Potential Measurements

ing subsections provide some commonly cited categories of software measurement and metrics.

2.3.1 Direct and Indirect Measurement

Measurement can be conducted both directly or indirectly. It would be a direct measurement of an attribute, if an entity involves no other attribute or entity [84]. For example line of code, duration of development process or number of defects; Indirect measure is often useful in making visible the integrations between direct measurements, such as productivity (LOC/effort), module defect density (number of defects/module size), or test effectiveness ratio (number of items covered/total number of items).

2.3.2 Process, Products and Resources Measures

The most popular categories of software metrics are *Process, Products* and *Resources* measures [88]. Fenton defined *Process* as collections of software-related activities. *Product* as any artifacts, deliverable or documents that result from a process activity. *Resources* are entities required by a process activity. All of these measures have both *internal* attributes and *external* attributes. Fenton [88] also provided a precise list of key components of software measurement as in Table 2.2 [88].

2.3.3 Jones' Software Measurement Categories

In his book [146] Capers Jones states software projects are influenced by as many as 250 different factors that can affect schedule, cost, quality and user satisfaction, and listed 6 sets of factors, each with 6 topics. Jones proposed these 36 items of data are important enough that they should be captured during all assessment and benchmark studies. Jones [146] introduced factors influencing software measurement in six majority categories: Classification factors, Project-specific factors, Technology factors, Sociological factors, Ergonomic factors, International factors. He also declared that all 36 of the factors presented are important, and each one can exert as much as $\pm 20\%$ impact on software schedules, costs or quality when there are extreme variances between best and worst casing situation.

ENTITIES	ATTRIBUTES	
Product	*Internal*	*External*
Specifications	size, reuse, modularity, redundancy, functionality, syntactic correctness,...	comprehensibility,maintainability,...
Design	size, reuse, modularity, coupling, cohesiveness, functionality,..	quality, complexity, maintainability,...
Code	size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness,...	reliability, usability, maintainability,...
Test data	size, coverage level,...	quality,...
...	...	...
Process	*Internal*	*External*
Constructing specification	time, effort, number of requirements changes,...	quality, cost, stability,...
Detailed design	time, effort, number of specification faults found,...	cost, cost-effectiveness,...
Testing	time, effort, number of coding faults found,...	cost, cost-effectiveness, stability,...
...	...	...
Resources	*Internal*	*External*
Personnel	age, price,...	productivity, experience, intelligence,...
Teams	size, communication level, structuredness,...	productivity, quality
Software	price, size,...	usability, reliability,...
Hardware	price, speed, memory size,...	reliability,...
Offices	size, temperature, light,...	comfort, quality,...
...	...	...

Table 2.2: Components of Software Measurement

2.3.4 Rubin's Software Measurement Categories

With the world wide focus on improvement to the software process in the 1990s, there is clearly a need for an understanding of its impact on software engineering productivity and quality. Regarding the above, Rubin [262] suggested ten basic categories of metrics that provide a reasonable universe for measuring a software engineering organization, its projects, and applications. These ten categories are: productivity metrics, quality metrics, delivery metrics, penetration metrics, work profile metrics, demand metrics, technology assimilation metrics, work distribution metrics, capability metrics and business oriented metrics.

Because of the variety of software metrics, we are not going to offer an exhaustive list, but only highlight metrics related to software productivity. As described in the preceding sections, the productivity measure is one of the most important aspect of software measurement [319], and is also closely relate to the topic of this thesis, since productivity estimation is an indirect estimation of software development cost in terms of effort. The next three sections focus on productivity, cost (in terms of effort), and size measures.

2.4 Software Productivity Measures

In economics, productivity is defined in a straightforward way. As defined by Oxford English Dictionary, 2nd edition:

> The rate of output per unit of input used especially in measuring capital growth, and in assessing the effective use of labor, materials and equipment.

The productivity measure most commonly used in software engineering is one of size of effort.

$$P = S/E \tag{2.1}$$

where P is *productivity*, S is *size*, which can be measured as LOC or FPs, and E is *effort* which is measured in person days or person months. The definition imply that *effort* and *size* are two basic measures of *productivity* measurement, and will be described in the next two subsections. As stated by Wrigley [313], the concept of software effort estimation and programmer productivity are inextricably linked.

Fenton [88] argued that the equation's simplicity hides the difficulty of measuring size and effort. As discussed in section 2.6, size could be measured in different ways (LOC, FPs, FFPs, etc.). Effort is not something that can be recorded as easily as a meter reading. For example, a "day" measure does not reflect whether the day consisted of 8, 12, or 16 hours; the contribution of a part time worker may be very different from 50% of a full-timer. Therefore, variations in size and effort estimation remain pragmatic problems of the definition. Moreover, the productivity equation views output only in terms of size, so that the real value of the output is ignored. We should consider productivity in light of the benefit we deliver, not just the size of the code. Although there have been some attempts at improving the definition of software *productivity*, for example consider measure input by more than *effort* or *time* (such as *tools and training*) that are consumed during the production process, the *size* over *effort* measure has continued to be used for many years.

The concepts of software effort estimation (in section 2.5) and programmer productivity are inextricably linked [313]. Implicit in all models of cost estimation are notions of factors which either increase or decrease software development effort. Programmer productivity gains are achieved by enhancing those factors which are believed to decrease effort and mitigating those that are believed to increase effort. Productivity however is a deceptively amorphous concept. In fields other than software engineering there is usually some reasonable metric for the output or yield from a process. Likewise, inputs can be measured. Economic productivity can be easily computed as yield/input. The problem we face in software development is that both the inputs and the outputs have not yet had a stable metric for comparison.

2.5 Software Cost/Effort Measures

When software engineers and project managers talk of "cost estimation", they usually mean predictions of the likely amount of effort, time and staffing levels required to build a software system. Because staff costs often dominate overall project cost, the terms "cost estimation" and "effort estimation" are sometimes used interchangeably.

Cost estimation are needed throughout the software life cycle. Preliminary estimates are required when bidding for a contract or determining whether a project is feasible in terms of a cost-benefit analysis. However, preliminary estimates are the

most difficult to obtain, and they are often the least accurate, because very little detail is known about the project or the product at an early stage.

Cost predictions support planning and resource allocation. The objective of making predictions is not just to provide an estimate, it is to provide a process by which accurate estimates can be generated.

There are two types [88] of models that have been used to estimate effort: *cost models* and *constraint models. Cost models* [24] provide direct estimates of effort or duration. Most cost models are based on empirical data reflecting factors that contribute to overall cost. By contrast, *constraint models* demonstrate the relationship over time between two or more parameters of effort, duration, or staffing level. The Rayleigh curve [229] is used as a constraint model in several commercial products, including Putnam's SLIM model [252]. A comprehensive description of cost models will be presented in categories within the following chapter.

2.6 Software Size Measures

Each product of software development is a physical entity, and can be described in terms of its size. The software size is the most important factor that affects the software cost. One of the purposes of measuring size is to use early product measures to predict subsequent product and process measures such as project effort.

Norman Fenton [88] suggested that software size can be described by three attributes: length, functionality and complexity. *Length* is the physical size of the product, such as Line of Code (LOC), and *functionality* measures the functions supplied by the product the user, relative metrics are different versions of Function Points (FPs) metrics. *Complexity* could be interpreted as problem complexity, algorithmic complexity, structural complexity or cognitive complexity. Halstead's "*software science*" [111] is one of the size measures related to complexity. This section describes some software size metrics used in practice.

2.6.1 Lines of Code (LOC)

Line of Code is the number of lines of the delivered source code of the software, excluding comments and blank lines and is commonly known as LOC [88]. Although LOC

is programming language dependent, it is the most widely used software size metric. Most models, will be presented in the following chapter, relate this measurement to the software cost. However, exact LOC can only be obtained after the project has completed. Estimating the code size of a program before it is actually built is almost as hard as estimating the cost of the program.

Although LOC is easy to collect and has been considered relatively objective compare with other size metrics, most users of the LOC metric have no idea at all of its subjectivity, randomness, and quirky deficiencies. The first complete analysis of the problems of LOC metrics was the study by Jones [140] in the IBM Systems Journal in 1978. In essence there are three serious deficiencies associated with Lines Of Code (LOC).

- There has never been a national or international standard for a line of code that encompasses all procedural languages.
- Software can be produced by such methods as program generators, spreadsheets, graphic icons, reusable modules of unknown size, and inheritance, wherein entities such as LOC are totally irrelevant.
- LOC metrics paradoxically move backwards as the level of the language gets higher, so that the most powerful and advanced languages appear to be less productive than the more primitive low level languages. That is due to an intrinsic defect in the line of code metrics. Some of the languages thus penalized include Ada, APL, C++, Objective C, SMALTALK and many more.

2.6.2 Software Science

Halstead proposed the code *length* and *volume* metrics [111], named as *Software Science*. Halstead's technique was an attempt to resolve the ambiguity and paradoxical nature of LOC by looking at the specific sub-elements of LOC, and divided code into two atomic units: the executable or command portion (which he termed "operators") and the data descriptive portion (which he termed "operands"). In software science, code length is used to measure the source code program *length* and is defined as:

$$N = N_1 + N_2 \tag{2.2}$$

Where N_1 is the total number of operator occurrences, and N_2 is the total number of operand occurrences.

Volume corresponds to the required storage space and is defined as:

$$V = N \times \log(n_1 + n_2) \tag{2.3}$$

where n_1 is the number of distinct operators, and n_2 is the number of distinct operands that appear in a program.

There have been some disagreements over the underlying theory that supports the software science approach [112, 275]. This measurement has received decreasing support in recent years.

2.6.3 Function Point

Alan Albrecht recognized the problem in size measurement in the 1970s, and developed a measurement based on the functionality of the program called Function Points (FPs) analysis [3] in 1979.The principle of function point analysis is that a system is decomposed into functional units [301] as the following:

1. Inputs: information entering the system
2. Outputs: information leaving the system
3. Enquiries: requests for instant access to information
4. Internal logical files: information held within the system
5. External interface files: information held by other systems that is used by the system being analyzed.

Each of these is analyzed and assigned a subjective *complexity* rating on a three-point ordinal scale: *simple*, *average* or *complex*. Then a weight is assigned to the item based on a standard Function Point Complexity Weights table. One example of this table can be found in Fenton's book [88]. There are 15 different varieties of the above five units, so we can compute the Unadjusted Function-point Counts (UFC) by multiplying

the number of functional units in a variety by the weight of the variety and summing over all fifteen:

$$UFC = \sum_{i=1}^{5} \times \sum_{j=1}^{3} \times N_{ij} \times W_{ij} \tag{2.4}$$

where N_{ij} and W_{ij} are respectively the number and weight of types of functional units i with complexity j. The weight W_{ij} are rated by past experiences and refer to the standard Function Point Complexity Weights table [88].

This initial function-point count is either directly used for cost estimation or is further modified by factors whose values depend on the overall complexity of the project. In the later situation, general systems characteristics are then analyzed to establish an adjustment factor. This will take into account the degree of distributed processing, the amount of reuse, the performance requirement, etc. Therefore to compute an Adjusted Function Point Count (FP), we need to multiply UFC by a *technical complexity factor*, called TCF. TCF involves the 14 contributing factors listed in a Components of the Technical Complexity Factor table which refers to Fenton's book [88]. The components in the table include Reusability, Installation ease, Online data entry, etc. Each component in the table is rated from 0 to 5, where 0 means the component is irrelevant, 3 means it is average and 5 means it is essential to the systems being built. The following formula combines the 14 ratings into a final technical complexity factor:

$$TCF = 0.65 + 0.01 \sum_{i=1}^{14} F_i \tag{2.5}$$

This factor varies from 0.65 (if each F_i is set to 0) to 1.35 (if each F_i is set to 5). The final calculation of Adjusted Function Point Count (FP) is [88, 301]:

$$FP = UFC \times TCF \tag{2.6}$$

The advantage of the function-point measurement is that it can be based on the system requirement specification in the early stage of software development. The UFC is also used for code-size estimation using the following linear formula:

$$LOC = a \times UFC + b \tag{2.7}$$

The parameters a, b can be obtained using linear regression and previously completed project data. The latest Function Point Counting Practices Manual is maintained by

the IFPUG (International Function Point Users Group) [129]. A brief history and evolution of FPs has been been detailed in Jones book [143].

The invention of this method was a considerable leap forward. It gives a measure that can be explained to users. Treble and Douglas [301] described that it measures the system in terms of its *information processing mass* and listed some problems of FPs. For examples, the measure is not continuous that a maximum size for any component can be reached but not exceeded would give problems with large, complex systems; FPs are difficult for the measurement of subsystems and sum the results to give a value for the system as a whole, etc.

As the number of users of function points continued to grow, more and more variations started to appear. The International Function Point Users Group (IFPUG) was created in 1986. The IFPUG counting practices generally following the IBM 1984 standards, although a number of variations and extensions have occurred. The major variations for counting Function Points are discussed in this thesis, for example, DeMarco's Function Metric,the British Mark II Function Point Method and IFPUG Standard Counting.

DeMarco's Function Metric

DeMarco's "Bang" Functional Metric was published in 1982 [74]. The name come from the vernacular phrase "*getting more bang from the buck*". The bang metric and the function point metric are somewhat different in form and substance, but are aimed at the same problem. The bang metric was the first attempt to apply functional metrics to the domain of systems and scientific software. It is a considerable superset of the Albrecht function point metric [3], and it contains such elements as data tokens and state transitions which are normally associated with the more complex forms of systems software such as operating systems and telecommunication systems.

Mark II FPs Metric

In 1988, Symons [298] published a description of his Mark II Function Point metric. This method aimed to reduce the subjectivity in dealing with files by measuring entities and relations among entities. Symons also added six factors to supplement 14 influential factors [3, 88] cited by Albrecht and IBM. These factors are:

1. Software with major systems software interfaces
2. Software with very high security considerations
3. Software providing direct access for third parties
4. Software with special documentation requirements
5. Software needing special user training
6. Software needing special hardware

When considering the same application, the resulting function point totals differ between the IBM and Mark II by in cases more than 30% [143], with the Mark II technique usually generating the larger totals. As more factors are added, the function point total will tend to change as more results are available for reasons that appear unjustified under the assumptions of the original IBM assertions. Unlike the IBM method, which is in the public domain, the Mark II method is a proprietary technique of the consulting group of Nolan, Norton & company, which is an obstacle to it gaining widespread acceptance.

Similar with calculating Albrecht's function point, to calculate the Mark II function point we need to calculate the Unadjusted Function Point count (UFP), and calculate the Technical Complexity Adjustment (TCA). However, the coefficient of TCA formula is 0.005 and the total number of influence factors is increased to 19 (or 20 if the site-specific one is defined) [301] rather than 14. Each factor is still scored between 0 and 5 degrees of influence on the basis from Not Present to Strong.

The IFPUG Standard Counting

The International Function Point Users Group (IFPUG) was founded in 1986. It is the working group charged with creating standard guidelines and resolving inconsistencies. The IFPUG counting practice generally follow the IBM 1984 standards, although a number of variations and extensions have occurred. At a technical level, the 1990 IFPUG counting rules have both extended and modified the 1984 IBM standard.

- External inputs: The IFPUG rules count duplicated inputs twice, whereas the IBM rules count the input only once.

- External outputs: The IFPUG rules count duplicated outputs for each occurrence, whereas the IBM rules count not specific.

- Internal logical files: The IFPUG rules on internal files generally agree with IBM, but they offer some extensions and additional refinements.

- External interface files. The potential variations between IBM and IFPUG are in the way external interface files are counted. The 1990 IFPUG rules count external interfaces as the "receiving" application but not as the "sending" application. The 1984 IBM rules, on the other hand, credit the interface file to both the sender and the receiver.

- External inquiries: Inquiries have long been among the most troublesome factors when counting function points, because of the need to consider both the input and output portions. The 1990 IFPUG rules provide significantly expanded sets of examples dealing with queries.

The IFPUG counting practices committee is starting to provide a true international forum for serious discussions about functional metrics, and thus will benefit the software community as a whole.

2.6.4 Extensions of Function Point

Feature Point

In 1986, Software Productivity Research Developed an experimental method for applying function point logic to system software such as operating systems and telephone switching systems [142]. To avoid confusion with the IBM function point methods, this experimental alternative was called feature points. The method has been applied to many kinds of software: system software, embedded software, real-time software, Computer-Aided Design (CAD), Artificial Intelligent (AI) and even Management Information System (MIS) software.

Feature point extends the function points to include algorithms as a new class [143]. An algorithm is defined as the set of rules which must be completely expressed to solve a significant computational problem. For example, a square root routine can be considered as an algorithm. Each algorithm used is given a weight ranging from 1

(elementary) to 10 (sophisticated algorithms) and the feature point is the weighted sum of the algorithms plus the function points. In his book, "Applied Software Measurement – Assuring Productivity and Quality", Capers Jones [143] described the procedure to count and weight algorithms, and also detailed the typical algorithms include sorting, searching, step-rate calculation function, feedback loops etc. This measurement is especially useful for systems with few input/output and high algorithmic complexity, such as mathematical software, discrete simulations, and military applications.

Full Function Point (FFP)

Another extension of function points is full function point (FFP) for measuring real-time applications, by also taking into consideration the control aspect of such applications. FFP introduces two new control data function types and four new control transactional function types. A detailed description of this new measurement and counting procedure can be found in [287].

Object Point

While feature point and FFP extend the function point, the object point [30] measures the size from a different dimension. This measurement is based on the number and complexity of the following objects: screens, reports and third-generation language (3GL) components. Each of these objects is counted and given a weight ranging from 1 (simple screen) to 10 (3GL component) and the object point is the weighted sum of all these objects. This is a relatively new measurement and it has not been very popular. But because it is easy to use at the early phase of the development cycle and also measures software size reasonably well, this measurement has been used in major estimation models such as COCOMO II [26] for cost estimation.

2.7 Summary

This chapter described the history and categories of software measurement and metric within the context of software engineering. It identified that measurement is critical in quality control, which is the core of software engineering. It discusses the current state of metrics and recognized that only a few basic metrics are qualified as candidates for

success.

This chapter highlighted the metrics related to productivity include *project cost* (in terms of *effort*), and *size* measures. These measures are closely related to the topic of this project and the content of the following chapter.

In summary, measurement, as any technology, must be used with care. Any application of software measurement should not be made on its own. Rather it should be an integral part of a general assessment or improvement programme, where the measures support the goals and help to evaluate the results of the action.

Chapter 3

Cost Estimation Models

3.1 Introduction

One of the basic goals of software engineering is the establishment of useful models and equations to predict the cost of any given programming project [10].

This chapter gives a general overview of software cost estimation techniques including the recent advances and evaluation criteria in the field. Estimation models are classified into categories regarding their underlying assumptions and modelling characteristics, which include expertise-based techniques, parametric models, statistically based techniques, learning-oriented techniques, and hybrid techniques. This chapter summarizes the comparative evaluation results reported in the literature and discusses common problems of cost estimation.

3.2 Software Cost Estimation Models

Accurate prediction of software development cost may have a vital economic impact. By contrast to their importance, software projects have an unfortunate reputation among the public at large for poor performance. The Standish Group research [106, 154] indicated that on average, only 16.2% of 175,000 software projects that are completed on time and on-budget. According to Sauer and Cuthbertson [270]'s report in UK between 2002 and 2003, the software project average overrun on budget is 18%; average overrun on schedule is 23%; average underachievement on scope/functionality is 7%; The number of projects hitting all their targets remains low at 16%.

3.2.1 Software Cost Estimation

Research on software cost estimation started independently in a number of companies and military organizations that built large software systems in 1960s [147]. A paper by Benington [22] in 1956 is the first published work on the subject. The main issue that led to formal research programs for software cost estimation was the difficulty encountered in completing large software applications on time and within budget.

Cost estimation [283] normally involves estimating the quantity of labor, materials, utilities, floor space, sales, overhead, time and other costs for a set series of time periods. Most researchers [88, 26] define software cost prediction as the process of predicting the

amount of effort required to build a software system. Fenton [88] proposed that the term "cost estimation" and "effort estimation" are sometimes used interchangeably. Boehm [26] stated the fundamental equation as follows:

$$Cost = \text{Effort} \times (Salary + Burden) \tag{3.1}$$

We know that the cost of developing software, up until the point that it is accepted, is only a fraction of the total cost of the system over the typical life cycle of the product. However, for the purpose of this study we will exclude the maintenance costs, and will speak only of the development costs up until acceptance, This position is consistent with that taken by those having done research in this field. This project will stand upon this point and consider the term "cost estimation" is synonymous with "effort estimation" or "effort prediction". As defined by Delurgio [293] a prediction is a probabilistic estimate or description of a future value or condition. Software cost estimation is a probabilistic estimate of a future value or a range of values of software effort in terms of the Man-Month metric.

3.2.2 Software Cost Models

In science and engineering, the term "model", as understood to refer to the ensemble of equations which describe and interrelate the variables and parameters of a physical system or process [18]. A software cost model refers to the ensemble of functions of project cost which describe and interrelate the variables and parameters of a software development process. Cost models provide direct estimation of effort and typically have a primary cost factor such as size and a number of secondary adjustment factors. Cost factors are characteristics of the project, process, products, or resources that influence effort. When predicting software project effort, we label these cost factors as predictor variables x_i, the variable is being predicted i.e., project effort(man-month) as the predicted variable E. The intention is to find out the underlying relationship between the predictor variables and the predicted variable. The requirement is to define a mapping that is extracted from a historical data set as follows:

$$E = f(x_i) \tag{3.2}$$

This functional relationship is then used to predict the new project's effort E' dependent upon the new project's attribute variables x_i'.

Typically, this relationship could be modeled as mathematical equations, statistical statements, rule based systems, tree structures, or even unknown relationships (e.g. neural net work predicting). The existing effort estimation models differ in two aspects: the selection of explanatory or cost factors x_i, and the form of the function f. We will first discuss the common cost factors used in these models, then characterize the models according to the form of the functions. The techniques, which have the capacity of constructing the function f, will be introduced in the following sections.

3.2.3 General Cost Factors

Project size is one of the widely accepted cost factors. Besides the software size, there are many other cost factors. The most comprehensive set of cost factors are proposed and used by Boehm *et al.* [26] in the COCOMO II model. These cost factors can be divided into four types:

- **Product factors**: required reliability; product complexity; database size used; required reusability; documentation match to life-cycle needs;
- **Computer factors**: execution time constraint; main storage constraint;computer turnaround constraints; platform volatility;
- **Personnel factors**: analyst capability; application experience; programming capability; platform experience; language and tool experience; personnel continuity;
- **Project factors**: multi-site development; use of software tool; required development schedule. The above factors are not necessarily independent, and most of them are hard to quantify.

In many models, some of the factors appear in combined form and some are simply ignored [191]. Also, some factors take discrete values, resulting in an estimation function with a piece-wise form.

3.3 Expertise-based Techniques

The problems of quantifying project size and the continual emergence of new technology in software effort estimation, present as a classic example of a weak theory domain where experience is key [73]. Expertise-based methods or Expert judgment techniques [115] are useful in the absence of quantified, empirical data. They capture the knowledge and experience of practitioners seasoned within a domain of interest, providing estimates based upon a synthesis of the known outcomes of all the past projects to which the expert is privy or in which he or she participated.

Expertise-based methods have some advantages compare with other predicting methods: The experts can factor in differences between past project experience and requirements of the proposed project; The experts can factor in project impacts caused by new technologies, architectures, applications and languages involved in the future project and can also factor in exceptional personnel characteristics and interactions. The drawbacks [88] to these methods are that an estimate is only as good as the experts' opinion, and there is no way usually to test that opinion until it is too late to correct the damage if that opinion proves wrong. Expertise-based methods are good for unprecedented projects and for participatory estimation, but encounter the expertise-calibration problems discussed above and scalability problems for extensive sensitivity analysis.

This section gives an overview of expertise-based methods, including Delphi, Parkinson, Price-to-win, Work Breakdown Structure, Top-Down and Analogy-based estimation methods.

3.3.1 Delphi Technique

The Delphi technique [116] was developed at The Rand Corporation in the late 1940s originally as a way of making predictions about future events - thus its name, recalling the divinations of the Greek oracle of antiquity, located on the southern flank of Mt. Parnassos at Delphi. As described by Dalkey [71] and Helmer [116], the general process of Delphi is that: Participants are asked to make some assessment regarding an issue, individually in a preliminary round, without consulting the other participants in the exercise. The first round results are then collected, tabulated, and then returned to each participant for a second round, during which the participants are again asked to

make an assessment regarding the same issue, but this time with knowledge of what the other participants did in the first round. The second round usually results in a narrowing of the range in assessments by the group, pointing to some reasonable middle ground regarding the issue of concern.

The original Delphi technique avoided group discussion. To provide a sufficiently broad communication bandwidth for the experts to exchange the volume of information necessary to calibrate their estimates with those of the other experts, a wide-band Delphi technique is introduced over standard Delphi technique by Boehm [24]. which accommodates group discussion between assessment rounds. This is a useful technique for coming to some conclusion regarding an issue when the only information available is based more on expert opinion than hard empirical data.

The wide band Delphi Technique has subsequently been used in a number of studies and cost estimation activities. It has been highly successful in combining the free discuss advantages of the group meeting technique and advantage of anonymous estimation of the standard Delphi Technique.

3.3.2 Parkinson

Using Parkinson's "Principle work expands to fill the available volume" [236], the cost is determined (not estimated) by the available resources rather than based on an objective assessment. If the software has to be delivered in 12 months and 5 people are available, the effort is estimated to be 60 person-months. Although it sometimes gives good estimation, this method is not recommended as it may provide very unrealistic estimates. Also, this method does not promote good software engineering practice.

3.3.3 Price-to-win

The software cost is estimated to be the best price to win the project. The estimation is based on the customer's budget instead of the software functionality. For example, if a reasonable estimation for a project costs 100 person-months but the customer can only afford 60 person-months, it is common that the estimator is asked to modify the estimation to fit 60 person-months effort in order to win the project. This is again not a good practice since it is very likely to cause a bad delay of delivery or force the development team to work overtime.

3.3.4 Work Breakdown Structure (WBS)

Long a standard of engineering practice in the development of both hardware and software, the Work Breakdown Structure [11] (WBS, is also called Bottom-up method) is a way of organizing project elements into a hierarchy that simplifies the tasks of budget estimation and control. It helps determine just exactly what costs are being estimated. Moreover, if probabilities are assigned to the costs associated with each individual element of the hierarchy, an overall expected value can be determined from the bottom up for total project development cost. Expertise is involve in this method in the determination of the most useful specification of the components within the structure and of those probabilities associated with each component. The leading method using this approach is COCOMO's detailed model [24].

3.3.5 Top-Down Estimating Method

Top-down estimating method is also called Macro Model. An overall cost estimation for the project is derived from the global properties of the software project, and then the project is partitioned into various low-level components. The leading method using this approach is Putnam's SLIM model [250] and Albrecht's Function Points based model [3]. Top-down estimating methods focus on system-level activities such as integration, documentation, configuration management, etc., many of which may be ignored in other estimating methods. It requires minimal project detail, and it is usually faster, easier to implement. This method is more applicable to early cost estimation when only global properties are known. However, it provides no detailed basis for justifying decisions or estimates either.

3.3.6 Analogy-Based Estimation

Estimating software effort by analogy [60, 280, 279, 221] is often considered as a form of Case Base Reasoning (CBR) [304] approach, which will be detailed in section 3.5.2, however, in some respects, it is also a systematic form of expert judgment since experts often search for analogous situations so as to inform their opinion.

The basic idea of the analogy-based estimation is to identify the completed projects that are the most similar to a new project. Actual data from the completed projects

are extrapolated to estimate the proposed project. Some of the analogy methods such as Mukhopadhyay *et al.*'s [221] require to access an expert in order to derive estimation rules and create a case base. However, Martin Shepperd *et al.*'s analogy method [279] differs in that no expert is used and a pure CBR strategy was adopted by calculating Euclidean distance between two cases.

Analogy-based estimation is a more intuitive method so it is easier to understand the reasoning behind a particular prediction. By contrast, there are also some problems with this method, Using this method, we have to determine how best to describe projects. The choice of variables must be restricted to information that is available at the point that the prediction required. Even once we have characterized the project, we have to determine the similarity and how much confidence can we place in the analogies.

In literature, there are both positive and negative reports about analogy-based estimation. Shepperd *et al.* [279] concluded that estimation by analogy tends to be the more accurate prediction method comparing with an algorithmic approach based upon stepwise regression analysis. While Briand *et al.* [44] demonstrated that analogy-based models appear to be less accurate than ordinary least squares regression, stepwise Analysis of Variance (ANOVA), and Classification and Regression Trees (CART). Both Shepperd *et al.* [280] and Briand et al. [44] used the software tool called ANGEL (ANaloGy Estimation tool) to run the analogy estimating approach.

3.4 Parametric Models

The parametric (or algorithmic) effort estimation method is designed to provide mathematical equations as functions of a number of variables, which are considered to be the major cost factors. These equations are constructed based on previous researches or historical data. As described in section 3.2.3, the cost factors could be source Lines of Code (LOC), number of functions to perform, language, design methodology, skill-levels, risk assessments, etc. The algorithmic effort estimation model has the general form as follows:

$$\text{Effort} = f(x_1, x_2, \ldots, x_n) \tag{3.3}$$

Where the $(x_1, x_2, \ldots, x_n)$ denote the cost factors.

Parametric models have the advantages of being objective(unbiased with respect to factors that are not parameters) and reliable(it is able to generate repeatable estimations). However, the accuracy of estimates generated by uncalibrated parametric models is relatively low [304]. Model calibration requires a sizable historical project database, which may not be available, A more fundamental problem with many quantitative models is that the estimation process is based on mathematical formulae whose parameters can be difficult for development managers to understand and manipulate [317, 304].

This section presents the typical forms of parametric cost models and an overview of Ordinary Least Squares Regression, Robust Regression, Analysis of Variance, SLIM, COCOMO-based, Functionality Based parametric models. A brief review of automated cost estimation tools including PRICE, SELECT, etc. is provided in this section.

3.4.1 Typical Forms of Parametric Models

Typical forms of algorithmic effort estimation models [10] including linear models, multiplicative models and power function models are introduced as the following.

Ordinary Linear Regression Models

Multiple Regression expresses the response (e.g. Man-Month) as a linear function of i predictors (e.g. LOC, product Complxity, etc.). The linear function is estimated from the data using the ordinary least squares approach discussed in numerous books such as Hair [107].

A linear multiple regression models have the form:

$$\text{Effort} = a_0 + \sum_{i=1}^{n} a_i \times x_i \tag{3.4}$$

where the coefficients $a_1, \ldots, a_n$ are chosen to best fit the completed project data. The most straight forward statistical algorithm is to assume a linear model as follows [10]:

$$\text{Effort} = a + b \times \text{Size} \tag{3.5}$$

that a represents fixed development costs (for example regression testing will consume a fixed amount of effort irrespective of the size of the software) and b represents productivity. This regression approach have been applied by Albrecht and Gaffney [4],

Behrens [20], Marouane and Mili [200], etc. Kok *et al.* [178] described how step-wise regression approach has been successfully utilized on the Esprit MERMAID project. The advantages and disadvantages of regression models were discussed in [41].

Multiplicative Models

Multiplicative models have the form:

$$\text{Effort} = a_0 \times \prod_{i=1}^{n} a_i^{x_i} \tag{3.6}$$

Again the coefficients $a_1, \ldots, a_n$ are chosen to best fit the completed project data. Walston and Felix [307] used this type of model with each x_i taking on only three possible values: -1, 0, +1. The Doty model [118] also belongs to this class with each x_i taking on only two possible values: 0, +1. The multiplicative models seem to be too restrictive on the cost factor values [41] that the small change of cost factors x_i's value makes exponential change of predicted variable's value.

Power Function Models

Power function models have the general form:

$$\text{Effort} = a \times S^b \tag{3.7}$$

where S is the code-size, and a, b are (usually simple) functions of other cost factors. This category contains two of the most popular algorithmic models in use, as COCOMO [24, 26] and SLIM [250].

3.4.2 Least Squares Regression (LSR) Model

Least Squares Regression is one of the common techniques used for software development effort prediction [44, 200].

For data on a predictor variable x (i.e. Size) and a predicted variable y (i.e. Effort) for n individuals (i.e. n project data sets), the statistical equation of the least-squares

regression line is:

$$\hat{y} = a + b \times x \tag{3.8}$$

$$b = r \times \frac{s_y}{s_x} \tag{3.9}$$

$$a = \overline{y} - b \times \overline{x} \tag{3.10}$$

The means and standard deviations of the sample data are $\overline{x}$ and s_x for x and $\overline{y}$ and s_y for y.

As demonstrated by Briand *et al.* [44], multivariate least squares regression analysis can be applied by fitting the data to a specified model that predicts effort. Marouane and Mili's [200] implemented the MicroTSP tool to carry on least square analysis for deriving their effort and schedule equations in TUCOMO model.

3.4.3 Robust Regression (RR)

The main concepts behind Ordinary Least Squares (OLS) regression also hold for Robust Regression (RR). But in order to remedy the sensitivity to outlying observations alternative minimizing methods are proposed. Instead of minimizing the sum of squares of absolute error, like in OLS regression, other methods are applied in robust regression. LMS (least median of squares), for example, minimizes the median of squares of absolute error [260]. Other examples are the LBRS (least-squares of balanced relative errors) method minimizing the sum of squares of balanced relative error, or LIRS (least-squares of inverted balanced relative errors) that minimizes the sum of squares of inverted balanced relative error [212]. Robust regression aims to be robust against outlying observations thus addressing an important issue in software engineering data. In contrast to OLS regression there is no closed-form formula that can be applied for some of the alternative algorithms [260]. Therefore, tool support is needed and some of the methods are computationally intensive.

3.4.4 Analysis of Variance (ANOVA) Model

The analysis of variance(ANOVA) procedure has been used for constructing software cost estimation models [44, 168, 187, 188, 189, 190]. This procedure can analyze the variance of unbalanced data and fit regression estimates to models with categorical

variables. Briand *et al.* [44] applied a Stepwise ANOVA model in the following form.

$$\text{Effort} = a \times \text{Size}^b \times F_1^c \times F_2^d \times \ldots \tag{3.11}$$

Where a is a constant which varies with the significant class variables, Size is the system size in function points, and F_x is a significant productivity factor (at the 0.05 level of significance, $\alpha = 0.05$). Interaction effects of class variables were taken into consideration. The detailed explanation is presented in section 4.4.3 and section 4.5.

3.4.5 Putnam's SLIM

In 1960s, the Rayleigh distribution was observed to provide a good approximation of the manpower curve for various hardware development process [239]. In 1968, Norden [229] proposed that the Rayleigh distribution is the best fit for the pattern of manpower buildup and phaseout in complex projects. Based on the nature of the software development process, Larry Putnam [250] [251] of Quantitative Software Measurement studied and tested the Rayleigh model extensively and extended it to software estimates, productivity measurements and quality forecasts. This is the well known Software Life Cycle Model (SLIM). Figure 3.1 demonstrates it is based on the Rayleigh distribution of project personnel level versus time. The curve is modeled by the differential equation:

$$\frac{\mathrm{d}\,y}{\mathrm{d}\,t} = 2 * K * a * t\mathrm{e}^{-at^2} \tag{3.12}$$

Which can be written as:

$$y' = (K/t_d^2) * t * e^{(-t^2/2t_d^2)} \tag{3.13}$$

Where,

y is cumulative effort.

y' is effort per time period, such as man-months per month.

K is the total effort to the end of the project, that is, the area under the Rayleigh curve, such as man-months.

t is elapsed time from the start of the cycle, such as months.

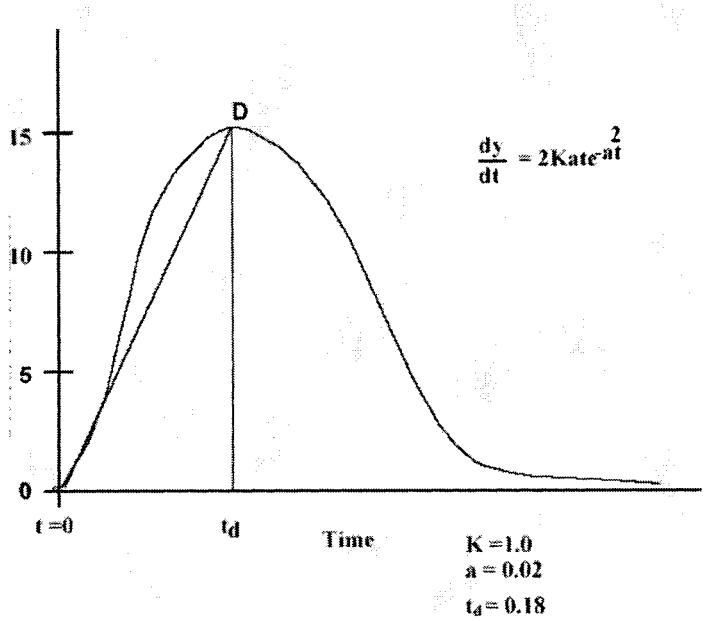

Figure 3.1: SLIM Model

a is a shape parameter that governs the rate at which the curve approaches peak manpower.

t_d is the point on the manpower utilization curve at which the effort rate is at a maximum, indicated as 39 percent of total effort.

The SLIM models is expressed as two equations describing the relation between the development *effort* and the *schedule*. Equation 3.14, called the software equation, and states that development effort is proportional to the cube of the size and inversely proportional to the fourth power of the development time. Equation 3.15 is the manpower buildup equation, states that the effort is proportional to the cube of the development time. A Manpower Buildup Index (MBI), which corresponds to the Manpower Buildup Parameter (MBP), and a Technology Constant or the Productivity Index (PI), which corresponds to Productivity Parameter (PP), are used to influence the shape of the curve. SLIM can record and analyze data from previously completed projects which are then used to calibrate the model; or if data are not available then a set of questions can be answered to get values of MBP and PP from the historical data sets. The project effort and project time can be calculated by solving the following

equations [251], which hold simultaneously:

$$\text{Effort} = ((\text{Size} * B^{(1/3)}/\text{PP})^3 * (1/\text{Time})^4) \quad (3.14)$$

$$\text{Effort} = 0.39 * \text{MBP} * \text{Time}^3 \quad (3.15)$$

Where,

Effort is the man-years of work by all job classifications for the software construction.

Size is the Project Size by Lines Of Code (SLOC). The size measured by function points can be change to LOC for a specified programming language.

B is a special skills factor that is a function of size. It provides for specialized skills for integration testing, documentation, and management as the size of the system increases. The value of B is between 0 and 1, which corresponds to the range of Size.

PP Productivity Parameter is a number obtained by calibration from past projects. A Productivity Index (PI) has been assigned for different range of Productivity Parameter.

Time is the elapsed calendar schedule in years for the software construction phase.

MBP is the Manpower Buildup Parameter, can be calibrate from past completed projects. Each range of MBP corresponds a Manpower Buildup Index (MBI).

Researchers, such as Pillai [239], have criticized the use of a Rayleigh curve as a basis for cost estimation on the grounds that Norden's original observations were not based in theory but rather on observations. Moreover his data reflects hardware projects. It has not been demonstrated that software projects are staffed in the same way. Software projects sometimes exhibit a rapid manpower buildup which invalidate the SLIM model for the beginning of the project.

3.4.6 Boehm's COCOMO and COCOMO II

One very widely used algorithmic software cost model is the Constructive Cost Model (COCOMO) [24], which was originally published in 1981, and became one of the most

popular parametric cost estimation models of the 1980s. The first version of COCOMO model was normally referenced as COCOMO'81. It has three different models that can be used throughout a project's life cycle:

- Basic Model - this model would be applied early in a project's development. It will provide a rough estimate early on that should be refined later on with one of the other models.
- Intermediate Model - this model would be used after more detailed requirements are available for a project.
- Advanced Model - when definition of a project is complete, this model can be applied to further refine the estimate.

Within each of these models there are also three different modes. The mode is chosen depending on the work environment, and the size and constraints of the project itself. the modes are:

- Organic - this mode is used for "relativity small software teams developing software in a highly familiar, in-house environment"
- Embedded - operating with tight constraints where the product is strongly tied to a "complex of hardware, software, regulations and operational procedures".
- Semi-detached - an intermediate stage somewhere in between organic and embedded. Projects are usually of moderate size of up to 300,000 lines of code.

One of the problems with using a model like COCOMO 81 today is that is does not match the develop environment of the late 1990's and 2000's. It was created in a time when batch jobs were the norm, programs were run on mainframes and compile times were measured in hours not seconds. It is outdated for use in today's development (for example, rapid application development, 4th generation languages etc.). There are some other problems of COCOMO 81 models as follows:

- In early phase of system life-cycle, the size is estimated with great uncertainty value. So, the accurate cost estimate can not be arrived at.

- The cost estimation equation is derived from the analysis of 63 selected projects. It usually have some problems outside of its particular environment. For this reason, the recalibration is necessary.

- According to Kemerer's research [160], the average error for all versions of the model is 601%. The detailed model and Intermediate model seem not much better than basic model.

For these reasons, The latest version, COCOMO 2.0, was developed and referenced as COCOMO II. The COCOMO II research effort was started in 1994 to address the issues on non-sequential and rapid development process models, re-engineering, reuse driven approaches, object oriented approaches etc. COCOMO II was initially published in the Annals of Software Engineering in 1995 [31].

COCOMO II

COCOMO II includes two underlying information models. The first is a framework for describing a software project, including models of the process, culture, stake-holders, methods, tools, teams, and the size/complexity of the software product. The second is an experience base that can be used to estimate the likely resources (effort and time) of a project from historical precedents [26]. For the most part estimates are obtained in pretty much the same way as COCOMO 81. The main changes have been in the number and type of cost drivers and the calculation of equation variables rather then the use of constants. The major new modelling capabilities of COCOMO II [28] are a tailorable family of software size models, involving object points, function points [12] [159] and source lines of code; nonlinear models for software reuse and re-engineering; an exponent-driver approach for modelling relative software diseconomies of scale; and several additions, deletions, and updates to previous COCOMO effort-multiplier cost drivers.

The COCOMO II model also has three sub-models [31, 25], *Applications Composition*, *Early Design* and *Post-Architecture*, which can be combined in various ways to deal with the current and likely future software practices marketplace.

- The*Application Composition* model is used to estimate effort and schedule on projects that use Integrated Computer Aided Software Engineering tools for rapid

application development. Typical components are GUI builders, database or objects managers, middle-ware for distributed processing or transaction processing, etc. and domain-specific components such as financial, medical or industrial process control packages. The Applications Composition model is based on Object Points [12, 159]. Object Points are a count of the screens, reports and 3 GL language modules developed in the application. Each count is weighted by a three-level (simple, medium, and difficult) complexity factor. This estimating approach is commensurate with the level of information available during the planning stages of Application Composition projects.

- The *Early Design* model involves the exploration of alternative system architectures and concepts of operation. Typically, not enough is known to make a detailed fine-grain estimate. This model is based on function points (or lines of code when available) and a set of five scale factors and 7 effort multipliers.

- The *Post-Architecture* model is used when top level design is complete and detailed information about the project is available and as the name suggests, the software architecture is well defined and established. It estimates for the entire development life-cycle and is a detailed extension of the Early-Design model. This model is the closest in structure and formulation to the Intermediate COCOMO'81 and Ada COCOMO models. It uses Source Lines of Code and/or Function Points for the sizing parameter, adjusted for reuse and breakage; a set of 17 effort multipliers and a set of 5 scale factors, that determine the economies/diseconomies of scale of the software under development. The 5 scale factors replace the development modes in the COCOMO'81 model and refine the exponent in the Ada COCOMO model.

A primary attraction of the COCOMO models is their fully-available internal equations and parameter values. Over a dozen commercial COCOMO'81 implementations are available. Some of the popular tools, i.e., Costar, also supports COCOMO II. We only

highlights the effort equation of COCOMO II as the following:

$$\mathrm{PM} = \mathrm{A} \times \mathrm{Size}^{\mathrm{E}} \times \prod_{i=1}^{n} \mathrm{EM}_i + \mathrm{PM}_{\mathrm{Auto}} \tag{3.16}$$

$$\mathrm{E} = \mathrm{B} + 0.01 \times \sum_{j=1}^{s} \mathrm{SF}_i \tag{3.17}$$

$$\mathrm{PM}_{\mathrm{Auto}} = \frac{\text{Adapted SLOC} \times (\frac{\mathrm{AT}}{100})}{\mathrm{ATPROD}} \tag{3.18}$$

The symbols having the following meaning.

Symbol	Description
A	Effort coefficient that can be calibrated.
AT	Percentage of the Adapted SLOC that is re-engineered by automatic translation
ATPROD	Automatic translation productivity
B	Scaling base-exponent for Effort that can be calibrated
E	Scaling exponent for Effort
EM	Effort Multipliers: 7 for the Early Design and 17 for the Post-Architecture modes
PM	Person Months effort from developing new and adapted code
$\mathrm{PM}_{\mathrm{Auto}}$	Person Months Effort from automatic translation activities
SF	5 Scale Factors
SLOC	Source Lines of Code

For a COCOMO model to be accurate it must be calibrated using historical data. The calibration process can be done by using a company's own data, but for the most part it requires more data than a single company would have. The calibration involves doing a statistical analysis on data set and then adjusting all cost driver values. Because of the need of a proper calibration there are standard calibrations released. COCOMO II has gone through two calibrations, COCOMO II.1997 and COCOMO II.1998 [29]. COCOMO II.1997 was based on 83 data points and was found that it only could come within 20% of the actual values 46% of the time. The COCOMO II.1998 calibration [60] was found to come within 30% of the actual values 75% of the time, this calibration was based on 161 data points.

3.4.7 COoperative Programming MOdel (COPMO)

In order to obtain a model that supports Brook's law (as the manager of the colossal OS/360 projects, Brooks proposed a law based on his experience [50]: "adding manpower to a late software project makes it later") COoperative Programming MOdel (COPMO) [61] including metrics concerning project size and staff size was proposed by Conte *et al.*.

$$\widehat{E} = u_i \times S + v_i \times \overline{P}^d \tag{3.19}$$

Symbol	Description
$\widehat{E}$	number of estimated man months (MM)
u_i, v_i	technology factors that are believed to affect productivity
S	Thousands of Delivered Source Instructions(KDSI)
$\overline{P}$	average full-time equivalent team size

3.4.8 An Adaptive COCOMO model - Tunisian Cost Model (TUCOMO)

In 1989, Marouane and Mili published a Tunisian Cost Model (TUCOMO) considering the data processing culture in Tunisia. They conducted a survey to derive specific COCOMO-like equations, which are called Tunisian Cost Model (TUCOMO). The survey was based on the Basic COCOMO and intermediate COCOMO model, however, as the authors declared [200], at that preliminary stage of the work, it was possible only to give the basic equations of TUCOMO; i.e. the Tunisian equivalent of basic COCOMO, re-calibrated using a project database containing 47 Tunisian MIS projects [200]. The data have been separated as two development modes: organic and embedded, corresponded to COCOMO's "organic" mode, and "semidetached and embedded" mode respectively.

The effort equation is:

$$\begin{aligned} \text{MM} &= 2.74 \times \text{KDSI}^{0.76} \quad \text{for Organic Mode} \\ \text{MM} &= 6.85 \times \text{KDSI}^{0.66} \quad \text{for Embedded Mode} \end{aligned} \tag{3.20}$$

Because of the re-calibration the constants have been set to 2.74 and 0.76 for the

Organic Mode and 6.85 and 0.66 for the Embedded Mode. MM stands for Man-Month, KDSI is Thousands of Delivered Source Instructions.

Regression analysis, including T-test and Standard deviation calculation, have been performed for each Tunisian equivalent and basic COCOMO to prove the TUCOMO equations are reasonably accurate. However, these results are only partialy reliable because of the small quantity and localized experimental data resources.

3.4.9 Functionality Based Methods

The Function Point Analysis [3] is another method of quantifying the size and complexity of a software system in terms of the functions that the systems delivers to the user. A number of proprietary models for cost estimation have adopted a function point type of approach, such as ESTIMACS [263], SPQR/20 [141] [311] and Checkpoint [143]. The most straightforward equation to calculate project effort based on measuring size in function points is as 3.21:

$$\text{Effort} = \text{Size}/\text{Productivity} \tag{3.21}$$

An example of above can be found in Treble and Douglas' book [301]. A further consideration such as measuring the complexity, adjusting the effort and schedule for currently existing projects, etc., have been made. The calibrated equation is as 3.22:

$$\text{Effort} = (\text{Size} * \text{TCA}/\text{Productivity})/\text{SCF} \tag{3.22}$$

Where, TCA is Technical Complexity Adjustment, SCF is Schedule Compression Factor, which can be obtained by dividing available time by estimated time.

The advantages of function point analysis based model are: function points can be estimated from requirements specifications or design specifications, thus making it possible to estimate development cost in the early phases of development; function points are independent of the language, tools, or methodologies used for implementation; and non-technical users could have a better understanding of what function points are measuring since function points are based on the system user's external view of the system.

Rubin's ESTIMACS

ESTIMACS is a propriety system designed to give development cost estimate at the conception stage of a project and it contains a module which estimates function points as a primary input for estimating cost. It was originally developed by Howard Rubin [263] in the late 1970s as Quest (Quick Estimation System).

Rubin has identified six important dimensions of estimation and a map showing their relationships, all the way from what he calls the gross business specifications through to their impact on the developers long term projected portfolio mix. The critical estimation dimensions include effort hours, staff size and deployment cost, hardware resource requirements, risk and portfolio impact. This functionality based cost estimation methods is the better approach especially in the early phases of development [160].

SPQR/20

SPQR/20 [141] is also derived from Albrecht's function point approach by Caper Jones in 1986. It is based on a modified function point method. Whereas traditional function point analysis is based on evaluating 14 factors, SPQR/20 separates complexity into three categories: complexity of algorithms, complexity of code, and complexity of data structures. It is an interactive package which provides similar outputs to ESTIMACS. The user is prompted by a series of product and project related questions. Then the project size estimate is input by the user in terms of function point categories. Wittig and Finnie [311] applied the SPQR/20 [141] software estimation tool to generate simulated software development project data, which has been declared that SPQR/20 has given relatively good results in estimating development development effort in a locally calibrated environment [132].

Checkpoint

Checkpoint is a knowledge-based software project estimating tool from Software Productivity Research (SPR) developed from Jones' studies [143]. It has a proprietary database of about 8,000 software projects and it focuses on four areas that need to be managed to improve software quality and productivity. Checkpoint uses Function Points (or Feature Points) [3] [298] as its primary input of size. SPR's Summary of

opportunities for software development is shown in Figure 3.2. QA stands for Quality Assurance; JAD for Joint Application Development; SDM for Software Development Metrics. It focuses on three main capabilities for supporting the entire software development life-cycle as outlined here:

Estimation: Checkpoint predicts effort at four levels of granularity: project, phase, activity, and task. Estimates also include resources, deliverables, defects, costs, and schedules.

Measurement: Checkpoint enables users to capture project metrics to perform benchmark analysis, identify best practices, and develop internal estimation knowledge bases (known as Templates).

Assessment: Checkpoint facilitates the comparison of actual and estimated performance to various industry standards included in the knowledge base.

The approach [143], which Software Productivity Research developed is to use multiple-choice questions built around a weighting scale – the software effectiveness level (SEL), was used in questionnaires to acquire the baseline data for measuring software quality(i.e. defect origins, defect severities,etc) and user satisfaction(i.e. user reported defects).

3.4.10 The PRICE-S model

The PRICE-S model was originally developed at RCA for use internally on software projects such as some that were part of the Apollo moon program. It was then released in 1977 as a proprietary model and used for estimating several US DoD, NASA and other government software projects. The model equations were not released in the public domain, although a few of the models central algorithms were published in [235, 66]. The tool continued to become popular and is now marketed by PRICE Systems, which is a privately held company formerly affiliated with Lockheed Martin. As published on PRICE Systems web site (http://www.pricesystems.com), the PRICE-S Model consists of three sub-models that enable estimating costs and schedules for the development and support of computer systems.

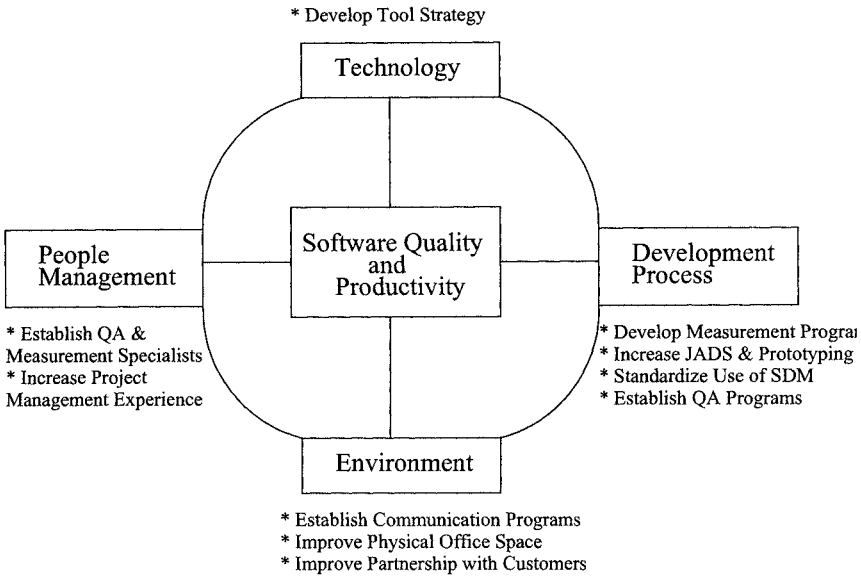

Figure 3.2: SPR Model

3.4.11 SEER-SEM Model

SEER-SEM [89] is a product offered by Galorath, Inc. of El Segundo, California. This model is based on the original Jensen model [137]. It has evolved into a sophisticated tool supporting top-down and bottom-up estimation methodologies. Its modelling equations are proprietary, but they take a parametric approach to estimation.

The scope of the model is wide. It covers all phases of the project life-cycle, from early specification through design, development, delivery and maintenance. It handles a variety of environmental and application configurations, such as client-server, stand-alone, distributed, graphics, etc. It models the most widely used development methods and languages. Development modes covered include object oriented, reuse, COTS, spiral, waterfall, prototype and incremental development. Languages covered are 3rd and 4th generation languages (C++, FORTRAN, COBOL, Ada, etc.), as well as application generators. It allows staff capability, required design and process standards, and levels of acceptable development risk to be input as constraints. Figure 3.3 is adapted from a Galorath illustration and shows gross categories of model inputs and outputs, but each of these represents dozens of specific input and output possibilities and parameters. The reports available from the model covering all aspects of input and output summaries and analysis number in the hundreds.

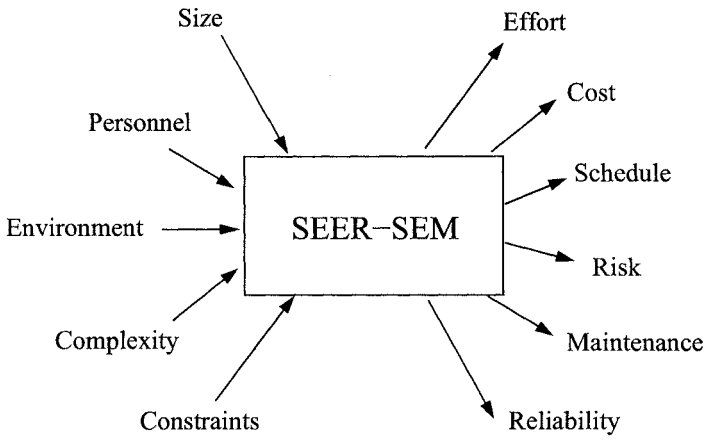

Figure 3.3: SEER-SEM Model

3.5 Learning-oriented Techniques

Software development is a complex production with many interrelated factors which affect development effort and productivity. Accurate forecasting has proved difficult [89, 131, 160, 221], since many of these interrelationships are not fully understood [167, 24]. As opposed to parametric prediction approaches, machine learning approaches are nonparametric, since they make no a priori assumptions about the form of the function being approximated. These learning oriented techniques, such as, Case Based Reasoning, Artificial Neural Networks, Genetic algorithms, etc., were introduced to this discipline in the mid 1990's, centering on artificial neural network estimation models. Results from studies such as Khoshgoftaar [163], Samson [269] and Wittig [311] showed highly promising results. However, the sensitivity of each learning system to certain configurations should be considered carefully. Clearly there is a need for further investigation, particularly in finding appropriate configuration heuristics for learning systems. Furthermore, non-parametric techniques inspire little confidence regardless of how excellent are the Mean of Magnitude of Relative Errors (MMREs) that have been obtained [289].

This section describes typical learning oriented techniques for software cost prediction, including Case Based Reasoning (CBR), Artificial Neural Networks (ANN), and Genetic Algorithms (GA).

3.5.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are recognized for their ability to provide good results when dealing with problems where there are complex relationships between inputs and outputs, and where the input data is distorted by high noise levels. ANNs are purely data driven models, which through training, iteratively transition from a random state to a final model. They do not depend on assumptions about functional form, probability distribution or smoothness, and have been demonstrated to be universal approximators [94, 122]. Back propagation is one of the most common learning algorithms that have been used by software metrics researchers [158, 302, 163, 272, 286, 311, 269, 128]. The most commonly adopted architecture, learning algorithm and activation function [128] are respectively the feed-forward Multi-Layer Perceptrons, the back-propagation algorithm and the Sigmoid function (see Equation 3.23), although many more sophisticated neural networks have been proposed.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.23}$$

Figure 3.4 illustrates a neural network architecture of three layers of Multiple Layer

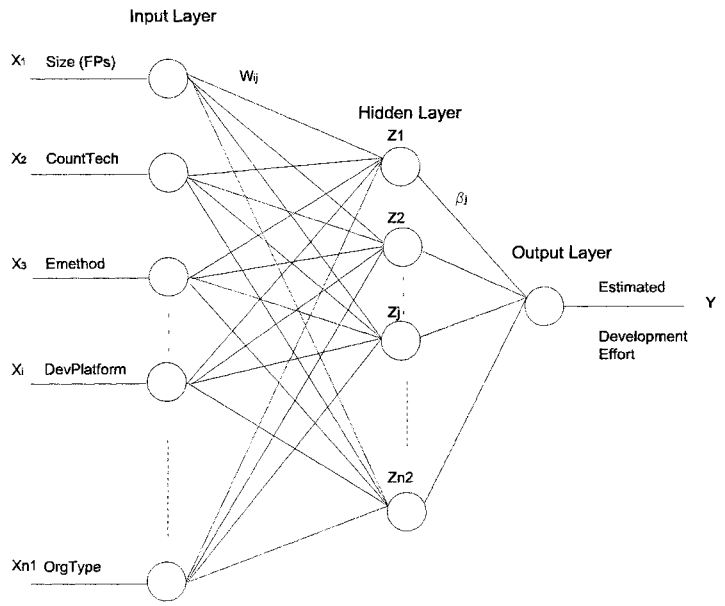

Figure 3.4: A Neural Network Architecture

Perceptron (MLP). It is organized in layers, with each layer composed of neurons or

processing elements and connections. The first layer called the input layer contains neurons that represent the state of input variables. The output layer contains neurons that represent the output variables. When the relationship between the input and output variables is nonlinear, the hidden layer helps in extracting higher level features and facilitates generalization. Connections between neurons have numerical weights associated with them; the weights are adjusted in the training process by repeatedly feeding examples from the training set. Each neuron has an activation level, specified by continuous or discrete values. The value coming into a neuron in the hidden or output layers is typically the sum of each incoming activation level times its respective connection weight. For the neurons in the inputs layer, the activation levels are determined in response to the input signals received from the environment. The back-propagation we introduced in this thesis assumes that weights in the network are initialized to small, random values prior to training. Accord to the architecture of the network in Figure 3.4, the effort is given by [128]:

$$\text{Effort} = \sum_{j=1}^{n2} Z_j \beta_j \text{ with } Z_j = f(\sum_{i=1}^{n1} W_{ij} X_i) \tag{3.24}$$

where f is the sigmoid function, W_{ij} are the weights of the connections from the input layer to the hidden layer and β_j are those from the hidden layer to the output layer.

Karunanithi *et al.* [158] reported the use of Artificial Neural Networks (ANN) for predicting software reliability, and conclude that both *feed forward* and *Jordan networks* with a cascade correlation learning algorithm, outperform traditional statistical models. More recently Wittig and Finnie [311] described the application of back propagation learning algorithms on a *multi-layer perceptron* in order to predict development effort. An overall error rate (MMRE) of 29% was obtained which compares favorably with other methods. However, it must be stressed that only a very small number of projects were selected from data sets for validation purposes [279]. Some outliers also appear to have been removed. This tends to confirm the finding of Serluca's [272] that neural nets seem to require large training sets and inputs must be filtered in order to give good predictions. Another study by Samson *et al.* [269] uses an *Albus multi-layer perceptron* in order to predict software effort. In this instance they used Boehm's COCOMO'81 data set. The work compares linear regression with a neural net approach. Both approaches seem to perform badly with MMREs of 520.7 and 428.1 percent, respectively. Srinivasan and Fisher [286] also reported on the use of a neural net with a back propagation learning algorithm and found neural nets outperformed

other methods. More recently, Shin and Goel [281] proposed an ANN modeling approach using Radial Basis Functions (RBFs) based on a well-known NASA data set. The obtained MMRE value was 0.25 and Pred(25%) value was 72.22%. The authors then claimed the RBFs methodology makes the modeling process objective, systematic and computationally fast.

Mair *et al.* [197] argued that most studies concerned with the use of ANNs to predict software development effort have focused on comparative accuracy with algorithmic models rather than on the suitability of the approach for building software effort prediction systems, for example, Wittig and Finnie's [311] experiments. Mair *et al.* [197] emphasized that explanatory value and configurability of ANNs are as important as input values and these also need investigation. The guidelines used in the experiments to develop the back-propagation networks models were presented in Hinton's [121] extensive experience. The recommendation of the suitable size of training set can be calculated by Baum and Haussler's method [19]. Mandischer [199] also pointed out the common approach to designing neural networks is to build a network, and test it for the desired function. The network is then refined by changing the structure and parameters and assessing the network until the evaluation criteria have been met.

There are two main advantages when using estimation by artificial neural networks [128]. First, it allows learning from previous situations and outcomes. The learning criteria is very important for cost estimation models because software development technology is supposed to be continuously evolving. Second, it can model a complex set of relationships between the predicted variable (such as effort) and the predictor variables (cost factors). However, there are some shortcomings that prevent it from being accepted as a common practice in cost estimation: Neural networks may be considered as 'black boxes'. Consequently, it is not easy to understand and to explain its process to the users; The ability of neural networks to solve problems of high complexity has been proven in classification and categorization areas whereas in the cost estimation field we deal with a generalization rather than a classification problem; In addition, there are no accepted guidelines for the construction of the neural networks topologies (number of layers, number of units per layer, initial weights, etc.). Several studies have proved that the neural network approach generates less accurate estimates when the training data and the test data come from separate databases [272, 286].

3.5.2 Case Based Reasoning (CBR)

Case Based Reasoning (CBR) [179, 310] is an artificial intelligent systems method which works by solving new problems by matching the new problems to past cases and reusing the solutions to the past cases. The assumption of case based study is that similar projects are likely to be subject to similar costs and schedules. When estimating a new project, these models will look for the similar cases (by constraining the source of the analogy to previously solved cases of the same problem class), and apply a series of analogy analysis to surmise (give the best guess to) the results of new project.

Early in 1984, Boehm *et al.* [32] recognized Case Based Reasoning as a useful technique in software estimation. As described in section 3.3.6, estimating effort by analogy [279] is also a form of Case Base Reasoning (CBR) approach. Silverman [282], a designer of the NASA systems, has identified analogy as the primary method use to estimate the size and execution time of new ground-based stellate control systems in 1985. However, the empirical literature on analogical reasoning in software estimation was virtually nonexistent, till 1990 Vicinanza *et al.* [304] proposed the Estor model. Mukhopadhyay *et al.* [221] also described some early work using a hybrid case based reasoning (CBR) and rule based system in 1992. They reported encouraging results based upon the data set collected by Kemerer [160]. Recent research had shown the feasibility of applying CBR to the problem of project cost estimation [92, 280, 23, 221],and a number of CBR applications applied to the problem of software cost estimation have been developed [249, 23, 280]. The most popular algorithm to calculate the similarity is the nearest neighbour matching used by Cognitive Systems ReMind software reported in Kolodner [179, 280]. The use of search techniques to help optimize a case-based reasoning system for predicting software project effort has been investigated by Kirsopp *et al.* [166] in 2002.

The case-based approach to problem solving is appropriate in task domains that have no strong theoretical model and where the domain rules are incomplete, ill-defined, and inconsistent [7]. It does not require an explicit domain model and so elicitation becomes a task of gathering case histories. Viewed in terms of task adaption, the structure of the task is always changing and the appropriate knowledge adaptations cannot reflect deep causal principles, rather, those structures permit effective access to the most appropriate, similar experiences encountered and need not rely on underlying

causal mechanisms [248]. As the domain of software effort estimation lacks a strong causal model based on deep principles and is situated within an often-changing, highly context dependent task environment, the case-based approach evidenced by the expert should indeed be an appropriate strategy to bring to bear [304].

As Vicinanza *et al.* [304] argued that the accuracy of the model strongly depends on the range of historical data sets, as the most appropriate case is only relatively "appropriate" selected from existing data sets, which may not have any "similar" cases at all. i.e., there were instances where no rules could be applied to map source to target. Sample size would limited the estimateability as well. Applying CBR in early stage software project cost estimation is not reliable [73] either because of the difficulties to generate a comprehensive case representation, when there is lack of features and data about the projects.

On the other hand, where there are too many subjective elements, approaches based on CBR require access to an expert in order to derive estimation rules and create a case base. For example, in the Estor model based on a verbal protocol of the processes involved in estimation provided by a human expert. The Estor model only provides a trace of its behavior. This subjective nature makes the case based reasoning estimation models difficult to be replicated. The numbers of people with first hand theoretical or practical experience of CBR is still small. Vicinanza *et al.* [303] used the Estor model to estimate the project effort using Kemerer's data set without formal algorithmic techniques and found the results outperformed the models in the original study. However, the MMRE ranges from 32 to 1107%. In addition, there is evidence in the literature that case based models such as the Estor model would almost certainly fail to accurately estimate projects from very different environments without additional domain knowledge [304].

Nearest neighbour and induction are two commonly used techniques for retrieving similar cases. Nearest neighbour provides a measure of how similar a new case is to an existing case, but it has one major weakness, retrieval speed. To find the best matching case a new case must be compared to every existing case in the case-base, and similarity must be calculated for every indexed attribute, thus for a case base with 100 cases and two indexed attributes, 200 similarity calculations must be performed, if a case base had 10 indexed features and grew to contain 100,000 cases, 1,000,000 similarity calculations would be required. This means that nearest neighbour can

become inefficient as either the case base and/or the number of indexed attributes increases. On the other hand inductive method retrieval times are extremely quick at runtime and only increase slowly as the number of cases increase, this is because inductive retrieval depends on pre indexing, since the decision tree is produced off-line before retrieval. The production of the decision tree is time-consuming for larger case-bases and has to be redone every time a new case is input into the case base. Inductive retrieval does have one major disadvantage however that is if any case data is missing or unknown it may not be possible to retrieve any case at all. Nearest neighbour is much less sensitive to missing, or noisy, case data.

This section highlights both a nearest neighbour and an induction techniques of CBR: K-Nearest Neighbour (KNN) and Classification and Regression Tree (CART). It also explains Rule Induction (RI) and the Estor model.

K-Nearest Neighbour (KNN)

Watson [310] states that at a conceptual level, nearest neighbour is a very simple technique, and has been applied in software cost estimation by Finie *et al.* [92, 91]. In K-Nearest-Neighbour prediction, the training data set is used to predict the value of a variable of interest for each member of a "target" data set. The data is structured such that there is a variable of interest (project *effort*, for example), and a number of additional predictor variables (*size, programming languages, programming platform, etc.*). Generally speaking, the algorithm is as follows: For each case (or project) in the target data set (the set to be predicted), locate the K closest members (the K nearest neighbours) of the training data set. A Euclidean Distance measure [279] is used to calculate how close each member of the training set is to the target row that is being examined; Find the weighted sum of the variable of interest for the K nearest neighbours (the weights are the inverse of the distances); Repeat this procedure for the remaining cases in the target set. The distance functions used in CBR-Works 4.0 beta [44] and ANGEL tool [279], are based on the un-weighted Euclidean Distance [1] using variables(project features) normalized between 0 and 1. The overall distance(P_i, P_j) between two projects P_i and P_j is defined as :

$$\text{distance}(P_i, P_j) = \sqrt{\frac{\sum_{k=1}^{n}(P_{ik}, P_{jk})}{n}} \tag{3.25}$$

Where n is the number of variables. The distance regarding a given variable k between two projects P_i and P_j is (P_{ik}, P_{jk}):

$$d(P_{ik}, P_{jk}) = \begin{cases} (\frac{|P_{ik}-P_{jk}|}{max_k - min_k})^2 & \text{if } k \text{ is continous} \\ 0 & \text{if } k \text{ is categorical AND } P_{ik} = P_{jk} \\ 1 & \text{if } k \text{ is categorical AND } P_{ik} \neq P_{jk} \end{cases} \quad (3.26)$$

Where the value max_k/min_k is the maximum/minimum possible value of variable k. The computing time increases as K increases, but the advantage is that higher values of K provide smoothing that reduces vulnerability to noise in the training data. In practical applications, typically, K is in units or tens rather than in hundreds or thousands.

There are alternatives to Euclidean distance measures such as *correlation coefficients* [156]. The correlation coefficients then measure inter-object similarity by measuring the correlation coefficient between two objects (rows) rather than between two variables (columns). Another option has been introduced in [289] when projects were measured on binary (or categorical) variables, a *matching-type* similarity measure was used. A binary variable may or may not present an attribute.

Classification and Regression Trees (CART)

Another technique for retrieval of cases used by CBR is a process called induction. The induction technique was developed by machine learning researchers to extract rules or create decision trees based on past cases. The decision tree approach classifies the data set in a tree structure. Figure 3.5 [286] illustrates a regression tree over Boehm's [24] 63 Software Project Descriptions [1]. Decision trees are referred to as classification or regression trees depending on whether they classify discrete variables or continuous variables. The common term for these trees in software estimation is "Classification and Regression Trees" (CART) [36]. In effort prediction, some predictor variables and effort are continuous whereas some predictors e.g. personnel capability is discrete. CART is suitable for automatically handling non-linearity and interactions when there are extensive data. CART analysis is a form of binary recursive partitioning. The term "binary" implies that each group of projects, represented by a "node" in a decision tree, can only be split into two groups. Thus each node can be split into two child nodes, in which case the original node is called a parent node. The term "recursive" refers to the

[1] Numbers in square brackets represent the number of projects classified under a node.

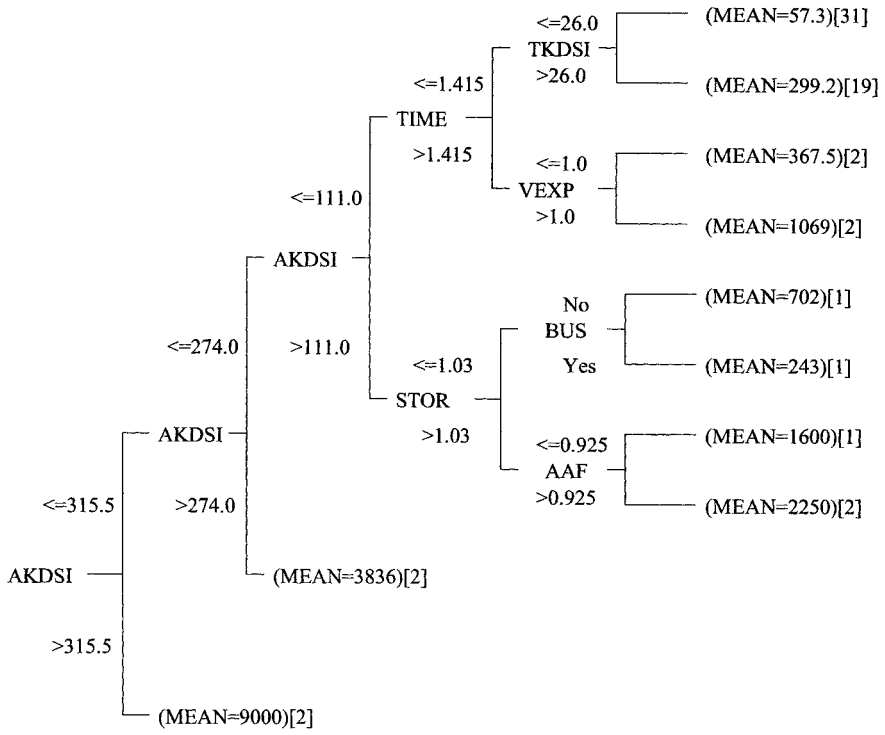

Figure 3.5: A Regression Tree Model

fact that the binary partitioning process can be applied over and over again. Output includes a "decision tree" which is immediately useful for prediction. CART analysis consists of four basic steps: building the tree, stopping the tree building, pruning the tree, cross validation.

- Step1: Building the tree, during which a tree is built using recursive splitting of nodes. Each resulting node is assigned a predicted class, based on the distribution of classes in the learning data set which would occur in that node and the decision cost matrix. The assignment of a predicted class to each node occurs whether or not that node is subsequently split into child nodes.

- Step2: Stopping the tree building process. the tree building process goes on until it is impossible to continue. The process is stopped when: (1) there is only one observation in each of the child nodes; (2) all observations within each child node have the identical distribution of predictor variables, making splitting impossible; or (3) an external limit on the number of levels in the maximal tree has been set by the user ("depth" option). At this point a "maximal" tree has been produced,

which probably greatly overfits the information contained within the learning data set. In other words, the maximal tree follows every idiosyncrasy in the learning data set, many of which are unlikely to occur in a future independent group of projects.

- Step3: Tree "pruning". In order to generate a sequence of simpler and simpler trees, each of which is a candidate for the appropriately-fit final tree, the method of "cost-complexity" pruning is used. This method relies on a complexity parameter, denoted α, which is gradually increased during the pruning process. Beginning at the last level (i.e., the terminal nodes) the child nodes are pruned away if the resulting change in the predicted misclassification cost is less than α times the change in tree complexity. Thus, α is a measure of how much additional accuracy a split must add to the entire tree to warrant the additional complexity. As α is increased, more and more nodes (of increasing importance) are pruned away, resulting in simpler and simpler trees.

- Step4: Optimal tree selection. During which the tree which fits the information in the learning data set, but does not overfit the information, is selected from among the sequence of pruned trees.

The effort for a new project is then predicted by identifying which group it fits with and then using the average effort of the projects in that group as an estimate [286].

CART analysis has a number of advantages over other classification methods, including multivariate logistic regression. It is inherently non-parametric, so there are no assumptions being made regarding the underlying distribution of values of the predictor variables. Thus, CART can handle numerical data that are highly skewed or multi-modal, as well as categorical predictors with either ordinal or non-ordinal structure. This is an important feature, as it eliminates analyst time which would otherwise be spent determining whether variables are normally distributed, and making transformations if they are not.

Recent research of applying CART in software cost model has been developed in a series of publications by Porter and Selby [241, 240, 242, 243, 244, 245]. They [244, 245] described the use of decision and classification trees in predicting aspects of the software development process. Results from this approach seem to be quite mixed and as with the neural net approach, results are quite sensitive to aspects such as the choice of

algorithm to derive the tree and tree depth [279]. Briand *et al.* [44] also introduced a combination method of CART with regression analysis for cost estimating. The approach involves the development of a regression tree, and the application of regression analysis to projects belonging to each terminal node. Thus, for each terminal node in a tree, the authors developed regression equations for predicting effort, instead of just using median values for prediction. Srinivasan and Fisher [286] described the application of a regression tree to predict effort using the Kemerer data set. They found that although it outperformed COCOMO and SLIM, the results were less good than using either a statistical model derived from function points or a neural net. Briand *et al.* [41] obtained rather better results (MMRE=94%) from their tree induction analysis. However, they also pointed out that a tree structure may force the modelling process to ignore some variables that could be useful for some predictions but includes irrelevant pieces of information.

Rule Induction (RI)

Rule Induction (RI) is a particular aspect of inductive learning in which algorithms produce rules as a result of modelling. RI is based on algorithms for induction which given a training set of examples, each of which is described by the values of an attribute and the outcome, will automatically build decision trees that will correctly classify not only all the examples in the training set, but unknown examples from the wider universe of examples of which the training set is presumed to provide a representative sample [162]. Inductive learning is the process of acquiring general concepts from specific examples. By analyzing many examples, RI may be possible to derive a general concept that defines the production conditions [197]. Examples of using RI to develop prediction systems include [72, 80, 271, 197, 278].

In order to produce a set of rules, induction works on a randomly, or algorithmically selected sub-set of the examples often referred to as the training set. These rules can be tested on the remainder of the examples (the test set) to assess how well they represent the data. RI can be used for a range of problems where there exist a set of suitable examples. Rules can be seen as decision tress where the leaf node contains the predicted value or range of values, numeric decision trees are generated by calculating the average outcome for the set of cases being considered at each node. An example fragment of rules generated from the Desharnais data set [197] is depicted below:

IF AdjFPs >= 266
IF ExpPM < 3
IF Transactions < 165
THEN effort = 3542

One advantage of inductive learning over neural network learning is that the rules are transparent and therefore can be read and understood. RI helps the estimator understand the prediction and underlying assumptions upon which it is based. Moreover, the rules may be rephrased and provided to offer a clearer explanation as to how a prediction has been made. The drawback is for a range of problems there may not exist a set of suitable examples.

The Estor Model

In 1990 Vicinanza *et al.* [304] proposed the Estor model. The purpose of Estor is to construct a computational model demonstrating the analogical reasoning strategy used by the most accurate human estimator. The Estor model incorporates five analogical problem solving process (construct, retrieve, transfer, map, and adjust) and these five generic processes were supplemented with the domain-specific knowledge of the referent expert. It used index cards to store source cases, and retrieve an appropriate case by calculating function point distance between source cases and target case (or the nearest neighbour). A human expert is needed to transfer the solution that achieved the goal in the source case to the target case, and the expert is also needed to map the source and target by examining each target attribute, recalling its value on the source and classifying it as either in correspondence or out of correspondence. Figure 3.6 depicts the overall architecture of the Estor model. The prior case knowledge and general domain knowledge (case based, rule based and analog retrieval heuristic) will be stored in long term memory. External memory contains information relating to the target projects and non-correspondence attributes lists. Working memory has been elaborated with pointers to information needed by the current analogical reasoning process [304]. The Estor model was evaluated against expert judgment, COCOMO, and Function Point Analysis (FPA) [304]. None of the models were able to improve on the performance of the expert. The estimation error of the expert and of the Estor model were considerably less than FPA and COCOMO. Despite the improvement the errors are still large with Estor's Mean Magnitude Absolute Relative Error (MMRE) greater than 50%. Although Vicinanza *et al.* [304] claimed that correlation between the

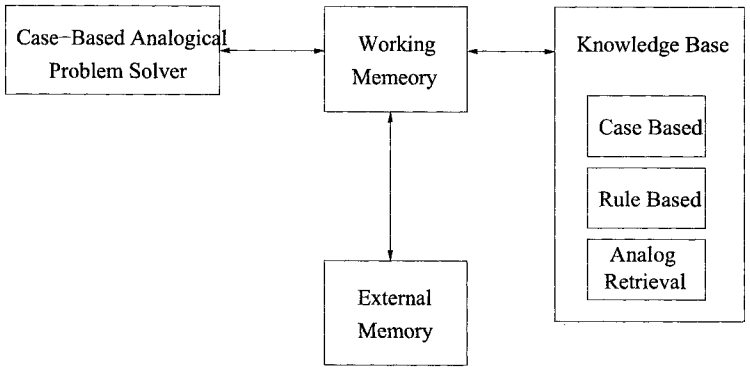

Figure 3.6: The Estor Model

actual effort and the estimates of the Estor model was 97%, however, the experiments were only based on 15 projects.

3.5.3 Genetic Algorithms

Genetic Algorithms (GA) has been applied for the calibration of COCOMO-like models within the context of software cost estimation by Cordero *et al.* [63]. Genetic Algorithms (GA) are local search methods well suited to the optimization of several kinds of functions. They are different from other mathematical approaches because they do not search a solution to the given problem using mathematical algorithms but making a population of solutions evolve towards better ones. These methods have been formulated from the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. Figure 3.7 demonstrates how the evolution exploited by GA is achieved:

- The starting point is a generic population where each individaul/solution has been evaluated with a fitness function. The individuals are encoded in strings representing in some way their DNA information.
- The next set is the creation of an intermediate population (the temporary population in the Figure 3.7), also called mating pool, which is composed by a subset of the solutions of the initial population. The solutions taken from the initial population are randomly selected according to the fitness function (solutions with a higher fitness are more likely to be selected).

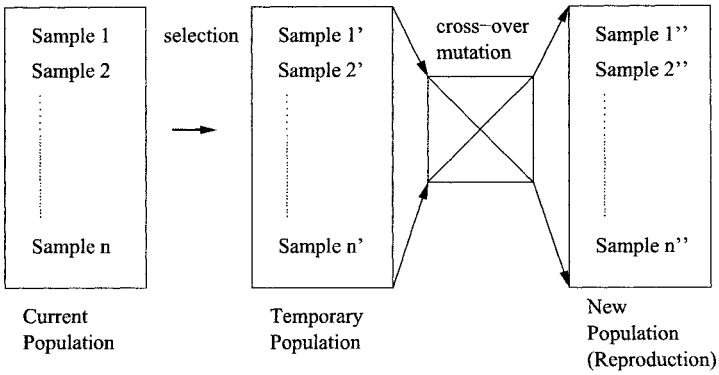

Figure 3.7: The Evolution Exploited by Genetic Algorithm

- When the mating pool is complete it is possible to start the creation of a new population. Pairs of solutions are extracted from this pool until it is empty. The two individuals of each pair (the parents) are then combined to create a completely new pair of solutions (the children), which are inserted into the new population. The combination of the parents is done by means of two operators: cross-over (the DNA of a new individual is created by mixing the DNA of the parents: this is necessary to preserve in the new population the fittest information) and mutation (the process by which a gene is casually altered: this is necessary to explore some new part of the solution space which may contain good solutions).

When the new population is completed, it can be evaluated (by the fitness function) and the process is ready to start again. The entire process will have virtually no end. When applied for searching a solution to a difficult problem a GA will be made to stop when either a sufficiently good solution has been found or the algorithm has gone on for an established number of generations. A simple example has been illustrated in [63].

Cordero *et al.* [63] proposed a new recalibration method which uses Differential Evolution (DE) [99], a recent development of the Genetic Algorithms family, as a search algorithm, instead of the traditional least-square regression. Cordero *et al.* concluded the model involved in Genetic Algorithm performs better than the original one and led to a new version of the cost model.

3.6 Hybrid Techniques

As discussed above, there are many pros and cons of using each of the existing techniques for cost estimation. The disadvantages of current cost estimation approaches have prompted some to urge the use of multiple estimation approaches together [24, 114, 180]. Hybrid techniques incorporate a combination of two or more techniques to formulate the most appropriate functional form for estimation have attracted more attention. This section introduces a few well-established hybrid techniques for software cost estimation.

Many researchers [24, 114, 289] have suggested the use of more than one technique in order to support effort prediction. One of the recent studies to directly explore this area is [169] based on 145 software projects. However, there has been little empirical evaluation till MacDonell and Shepperd's study [196] with the conclusion that using more than one effort prediction technique has some basis. They advise that a dominant prediction technique should initially sought. If so, there is little purpose in trying to combine more than one technique. Otherwise, to employ as diverse a set of techniques as possible. However, statistical analysis in MacDonell and Shepperd's study was unable to reveal any very clear patterns as to which conditions favored which technique. So far, this remains an open research question.

3.6.1 Bayesian-COCOMO II

The Bayesian analysis approach [34, 184] is a well-defined and rigorous process of inductive reasoning that has been used in many scientific disciplines. It provides an effective way of reasoning under uncertainty, and has a firm mathematical background in probability theory [34]. Bayesian analysis has been used to calibrate software cost models (i.e. COCOMO II) by Chulani *et al.* [59, 60]. Bayesian-COCOMO II estimation models presented in [60] provides a formal process for merging expert prior information with software engineering data.

A distinctive feature of the Bayesian-COCOMO II model is that it permits the investigator to use both sample (data) and prior (expert-judgment) information in a logically consistent manner in making inferences. This is done by using Bayes' theorem to produce a 'post-data' or posterior distribution for the model parameters. Using Bayes' theorem prior or initial values are transformed to post-data views. This trans-

formation can be viewed as a learning process. The posterior distribution is determined by the variances of the prior and sampling information, if the variance of the sample information is greater than the variance of the prior information, then a higher weight is assigned to the prior information. On the other hand, if the variance of the sample information is smaller than the variance of the prior information, then a higher weight is assigned to the sample information causing the posterior estimate to be closer to the sample information. Using Bayes' theorem, Chulani *et al.* [59, 60] combined two information sources (a-priori expert-judgment and sampling information)as follows:

$$f(\beta \mid Y) = \frac{f(Y \mid \beta) f(\beta)}{f(Y)} \tag{3.27}$$

Where β is the vector of parameters in which are of interest, and Y is the vector of sample observations from the joint density function $f(\beta \mid Y)$. In Equation 3.27, $f(\beta \mid Y)$ is the posterior density function for β summarizing all the information about β, $f(Y \mid \beta)$ is the sample information and is algebraically equivalent to the likelihood function for β, and $f(\beta)$ is the prior information summarizing the expert-judgment information about β. Equation 3.27 can be rewritten as:

$$f(\beta \mid Y) \propto l(\beta \mid Y) f(\beta) \tag{3.28}$$

In the Bayesian analysis context, the "prior" probabilities are the simple "unconditional" probabilities given sample and prior information.

The composite Bayesian-COCOMO II model discussed in Chulani, Boehm and Steece's paper [60] was claimed to enable stronger solutions to one of the biggest problems faced by the software engineering community: the challenge of making good decisions using data that is usually scarce and incomplete. They noted that the predictive performance of the Bayesian approach (i.e. within 30% of the actuals 75% of the time) is significantly better than that of the previous multiple regression approach (i.e. within 30% of the actuals 52% of the time) on a sample of 161 project data-points. By comparing and contrasting the two empirical approaches, they concluded that the Bayesian approach was better and more robust than the multiple regression approach (i.e. COCOMO II).

Although, Chulani, Boehm and Steece [60] concluded that both a calibration COCOMO II approach and Bayesian-COCOMO II technique are better solutions to resolve some of the counter-intuitive results produced by multiple regression models, both approaches require involvement of expert judgment, which might not be available for

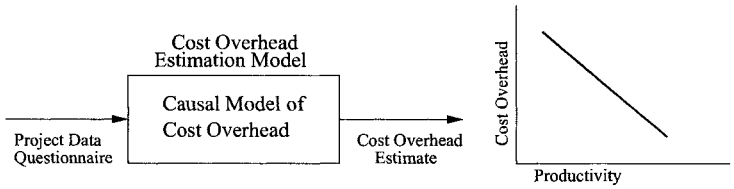

Figure 3.8: COBRA Model

every estimation. Moreover, the problems of data set, which could be the root of the counter intuitive results, still have not been dealt with at all.

3.6.2 Cost estimation, Benchmarking and Risk Analysis (COBRA)

Briand *et al.* [43] proposed a hybrid cost modelling method, COBRA: Cost estimation, Benchmarking and Risk Analysis. This method was based on expert knowledge and quantitative project data from six projects. Encouraging results were reported on a small data set. The core of the COBRA is to develop a productivity estimation model. This productivity estimation model has two components as illustrated in Figure 3.8 [43], the first component is a model that produces a cost overhead estimate. The second component is an productivity model that estimates productivity from this cost overhead. The cost overhead estimation model is developed based on project manager's experience and knowledge. The productivity equation is developed using past project data. Briand *et al.* provided a case study based on six projects in [43]. They ranked twelve important cost drivers from COCOMO cost drivers based on eleven experienced project managers' opinions, then developed a qualitative causal model, which reflects the collective experiences of the organization's senior project managers about the factors that affect cost of development. To use the cost overhead estimation model, the project managers will have to be able to characterize their projects in terms of the factors in the causal model. This is achieved through a questionnaire [43]. After collecting qualitative values of each factor in the casual model, Briand *et al.* quantified the model by determining the magnitude of each of the relationship in the qualitative causal model. A set of multipliers reflecting the expert opinion and its inherent uncertainty were obtained for each of the relationships in the causal model. These multipliers and the project questionnaire variables have to be related in a formal way to cost overhead as a set of equations expressed in [43]. For the model to be used, Monte

Carlo simulation techniques are applied to sample the cost overhead distribution. One can select, for example, the mean of this distribution as the estimated value of cost overhead for a project. Finally the cost estimation was performed by building a linear model that minimized the sum of squares criterion.

Although Briand *et al.* [43] claimed the average absolute relative error of COBRA in predicting cost was 0.09, which is a good accuracy, there were only six samples examined. Like other expert-judgment related methods, the accuracy of the COBRA unavoidably depends on expert's experience, and the method is not easily repeatable.

3.6.3 Fuzzy Logic Related Cost Models

In software metrics, specifically in software cost estimation, many factors, such as the experience of programmers and the complexity of modules, are measured on an ordinal scale composed of qualifications The conventional definitions of categorical cost drivers could cause a serious problem in that it can lead to a great difference between the estimations of two analogous projects. The assignment of linguistic values to the cost drivers uses conventional quantization where the values are intervals. Two similar numerical values of one cost factor could fall into different categories, which leads to very different prediction results. A simple example was illustrated in [126] that two projects have identical cost driver values except DATA factor, which have very close ratio values (9.99 and 10.01 respectively), could have different adjusted effort from 15 Man-Month to 52 Man-Month by the conventional intermediate COCOMO'81 model.

Fuzzy logic [315, 316] has proved sufficient to deal with above problem, therefore, it has been considered to have great potential to improved the utility of traditional cost models. Figure 3.9 [126] shows the linguistic value 'too expensive'; the first using a fuzzy set, and the second using a classical set. The combination between fuzzy logic and conventional cost models allows tolerating imprecision in inputs (cost drivers) and consequently generating more gradual outputs (effort). More recently, some composite cost models involved in fuzzy logic have been established, for example, COCOMO-Fuzzylogic Model [127], Analogy-Fuzzylogic Model [126] and Neurofuzzy model [95]. A drawback [283] of the fuzzy approach is that the relationships are developed from qualitative information of the cost estimating problem, usually elicited from knowledgeable persons. Fuzzy relationships are not primarily empirical models like regression and neural networks.

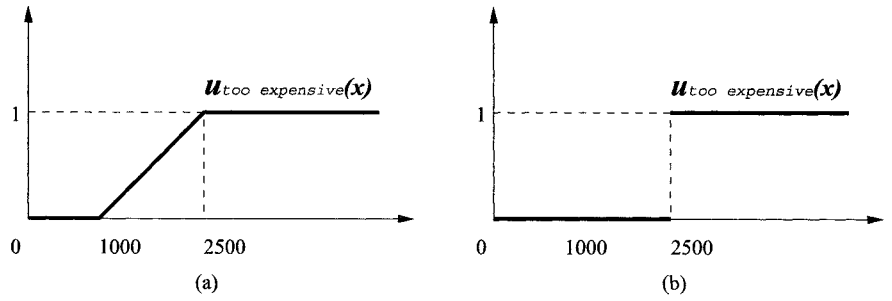

Figure 3.9: Fuzzy set (a) and Classical set (b) for the linguistic value 'too expensive'

COCOMO-Fuzzylogic Model

Idri *et al.* [127] investigated the compatibility of COCOMO'81 with the fuzzy logic system. They re-constructed the intermediate COCOMO'81 model using equation and effort multipliers obtained from fuzzy sets (FC_{ij}) rather than from the classical C_{ij}.

$$\text{MM}_{est} = A \times \text{Size}^B \prod_{i=1}^{i=15} FC_{ij} \quad j = 1 \ldots k_i \tag{3.29}$$

where, MM_{est} is the Man-Month required for development, and Size is the code size measured in SLOC. A and B are constants which are specific to each project mode (organic, semi-detached or embedded) and C_{ij} are the effort multipliers associated with the j^{th} selected rating for the i^{th} cost driver attribute.

FC_{ij} is calculated from the classical C_{ij} and the membership function μ defined for the various fuzzy set associated with the cost drivers:

$$FC_{ij} = \sum_{j=1}^{k_i} \mu_{A_j}^{V_i}(P) \times C_{ij} \tag{3.30}$$

where, the $\mu_{A_j}^{V_i}$ is the membership function of the fuzzy set A_j associated with the cost diver V_i.

According to the above, the corresponding fuzzy sets were defined for each cost driver and its associated linguistic values in Idri *et al.*'s model [127]. The experimental results reported that this 'fuzzy' intermediate COCOMO'81 is less sensitive to the change of the cost drivers. However, there are still many other aspects (such as the definition of project size) of COCOMO remain incompatible with the fuzzy set theory.

Fuzzy Analogy Model

Similar with COCOMO-Fuzzylogic theory, Idri *et al.* [126] integrated fuzzy logic into Shepperd's analogy model called a Fuzzy Analogy model. This model is based on reasoning by analogy and fuzzy logic and extends the classical analogy in the sense that it can be used when the software projects are described either pair numerical or categorical data. Fuzzy Analogy is a fuzzification of the classical analogy procedure. It is also composed of three steps: identification of cases, retrieval of similar cases and case adaption; each step is a fuzzification of its equivalent in the classical analogy procedure. These steps are further detailed in [126]. Although this new model is theoretically sound, there is no comparison research conducted to evaluate this new model. It is still in its infancy.

Neurofuzzy in Software cost Estimating

Machine learning techniques and in particular, artificial neural networks have been successfully used to estimate software engineering costs. However, due to their complexity, the information stored inside the network is not transparent to the designer, making the model difficult to interpret or validate. Neurofuzzy cost models [51, 308, 95], which combined the valuable learning and modelling aspects of neural networks with the linguistic properties of fuzzy systems, enabling adaptive, transparent models to be built.

By comparing with three other estimation techniques and models: Least squares multiple linear regression model, ANGEL, and a general in-house ANN tool, Garratt and Hodgkinson [95] concluded that the neurofuzzy models were quantitatively as least as accurate and consistent as the existing models. Problems related to this research can be summarized as follows: sample size is limited; only two cost drivers (size and duration) were applied; the process is rather complicated.

3.6.4 Optimized Set Reduction

Optimized Set Reduction (OSR) was proposed by Briand, Basili and Thomas [41]. It is a pattern recognition method applied to analyze trends in software engineering data sets build upon Basili's previous work [14, 17]. Based on machine learning and statistics, OSR builds estimation models usable to classify and assess objects. Regarding the

specific needs of software engineering data [41] and the weakness of regression analysis and classification tree analysis, OSR can be used to assess particular characteristics of a factor (e.g., *effort of a project*). It builds an individual model for each object to be assessed. The algorithm selects subsets from historical data that are "optimal" to provide the best characterizations of new objects (i.e., *projects*) to be assessed. These subsets are characterized by models in the form of logical expressions that represents trends in a data set that are relevant to the estimation at hand.

To determine which subsets of the historical data yield the "best" probability distributions on the predicting *effort* range, Briand, Basili and Thomas [41] recommended that a good probability distribution on the *effort* value domain is one that concentrates a large number of pattern vectors in either a small part of the range (if *effort* is continuous) or a small number of predicted variable categories (if *effort* is discrete). One of the commonly used probability distribution evaluation functions - the information theory entropy H [209, 253] was adopted in OSR.

In a historical data set, each of the subsets yielding "optimal" distributions, referred to as optimal subsets, are characterized by a set of conditions that are true for all objects in that subset. Each set of conditions characterizing a subset is called a pattern. Figure 3.10 [41] shows an example of a pattern and its associated probability distribution in the data set. The pattern is composed of three conditions where the variable to be assessed is "productivity". Figure 3.10 shows if these conditions (i.e., ComPLeXity = Nominal, RELiabilitY = Low, DATA base size = High) are true for a new project, then this new project's productivity is most likely to be in the second productivity class. The extracted subsets from various probability distributions across the predicted variable range, and may show different trends. For each extracted subset, a predicted variable prediction is performed by considering the subset extracted as past experience representative of the current problem, using the probability distributions of the extracted subset. The prediction rules are based on Bayesian [184] [300] probability theory. Each pattern prediction is used to make a final global prediction based on redefined decision rules.

OSR dynamically builds a different model for each project to be estimated. For each such project, subsets of "similar" projects are extracted from the underlying historical data set. This is a stepwise process and, at each step, a project attribute is selected to reduce the subset further. The projects that belong to the same class of values as

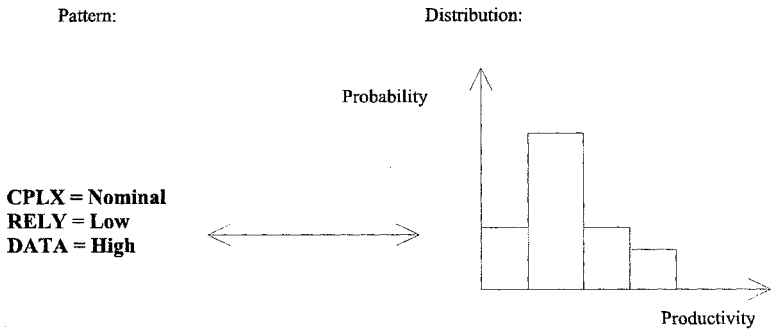

Figure 3.10: An Example of Using OSR Model

the project to be estimated are retained. One major difference to other set reduction methods, such as decision trees, is that the selected variable only has to be a good predictor in the value range relevant to the project to estimate. The subset generation does not partition the data set, but derives a hierarchy. Thus, the generated subsets may overlap, as in Figure 3.11 [41]. Briand, Basili and Thomas [41] analyzed the

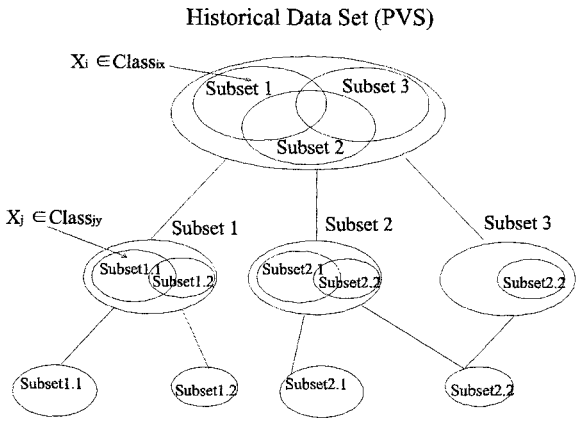

Figure 3.11: OSR Hierarchy

relationship between MRE and entropy, and noted a correlation of 0.71 between these

two variables. A nonparametric Mann-Whitney U test [53] was deployed to test the hypothesis that the MRE's are smaller in classes with low entropy, and a satisfactory level of significance was obtained. They concluded that the lower the entropy values, the more precise the estimates.

3.7 Recent Advances

This section introduced new advances in parametric techniques, non-parametric techniques and other relevant issues of software cost estimation.

3.7.1 Recent Advance in Rayleigh Curve Based Models

Putnam [250] pioneered the use of the nonlinear Norden/Rayleigh model [228] for software cost estimation. As an alternative to the Rayleigh curve, Pillai and Nair [239] proposed an alternative model based on the gamma distribution. In contrast to the Rayleigh curve, the gamma distribution is more symmetrical, and it gives positive manpower values when time equals zero. The Parr/Nair model has the potential for tracking the early stages of manpower build-up better than the Rayleigh model. However, the estimation of the parameters of the Parr/Nair model is not as straightforward as for the Putnam model, and so this model has seen limited practical. More recently, Knafl and Gonzales [175] defined the generalized Weibull model as an extension of the Rayleigh and the Pillai/Nair software manpower estimation models. The authors propose this extended model provided better prediction results and a more general framework for conducting software manpower estimation.

3.7.2 Recent Advance in COCOMO-based Models

Recent advances of COCOMO-based models have focused on adjusting cost factors to cope with change in software development, for example, new domains (e.g., Web development projects) and new development features (e.g., task assignments of project team, CASE tools). These improve the accuracy of estimation models. Machine learning techniques (e.g., Bayesian Analysis) have been introduced into the calibration of COCOMO II models (e.g., calibrate the best-fit weight values of cost factors).

Boehm *et al.* [33] has continuing the calibration and refinement of COCOMO II models, and explained how to use the COCOMO II models (application composition, early design, post-architecture) to cope with change in software development. The emerging extensions of COCOMO II include COCOTS, CORADMO, Applications Composition and other models [26] represent hypotheses of how to model the cost, schedule and quality effects of current and future trends in software engineering practice.

Hale *et al.* [108] proposed estimation adjustment factors based on the task assignments of project team members that can be used to improve the accuracy of existing cost estimation models. Their research demonstrated improvements in the predictive abilities of both COCOMO'81 and COCOMO II when these factors are included.

Reifer [254] proposed a cost model for Web development projects that combines a size measure based on Halstead's Volume measure and a function-point-like table of complexity weights. Size estimates are used in COCOMO-like equations to produce estimates of effort and duration. Reifer [254] proposes eight cost drivers for the effort adjustment factors. As pointed out by the author, there are still a large number of open issues to be resolved. However, his article shows an approach to developing estimation models for this important and growing domain of software engineering.

In 2002, Baik , Boehm and Steece [9] disaggregated and calibrated the CASE Tool variable in COCOMO II. They proposed an extended set of tool rating scales based on the completeness of tool coverage, the degree of tool integration, and tool maturity/user support. Those scales are used to refine the way in which CASE tools are effectively evaluated within COCOMO II. In order to find the best fit of weighting values for the extended set of tool rating scales in the extended research model, a Bayesian approach is adopted to combine two sources of (expert-judged and data-determined) information to increase prediction accuracy.

3.7.3 Recent Advance in Machine Learning Techniques

Learning oriented techniques have been continually investigated in software cost estimation. As introduced in preceding sections, these techniques include Bayesain analysis, Fuzzy logic analysis, Genetic Algorithm (GA), Case Based Reasoning (CBR, e.g., Analogy, Regression Trees, CART), Artificial Neural Networks (ANN). Bayesain anal-

ysis have been adopted in OSR by Briand *et al.* [41, 39, 47] and in extended COCOMO II models by Reifer [254, 256].

Bayesain probability distributions for assessing measurement of subjective software attributes has been studied by Moses *et al.* [217, 218]. Moses and Farrow [219] also used Bayesian inference and the Bayesian statistical simulation program, BUGS, to impute missing data avoiding deletion of observations.

Fuzzy logic techniques have been incorporated into existing models such as COCOMO, Analogy by Idri *et al.* Such hybrid models include COCOMO-Fuzzylogic Model [127] and Analogy-Fuzzylogic Model [126]. Garratt and Hodgkinson also proposed a Neurofuzzy model [95].

The use of the neural networks to estimate software development effort has been viewed with skepticism by the majority of the cost estimation community. Although, neural networks have shown their strengths in solving complex problems, their shortcomings of being 'black boxes' has prevented them being accepted as a common practice for cost estimation. In 2002, Idri, Khoshgoftaar and Abran [128] further investigate the usability of ANN in cost estimation by mapping ANN to a fuzzy rule-based system. Idri *et al.* made a hypothesis that if the obtained fuzzy rules are easily interpreted, the neural network will also be easy to interpret. Their experiments showed that the capability of explaining the meaning of the output and the propositions composing the premise of each fuzzy rule. However, their operator seems to be inappropriate to combine the effects of the various fuzzy propositions on the output of the fuzzy rules in software cost estimation. Consequently, further research is needed in order to extract more comprehensive fuzzy rules from the type of neural network used in their experiment. Although the conclusion is not very encouraging, it was the first attempt at interpreting ANNs within software project effort estimation context.

CBR in effort estimation has been continuously investigated, and focus on improving search methods for optimal feature subsets. Shepperd *et al.* [157, 166, 164, 246] have pioneered the research, and present a series of search techniques to help identifying key features which are essential for constructing CBR cases. These search techniques including random searching, hill climbing, forward sequential selection (FSS), using meta-data analysis to guide retrieval in CBR for software cost prediction, etc.

The OSR [43] algorithm has been generalized and enhanced [49]. OSR is now

applicable in solving classification and regression problems and thus builds a set of algorithms, rather than one single algorithm.

Data clustering approaches have been introduced to the software estimation field by Tian [299] in 2002 to provide better reliability assessment and prediction. The new approach models software reliability by grouping data into clusters of homogeneous failure intensities. This series of data clusters associated with different time segments can be directly used as a piecewise linear model for reliability assessment and problem identification, providing meaningful results early in the testing process.

Jeffery *et al.* [134] performed a replication of the studies (Briand *et al.* [46], 1996 and Briand *et al.* 2000 [45]) on a data set available from the International Benchmarking Standards Group (ISBSG). This version of the data set contains data worldwide collected on more than 700 software projects, mainly management information systems. In addition to the estimation methods used in the two previous studies, robust regression was applied, which performed most accurately. One notable difference was that analogy performed relatively well.

An evolutionary computation method, Grammar Guided Genetic Programming (GGGP) was applied by Shan *et al.* [273] to fit models, which aims at improving the prediction of software development costs. As reported in [273], results are significantly better than those obtained by simple linear regression and by Log regression.

3.7.4 Other Advanced Techniques and New Directions

More critical evaluation of current effort prediction methods, published results, evaluation criteria has been conducted. For example, general comments on prediction methods by Jørgensen *et al.* [152, 149, 150, 151, 153], studies on evaluation criterion MMRE by Foss *et al.* [93] and Stensrud *et al.* [290, 291], and evaluation of several nonparametric bootstrap methods to estimate confidence intervals for software metrics by Lei *et al.* [185].

The following highlights emerging developments including combining techniques to optimize effort predictions; analyzing historical data sets with missing values or sparse data; simulation techniques in software effort estimation; cost estimation for web applications etc.

The Meta Level Learning For Software Cost Estimation

As Shepperd *et al.* [198] pointed out within the field of empirical software engineering, attention is turning to how to combine results in order to construct bodies of knowledge. There is a lack of cohesion picture of which type of prediction system is to be preferred with no single technique dominating, and a realization that there is marked relationship between the study setting or context and the performance of different prediction systems. Meta level learning has been considered as a means of finding models relating variables such as prediction technique, data set size and amount of noise to accuracy. One of the most recent EPSRC funded projects (dec.bmth.ac.uk/ESERG/MeLLow) led by Shepperd *et al.* has focusing on meta level learning for software cost estimation. Shepperd and Mair [198] are undertaking an initial research based on 12 seemingly similar published studies, for the meta analysis within empirical software engineering. Shepperd and Mair found a number of difficulties in making a comparison and integrating the data based on those 12 studies, and concluded there is an overwhelming need for an agreed protocol and mechanism for publishing and integrating results. A minimal list of how to present results in predicting effort was presented in their research [198].

Analyzing historical data sets with missing values or sparse data

Missing values are often encountered in data sets used to construct effort prediction models. This causes two significant problems. First, it means information is wasted. Second, missing data risks biased analysis and so inaccurate models. Handling missing data [224, 56, 276, 219] in historical data sets become one of the new interests in the field.

Myrtveit *et al.* [224] evaluated four missing data techniques (MDTs) within the context of software cost modeling: listwise deletion (LD), mean imputation (MI), similar response pattern imputation (SRPI) and full information maximum likelihood (FIML). Myrtveit *et al.* applied the MDTs to an EPR data set, and thereafter constructed regression-based prediction models using the resulting data sets. The evaluation suggested that only FIML is appropriate when the data are not missing completely at random (MCAR). Otherwise, the other three techniques would be appropriate.

Strike *et al.* [296] evaluated three techniques for dealing with missing data in the context of software cost modeling, namely, they are listwise deletion, mean imputation,

and eight different types of hot-deck imputation. Strike *et al.*'s results indicated that all these missing value techniques perform well with small biases and high precision. The authors suggested that the listwise deletion is the simplest technique, while hot-deck imputation with Euclidean distance and a z-score standardization is the consistently best performed technique. Lately, Cartwright *et al.* [56] also introduced a number of imputation techniques to deal with missing values, for example, ignoring methods, mean imputation, hot deck methods, multiple imputation, likelihood methods.

Moses and Farrow [219] carried on the research on estimate effort using data sources with missing data from a different angle - adding in simulated data. They used Bayesian inference and the Bayesian statistical simulation program, BUGS, to impute missing data avoiding deletion of observations, and increasing sample size from 339 to 616. Providing that by imputing data distributional biases are not introduced, the accuracy of inferences made from models that fit the data will increase. As a consequence of the imputation, models that fit the data and explain about 59% of the variability in actual effect are identified.

Shepperd and Cartwright [276] introduced a sparse data method (SDM) based upon a pairwise comparison technique and Saaty's Analytic Hierarchy Process (AHP). AHP has been widely used for multi-criteria decision making [267, 268]. Shepperd and Cartwright claimed that based on the minimum data requirement (a single known point), the SDM has capacity of adding value to expert judgment by producing more accurate and less biased results. They concluded the SDM is promising and may help overcome some of the present barriers (e.g. the absence of reliable and systematic historic data) to effective project prediction.

Simulation techniques in software effort estimation

To assess the ability of effort prediction techniques in large complex environments a large training set of project data containing details of numerous development attributes is required. Generally, this data has not been available. Since it has been proven difficult [277, 165] to obtain significant results over small data sets, some researchers have advocated the use of simulated data [238]. Wittig and Finnie [311] applied the SPQR/20 [141] software estimation tool to generate simulated software development project data, and claimed that SPQR/20 has given relatively good results in estimating development development effort in a locally calibrated environment [132].

Angelis and Stmelos [5, 6] presented a statistical simulation tool, namely the bootstrap method, which helps the user in tuning the analogy approach in software effort estimation. As claimed by the authors, the using of simulation procedure improves the applicability and the reliability of the estimation by analogy methods for software projects.

The idea of artificially generated data with known properties to explore software engineering data set modeling techniques was also proposed by Pickard *et al.* [238]. The use of the simulation procedure to create data sets with a known underlying model and with non-Normal characteristics that are frequently found in software data sets: skewness, unstable variance, and outliers and combinations of these characteristics.

Followed Pickard *et al.*'s suggestion [238]. Shepperd and Kadoda [278] generated random data using a simple Gamma to allow both control and the possibility of large (1,000) validation data sets. Shepperd and Kadoda then compared stepwise regression (SWR), rule induction (RI), case-based reasoning (CBR) and artificial neural nets (ANN) based on three simulated data sets (which distinguished by predicted variable's characteristics - continuous, discontinuous and random). They concluded that SWR tended to do best with the continuous "cost function", while the Marching Learning (ML) approaches performed better when the "cost function" was discontinuous; and SWR is to be preferred when the data set is approximately normally distributed, while CBR is strongly to be preferred if the data set has outliers and collinearity.

As discussed in the above, simulation modeling is increasingly being used [5, 6, 277, 278, 165, 93, 185, 219], and researchers have published promising results, but using simulation techniques in cost estimation is still at its early stage. Many important lessons have emerged, which have been discussed in [278]. For example, there is still need to explore a richer set of "cost" functions. For instance, Gamma distribution which is quite prevalent in software project data [175, 278] to simulate prediction data sets.

Cost Estimation for Web Applications

Currently, effort estimation for Web applications is gaining increasing interest among academic researchers and practitioners. Mendes *et al.* [205, 206] illustrated several case studies on the evaluation of the development of Web applications by using length

and complexity metrics in ordinary least square prediction models. Their experiments consider only the resulting systems (Web pages and links), while Baresi *et al.* [13] focus their analysis on effort of designing artifacts. Baresi *et al.* identified a number of predictors (subjects, applications, availability of tools, notation, classes and construct validity) which may influence the effort estimation for the design of Web applications. In their extended research, Mendes *et al.* [207, 208] compared the prediction accuracy of three case-based reasoning techniques to estimate the effort to develop Web hypermedia applications. The COSMIC-FFP [130] approach was used to measure functional size.

Reifer [254, 256] tailors the COCOMO II model to Web development by revising the cost factors, and introduced Web objects, application points and multimedia points as new sizing metrics for Web applications. The Web objects metric seems the most appropriate [13, 230]. Reifer [255] then proposed a new estimation model called the WEBMO cost model as an extension of the COCOMO II early design model. Similarly, Rule *et al.* [265, 266] proposed an adaptation of the COBRA method [43] to the Web domain and applied it to the data from 12 projects developed by a small Australian company.

Ochoa *et al.* [230] present a method to estimate fast the development effort of Web-based information systems. The method is called Chilean Web Application Development Effort Estimation (CWADEE) and it is claimed that it could get effort estimations in a short period of 24 to 72 hours using limited information. This method is simple and specially suited for small or medium-size Web-based information systems, and requires historical information. Ochoa *et al.* defined another new size metric called data web points (DWPs) in implementing their CWADEE method.

Moløkken- Østvold and Jørgensen [215] studied estimation of Web application development from a very different angle. They studied and compared the prediction accuracy between two main stake-holders: technical (e.g. programming) and non-technical (e.g. user interaction design) roles in Web development. Surprisingly, their results suggested that people with technical competence provided less realistic project effort estimates than those with less technical competence. This means that more knowledge about how to implement a requirement specification does not always lead to better estimation performance.

Although some research in estimating Web application effort has been done, it is still immature. This is reflected on several issues, such as there are no standard sizing

measures; most of above research only yields pilot results, which are based on small and medium-size projects (mainly students' projects).

3.8 Cost Model Evaluation Criteria

It has been summarized by Shaw [274] in Carnegie Mellon University that, "*Science and engineering research fields can be characterized in terms of the kinds of questions they find worth investigating, the research methods they adopt, and the criteria by which they evaluate their results*". Good research requires not only a result, but also clear and convincing evidence that the result is sound. This section lists the common evaluation criteria.

3.8.1 Relative Error (RE) and Mean RE ($\overline{RE}$)

To show how the actual value of an estimate (E) and predicted value $\widehat{E}_i$ relate to each other, the Relative Error (RE) [61] is defined as:

$$RE_i = \frac{E_i - \widehat{E}_i}{E_i} \tag{3.31}$$

and mean relative error of a set of n projects by the formulae:

$$\overline{RE} = \frac{1}{n}\sum_{i=1}^{n} RE_i \tag{3.32}$$

If a model is a good representation of effort expenditure, it will lead to small values of Relative Error (RE) and generally to a small $\overline{RE}$. However, since it is possible that large positive REs can be balanced by large negative REs, a small $\overline{RE}$ may not imply that a model is a good one. Hence, this measure may not be useful in practice [61]. In view of the problem with RE and $\overline{RE}$, Conte *et al.* [61] defined the Magnitude of the Relative Error as MRE.

3.8.2 Magnitude of Relative Error (MRE) and Mean MRE (MMRE)

The Magnitude of Relative Error (MRE) is one of the most commonly used criteria of the evaluation of cost estimation models [61] [44] [279]. A large positive MRE would

suggest that the model generally overestimates the effort, while a large negative value would indicate the reverse. MRE is defined as:

$$MRE_i = \frac{\mid E_i - \widehat{E}_i \mid}{E_i} \tag{3.33}$$

The MRE value is calculated for each observation i whose effort is predicted. The aggregation of MRE over multiple observations, say n, can be achieved through the Mean MRE (MMRE), which is the mean of absolute percentage errors:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\mid E_i - \widehat{E}_i \mid}{E_i} \tag{3.34}$$

This is a widely used criterion. A weakness of MMRE is that it is asymmetric since underestimates cannot exceed 100% whereas overestimates are unbounded [277]. Another problem of MMRE is that small actuals (small $\widehat{E}_i$) will tend to give a large MMRE [93, 291].

3.8.3 The Median of MRE (MdMRE)

Notwithstanding the above, the MMRE is sensitive to individual predictions with excessively large MREs (observation outliers). Therefore, an aggregate measure less sensitive to extreme values should also be considered. namely the median of MRE values for the n observations (MdMRE).

3.8.4 The Balanced MMRE (BMMRE)

There has been some criticism of the MMRE measure, in particular that it is unbalanced and penalized overestimates more than underestimates. For this reason, Miyazaki *et al.* [211] also proposed a balanced mean magnitude of relative error measure as follows:

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\mid E_i - \widehat{E}_i \mid}{min(E_i, \widehat{E}_i)} \tag{3.35}$$

Other evaluation metrics such as the mean of the balanced relative error BRE and the inverted balanced relative error IBRE were proposed by Miyazaki *et al.* [212].

3.8.5 MER and Mean MER (MMER)

More recently, an alternative evaluation metrics MER that was proposed by Kitchenham [170] is claimed to be preferable to MRE [170, 93] since it measures the error relative to the estimate. MER is defined as:

$$MRE_i = \frac{\mid E_i - \widehat{E}_i \mid}{\widehat{E}_i} \tag{3.36}$$

The MER value is calculated for each observation i whose effort is predicted. The aggregation of MER over multiple observations, say n, can be achieved through the Mean MER (MMER), which is the mean of absolute percentage errors:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\mid E_i - \widehat{E}_i \mid}{\widehat{E}_i} \tag{3.37}$$

3.8.6 The Median of MER (MdMER)

The MMER is sensitive to individual predictions with excessively large MERs (observation outliers). Therefore, an aggregate measure less sensitive to extreme values should also be considered. namely the median of MER values for the n observations (MdMER).

3.8.7 The Coefficient of Multiple Determination (R^2)

The coefficient of multiple determination (R^2) [4, 61] is among the most popular measures used to indicate the extent to which E_i and $\widehat{E}_i$ are linearly related. R^2 is defined for n pairs of E_i and $\widehat{E}_i$ values as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(E_i - \widehat{E}_i)^2}{\sum_{i=1}^{n}(E_i - \bar{E})^2} \tag{3.38}$$

R^2 is meaningful when only one predictor variable is involved. When $\widehat{E}_i$ is obtained through multi-linear regression, R^2 is frequently used to denote the percentage of variance accounted for by the predictor variables used in the regression analysis. A high value of R^2 suggests either that a large percentage of variance is accounted for, or that the inclusion of additional predictor variables in the model is not likely to improve the model much. The value of R^2 does not reflect how closely E_i and $\widehat{E}_i$ correspond to one another in absolute sense.

3.8.8 The Prediction at Level l (PRED(l))

A complementary criterion that is commonly used is the prediction at level l [45, 280, 286, 89, 131, 160, 221, 23], known as Pred(l),

$$PRED(l) = \frac{1}{n}\sum_{i=1}^{n} a_i \quad a_i = \begin{cases} 1 & \text{if MRE}_i \leq l \\ 0 & \text{if MRE}_i > l \end{cases} \tag{3.39}$$

where a_i is the number of observations where MRE is less than or equal to l. MMRE and Pred(l) are, respectively, measures of the spread and the kurtosis of the variable z, where z = estimate/actual [170]. In this research $z = E_i/\widehat{E}_i$.

3.8.9 Mean Square Error ($\overline{SE}$) and the Relative Root Mean Squared Error ($\overline{RMS}$)

$$\overline{SE} = \frac{1}{n} \times \sum_{i=1}^{n} (E_i - \widehat{E}_i)^2 \tag{3.40}$$

The argument for using the mean square error is that if you are risk averse it penalizes large errors more than small errors, however, it suffers from the problem of being difficult to compare values across data sets [277]. Moreover, $\overline{SE}$ is meaningful for regression models only [61]. It represents the mean value of the error minimized by the regression model. For $\overline{SE}$, we can compute the root mean square error (RMS) as:

$$RMS = (\overline{SE})^{1/2} = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (E_i - \widehat{E}_i)^2} \tag{3.41}$$

and the relative root mean square error ($\overline{RMS}$) defined by

$$\overline{RMS} = \frac{RMS}{\frac{1}{n} \times \sum_{i=1}^{n} E_i} \tag{3.42}$$

Conte *et al.* [61] suggested that $\overline{RMS} \leq 0.25$ could be an acceptable performance for a model.

3.8.10 Standard Deviation (SD)

A measure of residual error is the standard deviation (SD). It is computed as follows [93]:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(E_i - \widehat{E}_i)^2}{n-2}} \tag{3.43}$$

SD is positively correlated with size because software project data sets are often heteroscedastic.

3.8.11 Relative Standard Deviation (RSD)

Relative Standard Deviation is computed as follows [93]:

$$RSD = \sqrt{\frac{\sum_{i=1}^{n}(\frac{E_i - \widehat{E}_i}{x_i})^2}{n-1}} \tag{3.44}$$

The variable x could be project size (e.g. Function Points). The rationale behind RSD is to measure the dispersion relative to the x value rather than relative to the E (effort) value to avoid one of the problems with MMRE that small actual (small $\widehat{E}_i$) which has been divided by a small number tends to be a large number [93, 291]. As opposed to SD, RSD is almost uncorrelated with size. However, we cannot ignore that RSD is limited to models with a single predictor variable [93].

3.8.12 Logarithmic Standard Deviation (LSD)

Logarithmic Standard Deviation is computed as follows [93]:

$$LSD = \sqrt{\frac{\sum_{i=1}^{n}(e_i - (-\frac{s^2}{2}))^2}{n-1}} \tag{3.45}$$

The term s^2 is an estimator of the variance of the residual e_i, where e_i is given by

$$e_i = lnE_i - ln\widehat{E}_i \tag{3.46}$$

LSD is useful for comparing multiplicative models but it may be inappropriate for comparing additive models [93].

3.8.13 Discussion on Software Cost Model Criteria

The accuracy of cost estimation models is commonly measured with two or three indicators. The fact is that these accuracy indicators are based on only two terms [63]: the actual and estimated effort. These two terms have allowed easy comparison of the estimation accuracy of different methods. It is suggested by Fenton [88] and Conte *et al.* [61] that a model is acceptable if 75% of the predicted values fall within 25% of their actual values. Unfortunately, there is no generally accepted standard, and most models reported have failed based on this criterion. Complexity and size estimation are considered the main cause for such inaccurate estimation. For example, Boehm [25] stated that most researchers working on developing models of cost estimation all faced the same dilemma: as software grew in size and importance it also grew in complexity, making it very difficult to accurately predict the cost of software development.

As presented in the preceding subsections, various evaluation criteria have been proposed, however, it is unfortunate that these criteria are often not in agreement for a set of models and projects in the sense that we cannot say which model is best without making a subjective judgment on the relative importance of the evaluation criteria. Conte *et al.* [61], suggested that we can chose the models which have smaller average errors, and more predictions that fall within 25% of the actual values.

Many inconsistent effort estimation model evaluation results have been reported across the field, and researchers [93, 290, 277] have started to question the effectiveness of the fundamental evaluation criteria such as Magnitude of Relative Error (MRE). In these studies, the conclusions partly discourage the use of MRE and MMRE and propose other evaluation criteria. For example, Stensrud *et al.* [290] reported a negative correlation between MRE and project size based on four data sets: 1) Albrecht data set, 2) Kemerer data set, 3) Finnish data set, and 4) DRM and Accenture-ERP data set. The authors then concluded MRE might mislead the results analysis, and MMRE is not an appropriate measure of the expected MRE for small and large projects. They recommended that the data set be partitioned into two or more subsets and that MMRE is reported per subset. A year later, the same authors [291] conducted a further empirical evaluation by plotting MRE against the predicted effort rather than against the actual effort based on the same four data sets. They obtained very different results from their previous study, then amended their conclusions, stating that there are no reasons to dismiss the use of MRE and MMRE. However, it is a necessary,

but not sufficient, condition that MRE and project size are uncorrelated. There are a number of other requirements that need to be met in order to ensure MRE is a reliable evaluation metric. Other studies by Kitchenham *et al.* [170] and by Foss *et al.* [93] suggested there are other serious flaws with MMRE.

Foss *et al.* [93] recommend not using MMRE to evaluate and compare prediction models. Based on their research, MRE related metrics (e.g. MdMRE) are less reliable than SD, RSD and LSD. However, they do not have any universal replacement for MMRE, and only suggested using a combination of theoretical justifications of the models that are proposed in [93], including Standard Deviation (SD), Relative Standard Deviation (RSD), and Logarithmic Standard Deviation (LSD).

3.9 Comparative Evaluation of Cost Modelling Techniques in Literature

Although a large number of software estimation approaches have been proposed, attempts at validating them have been largely unsuccessful. For example, uncalibrated models may average as much as 600 percent relative error [160]. Even with local calibration and attention, their use in industry is often restricted to the verification of estimates generated by manual techniques [317].

The comparative evaluation of effort modelling techniques has been the focus of many studies. The main purposes of these researches fall into the following categories:

1. Determine which technique has the greatest effort prediction accuracy [160, 264, 313, 201, 180, 89, 114, 91, 286, 306, 44, 45, 289],

2. Propose new or combined techniques that could provide better estimates [92, 138, 41, 43, 61, 279, 292, 59, 60, 175, 126], and,

3. Relatively few researchers [44, 45, 133, 134] examine the difference between estimates derived from multi-company or company specific data sets.

Many researchers [174, 264, 160, 313, 180, 120, 83, 114, 163, 43, 289] have discussed the pros and cons of software cost estimation techniques and presented analysis results. Experimental [213, 264, 180, 114] results reported in the field showed that estimates

made by the different models for the same project exhibited wide variation. As discussed by Jeffery *et al.* [134], due to the differences in the questions addressed, the techniques applied, the data sets used, and the designs applied for the companies, it is difficult to draw general conclusions from the results obtained. In this section, we only highlight representational researches which have been cited frequently in the field.

Kitechenham and Taylor [173, 174] compared the performance of the SLIM model and COCOMO model by plotting actuals vs estimates. However, the plots showed little systematic relation. Their conclusions were to reject the use of either model under ICL and British Telecom (BT) environment (e.g. a real-time and multi-sites development) and to establish their own historical database. The ICL/BT data set only contains 33 data points. Kitechenham and Taylor suggested that cost models such as SLIM and COCOMO cannot be used indiscriminately in any environment. Another study on COCOMO also found high error rates, averaging 166% [210].

Kemerer [160] performed an empirical validation of four parametric models (SLIM, COCOMO, Estimacs, and FPA). No re-calibration of models was performed on the project data, which was different from that used for model development. Most models showed a strong over estimation bias and large estimation errors, ranging from a MMRE of 85% to 772%. All of these four models tested failed to sufficiently reflect the underlying factors affecting productivity. Due to the limited sample size, Kemerer's experimental results only suggest certain hypotheses that may verifiable with further research [160]. Inappropriate choice of the technology factors (predictor variables) makes estimation (e.g. SLIM) extremely unreliable [173].

A similar study was carried out by Rubin [264]. A project description was sent to Jensen (Jensen's model), Green (Putnam's SLIM), Rook (GECOMO) and to himself (Rubin's Esimacs). The main purpose was to compare and contrast the different sort of information required by the four models. However, the estimates vary significantly, MMRE ranged from 20% (Estimacs estimate) to 840% (SLIM estimate). Rubin explained that the models are based on different databases of completed projects and have not been calibrated and the four participants made different assumptions in choosing the settings of the cost driver.

Wrigely's research [313] clarified and compared some of the more widely cited models which had been proposed during the seventies. These approaches were categorized as Expert judgment, Algorithmic, Bottom-up and Top down approaches. The

models were reviewed including Putnam's SLIM, Albrecht's Function Point analysis, Boehm's COCOMO'81, Bailey-Basili's meta model, Rubin's ESTIMACS model, Jones' SPQR/20 estimating product and Vosburgh's work in investigating the factors affecting productivity. Based on the empirical review of above researches, Wrigely [313] mainly criticized the factors of sizing, effort, scheduling, historical data, design and factor independence as the following

- The rationality of using lines of code, or any other output metric of a system development effort, as an input into an estimating model is questionable.
- There are insufficient studies to accurately assess the effects of predictor factors on the development effort.
- The change of software development life cycle (e.g. shifts in effort away from coding towards more analysis and design) made traditional schedule estimation inappropriate.
- The major problem with using historical data is that much of the development effort is technology dependent (e.g. languages, compilers, automated tools). Changes in technology in recent years may dominate any information that historical data may provide.
- The issues of design quality has been largely ignored in all of the models reviewed.
- There is the implicit assumption in all estimating models to date that the factors which affect development effort are orthogonal. This may be due to insufficient data points to check for interdependencies.

Kusters *et al.* [180] conducted a comparative analysis among Function Point analysis, COCOMO'81, Price and Estimacs based on data from 14 projects. On the basis of the differences found between the estimates and reality, Kusters *et al.* concluded that it has not been shown that the selected models can be used for estimating projects at an early stage of system development. They pointed out simply using a model without adapting it to the environment in which it is used will not lead to accurate results.

Jeffery and Low [131] investigated the need for model calibration at both the industry and organization levels. Without model calibration, the estimation error was large, with MMRE ranging from 43% to 105%. Jeffery, Low and Barnes [132] later

compared the SPQR/20 model to FPA using data from 64 projects from a single organization. The models were recalibrated to the local environment to remove estimation bias. Improvement in the estimate was observed with a MMRE of 12%, reflecting the benefits of model calibration.

Hihn's survey [120] in 1991 indicated the majority of the technical staff estimating software costs used informal analogy and high level partitioning of requirements.

Heemstra [114] reviewed and compared 29 cost models in 1992. To conduct a comparative analysis, Heemstra defined five model requirements, two application requirement and six implementation requirements. After a severe selection, only the software packages Before You Leap (BYL) and Estimacs models were considered qualified. BYL is a commercial package based on a link-up between FPA and COCOMO. However, the experiments of Heemstra showed neither BYL nor Estimacs can be used for a reliable estimation tool at an early stage of software development. He suggested data collection is necessary and important and using more than one estimation techniques might improve the prediction accuracy.

Ferens and Gurner [89] evaluated three models (SPANS, Checkpoint, and COSTAR) using 22 projects from Albrecht's database and 14 projects from Kemerer's data set [89]. The estimation error was also large, with MMRE ranging from 46% for the Checkpoint model to 105% for the COSTAR model. The study showed that the three function point models can be useful in effort estimation, especially if they can be adjusted for bias. Based on the percentage tests, the unadjusted Checkpoint model was more accurate than the other two models. However, when models were adjusted for bias, the results were very similar.

Learning oriented techniques were introduced to the discipline of software estimation in the mid 1990's, centering on artificial neural network estimation models. Results from studies such as Khoshgoftaar [163], Samson [269] and Wittig [311] showed highly promising results. For example, Finnie and Wittig [90] applied artificial neural networks (ANN) and case-based reasoning (CBR) to estimation of effort. Based on an early version of ISBSG data set, ANN was able to perform within 25% of the actual effort in more than 75% of the projects and with a MMRE of less than 25%. However, the result from CBR were less encouraging. In 73% of the cases, the estimates were within 50% of the actual effort and for 53% of the cases, the estimates were within 25% of the actual. In a separate study, Mukhopadhyay [221] found that an expert system

based on analogical reasoning outperformed other models. Srinivasan and Fisher [286] used machine learning approaches were found to be competitive with SLIM, COCOMO and function points compared to the previous study by Kemerer [160].

Smith and Mason [283] examined the performance, stability and ease of cost estimation modeling using regression versus neural networks to develop cost estimation models. Results showed that neural networks have advantages when dealing with data that does not adhere to the generally chosen low order polynomial forms, or data for which there is little a priori knowledge of the appropriate relationship to select for regression modeling. However, in cases where an appropriate relationship can be identified, regression models have significant advantages in terms of accuracy, variability, model creation and model examination.

Walkerden and Jeffery [306] studied several methods of analogy-based software effort estimation compare with a simple linear regression model. The results showed that people are better than tools at selecting analogues for the data set used in the study. Estimates based on analogues selected by people (25 Masters students), with a linear size adjustment to the analogue's effort value, proved more accurate than estimates based on analogues selected by tools, and also more accurate than estimates based on the simple regression model. In terms of MMRE, the best performance was 39%, while OLR performed MMRE 67%. The data set used in Walkerden and Jeffery's study consists of 19 projects with function point counts, total effort and a collection of other project related metrics.

In 1999, Briand *et al.* [44] assessed and compared the accuracy of a selection of common cost modeling techniques, namely, Ordinary Least Squares regression (OLS), Stepwise ANalysis Of VAriance (ANOVA), Analogy, and CART. Their research also used organizational specific data as compare to multi-organizational data. The study was based on a database consisting of 206 business software projects from 26 companies in Finland. The experimental results showed that the selected modeling approaches did not show large differences according to the three standard evaluation criteria (i.e. MMRE, MdMRE and Pred(0.25)). CART models performed MMER 52% based on multi-organizational data and 56.9% based on organizational specific data. Briand *et al.* [44] concluded that CART model seemed to be a good estimation model, both from accuracy and interpretability points of view; and local models developed using the one-company database did not performed significantly better than

the models developed using external multi-organization data. Late in the same year, Briand *et al.* [42] replicated a comprehensive comparison of the above four techniques and two new combined approaches within different organizational context, using data from European Space Agency. These two new approaches are combined CART and OLS and combined CART and Analogy. Their research results showed that traditional techniques, namely, ordinary least squares regression and analysis of variance outperformed analogy-based estimation and regression trees. Consistent with the results of the replicated study, no significant difference was found in accuracy between estimates derived from company-specific data and estimates derived from multi-organizational data. In terms of MMRE, OLS performed 0.39% based on multi-organization data, and 0.43% based on company-specific data. While Analogy performed MMRE 273% and 103%; combined CART and OLS performed MMRE 101% and 107%; combined CART and Analogy performed MMRE 97% and 115% respectively.

Briand *et al.* [44] stated that Conte *et al.* used six data sets from widely differing environments and reported an MMRE variation between 70% and 90% for their three tested models: SLIM, COCOMO, and Jensen's model [136]. As a result of their investigation, Conte *et al.* proposed a new model COPOMO [61] calibrated separately to the six data sets. This new model yielded a MMRE of 21%.

Mair *et al.* [197] outlined comparative research in to the use of artificial neural networks (ANNs), case-based reasoning (CBR) least squares regression (LSR) and rule induction (RI). Examination of the results showed that the ANN technique appeared to perform best followed by CBR and LSR and lastly RI. However, Mair *et al.* found it took considerable effort to configure the neural net and it required a fair degree of expertise. It was difficult to see how ANN techniques could be easily be used within the project estimation context by end-users. Mair *et al.* pointed out [197] that if one adopts a broader perspective (accuracy, explanatory value, and configurability) than merely focusing upon accuracy, neural nets no longer become the obvious choice for building prediction systems.

Although learning oriented techniques have been introduced and are considered to have great potential in effort prediction, Stensrud [289] argued in 2001 that only statistical techniques deal properly with stochastic variation. Therefore, the effort predictions by non-parametric approaches inspire less confidence, however promising the empirical results are, comparing with predictions by parametric approaches. If the effort predic-

tions are to be trustworthy, the non-parametric approaches should be combined with statistical techniques such as e.g. Briand *et al.* [41, 40] and Jørgensen [148]. Thus, Stensrud concluded that ANGEL,ACE, CART and OSR may add value in exploratory data analysis whereas Artificial Neural Networks (ANN) do not [289]. By comparing ANGEL and regression, Stensrud [289] also observed that regression analysis assists the user in understanding the historical data by synthesizing the data, i.e. by data reduction, whereas ANGEL adds value by drilling down in the data. Stensrud [289] suggested these two approaches may then be seen as complementary aids to a project manager.

As discussed above, despite the intense research activities, few generalizable conclusions can be drawn. There are some common weakness of most of the above studies that only small subsets of modelling techniques were explored; based on small data sets, Table 3.1 summarizes such problems from the literature, and initial evaluations are rarely replicated, hence one can only have limited confidence in the conclusions drawn.

As advocated by Briand *et al.* [42], the research in software cost engineering should perhaps put less emphasis on investigating ways of achieving better predictive accuracy through new data modeling procedure, but re-direct that research effort to other questions that call for more investigation, such as subjective effort estimation [123], modeling based on expert knowledge elicitation [43], techniques combining expert opinion and project data [60]. Shepperd and Kadoda [276] suggest that there is a strong relationship between the success of a particular technique and training set size, nature of the "cost" function and characteristic of the data set (outliers, collinearity, number of features, number of cases etc.), concluding that the "best" prediction technique might not be the right idea to follow. Data analysis has gaining increasing interest among researchers [313, 114, 168, 223, 202, 203, 188] e.g. identifying significant cost factors, removing outliers which strongly affect the results of the prediction, dealing with interrelated factors, etc.

3.10 Common Problems of Cost Estimation

Current cost estimation techniques have a number of drawbacks. For example, developing algorithmic models requires extensive past project data [43]. Off-the-shelf models

Year	Representative and References	Model	Data Set	Sample Size	Preliminary Analysis
1966	Nelson [228]	SDC Regression Model		169	No Evidence
1977	Walston *et al.* [307]	IBM-FSD Model	Self-collected	60	No Evidence
1981	Bailey [10]	Meta-Model	Self-collected	18	No Evidence
1981	Boehm [24]	COCOMO 81	COCOMO 81	63	No Evidence
1983	Albrecht [4]	Function Points	IBM Data Set	22	No Evidence
1983	Behrens [20]	Function Points	Self-collected	25	Regression Analysis
1983	Rubin [263]	ESTIMACS	Proprietary	N/A	Proprietary
1984	Vosburgh [305]	Multivariate Regression Model	Company-supplied	44	Regression Analysis
1985	Kitechenham *et al.* [174]	SLIM and COCOMO	ICL and BT	33	Regression Analysis
1986	Conte *et al.* [61]	SLIM, COCOMO, SOFTCOST, and Jensen	Business, Space, Uni	6	No Evidence
1987	Kemerer [160]	SLIM, COCOMO, FPs and ESTIMACS	Self-collected	15	Peer review
1990	Kusters *et al.* [180]	FPA, COCOMO, Price and ESTIMACS	Self-collected	14	No Evidence
1991	Jenson *et al.* [138]	An Object0Oriented Model	A British Company	16	Regression Analysis
1993	Jeffery *et al.*[132]	SPQR/20 and FPA	Company-supplied	24	No Evidence
1995	Khoshoftaar *et al.* [163]	Regression models and ANNs	Company-supplied	152	Principal Component Analysis and Multiple regression model selection
1997	Boehm *et al.* [29]	COCOMO II	COCOMO II.1997	83	No Evidence
1998	Boehm *et al.* [28]	COCOMO II	COCOMO II.1998	161	No Evidence
1998	Knafl [175]	Generalized Weibull	Warburton [309]	116	Regression Analysis
1998	Briand *et al.* [43]	COBRA	Self-collected	6	Experts' opinions
1999	Walkerden *et al.* [306]	6 Analogy models and OLS	Company-supplied	19	No Evidence
2000	Briand *et al.* [45]	OLS, ANOVA, Analogy, and CART	European Space Agency	166	Removed missing data

Table 3.1: A Summary of Comparative Evaluation of Software Effort Estimation Based on Small Data Sets

have been found to be difficult to calibrate but inaccurate without calibration [131]. Because of its lightweight nature, the Expert-based estimation suffers from being highly dependent upon competent estimators. Informal approaches based on experienced estimators depend on estimator's availability and are not easily repeatable, as well as not being much more accurate than algorithmic techniques [303, 43]. Although results from studies of non-parametric methods [163, 269, 311] showed promising results, the sensitivity of each learning system to certain configurations should be considered carefully. Non-parametric techniques inspire little confidence regardless of how excellent are the Mean of Magnitude of Relative Errors (MMREs) that have been obtained [289]. The need for reliable and accurate cost predictions in software engineering is an ongoing challenge [254, 33].

Almost all of these cost estimation techniques share a common attribute: they require historical data of completed projects [173, 24, 14, 17, 286, 269, 289]. A general process of constructing cost prediction models from a historical data set is illustrated in section 1.2. Analysis of historical data provides intuition about the development environment (e.g., which variables affect effort). This would help an organization improve its development processes and management technique by focusing attention on influential factors in that organization. However, lack of data on completed software projects has been an impediment for many years [313, 114, 262]. Table 3.1 lists sample size of some data sets used in the last twenty years. Moreover, there are always problematic data (outliers, interrelated factors, missing values, etc.) existing in historical data sets. The problems concerning the data set are specifically addressed in chapter 4. Furthermore, at the early stage of the prediction, we often loosely define the model as mapping all of the available predictor variables (project attributes) in a historical data set to the project effort, since we have little precognition of how many factors (x_i) exactly we need for prediction, nor what are these factors. Consequently, constructing a specific model based on such loose definitions makes the prediction neither reliable nor efficient [173, 313]. However, there are neither formal approaches to identify that how many samples (projects or rows) in a historical data set are meaningful, nor how many predictor variables (project attributes or columns) in each sample are sufficient for predicting project effort. Therefore, there is a clear demand that a formal procedure should be established to deal with the problematic data and select appropriate (significant) predictor variables in historical data sets, that is, deduct rows which appear as outliers; and deduct columns which appear as non-dominant or inter-related factors.

The implication of the above problems is that empirical results in software effort prediction provide weak evidence. Several reasons for this lack of conclusive evidence include:

- Rapid changes in software development are a problem for a stabilization of the estimation process. (e.g. it is difficult to determine cost factors in operation.)
- Lack of sufficiently representative historical data [313, 114, 289]
- Inconsistent or out of date characteristics of software and software development make estimating difficult [313, 237].
- Subjectivity in data collection and analysis [223]
- Little use of preliminary data analysis. Outliers and interrelated predicting factors degrade the effort estimation. [168, 223, 202, 203, 188]
- Difficult replication of experiments in validating or evaluating cost models [42, 44]
- Missing values in the data sets
- Continuous evolution of development environment and inadequacies with the use of ad hoc measures like MMRE and PRED(l) [171]

3.11 Summary

This chapter gave an overview of the leading cost estimation techniques, evaluation criteria and discussed their strengths and weaknesses. Despite the intense research activities in software cost estimation, few generally applicable conclusions can be drawn. An accurate estimate cannot be guaranteed and so using more than one methods of estimation is recommended [25] for verification of an estimate.

This chapter stressed common problems in software cost estimation and identified a clear demand that a formal procedure should be established to deal with the problematic data and select significant predictor variables in historical data sets for construction of cost estimation models.

Exploitation of Heterogeneous and Homogeneous Data for Cost Estimation needs further investigation, since only a few researches [45, 134] have been conducted.

Chapter 4

Data Analysis Methods

4.1 Introduction

Pfleeger *et al.* [237] argued in 1997 that "*In the past, (software) measurement research has focused on metric definition, choice, and data collection. As part of a larger effort to examine the scientific bases for software engineering research, attention is now turning to data analysis and reporting...*". Researchers have been using statistical techniques in attempts to: identify the major factors contributing to software development costs [297, 305]; remove outliers [203]; deal with intercorporate cost factors [203]; analyze unbalanced data sets [168]; impute missing values [224, 56, 276, 219]. Although some systematic procedures have been proposed, none of them have been evaluated based on public data sets. Moreover, no standard preliminary data analysis procedures have been defined.

This chapter introduces constraints related to software engineering data, describes types of variables in project *effort* or *cost* estimation, discusses related work of preliminary data analysis, and elaborates problems with existing analysis procedures.

4.2 Constraints Related to Software Engineering Data

In the literature, we observed that software effort estimation data sets have some common features such as the following:

- containing predicted variable (i.e., effort) which is always positive, with a right skewed distribution [41, 63, 60, 48];
- containing both continuous (such as project size or duration) and discrete (such as application types) variables [41];
- and containing significant correlated or statistical not significant predictor variables [59, 202, 203].

Software engineering data sets tend to suffer from a number of problems such as the following:

- The difficulty and expense in collecting. Given the nature of software development, large amounts of data cannot be collected in an experimental setting.

- The limited sample size [313, 114, 262]. The timescale of a typical software project means that relatively few data points will be collected from each organization per year.
- Outliers have a great influence over estimation [41, 63, 60];
- The interdependencies of predictor variables can affect the understandability of models [59, 202, 203], but are not always harmful to their accuracy [41] (e.g. regression models [79]);
- Academics have restricted access to industrial project data, due in part to the reluctance of companies to release such data into the public domain, see Table 3.1.
- Missing values are a widespread problem as a result of lack of motivation on the part of some software engineers, some variables being more difficult to collect than others [224, 56, 276, 219].

In addition, when many variables are measured, however, some practical problems arise. For example, with as few as 10 variables there are 45 correlations that must be considered; with 20 variables there are 190 correlations; with 40 variables there are 780; and the number of correlation coefficients rises rapidly as the number of variables increases [79]. Obviously, with large numbers of variables the number of relationships is so large as to be beyond comprehension, and data reduction techniques that can systematically summarize large correlation matrices is clearly needed.

4.3 Scales of Variables

Kitchenham *et al.* [172] argued that "*Analyst often plunge directly into using statistical techniques without first considering the nature of the data themselves. It is important to look first at the organization of the data, to determine whether any results might be due to outliers or data points that have an unreasonable influence.*"

Not all data is equal, that is some data types may contain more information than others. For example, an ordinal data type might indicate the programming language used on a project. In itself, the number allocated to a particular language has no meaning, it only has meaning insofar as it is different to the numbers allocated to all

the other programming languages. This concept extends to other data types and is thoroughly discussed in Pfleeger *et al* [237].

Following the discussion in Pfleeger*et al* [237] and Hair *et al.* [107], we recognize several scales of measurement - nominal, ordinal, interval, and ratio. Each captures more information than its predecessor. The data scales relate to different data types, each of which has the following characteristics.

- **Nominal Scales** Nominal Scales, also known as categorical scales, provide the number of occurrences in each class or category of the veritable being studied. This scales relate to qualitative variables or attributes, such as gender or blood group, and are records of category membership. Nominal data is usually in the form of numbers. However, these numbers have no intrinsic meaning of their own but are in contrast only indicative. In software engineering data, variables such as *business sector*, *application type*, *programming languages*, etc., are nominal scaled data. These variables differ in kind only, they have no ratio sense. There is no meaningful order. Most of the variables relate to a software project are nominal data which has less statistical power compare with other data types.

- **Ordinal Scales** Ordinal scales are the next higher level of measurement precision. Variables can be ordered or ranked with ordinal scales in relation to the amount of the attribute possessed. Every subclass can be compared with another in terms of a "greater than" or "less than" relationship. For example, *Data Quality Rating* is an ordinal scaled variable. It is correct to say rate 1 is more reliable than rate 4. However, equal differences between ordinal values do not necessarily have equal quantitative meaning. Numbers utilized in ordinal scales are non-quantitative, because they indicate only relative positions in an ordered series. Variables such as *standards use*, *method use*, etc., are ordinal scaled variables.

- **Interval Scales** The value of an interval scaled variable can be ranked in order. In addition, equal distances between scale values have equal meaning. However, the ratios of interval-scale values have no meaning. The only real difference between interval and ratio scales is that interval scales have an arbitrary zero point, while ratio scales have an absolute zero point. The most familiar interval scales are the Fahrenheit and Celsius temperature scales. For example, an 80 Fahrenheit degrees data cannot correctly be said to be twice as hot as 40 Fahrenheit degrees day. *Project start year* is another example of the interval scaled variable.

- **Ratio Scales** Ratio scales represent the highest form of measurement precision, because they possess the advantages of all lower scales plus an absolute zero point. All mathematical operations are permissible with ratio scale measurements.Variables such as software project *effort*, application *size*, and *duration* are measured using ratio scales. Ratio scaled variables can be ranked in order, equal distances between scale values have equal meaning, and the ratio of ratio scale values make sense.

Numerical or continuous data are ratio or interval in the nature. In contrast, the categorical or discrete data are ordinal or nominal in nature.

4.4 Statistical Methods for Software Estimation

Statistical methods and tools have been used in software engineering data analysis since the 1970's. Vosburgh [305] extracted 5 out of 100 factors by statistical analysis, which explained 65% of the variance. Conte *et al.* [61] introduced some basic data analysis and evaluation statistical methods in the book "Software Engineering metrics and models". Another dedicated book is Burr *et al.*'s [52] "Statistical Methods for Software Quality– Using Metrics for Process Improvement". Putnam [251] also introduced the use of scatter diagrams, and statistical trend lines in software project's data analysis. From the application of basic scatter diagrams, and statistical trend lines [251], to Analysis of Variance (ANOVA) [202] and testing residuals [168, 238], researchers have endeavored to investigate the application of statistical techniques in analyzing software engineering data. Some systematic procedures have been introduced in the literature, for example Kitchenham's "A procedure for analyzing unbalanced data sets" [168] in 1998 based on Bailey and Basili's work [10] and Maxwell's [203] "Applied statistics for software managers" in 2002 based on her previous work for European Space, Military and Industrial Applications [202].

This section introduces general statistical methods for software estimation, including correlation coefficient analysis, multivariate analysis, analysis of variance, chi-square test, student t test, Mann-whitney U test, Wilcoxon matched-pairs test, and data transformation techniques. These systematic procedures will be described in the following sections.

4.4.1 Correlation Coefficient Analysis

Correlation coefficient [156] measures the strength and direction of the relationship between two continuous variables (data should be ratio or interval in nature). The correlation coefficient r can have any value between -1 and $+1$. If r is -1, this means that the two variables are perfectly negatively correlated. If r is $+1$, this means that the two variables are perfectly positively correlated. If r is 0, this means that the two variables are not correlated at all.

Correlation is probably the most widely used statistical method in assessing relationships among predicted variable(e.g., effort) and predictor variables (e.g., size, duration, etc.). However, caution must be exercised when using correlation, otherwise the true relationship under investigation may be disguised or misrepresented. Stephen [294] pointed out most of the time when one mentions correlation, it means linear correlation. "If a correlationship between two variables is weak, it simply means there is no linear relationship between the two variables. It does not mean there is no relationship of any form." Second, if the data contains noise (due to unreliability in measurement) or if the range of the data points is large, more likely than not the correlation coefficient (Pearson correlation) will show no relationship. The rank-order correlation methods, such as Spearman's rank-order correlation, are recommended.

There are three key assumption [156] made when using correlation coefficient analysis: 1) The correlation coefficient r is only appropriate for measuring the degree of relationship between variables which are linearly related. 2) When using the correlation coefficient analysis, the variables are random variables and are measured on either an interval or ratio scale. 3) The two variables have a joint normal distribution.

4.4.2 Multivariate Analysis

Multivariate analysis is not easy to define. Broadly speaking, it refers to all statistical methods that simultaneously analyze multiple measurements on each individual or object under investigation [107]. One reason for the difficulty of defining multivariate analysis is that the term multivariate is not used consistently in the literature. To some researchers, multivariate simply means examining relationships between or among more than two variables. Others use the term only for problems where all the multiple variables are assumed to have a multivariate normal distribution.

In software project development effort prediction, the goal of multivariate analysis is to determine what levels of predictive accuracy can be achieved in terms of cost estimation. In addition, it tells us about what kind of design measurement seems to play a more predominant, practically significant role for cost prediction [48]. In this subsection, we fellow Hair *et al.* 's concepts [107], and describe two multivariate analysis methods: multiple regression analysis and Principal Components Analysis, which have been used within software cost estimation context.

Multiple Regression Analysis

Regression analysis is by far the most widely used and versatile dependence technique. In particular, regression analysis is the foundation for forecasting models. Multiple regression analysis, also known as general linear modeling, is a general statistical technique used to analyze the relationship between a single predicted variable and several predictor variables. It has a general form as the follow:

$$Y = a_0 + \sum_{i=1}^{n} a_i \times x_i \tag{4.1}$$

where the coefficients $a_1, \ldots, a_n$ are denoting the relative contribution of each predictor variables to the overall prediction. These coefficients also facilitate interpretation as to the influence of each variable in making the prediction.

The most direct interpretation of the regression variate is determination of the relative importance of each predictor variable in the prediction of the predicted variables. Many researchers have used this method to assess the importance of (numerical) variables in effort prediction [203, 163, 103]. Numerical predictor variables' impact on the predicted variable can be assessed using simple linear or stepwise regression analysis.

Because the prediction accuracy for multiple variables decreases as multicollinearity (which occurs when any single predictor variable is highly correlated with a set of other predictor variables) increases, to maximize the prediction from a given number of predictor variables, the analyst should look for predictor variables that have low collinearity with the other predictor variables [107].

Ordinary Least Squares (OLS) regression is a general method of estimating the parameters of a regression curve, and has been used in software development effort prediction by Marouane *et al.* [200] and Briand *et al.* [44]. The parameters of the

regression equation are obtained by minimizing the sum of square of deviations. Two recently published studies [44, 45] have found that ordinary least square regression models produce better results than a set of other techniques including stepwise ANOVA, CART, and analogy-based analysis.

Principal Components Analysis

Principal Components Analysis (PCA) [79, 139] transforms the original set of variables into a smaller set of linear combinations that account for most of the variance of the original set. The purpose of PCA is to determine factors in order to explain as much of the total variation in the data as possible with as few of these factors as possible. Principal Components Analysis is a method for identifying the factor dimensions of the data and not as a formal statistical model. This method has been applied in effort estimation by Khoshoftaar *et al.* [163] in 1995 to deal with the common problems in software estimation data sets. These data sets often contain large number of variables (metrics), and tend to be highly correlated (or so called multicollinearity), which violate an assumption of multiple regression modelling.

Give an $n \times m$ matrix of multivariate data, principal components analysis can reduce the number of columns. For example, in our research, n could represents the number of completed projects for which m attributes (or metrics) have been collected. Using principal components analysis, the $n \times m$ matrix is reduced to an $n \times p$ matrix (where $p < m$) by extracting linear combinations of the original data in such a way that they account for most of the original variation in the data set. The linear combinations are orthogonal to one another and account for successively smaller amounts of the explained variation. Usually, the first few principal components explain a large proportion of the total variance [107]. By restricting analysis to a relatively small number of principal components, it is possible to reduce the dimensionality of the problem without a significant loss of the explained variance. Briand and Wüst [48] claimed PCA helped them better interpreted the relationship between the design measures and size.

It is customary first to transform the raw data matrix to either a covariance matrix or a correlation matrix prior to principal components analysis. In computing a correlation coefficient between two variables, differences due to both the mean and the dispersion of the variables are removed. Thus, the transformation makes the variables

directly comparable [79]. One of the products of the principal components analysis is a standardized transformation matrix, which describes the relationship between each of the m variables and the p significant principal components. These p orthogonal components can then be used to develop a multiple regression model.

The main limitations of applying PCA in software estimation that have been observed are:

- The regression models using principal components [163] are concerned with components which are linear in the original variables. For obviously non-linear data, the golden rule recommended by Dillon *et al.* [79] is to linearize the data from the outset, e.g. by transformations of the square-root or logarithmic type or more complicated functions if necessary.
- Principal Components Analysis (PCA) is intended for use with ratio or interval scale measures [168]. It cannot be used for nominal scale factors and can only be used for ordinal scale measures if we treat them as if they are interval scale.

4.4.3 Analysis of Variance (ANOVA)

Analysis of Variance (ANOVA) refers to a set of well-defined procedures for partitioning the total variation of a data collection into its component parts [156]. It is concerned with the testing of hypotheses about means. The ANOVA F statistic is calculated by dividing an estimate of the variability between groups by the variability within groups.

$$F = \frac{V_b}{V_w} \tag{4.2}$$

Where V_b is the Variance between groups and V_w is the Variance within groups. The null hypotheses H_0 states that there is no difference between the populations means. If we reject H_0, then we must infer that the experimental manipulation does have an effect. Since the F distribution is a well-defined probability distribution, we can determine the probability that a given F will be above a certain value, or conversely, we can determine the value of F which will be exceeded, say, 5% or 1% of the time by chance alone [156]. These critical F values for various pairs of degrees of freedom can be obtained from standard statistical tables.

If there are large differences among the predictor variables' means then the numerator of F will be inflated and the null hypothesis is likely to be rejected. Therefore,

we can use ANOVA to test whether there is an effect on the predicted variable of each of the explanatory variables (mainly categorical varaibles). A significant $F - value$ (the conventional significance level is usually 0.05 or 0.01) tells us that the population means are probably not all equal.

Based on the above description, ANOVA has been used to test the independence between the numerical and categorical variables [168, 202, 203]. The analysis of variance(ANOVA) procedure has also been used for constructing models to predict effort, for example, Briand *et al.* [44] and Kitchenham [168] used ANOVA to analyze the variance of unbalanced data and fit regression estimates to models with categorical variables.

4.4.4 Chi-Square Test of the Independence

Two events are independent whenever the probability of one happening is unaffected by the occurrence of the other. This concept can be extended to nominal variables. The Chi-square test [156] for independence compares the actual and expected frequencies of occurrence to determine whether or not two nominal variables are independent.

The assumption underpinning the Chi-square test for independence is that, if two variables are independent, the proportion of observations in any category should be the same regardless of what attribute applies to the other variable. The null hypothesis is that there is no relationship between the two variables. The Chi-Square test is used to test the independence between categorical variables in software engineering data analysis [203, 187, 188].

4.4.5 Student's t-test

For comparing software effort prediction accuracy based on different methods, or data sets, we use formal tests of significance. We use the Student's t-test [53] for comparing the means of two treatments, even if they have different numbers of replicates. In simple terms, the t-test compares the actual difference between two means in relation to the variation in the data (expressed as the standard deviation of the difference between the means).

Stensrud *et al.* [292] used the t - test to test whether the prediction accuracy of one

set of results was significantly different from another one. This test is appropriate to test Hypothesis 1 in this thesis.

4.4.6 Mann-Whitney U Test

One of the most useful of the two-sample nonparametric tests is the Mann-Whitney U test [53]. It requires data that are at least ordinal in level. It is a powerful alternative to the parametric t test for means, when the parametric assumptions for that test are not met or are in doubt. The t-test requires the assumption that the two population standard deviations are equal. This is often a questionable assumption. In software effort estimation, Shepperd [277] used the Mann-Whitney U test to determine the difference between two competing regression models with R-square values ranging from 25 to 28% respectively. In this thesis, the detailed application of the above significance testing is discussed in section 6.6.

4.4.7 Wilcoxon Matched-pairs Test

Wilcoxon matched-pairs signed ranks test [156] is often called Wilconxon T test. This test can be used to matched pairs to check if either model has consistently smaller errors. This test compares the magnitude of each model's estimation error for each project. The magnitude is the absolute value of the estimation error (|Actual-Estimate|). Both the Wilcoxon T test and Mann-Whitney U test are powerful alternatives to the parametric t-test for means. The Wilcoxon T test can only be used when some pre-conditions [53] are fulfilled that:

1. Each observation in the first sample must be paired with an observation in the second sample.

2. We must be able to determine which observation in each pair has the higher (or lower) score.

3. We must be able to determine the difference between the scores for each pair.

4. We must be able to rank the difference by magnitude.

5. We must have at least 10 pairs of scores.

Both Stensrud *et al.* [292] and Shepperd *et al.* [277] used the Wilcoxon matched-pairs test to test whether the prediction accuracy of one set of results was significantly different from another one. This test is appropriate to test Hypothesis 2 in this study, which competing prediction results fulfil all above condition. The detailed application of the above significance testing is also discussed in section 6.6.

4.4.8 Data Transformation

Many statistical techniques assume that the underlying data is normally distributed. However, the distributions of some variables, for example, software project effort and size are not normally distributed, the distributions are skewed to the right. They might also have a wide range. For example, software project size of the data set used in case study ISBSG R9 in this thesis spreads from 31 to 4887 function points. For that reason, to approximate a normal distribution, transformation techniques are frequently be used [10, 79, 107, 168, 203]. Log-linear ordinary least-squares (OLS) regression models are frequently used in software engineering cost estimation models. For example, one can fit the regression equation $\ln y = \beta_0 + \beta_1 \ln x_1 + ... + \beta_n \ln x_n$. Chulani, Boehm and Steece [60] linearized the COCOMO II equation by taking logarithms on both sides of the equation. Alternative transformations include taking data's standard deviation values or square-root [107, 79]. Transformation methods generally make large values smaller and bring the data closer together.

Briand and Wüst [48] pointed out that logarithmic transformation introduces a systematic bias in the predictions, i.e., the sum of predicted and actual effort values over all observations are not equal. As Briand and Wüst noticed this is due to the fact the $E(\ln y)$, the expected value of $\ln y$, tends to be lower than $\ln E(y)$ (Jensen's inequality). Unbiased prediction is a crucial property in prediction context, since the sum of the predicted effort over all classes is used as an estimate for the total project *effort*. To compensate for this bias, Briand and Wüst [48], modeled y directly using a non-normal distribution, i.e. Poisson distribution [193]. However, Graves and Mockus [102] do not assume a Poisson distribution because *effort* values need not be integers. The only critical part of the Poisson assumption is that the variance of a random *effort* is proportional to its mean. In general, it is difficult to a priori decide about the distributions of data, the form of relationships between cost drivers and cost expenditures, and the size of the data set.

4.5 Kitchenham's Data Analysis Framework

Kitchenham [168] noticed a problem concerning the direct use of categorical attributes in cost models, which is that the data sets including many factors are unbalanced, i.e. not all possible combinations of the factor levels appear in the characterization of a project. Kitchenham elaborated the possible concealed impacts and spurious impacts caused by unbalanced data sets, and proposed a data analysis framework procedure to cope with those negative impacts. Kitchenham's analysis procedure (namely, forward pass residual analysis) uses Ordinary Least Squares(OLS) or Analysis of Variance (ANOVA) depending on the scale of the predictor variable to build models with the most suitable factors. This analysis procedure used some ideas of Miyazaki and Mori [210] and Bailey and Basili [10], in particular, Bailey and Basili's idea of analyzing residuals. It consists of 9 steps:

1. If necessary, transform the productivity variable to improve normality, in particular the stability of the variance.

2. Apply simple analysis of variance (ANOVA) using each factor in turn.

3. Identify the most significant factor, i.e., from the set of factors which have a statistically significant effect on productivity, choose the one that gives the smallest error term.

4. Remove the effect of the most significant factor and obtain residuals.

5. Apply ANOVA using each remaining factor in turn on the residuals.

6. Identify redundant factors. Factors that were significant in the original analysis but no longer have significant effect on the residuals are factors that are confounded with the first significant factor. Confounded factors must be identified and excluded from subsequent analysis.

7. Identify the next most significant factor impacting residuals. Only factors that have a significant impact based on the analysis of variance of the residuals should be considered.

8. Remove the effect of the next most significant factors and obtain second residuals.

9. Repeat the analysis process until all significant factors are removed, or there are insufficient degrees-of-freedom available to continue.

The Kitchenham's analysis procedure is particularly useful in handling data of a mix of the different scale types including nominal, ordinal and other scale types. However, there are some limitations inherent in this analysis procedure. A major concern is that a forward pass analysis procedure does not guarantee to produce the best possible model. Another problem is that the analysis procedure ignores any interactions between predictor variables. Finally the analysis procedure depends in the predicted variable being distributed approximately normally.

This procedure was applied to Boehm's COCOMO 81 data set [24], and the achieved MMRE is 0.36 and Pred(0.20) is 0.49. The result of using this analysis procedure on the COCOMO model leads to a model that is less accurate than the COCOMO model. Thus, it may not be clear whether the statistical model is particularly useful. In addition, most of demonstrations in Kitchenham's [168] study are based on hypothetical (or artificial) data sets, which means that little empirical evaluation have been conducted.

4.6 Jeffery *et al.*'s Data Analysis Methods

The accuracy of the estimates based on the ISBSG R5 data set repository was compared with the results obtained from using a one-company data set from a company called Megatec. Ordinary least squares regression and an analogy-based tool were applied on the two data sets.

Jeffery *et al.* [133] evaluated the accuracy of the OLS model based on the ISBSG release 5 data set and the Megatec data set in 1999. A set of techniques used to identify dominate cost factors were described in Jeffery *et al.*'s study, which can be summarized as follows:

1. Correlation Analysis: to discover if there are metrics in the data repository which are highly associated.

2. Stepwise Linear Regression: at each step of the regression, variables in the equation are evaluated according to the selection criteria for removal; variables not in

the equations are evaluated for entry. This process repeats until no variable in the block is eligible for entry or removal.

3. One-way ANOVA: to test whether certain categories of categorical variables in the ISBSG repository have an impact on effort.

4. t-tests for binary variables (i.e. the use of CASE tools) to determine whether there is a significant difference in a predicted variable.

5. Remove projects contain missing values.

6. Split the data set into subsets of dependence on certain metrics and their values. These subsets can then be used for further estimation.

Jeffery *et al.* [133]'s data analysis procedure is not formally defined. For example, they selected the subsets "dependence on certain metrics and their values", but did not clarify the metrics and their values. Moreover, after processing the raw data based on the above procedure, the metrics were manually chosen by researchers. Although the definition of data analysis procedure is ambiguous, Jeffery *et al.* applied the procedure to ISBSG R5 data and Megatec (a company-specific) data set. They also evaluate the accuracy of the estimates by comparing ordinary least squares regression and an analogy-based tool. However, they simply compared the accuracy of the estimates between analyzed data sets, but not between the raw data sets and analyzed data sets. In addition, ISBSG R5 data was not the latest version in 2000, and Megatec data could not be accessed by other researchers. Therefore, Jeffery *et al.*'s research did not provide an appropriate empirical evaluation of the data analysis procedure.

4.7 Briand and Wüst's Data Analysis Procedure

Briand and Wüst [48] presented a five-step data analysis procedure in 2001. This procedure can be summarized as follows:

1. Look at the frequency distribution of design metrics.

2. Use Principal Components Analysis (PCA) to determine the dimensions captured by their design measures.

3. Univariate regression analysis considers the relationships between each of the design measures. This is a first step to identify potential cost predictors. A log-linear model assuming Possion distributed effort predictions was used in this step.

4. Multivariate analysis also considers the relationships between design measurement and cost, but does not consider design measures in combinations.

5. Apply cross validation in order to get a more realistic estimate of the predictive power of the multivariate predictions models.

Similar to *Jeffery et al.*'s [133] study, Briand *et al.* [48]'s research also focuses on comparing the accuracy of the estimates between analyzed data sets. There are a few aspects of Briand *et al.*'s analysis procedure that should be further considered:

- The detection and removal of outliers is not addressed.
- The Principal Components Analysis (PCA) is descriptive rather than analytic.

Although Briand and Wüst claimed PCA helped them better interpreted the relationship between the design measures and size, they did not produce a transformed matrix [163], nor use these results to do any further research. Briand and Wüst only observed predominate predictor variables (e.g. normalized cohesion measures of OO systems) based on the highest loadings within each principal component in the rotated component pattern. In addition, step 5 is not a direct part of the data analysis, but an evaluation method for assessing accuracy of prediction models. Typically, Jeffery *et al.* and Briand *et al.*'s use preliminary data analysis on an informal and ad hoc basis rather than as an independent, intrinsic procedure.

4.8 Maxwell's recipe

Compared with the above researches, Maxwell's analysis procedure is formally prescribed and well-defined. Maxwell's book "Applied Statistics for Software Managers" was the first complete guide to using statistical methods in software project management and process improvement. As the name implies, the book presents an easy-to-use

methodology that enables project managers to obtain results without deep mathematical expertise. However, some definitions (e.g., outliers) are quite informal. Maxwell's book is a textbook for project managers rather than a rigorous academic study. It is significant however, that throughout the book, model evaluation and validation is limited to testing residuals based on a single data set. Both Maxwell and Kitchenham's researches have the same origin that they all build upon Bailey and Basili's work of analysis of variations on the cost model in 1981 [10].

Maxwell's approach [203] can be summarized by the following three stages:

1. Validating software project data: data visualization and transformation.
2. Analyzing the variance of the data: build a multi-variable model.
3. Evaluating the data: extract the equation and test the residuals.

The following sections elaborate these three stages in further detail.

4.8.1 Validating software project data

To carry out a valid statistical analysis, we should remove outliers and vectors with missing values, and test the normality of the data. To visualize data, we can plot histograms or boxplots for the predictor variables and the predicted variable individually. In the test of normality, some variables were found to violate the statistical test. In each case, transformation by taking logarithms was applied.

4.8.2 Analyzing the variance of the data

Maxwell's approach expects to identify the relative significance of predictor variables which explain the most variation in the predicted variable by building a multi-variable model step by step. Adjusted R^2 values will be calculated to determine this relative significance. To acquire adjusted R^2 values, correlation analysis is applied for the ratio or interval variables, and ANOVA procedures for the nominal variables.

The procedure is to perform a stepwise analysis procedure, which allows us to determine the influence of explanatory (independent) variables on the response (dependent) variable. The model starts "empty" and then the variables most related to predicted

variable are added one by one in order of importance until no other variable can be added to improve the model.

To find the predictor variable which explains the most variation in the predicted variable (project effort), we perform regression procedures for the ratio or interval scale variables, and ANOVA procedures for the nominal variables. The importance of the predictor variables to the predicted variable depends on the significant value, $P > |t|$ for regression procedures (between numerical variables) and $Prob > F$ for ANOVA procedures (between categorical variables, or between numerical variables and categorical variables).

4.8.3 Evaluating the data

To evaluate the independence between predictor variables, correlation coefficient analysis, analysis of variance(ANOVA) procedure and Chi-square test are applied according to the variables' scale.

To evaluate the goodness of fit of the model, Maxwell's approach [203] extracts the final model and tests the residuals. In a well-fitted model, there should be no pattern to the errors (residuals) plotted against the fitted values. The term "fitted value" refers to the project effort predicted by the model. The term "residual" is used to express the difference between the actual effort and the predicted effort for each project. Therefore, there should be no pattern in the residuals of our final model.

4.9 Summary

This chapter described the types and constraints related to software engineering data, explained the common statistical methods that have been applied, and reviewed related work.

A common problem in software estimation is the lack of reliable and accessible historical data sets. The major problems with historic data sets are: Limited sample size, Inconsistent or out of date attributes, Missing values, Limited accessability, etc.

Several common statistical techniques are discussed in this chapter as a basis of the data analysis framework. Although statistical techniques in analyzing software

engineering data have been studied from the 1970s, there has been little emphasizes on forming systematic procedures until 1998.

By reviewing past work on preliminary data analysis which assisted project effort prediction, we identified the main results of those researches are:

- remove outliers.
- identifying inter-correlated predictor variables
- identifying statistic significant predictor variables, or subsets contains statistic significant predictor variables.

The various procedures are based on a same underlying principle that to construct cost estimation models based on as few as possible predictor variables from historical data sets. These predictor variables are selected by their statistic power of explaining predicted variable's variation. The more variation of the predicted variable is explained by a predictor variable, the more significant this predictor variable is.

In addition, the related studies discussed in this chapter have neither been empirically evaluated, nor applied in more than one standard prediction technique. Therefore, a formalized, empirically evaluated, systematic preliminary data analysis framework would represent a significant and positive contribution to the resolution of software cost estimation.

Part II

A Preliminary Data Analysis Framework

Chapter 5

Preliminary Data Analysis Framework

Everyone seems to have been surprised by the stickiness of the problem (in software estimation), and it is hard to discern the nature of it. But we must try to understand it if we are to solve it

Fred Brooks [50].

5.1 Introduction

Many estimation methods have been developed as presented in chapter 3, e.g. mathematical, statistical, rule based, machine learning and hybrid methods. However there is no formal procedure to define what project predictors and how much project data from a historical data set should be used when building a cost model. Some systematic data analysis methods in software engineering, which attempted to achieve the above goal, are discussed in chapter 4. However, these methods have not been evaluated by their application to a standard and extensive historical data sets, nor have they been applied in more than one cost estimation technique. To overcome the limitations of building cost models from historical data sets, this thesis proposes a preliminary data analysis framework, that is an extension of Maxwell's [203] analysis of variance.

This chapter outlines the general principles of the preliminary data analysis framework, describes the proposed extension of the proposed framework, and provides an overview including a flowchart of the framework.

5.2 General Principles

The main motivation behind this research is to define a preliminary data analysis framework which provides a formal basis for building software project effort prediction models based on historical data sets. The proposed framework defines a formal procedure to identify what project attributes and how much project data from a historical data set should be used to build project effort prediction models. Based on statistics methods, the preliminary data analysis framework reduces large sets of data to smaller sets that describe the original observations without sacrificing critical information. In other words, the proposed framework summarizes large sets of observations into more compact and interpretable forms. The application of the framework allows a better exploitation of data sets and identifies significant predictor variables. The framework selects subsets from historical data that are "optimal" to provide the best characterizations of the historical data sets.

As illustrated in Figure 5.1, a historical data set for effort prediction is defined as a project vector set (PVS). This data set contains j project vectors (PV_j) and each vector consists of i predictor variables (x_i) and one predicted variable (Y). First, the proposed

framework eliminates outliers and obtains a sub set of PVS as PSS which contains $m(m \leq j)$ project vectors (PV_m) and each vector consists of i factors (x_i). PSS is a row subset of PVS, we also call it the raw data set without outliers. Second, the proposed framework ranks the predictor variables by their statistical explanatory power [1] on project effort, i.e. the more the variable explains the variation of project effort, the more significant the variable is. The outcome is a column subset of PSS. This subset is defined as EPSS, which contains $m(m \leq j)$ project vectors (PV_m) and each vector consists of $k(k \leq i)$ factors (x_k). We call EPSS the (final) framework processed data set containing extracted predictor variables (or cost factors) and project samples for construction of cost prediction models. The predictive accuracy of the framework processed data

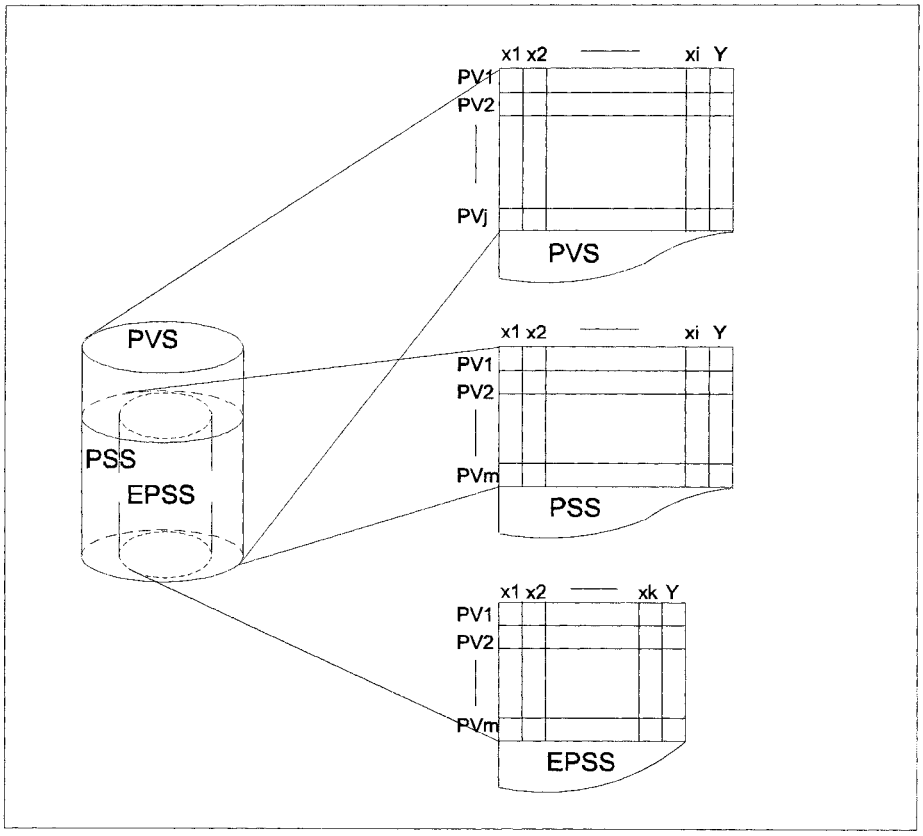

Figure 5.1: Extracting Optimal Subsets by the Proposed Framework

is empirically evaluated by defining a systematic and repeatable evaluation method outlined in section 6.3.

[1] actually the variance

5.3 Extension of Maxwell's Study

The proposed framework is built upon Maxwell's research. The proposed framework extends Maxwell's analysis procedure in ways that

- Formally define reduction rules that facilitate:
  - criteria for removing outliers;
  - rules for incorporating inter-correlated variables;
  - clarification of statistical methods that handle differently scaled variables e.g. ratio, ordinal, and nominal.
- Further analyze potential inter-correlated predictor variables, e.g. adopts univariate analysis to identify collinearity between categorical predictor variables.

In addition the extended framework is empirically evaluated against commonly used prediction methods, namely Ordinary Least-Square Regression (OLS), Robust Regression (RR), Classification and Regression Trees (CART), K-Nearest Neighbour (KNN), and is also applied to both heterogeneous and homogeneous data sets.

5.4 The Preliminary Data Analysis Framework

5.4.1 Terminology

This section specifies the terms will be used in the proposed framework and how they will be used.

Predictor and Predicted Variable

A prediction model can be developed by exploiting past experience from a historical data set. The model could be mathematical functional relationships, statistical statements, rule based systems, tree structures or even indefinable relationships, for instance, neural networks. To clarify the roles of the data, constructing a software effort prediction model means finding a mapping of project attributes to project effort.

In effect we are looking for some form of functional relationship as:

$$\text{Effort} = \text{Function(project attributes)} \tag{5.1}$$

We call *project attributes* the predictor variables x_i, the project *Effort* as the predicted variable Y. One of the aims of this research is to identify the influential predictor variables x_k with $(k \leq i)$, which contribute to the underlying relationships between the predictor variables and the predicted variable.

$$Y = f(x_k) \tag{5.2}$$

Adjusted Coefficient of Determination (adjusted R^2)

Adjusted R^2 [156] is a modified measure of the coefficient of determination that takes into account the number of predictor variables included in the regression equation. This statistic is used for comparison between equations with different numbers of predictor variables. For example, compare the equations with all available predictor variables and with selected number of predictor variables. The higher the value of the adjusted R^2, the more variation of the predicted variables is explained, and therefore the better the prediction of the set of the predictor variable.

Coefficient of Determination (R^2)

R^2 [156] is a measure of the proportion of the variance of the predicted variable about its mean that is explained by the independent, or predictor, variables. The coefficient can vary between 0 and 1. If the regression model is properly applied and estimated, the analyst can assume that the higher the value of R^2, the greater the explanatory power of the regression equation, and therefore the better the prediction of the criterion variable.

Correlation coefficient (r)

Correlation coefficient (r) [156] indicates the strength of the association between the dependent and the predictor variables. The sign (+ or -) indicates the direction of the relationship. The value can range from -1 to +1, with +1 indicating a perfect positive relationship, 0 indicating no relationship, and -1 indicating a perfect negative or reverse relationship.

Collinearity

Collinearity [156] is an expression of the relationship between two (collinearity) or more predictor variables (multicollinearity). Two predictor variables are said to exhibit complete collinearity if their correlation coefficient is 1 and a complete lack of collinearity if their correlation coefficient is 0. Collinearity is exchangeable with inter-correlated or interdependent. Multicollinearity occurs when any single predictor variable is highly correlated with a set of other predictor variables.

Multicollinearity

See collinearity.

Multiple Correlation Coefficient (R)

It is usually designated by the capital letter R to distinguish Multiple Correlation Coefficient [156] from the correlation coefficient r that exists between two random variables. R also measure the correlation between two variables, it is simply that one of the variables is derived as a weighted combination of several others.

Numerical Variables and Categorical Variables

As suggested by Maxwell [203], ratio, interval and ordinal scaled variables are defined as numerical variables. Nominal scaled variables are then defined as categorical variables within the proposed framework. The ratio, interval, ordinal and nominal scaled variable are defined in section 4.3.

Outliers

Outliers [156] are observations with a unique combination of characteristics identifiable as distinctly different from the other observations. Outliers are identified from a typical approach that first converts the data values to standard scores, which have a mean of zero and standard deviation of one. Based on the guidelines [107], for small samples (80 or less), we identify those cases with standard scores of 2.5 as outlier. When the sample sizes are larger (greater than 80), we adopt the threshold value of standard scores of 3.

Because most of the numerical variables in software estimation are positive, removing outliers by standard scores only eliminates the maximum extremes but ignored the minimum ones. An important and useful extra measure is the 5% trimmed mean [96] where the lowest and highest 5% of the observations are omitted and the mean is re-calculated. Therefore, we also applied the above measure and removed outliers. If there are categorical predictor variables, the project cases contain any categories appearing less than 3 times will be considered as outliers [203] as well. For example, one of the categorical variables - EMthod - in the historical data set contains 300 project vectors, which consists of two categories - YES and NO. If there are 280 YES and 2 NO, the two vectors with EMethod = NO, will be identified as outliers.

Residuals

Residuals [107, 44] are the portion of a predicted variable not explained by a multivariate technique, i.e. residual = (predicted value - actual value). Residuals can be used in diagnostic procedures that examine the unexplained portions to identify problems in the estimation technique or unidentified relationships. According to [203], if the residuals are normally distributed, the model is considered to be well fitted.

5.4.2 The Analysis Process

The preliminary data analysis framework is depicted as a flow chart in Figure 5.2; it is a sequence of steps. the proposed framework captures the impact of, and be effective in, integrating all predictor variables regardless of their type, i.e., ratio, interval, ordinal and nominal scaled variables. Also, the proposed framework provides a consistent way of interpreting each variable's effects on the predicted variable. It accounts for interdependencies among the predictor variables by providing the context within which each parameter of the model appears to be a relevant and significant piece of information. Thus, the proposed framework allows the underlying data to be used more efficiently for effort prediction.

Before analyzing software projects data, the most important step is data validation. To avoid invalid or incorrect data misleading the conclusions, it is necessary to get a precise idea of exactly what data is available and how much they can be trusted before starting the analysis. Statistical methods and tools are usually applied in validating

data. Step 0 to Step 3 in the proposed framework focus on data validation.

- Step 0: **Data Preparation.**

  Set up raw data set - Project Vector Set (PVS).

  If there are projects containing missing values, we remove these project vectors.

  If there are categorical variables with categories in text, we transform these categories into nominal numbers.

- Step 1: **Remove Outliers by Measuring Numerical Variables.**

  Use a summary function for each numerical (ratio, interval or ordinal scaled) variable in the PVS data set, and remove outliers. If the sample size is greater than 80, outliers are those cases with standard scores of 3, otherwise they are those cases with standard scores of 2.5. Because most of the numerical variables in software estimation are positive, removing outliers by standard scores only eliminates the maximum extremes but not the minimum ones. An important and useful extra measure is the 5% trimmed mean where the lowest and highest 5% of the observations are omitted and the mean is re-calculated. Therefore, we also apply the above criteria and remove these outliers.

- Step 2: **Data Transformation.**
  Plot the histograms for each numerical variable in the PVS data set to see if the variables are normally distributed (in a bell-shaped curve). If they are not normally distributed, a common transformation technique that takes the data's natural log is used to approximate a normal distribution. These methods make large values smaller and bring the data closer together. Transformed data must be re-visualized (e.g. histogram) and the outliers must be removed.

- Step 3: **Remove Outliers by Measuring Categorical Variables.**
  Tabulate each categorical (nominal scaled) variable to check how many projects are in each category, and remove projects contain any categories appear less than 3 times. This step yields the PSS data set, that is a row reduced subset of the PVS data set. Then, re-validate the PSS, e.g. re-calculate numerical variables' means; re-plot each variables (histogram and Q-Q normality Plot) to catch possible outliers, until there are no outliers.

- Step 4: **Test Collinearity Between Numerical Predictor Variables.**
  A common assumption made when building a multi-variable model is that predictor variables are independent of each other. To check if the ratio or interval variables are independent, a correlation analysis is used. If the absolute value of the correlation coefficient is greater than or equal to 0.75, and the $Pr > |t|$ value equals 0.05 or less, then the two variables are strongly correlated and should not be included in the final model together. If two variables are inter-correlated, We keep the one with higher statistical explanatory power, i.e. keep the one that has the greater correlation coefficient value with the predicted variable. Strongly related categorical variables can cause problems similar to those caused by numerical variables. Unfortunately, strong relationships involving categorical variables are much more difficult to detect. Examining the independence between categorical data is described in the step 7.

- Step 5: **Stepwise Regression Analysis.**
  Test the correlation between the numerical predictor variables and the predicted variable by running a backward stepwise regression analysis. This procedure allows us to determine the relative importance of each numerical predictor variable's relationship to the predicted variable. This step is normally applied in step 6, and for information only.

- Step 6: **Stepwise ANOVA Analysis to build a k-variable model.**
  The following sub-steps are to be followed.

  - Identify the best one-variable model. If there are i, where $i \geq 1$ predictor variables, then we need to build i one-variable models. The best one-variable model will be determined by the adjusted R^2 value. This value is calculated for each predictor variable with the predicted variable (project *effort*). The regression adjusted R^2 value will be calculated for numerical variables and the ANOVA adjusted R^2 value will be calculated for categorical variables. The predictor variable that has the greatest adjusted R^2 value explains most of the variation in the predicted variable and is kept in the model, we call it z_1, where $z_1 \in \{z_k\}$ and $\{z_k\} \subseteq \{x_i\}$.
  - Identify the best two-variable model by the same procedure. Determine which variable, x_i, in addition to z_1, explains the most variation in the predicted variable. This means that $i - 1$ two-variables models must be

built. The most significant variable is selected as the second variable to be added into the model: z_2 , where $z_2 \in \{z_k\}$ and $\{z_k\} \subseteq \{x_i\}$.

- Identify the best three-variable model by the same procedure, in addition to z_1 and z_2. The process is iterative, and the number of the variables in the model is increasing until no further improvement in the model is possible. If there are two n variable $1 \leq n \leq i$ models that explain nearly the same amount of variation in the predicted variable then develop $n + 1$ variable models based on each of them.
- Finally, we will identify the best k-variable model as equation 5.3

$$Y = f(z_k) \tag{5.3}$$

where $\{z_k\} \subseteq \{x_i\}$ and $k \leq i$.

- Step 7: **Test Collinearity Between Extracted Predictor Variables.**
The independence between the numerical variables has been tested in Step 4, however, at this stage it is not known whether these numerical variables are correlated to any other categorical variables. Also, it must be established whether the categorical variables are independent.

  As suggested by Maxwell [203], to determine if there is a relationship between a categorical variable and a numerical variable, analysis of variance (ANOVA) procedures can be used. To determine if there is a relationship between two categorical variables, the Chi-square test can be used.

  If significant relationships have been identified between two predictor variables they must be studied closely and a further judgment must be made. Although Maxwell suggests making a 100% bar chart showing the percentage of each factor level of both variables, we adopt the application of a Univariate analysis to examine the interaction between and among suspected predictor variables against the response variable. If these variables are inter-correlated, We keep the one with the higher rank of significance based on Step 6. This step yields the EPSS data set, that is a column reduced subset of the PSS data set.

- Step 8: **Extract and interpret equations.**
Normally, there are transformed variables or categorical variables which have different multiplier (coefficients) for different levels. Therefore we need to extract

the equation of identified k-variable model. For example, if the equation read from the final model's output is

$$\begin{aligned} \text{LnEffort} &= -2.406596 - (0.772367 \times \text{LnDuration}) + \\ &\quad (0.4475299 \times \text{LnSize}) - \\ &\quad (0.190931 \times t09) \end{aligned} \tag{5.4}$$

this can be transformed into the following non-linear equation for effort.

$$\begin{aligned} \text{Effort} &= (0.091 \times \text{Duration}^{-0.7224}) \times \\ &\quad (\text{Size}^{0.4475} \times e^{-0.1909 \times t09}) \end{aligned} \tag{5.5}$$

- Step 9: **Test residuals.** In a well-fitted model, there should be no pattern to the errors (residuals) plotted against the fitted values. The term "fitted value" refers to the project effort predicted by the model, the term "residual" is used to express the difference between the actual effort and the predicted effort for each project. There should be no pattern in the residuals of our final model. Finally, plot a histogram and a boxplot of the residuals, if they are normal distributed, then the model is well fitted. In other words, extracted predictor variables are valid.

A schematic flowchart of the proposed framework is shown in Figure 5.2.

5.5 Summary

This chapter described the general principles of the proposed preliminary data analysis framework, and the extension to Maxwell's analysis procedure. This chapter explained related researches [168, 133, 48, 202, 203] and identified their commons and drawbacks respectively.

Compare with these researches, the proposed framework inherits Maxwell's original intention and extends it to a rigorous procedure for building data-driven software effort prediction models. In other words, this extension is a formalization and specialization of previous studies, in particular Maxwell's. For the first time, this study provides a formal basis for building data-driven software effort prediction models, which allows prediction at the early stage of developing projects.

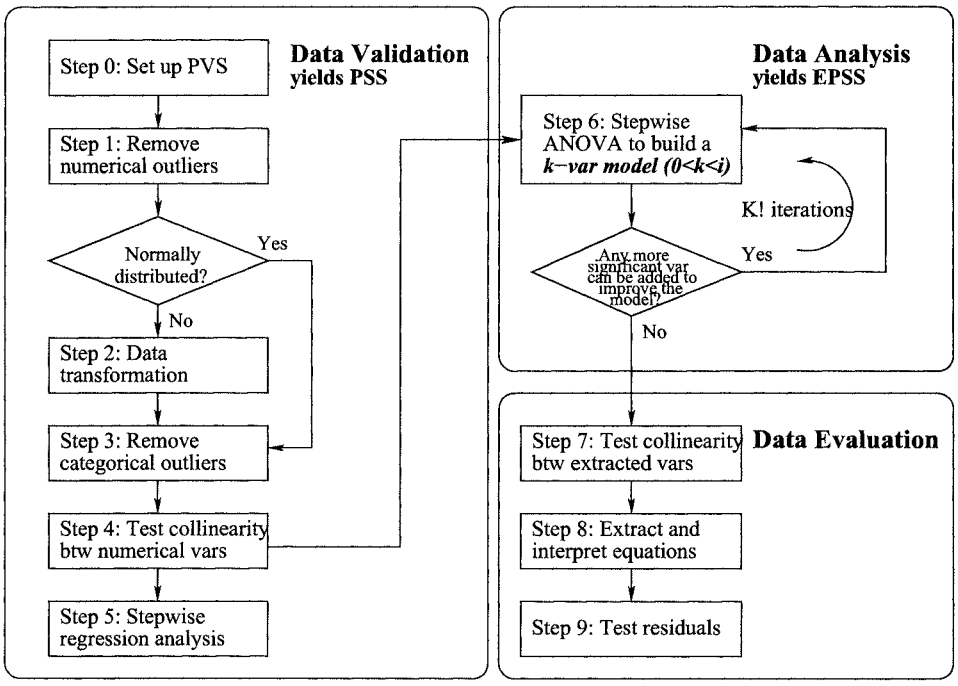

Figure 5.2: Flow Chart of the Preliminary Data Analysis Framework

Chapter 6

Empirical Evaluation Method

6.1 Introduction

The intention of the evaluation in this thesis is to derive conclusions about (1), the accuracy and efficiency of the proposed framework based on commonly used estimation techniques, and (2), the feasibility of the proposed framework based on heterogenous and homogenous data sets.

This chapter describes the research hypotheses tested in the case studies performed. It explains the comparative design of the studies and thus contributes to addressing the problems stated in section 1.6.1 and 1.6.2. To evaluate the accuracy and efficiency of the preliminary data analysis framework, this chapter includes a comprehensive cross-section of common estimation methods in comparison and provides the high-level selection criteria used for limiting the number of methods applied. In order to support the replication of the evaluation, the configurations for the selected methods and evaluation criteria are described in detail.

6.2 Detailed Research Hypotheses

Related to the objectives stated in section 1.5 the following research hypotheses are tested:

Hypothesis 1 The predictions based on the data set yielded by a preliminary analysis performs as well as or better than these based on both raw data set or raw data set with the most significant predictor (e.g., size).

Hypothesis 1a When using framework processed data (EPSS), the effort prediction model performs no worse than the same model using raw data (PVS).

Hypothesis 1b When using framework processed data (EPSS), the effort prediction model performs no worse than the same model using the subset of the raw data which only contains the most significant variable (e.g., size) and effort variable (PVS_{Size}).

Hypothesis 2 The predictor variables extracted by the proposed framework are statistic significant (at 95% confidence level).

Hypothesis 2a When using framework processed data (EPSS), the effort prediction model performs no worse than the same model using the raw data set without outliers (PSS, PSS $\subseteq$ PVS, the row reduced subset of PVS).

Hypothesis 2b When using framework processed data (EPSS), the effort prediction model performs no worse than the same model using the subset of EPSS, which only contains the most significant variable (e.g., size) and effort variable (EPSS_{Size}).

Hypothesis 3 Both the heterogeneous data sets and homogenous data sets benefit from the application of the proposed preliminary data analysis framework for improving project effort prediction accuracy.

Hypothesis 3a Both Hypothesis 1 and 2 are true for the heterogeneous data set (ISBSG R9 Data) in this study.

Hypothesis 3b Both Hypothesis 1 and 2 are true for the homogenous data set (Bank63 Data) in this study.

6.3 Comparative Design

This section presets a systematic and repeatable evaluation method of the proposed framework and how the data used for the performed case studies is partitioned in order to achieve the intended objectives.

The framework processed data set (EPSS) is compared to other data sets, including the raw data set (PVS), raw data set with the most significant predictor variable (PVS_{Size}), the raw data set without outliers (PSS) and the framework processed data set with the most significant predictor variable (EPSS_{Size}). These comparisons are repeated for a set of commonly used estimation methods, which are selected based on high-level criteria in section 6.4. The research adopts both classic criteria (i.e., BMRE, MMRE, MdMRE, Pred(l)) and more recently proposed criteria (i.e., MMER, MdMER) for accuracy evaluation. The detailed definition of these selected criteria are presented in section 6.5. Two case studies in chapter 7 and 8 are provided based on a multi-company data set and a company-specific data set. Both data sets are in the public-domain.

Some methods, such as Ordinary Least Squares (OLS) regression, Robust Regression (RR), cannot cope with missing data related to the parameters used in their specification. To ensure a fair comparison of all the applied techniques and the use of the same projects as the basis for comparing predictions, no missing values will be considered in this research. Therefore all cases containing missing values are removed from the historical data set before the analysis.

Cross validation [295] is an established techniques for estimating the accuracy of a predictor. Using a particular data set to build a cost estimation model, and then computing the accuracy of the model using the same data set, will lead to an optimistic accuracy evaluation (i.e., the error will be artificially low, and does not reflect the performance of the model on another unseen data set) [113]. A cross-validation approach gives more realistic accuracy measures. It has been proved that cross validation is a model evaluation method that is better than residual evaluation [177]. The problem with residual evaluations is that they do not give an indication of how well the prediction model will do when it is applied to make new predictions for unseen data. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is complete, the data that was removed can be used to test the performance of the trained model. The cross-validation approach divides the whole data set into multiple train and test sets, calculating the accuracy for each test set and then aggregating the accuracies across all the test sets. This is the basic idea for a whole class of model evaluation methods called cross validation. The *holdout* method is the simplest kind of cross validation, *K-fold cross validation* is one way to improve upon the *holdout* method. The process of splitting data by *K-fold cross validation* is illustrated in Figure 6.1. And *leave-one-out cross validation* is a K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set.

For a K-fold cross-validation, the data are partitioned into K equal parts. The first part is used as testing data set; the rest is used as training data set. Then, the second part is used for the test data and the rest is used for a new training. This procedure is repeated K times and the predictions of the K test data are averaged, as Figure 6.1 illustrates. It is essential that no knowledge of the models is transferred from fold to fold. There exist no clear rules regarding the number of folds to use for the cross-validation, whereby the simplest and clearest way of performing cross-validation is to leave one sample out at a time. Although *leave-one-out* cross validation often

works well for estimation generalization functions, it may work poorly for discontinuous error functions. *Leave-one-out* cross validation can also meet trouble with various model selection methods. One problem is the lack of continuity [35], a small change in the data can cause a large change in the model selected. For choosing subsets of inputs in linear regression, Breiman and Spector [37] found 10-fold and 5-fold cross validation show better performance than leave-one-out. Kohavi [177] also obtained good results for 10-fold cross-validation with empirical decision trees. Therefore, this research employs 10-fold cross validation, which is also consistent with Briand, *et.al*'s recent research [44] in assessment and comparison of common software cost estimation modeling techniques. Table 6.1 summarizes the hypotheses tested in the empirical

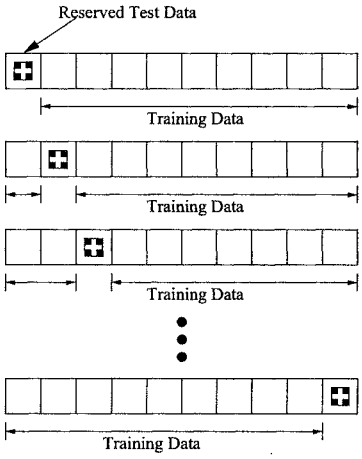

Figure 6.1: K-fold Cross Validation

evaluation of the proposed framework, their underlying questions, the corresponding types of analysis, and the corresponding references for the results described.

Hypothesis	Underlying Question	Analysis Procedure	Evaluation Results
Hypothesis 1a Hypothesis 1b	Which data set provides the most accurate prediction: PVS_{Size}, PVS, or EPSS?	For each technique: Accuracy comparison among data sets in 1a and 1b.	Case Study A: section 7.4, section 7.5; Case Study B: section 8.4, section 8.5
Hypothesis 2a Hypothesis 2b	Which data set provides the most accurate prediction: PSS, EPSS, or $EPSS_{Size}$?	For each technique: Accuracy comparison among data sets in 2a and 2b.	Case Study A: section 7.4, section 7.5; Case Study B: section 8.4, section 8.5
Hypothesis 3a Hypothesis 3b	Whether the prediction accuracy is improved based on both heterogeneous and homogenous data sets by applying the proposed framework?	Accuracy comparison between data sets in 3a and 3b.	Case Study A: section 7.4, section 7.6; Case Study B: section 8.4, section 8.6

Table 6.1: Research Hypotheses vs. Analysis vs. Results

6.4 Estimation Methods Applied

This section provides descriptions of how a selection of estimation methods was concretely applied within the empirical studies. General descriptions of each method's assumptions, strengths and weaknesses were provided in section 3.4, section 3.5 and section 3.9.

6.4.1 High-Level Selection Criteria

As evidenced in Chapter 3 there have been numerous estimation methods developed and applied to software cost estimation. In order to review methods will be applied in this thesis, focus is directed at the methods that fulfill the following high-level criteria:

Currency: It will exclude methods that have been developed more than 15 years ago and have not been updated since then. Therefore, we will not apply models like the Walston-Felix Model [307], the Bailey-Basili Model [10], the SEER (System Evaluation and Estimation Resources) model [136], the COPMO (Comparative Programming MOdel) model [61], etc.

Level of Description: The focus is on methods that are not proprietary. We only apply methods for which the information is publicly available, unambiguous, and somewhat unbiased. Therefore, we will not consider generic proprietary methods ("black-box" methods). The level of detail of the description for this kind of methods depends on the level of available, public information. Examples of proprietary estimation methods (and tools) are PRICE-S [66], Knowledge Plan [145], ESTIMACS [264, 160, 180], etc.

Automation: It considers methods that can substantially be automated since the intention is to easily apply and repeat the estimation process. This is important in practice, because automated methods are more likely to be used [105]. Moreover, this validation demanded intensive computational procedures. Thus, performing replications is facilitated if automatable methods are applied.

Suitable Input Requirements: The variables collected in the data sets used should be adequate and suitable as inputs to the effort estimation methods. Thus, although the COCOMO method is a well-known method and was very often compared in empirical studies (see Section 3.9), the COCOMO input requirements

were incompatible with the data supplied by the data sets used. Therefore, this method was not selected for investigation within this thesis.

Interpretability: The resulting estimation method output should be able to be reasonably understood and assessed. Non-interpretable, too complex results are not very likely to be used in practice, as practitioners usually want to have clear justifications for an estimate they will use for project planning [105]. Therefore, Artificial Neural Networks which compromise interpretability [91, 104] will not be evaluated in this thesis.

Independency: To evaluate the performance of the proposed preliminary data analysis framework, the estimation methods should be independent with the proposed framework. Therefore the Stepwise Analysis of Variance (ANOVA) prediction method [168], and Optimized Set Reduction (OSR) which embedded an initial utility data analysis procedure will not be evaluated in this thesis. The main method of the stepwise ANOVA analysis procedure is identical to the proposed framework proposed in this research.

Based on the above high-level estimation model selection criteria, only a few cost estimation methods are available for evaluating the preliminary analysis framework. The selected methods are Ordinary Least-Square Regression (OLS), Robust Regression (RR), Classification and Regression Trees (CART) and K-Nearest Neighbour (KNN).

6.4.2 Ordinary Least Squares (OLS) Regression

We applied multivariate OLS regression analysis [44, 200] fitting the data to specified models. A linear relationship between effort and all predictor variables in each data set (i.e., PVS, PVS_{Size}, PSS, EPSS and $EPSS_{Size}$) are modeled. As many statistical techniques, the OLS regression method assumes that the underlying data is normally distributed. However, the distributions of some variables, for example, software project effort and size are not normally distributed, the distributions are skewed to the right. Section 4.4.8 describes solutions to approximate a normal distribution by applying transformation techniques. In this research, we apply the log-linear transformation (i.e., exponential functional form). Exponential relationships have been also modeled in many other investigations [79, 107, 231, 168, 203].

6.4.3 Robust Regression (RR)

We used an iterative robust procedure implemented in the R 2.0.1 tool [62]. As independent variables we used the ones identified as significant in the applied stepwise OLS regression procedure. This variant of robust regression iteratively performs weighted OLS regressions. This corresponds to minimizing the sum of the weighted squared residuals. It calculates weights based on absolute residuals, and regresses again using those weights. Outlying observations are given lower weights than normal observations. The iterations stop when the maximum change in weights drops below a certain tolerance. The weights are derived from a combination of two weight functions (Huber's function and bi-weights). Both weighting functions are used because Huber weights have problems dealing with severe outliers, while bi-weights sometimes fail to converge or have multiple solutions [288].

6.4.4 Classification and Regression Trees (CART)

Regression trees were generated based on the CART [36] algorithm using rpart and rpart.control commands in R 2.0.1 software package. We applied project *effort* [286] as the dependent variable. The stopping criterion was set to a minimum of ten and five observations in each terminal node, for the ISBSG R9 and the Bank63 study, respectively. Predictions were based on the median value of the response variable in a terminal node. The current settings are summarized in Table 6.2.

CART	Settings for Evaluation in This Study
Splitting Criterion	Analysis of Variance (ANOVA)
Splitting Index	Gini by default in R
Stopping Criterion	$\geq$ 10 (ISBSGR9), $\geq$ 5 (Bank63) observations in a terminal node
Predicted Value	Based on median of effort variable in terminal node

Table 6.2: CART Application Configuration

6.4.5 K-Nearest Neighbour (KNN)

K-nearest neighbour classification for test set from training set. For each row of the test set, the 'K' nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the 'K'th nearest vector, all candidates are included in the vote. That is if more than one similar neighbour was retrieved due to equal similarity values, then

the average predicted value was used. This choice was used for the sake of simplicity. Moreover, Shepperd and others [279] report nearly equivalent accuracy when using more than two similar projects or using weighted averages. Another recent study [5] concluded that the best choice for the number of analogs was one project. Their study show that an increase of the number of analogs causes an increase in the mean MRE. In this study, to identify the best 'K' of KNN, a set of prediction models based on KNNs are built, where K=1 to 10; for each competing data set, the best result will be selected for comparison.

6.5 Evaluation Criteria

As described in section 3.8, that there are various evaluation criteria proposed (e.g., RE, MER, MMER, MdMER, BMMRE, Pred(l) etc.), however, these criteria are often not in agreement for a set of models and projects in the sense that we cannot say which model is best without making a subjective judgment on the relative importance of the laudation criteria. Conte *et al.*, suggested that we can chose the models that have smaller average errors, and more predictions that fall within 25% of the actual values.

This section discusses a selection of the most commonly used (i.e., MRE, MMRE, MdMRE, Pred(l) etc.) and the most recent(i.e., MER, MMER, MdMER, and BMMRE,) evaluation criteria. All of the above criteria will be applied to evaluate the proposed framework. A performance matrix will be produced to illustrate which model based on what data set performs the best under each of the criterion. For example, Table 7.15 in section 7.5.1 and 8.14 in section 8.5.1 are such matrices.

6.5.1 Magnitude of Relative Error (MRE) and Mean MRE (MMRE)

The Magnitude of Relative Error (MRE) is one of the most commonly used criteria of the evaluation of cost estimation models [61] [44] [279]. A large positive MRE would suggest that the model generally overestimates the effort, while a large negative value would indicate the reverse. MRE is defined as:

$$MRE_i = \mid \frac{E_i - \widehat{E}_i}{E_i} \mid \tag{6.1}$$

The MRE value is calculated for each observation i whose effort is predicted. The aggregation of MRE over multiple observations, say n, can be achieved through the Mean MRE (MMRE), which is the mean of absolute percentage errors:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{6.2}$$

6.5.2 The Median of MRE (MdMRE)

The MMRE is sensitive to individual predictions with excessively large MREs (observation outliers). Therefore, an aggregate measure less sensitive to extreme values should also be considered, namely the median of MRE values for the n observations (MdMRE).

6.5.3 The Balanced MMRE (BMMRE)

There has been some criticism of the MMRE measure, in particular that it is unbalanced and penalized overestimates more than underestimates. For this reason, Miyazaki *et al.* [211] also proposed a balanced mean magnitude of relative error measure as follows:

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n} \frac{\mid E_i - \widehat{E}_i \mid}{min(E_i, \widehat{E}_i)} \tag{6.3}$$

6.5.4 MER and Mean MER (MMER)

An alternative evaluation metrics MER was proposed by Kitchenham [170] is claimed to be preferable to MRE [170, 93] since it measures the error relative to the estimate. MER is defined as:

$$MRE_i = \frac{\mid E_i - \widehat{E}_i \mid}{\widehat{E}_i} \tag{6.4}$$

The MER value is calculated for each observation i whose effort is predicted. The aggregation of MER over multiple observations, say n, can be achieved through the Mean MER (MMER), which is the mean of absolute percentage errors:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} \frac{\mid E_i - \widehat{E}_i \mid}{\widehat{E}_i} \tag{6.5}$$

6.5.5 The Median of MER (MdMER)

The MMER is sensitive to individual predictions with excessively large MERs (observation outliers). Therefore, an aggregate measure less sensitive to extreme values should also be considered. namely the median of MER values for the n observations (MdMER).

6.5.6 The Prediction at Level l (Pred(l))

A complementary criterion that is commonly used is the prediction at level l [45, 280, 286, 89, 131, 160, 221, 23], Pred(l),

$$Pred(l) = \frac{1}{n}\sum_{i=1}^{n} a_i \quad a_i = \begin{cases} 1 & \text{if MRE}_i \leq l \\ 0 & \text{if MRE}_i > l \end{cases} \tag{6.6}$$

where a_i is the number of observations where MRE is less than or equal to l. MMRE and Pred(l) are, respectively, measures of the spread and the kurtosis of the variable z, where z = estimate/actual [170]. In this research $z = E_i/\widehat{E}_i$. For the experiments are implemented in this research, we obtained three levels of Pred as Pred(25%), Pred(10%), and Pred(5%) of each models for the comparison purposes.

6.6 Significance Testing

Stensrud *et al.* [292], Shepperd *et al.* [277] established an empirical study of conducting formal tests of significance comparing competing prediction results. By following the empirical evaluation guidelines from Shepperd *et al.*'s [277], we adopted t-test, Wilcoxon Matched-pairs test and Mann-Whitney U test to test the significance of the MREs from the competing predicting results based on selected data sets. As discussed in section 4.4.5, 4.4.6 and 4.4.7, the selection of appropriate significance tests depends on the characteristics of the competing prediction results. To test Hypotheses 1 and 2 presented in section 6.2, two sets of significance tests at 95% confidence interval are defined as the following:

Test A Run t-tests and Wilcoxon Matched-Pairs Signed Rank tests to MRE values for comparison between the model constructed from framework processed data

set (EPSS) and from other data sets (PVS, PVS_{Size}, PSS, and $EPSS_{Size}$) for each prediction technique (OLS, RR, CART and KNN).

Test B Run Mann-Whitney U tests to MRE values for comparison between the prediction models constructed by different techniques (OLS, RR, CART and KNN) from the same framework processed data set (EPSS).

Test A is designed to test the Hypotheses 1 and 2 in section 6.2. It tests whether the prediction accuracy based on the framework processed data set (EPSS) is significantly different from other data sets (PVS, PVS_{Size}, PSS, and $EPSS_{Size}$). When the other data set has different sample size with the proposed framework processed the data (i.e., EPSS vs. PVS and EPSS vs. PVS_{Size}), the t-test is applied. Otherwise, the matched-pairs Wilcoxon test is applied. This process is repeated for project effort prediction models constructed by each prediction technique including OLS, RR, CART and KNN.

T-tests of MRE values in the Test A test Hypothesis 1a that EPSS performs no worse than PVS and Hypothesis 1b that EPSS performs no worse than PVS_{Size}. According to the principle that the lower the MRE the better the prediction performance, we proposed that:

$\boldsymbol{H_0}$ $\mu_1 > \mu_2$

$\boldsymbol{H_a}$ $\mu_1 \leq \mu_2$

where μ_1 is the mean number of MRE values of EPSS data set and μ_2 is the mean number of MRE values of PVS for Hypothesis 1a and the mean number of MRE values of PVS_{Size} for Hypothesis 1b. For example, if we run a t-test of MREs between EPSS and PVS, and obtain P-value less than 0.05 at 95% confidence level, we can reject the null hypothesis H_0 and conclude that EPSS outperforms PSS data set, at the 0.05 level of significance. If both t-test results for hypotheses 1a and 1b are significant, we can conclude Hypothesis 1 is true.

Wilcoxon-tests of MRE values in the Test A test Hypothesis 2a that EPSS performs no worse than PSS. than $EPSS_{Size}$.

$\boldsymbol{H_0}$ MREs of EPSS are higher than MREs of PSS.

$\boldsymbol{H_a}$ MREs of EPSS are not higher than MREs of PSS.

Wilcoxon-tests of MRE values in the Test A test Hypothesis 2b that EPSS performs no worse than EPSS_{Size}.

$\boldsymbol{H_0}$ MREs of EPSS are higher than MREs of EPSS_{Size}.

$\boldsymbol{H_a}$ MREs of EPSS are not higher than MREs of EPSS_{Size}.

If both Wilcoxon-test results for hypothese 2a and 2b are significant, we can conclude Hypothesis 2 is true.

Test B is designed to test whether the prediction accuracy of one prediction technique is significantly different from others, when they are all based on the same framework processed data set (EPSS). Although MER values in test B are also naturally paired (all MRE shared the same actual project values of the same data set), unlike Test A, we are not able to assume which observation in each pair has the higher (or lower) value. That is, we cannot assume which technique performs better than the others. Therefore the Wilconxon test is inappropriate for comparisons between models constructed by different prediction techniques, we use Mann-Whitney U tests instead. If the proposed framework is validate and effective, ideally, we should detect significant difference in test A and no significant difference in Test B. If the above is true for homogenous data set in this study, we conclude Hypothesis 3a is true. If the above is true for heterogenous data set in this study, we conclude Hypothesis 3b is true.

To evaluate Hypothesis 3, two case studies will be conducted based on a multi-company data set (ISBSG R9) and a company-specific data set (Bank63). Both data sets are in the public-domain.

6.7 Summary

This chapter elaborated the research hypotheses and gave a general description of how the empirical studies, described in section 1.6.2, will be performed. It selected and defined the accuracy evaluation criteria including both classic criteria (i.e., BMRE, MMRE, MdMRE, Pred(l)) and more recently proposed criteria (i.e., MMER, Md-MER).

The efficiency evaluation of the proposed framework is performed in comparison between the framework processed data set (EPSS) and other data sets, including the

raw data set (PVS), raw data set with the most significant predictor variable (PVS_{Size}), the raw data set without outliers (PSS) and the framework processed data set with the most significant predictor variable ($EPSS_{Size}$). These comparisons are repeated for a set of selected estimation methods based on the high-level estimation model selection criteria. These methods are Ordinary Least-Square Regression (OLS), Robust Regression (RR), K-Nearest Neighbour (KNN) and Classification and Regression Trees (CART).

Finally, this chapter outlined the evaluation measures and testing procedures, which constitute a systematic and repeatable evaluation method.

Part III

Case Studies

Chapter 7

Case Study A: ISBSG R9

7.1 Introduction

This first empirical study uses raw data from the International Software Benchmarking Standards Group (ISBSG). The ISBSG has developed and refined its data collection standards over a ten year period based on the metrics that have proven to be the most useful in helping to improve software development management and processes.

This chapter outlines the data set and preliminary data analysis using the framework proposed in chapter 5. It explains how the comparative evaluation described in chapter 6 is applied. It presents the results obtained from the application of each extracted subset and the raw data set based on a selection of estimation methods. Finally there is a discussion of the result. A free statistical software package R 2.1.0, which was developed at University of Minnesota [62], was used.

7.2 Data Set Description

The ISBSG R9 is released in 2005 containing data from 3,024 software projects. The data in the repository has come from twenty countries, with 70% of the projects being less than six years old. The major contributors are indicated in Figure 7.1. The application context is world-wide and multi-organizational. The ISBSG data has been used in many studies [188, 190, 219, 134, 133, 192, 311, 231, 232].

As recommended by ISBSG, only Quality Rating (QA) of A or B data were selected for this study. There are 2,790 data that fulfill the above two conditions. We further eliminate less reliable project data based on the rating of the Unadjusted Function Point (UFP) value; thus, from the above 2,790 data, only the data with UFP Rating equal to A are selected. There are 2,157 data that fulfill the above three conditions. After removing the projects and attributes containing missing values, we finally select 560 project data with 16 project attributes for this study. Therefore, this selected subset of ISBSG R9 data consists of 1 predicted variable - project *effort*, and 15 predictor variables. The above set of 560 project data is defined as the raw data set for this study which we call Project Vector Sets (PVS), containing 3 numerical variables and 13 categorical variables. Five out of the 13 categorical variables consist of more than 5 categories. These five categorical variables are Organization Type (36 categories), Business Type (40 categories), Application Type (41 categories), Primary Program-

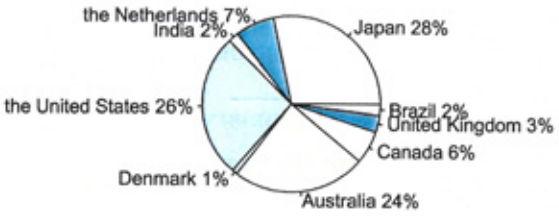

Figure 7.1: Case Study A: ISBSG R9 Data Set by Country

ming Language (33 categories), and Main Data Base (32 categories). For a description of the variables see Tables B.1 in Appendix B. This section only presents four pie charts of the organization type, application type and main programming language as illustrated in Figures 7.1 to B.10. The detailed data set profile is presented as the outcomes of step 0 to step 4 of the preliminary analysis framework in the following section.

7.3 Extracting Optimal Subsets

This section details the application of the preliminary data analysis framework to extract optimal subsets of ISBSG R9.

- Step 0: **Data Preparation.**
  As described in the preceding section, PVS is set up as a matrix containing 560 rows (project vectors) and 16 columns (attributes), after removed project vectors with missing values removed. There are 3 numerical variables and 13 categorical variables in the PVS.

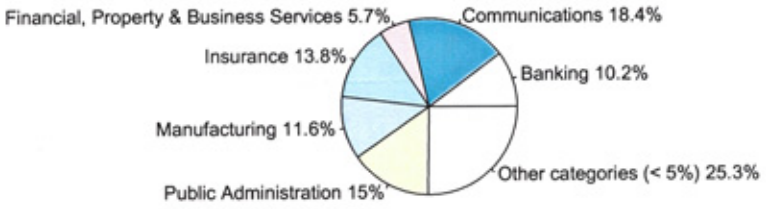

Figure 7.2: Case Study A: Main Categories of OrgType

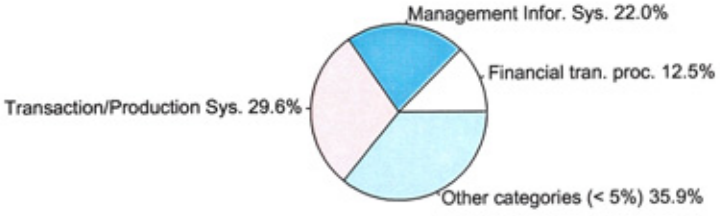

Figure 7.3: Case Study A: Main Categories of AppType

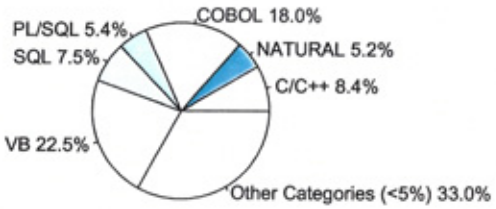

Figure 7.4: Case Study A: Main Categories of PriLang

Categories in text are transformed to nominal numbers. The allocated numbers and corresponding categories are detailed in step 3. Table 7.1 illustrates a fragment of the PVS after category transformation.

Numbers	Attributes	Project 1	Project 2	Project i	Project 560
1	Size	6000	755	...	41
2	Effort	55860	7350	...	498
3	Duration	20	24	...	3
4	CountApp	1	1	...	1
5	DevType	1	1	...	1
6	OrgType	5	5	...	32
7	BusiType	15	15	...	27
8	AppType	21	21	...	9
9	Architecture	3	2	...	2
10	DevPlatform	1	2	...	1
11	LangType	3	4	...	4
12	PriLang	5	5	...	1
13	MDBS	6	6	...	16
14	UMethod	2	1	...	2
15	RMethod	1	1	...	1
16	RLevel	1	1	...	1

Table 7.1: Case Study A: A Fragment of PVS with Transformed Categorical Variables

- Step 1: **Remove Outliers by Measuring Numerical Variables.**
  Table 7.2 summarizes variables with numerical values. This table shows the vari-

able name (Var. Name), the Observations (Obs.), the minimum, the maximum, the average (Mean) and the standard deviation (Std. Deviation). The *size* of projects in this database (PVS) ranges from 6 function points to 13,580 function points; the average project *size* is 574 function points. *Effort* ranges from 17 man-hours to 150,040 man-hours with an average of 5,017 man-hours. *Duration* ranges from 1 month to 84 months; the average duration is 8 months. Because the sample size of ISBSG R9 is 560 (> 80), we use the threshold value of standard scores of 3. Regarding the characteristic of the software project data (i.e. project effort, size and duration are always positive), we also measure the 5% trimmed mean and remove the lowest and highest 5% of the observations.

Var. Name	Obs.	Minimum	Maximum	Mean	Std. Deviation
Size	560	6	13,580	574	977
Effort	560	17	150,040	5,017	9,893
Duration	560	1	84	8	8
Valid Obs.	560				

Table 7.2: Case Study A: Summary of the Numerical Variables in PVS

- Step 2: **Data Transformation.**
  We plot the histograms for each numerical variable in PVS to see if the variables are normally distributed. Three numerical variables include project *effort*, *size*, and *duration* are examined. Figures 7.5, 7.7 and 7.9 demonstrate that none of these numerical variables are normally distributed. The PVS contains a few projects with a very high effort, or a very big size or a very long duration. It also contains many low effort, small size and short duration projects. This is typical in a software development project data set. To approximate a normal distribution we must transform these variables.

  We create three new numerical variables by taking each original numerical's log values, and name new variables as LnEffort, LnSize and LnDuration. We recreate histograms of each transformed variable in Figures 7.6, 7.8 and 7.10. This transformation method makes large values smaller and brings the data closer together. These transformed numerical variables will be used in the following steps.

- Step 3: **Remove Outliers by Measuring Categorical Variables.**
  First, we tabulate each categorical variable that has been transformed in step 1 to check how many projects are in each category. Tables B.2 to B.14 and

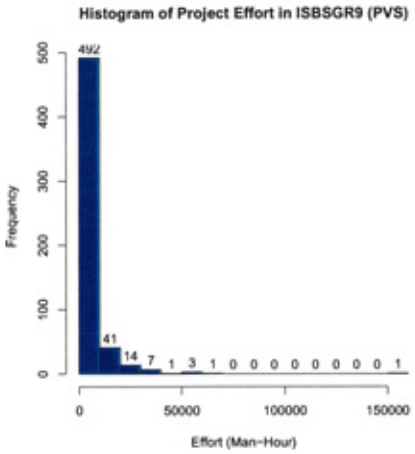

Figure 7.5: Hist Effort in PVS

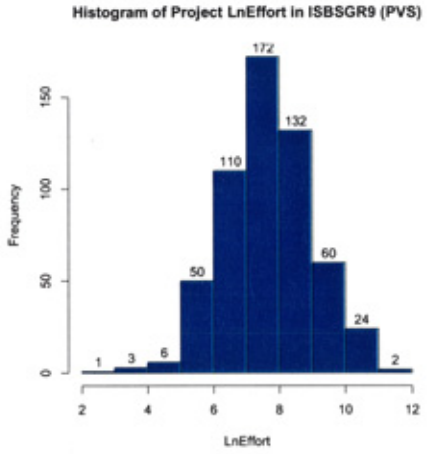

Figure 7.6: Hist LnEffort in PVS

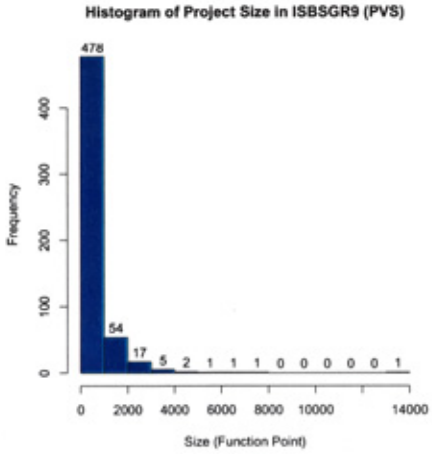

Figure 7.7: Hist Size in PVS

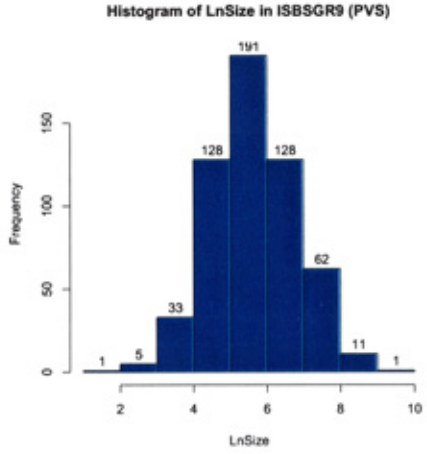

Figure 7.8: Hist LnSize in PVS

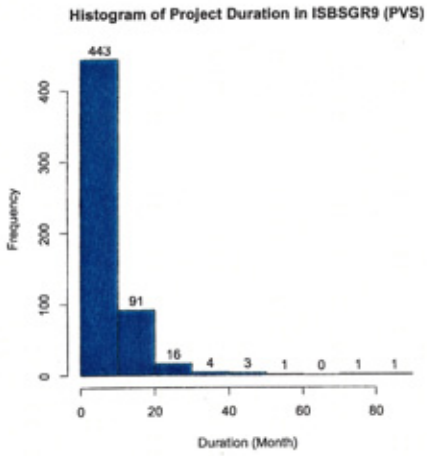

Figure 7.9: Hist Dur. in PVS

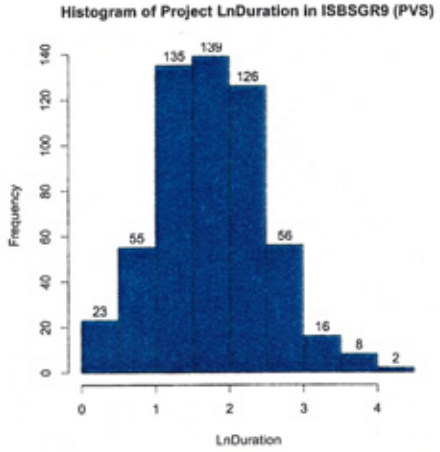

Figure 7.10: Hist LnDur. in PVS

Figures B.1 to B.13 in Appendix B demonstrate the number and the percentage of observations in each category. For example, AppType (Application Type) mainly consists of Transaction/Production Systems (29.6%), Management Information Systems (22.0%), and Financial transaction process (12.5%).

Second, we remove projects that contain any categories appearing less than 3 times. There are 198 outliers removed so far. This step yields PSS, that is a row reduced subset of PVS.

Third, we repeat steps 1 to 3 for PVS until there are no more outliers. Table 7.3, Figures 7.11 to 7.16 are the histograms and Q-Q normality plots for these three transformed numerical variables in PSS. These figures demonstrate that the transformed variables are normally distributed and there are no remaining numerical outliers. After processing in the previous steps, we now observe that the *size* of projects in this row reduced data set (PSS) ranges from 25 function points to 1,924 function points with the average being 375 function points. *Effort* ranges from 90 man-hours to 18,920 man-hours, with an average of 2,760 man-hours. *Duration* ranges from 1 months to 16 months with the average being 6 months. It is important to understand the range of numerical variables in a data set. Maxwell suggests not using data set to predict new projects with numerical variables out of the above identified ranges, e.g., not to apply to projects bigger than 1,924 function points, or less than 25 function points for the models developed from ISBSG R9 data. There are not categorical outliers in PSS, we

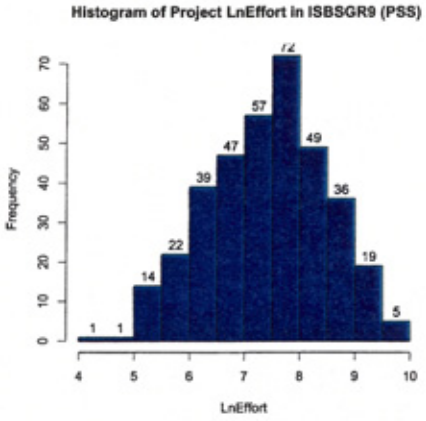

Figure 7.11: Hist LnEffort in PSS

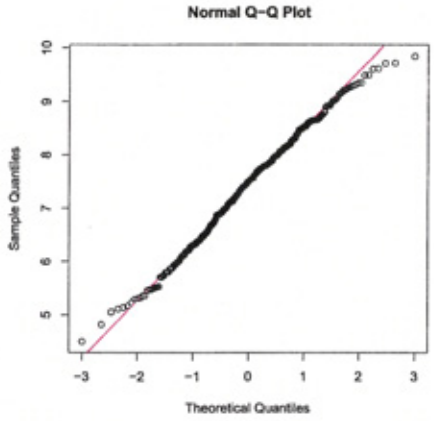

Figure 7.12: Q-Q Plot LnEffort in PSS

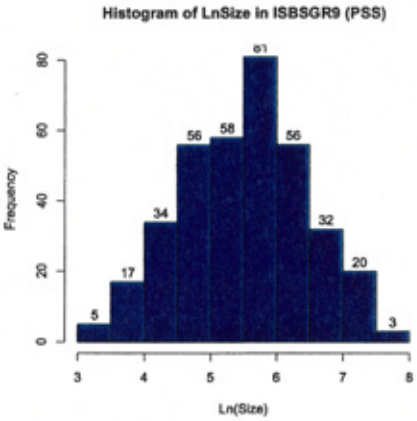

Figure 7.13: Hist LnSize in PSS

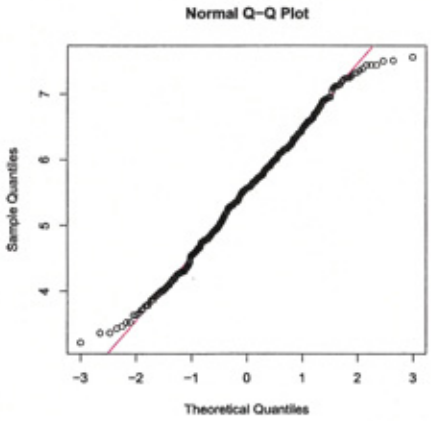

Figure 7.14: Q-Q Plot LnSize in PSS

can either boxplot or tabulate categorical variables to confirm it. Figure 7.17 indicates that there are no remaining categorical outliers.

Var. Name	Obs.	Minimum	Maximum	Mean	Std. Deviation
Size	362	25	1,924	375	368
Effort	362	90	18,920	2,760	2,985
Duration	362	1	16	6	3
Valid Obs.	362				

Table 7.3: Case Study A: Summary of the Numerical Variables in PSS

- Step 4: **Test Collinearity Between Numerical Predictor Variables.**

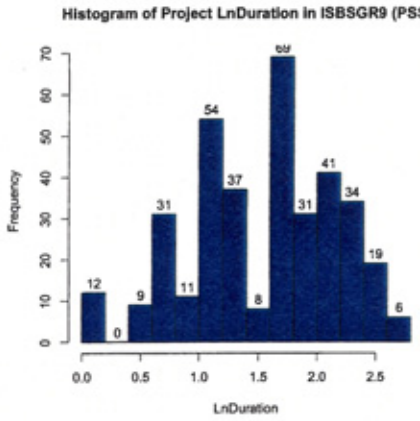

Figure 7.15: Hist LnDuration in PSS

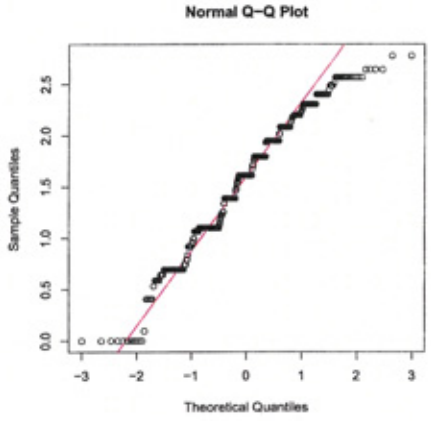

Figure 7.16: Q-Q Plot LnDuration in PSS

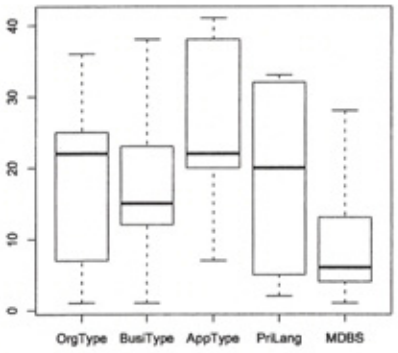

Figure 7.17: Boxplot Complex Catg. Var.s in PSS

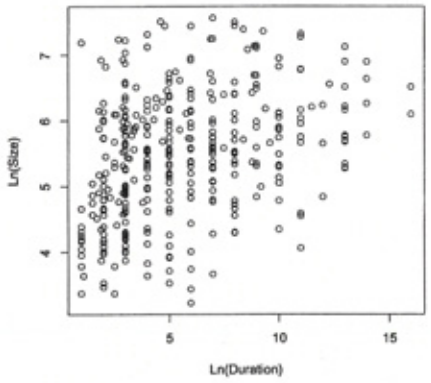

Figure 7.18: LnSize vs LnDuration in PSS

We present a 2-dimensional plot diagram for each pair of transformed numerical variables in Figures 7.18,7.19 and 7.20, and observed there are correlation between LnEffort, LnDuration and LnSize. We need to further examine these three variables, in particular to identify whether there is a strong relationship between LnSize and LnDuration. Table 7.4 indicates LnSize and LnDuration are not strongly correlated ($rho = 0.38 < 0.75$), and therefore they can be included in the same model to predict project Effort. The results presented in this table also confirmed the preceding observation that both LnSize and LnDuration are correlated to LnEffort, the correlation coefficients are 0.68 and 0.53 respectively.

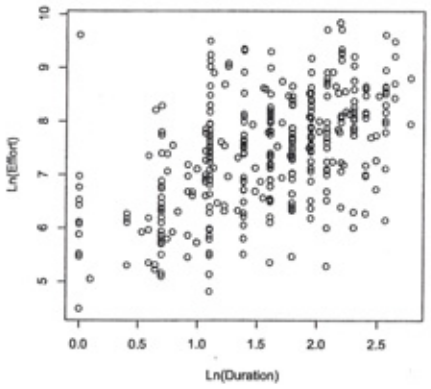

Figure 7.19: LnDuration vs LnEffort in PSS

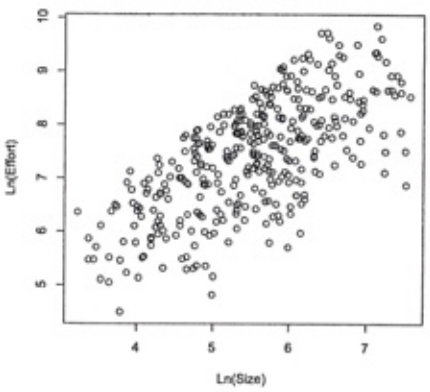

Figure 7.20: LnSize vs LnEffort in PSS

Variables	Correlation Coefficient (rho)
LnEffort and LnSize	0.68
LnEffort and LnDuration	0.53
LnSize and LnDuration	0.38

Table 7.4: Case Study A: Correlation Analysis of Numerical Variables in PSS

- Step 5: **Stepwise Regression Analysis.**
  In this step, we test the correlation between the numerical predictor variables and the predicted variable by running a backward stepwise regression analysis. This procedure allows us to determine the relative importance of each numerical predictor variable's relationship to the predicted variable. Based on the results presented in Table 7.5, there are no numerical variables removed. Compared with the project *duration*, *size* explained more variation in effort. That is, the most significant numerical variable is project *size*, the second significant numerical variable is project *duration*. The best model with numerical variables (*size* and *duration*) alone explained 54% of the variation of project *effort*. We will explore whether any of the categorical variables explain more of the variation of project *effort* in the following step.

- Step 6: **Stepwise ANOVA Analysis to build a k-variable model.**

Model	R	R^2	Adjusted R^2	
1	0.677(a)	0.459	0.457	
2	0.732(b)	0.536	0.534	
a Predictors: (Constant), LnSize				
b Predictors: (Constant), LnSize, LnDuration				
	Est. Coef.	Std. Er.	t value	$Pr(>\|t\|)$
(Intercept)	3.16545	0.22486	14.077	$< 2e-16$
LnSize	0.63067	0.04344	14.518	$< 2e-16$
LnDuration	0.49877	0.06448	7.735	$1.06e-13$

Table 7.5: Case Study A: Stepwise Regression Summary of PSS

The following sub-steps are to be followed.

- Identify the best one-variable model. There are 15 predictor variables, thus we need to build 15 one-variable models. The best one-variable model will be determined by the adjusted correlation coefficient square (R^2) value. This value is calculated for each predictor variable with the response variable (project *effort*). The regression adjusted R^2 value will be calculated when the predictor variable is type ratio or interval (i.e., *size* and *duration* in PSS) and the ANOVA adjusted R^2 value will be calculated when the explanatory variable is ordinal or nominal (the remain 13 variables in PSS). The predictor variable that has the greatest adjusted R^2 value explains most of the variation in the predicted variable and is kept in the model, In this case it is (transformed) project *size* with adjusted R^2 value equals to 0.4574.

- Identify the best two-variable model by the same procedure. Determine which variable, x_i, in addition to LnSize, explains the most variation in the predicted variable. This means that 14 two-variable models must be built. The most significant variable - LnDuration is selected as the second variable to be added into the two-variable model with LnSize. The adjusted R^2 value of adding LnDuration equals to 0.5336. Then we keep both LnSize and LnDuration in the model to build the best 3-variable model. This process is continued iteratively until there is no significant variables that can be added to improve the model. Table 7.9 presents the results of this iterative process.

- Finally, we obtain the best 5-variable model, which explains 62.22% of the

variance of project *effort*, see equation 7.1.

$$\begin{aligned} \text{LnEffort} &= f(\text{LnSize, LnDuration, LangType,} \\ &\quad \text{Architecture, PriLang}) \end{aligned} \tag{7.1}$$

- Step 7: **Test Collinearity Between Extracted Predictor Variables.**
  The independence between the numerical variables (LnSize and LnDuration) has been tested in step 4, however, at this stage it is not known whether these numerical variables are correlated to any other categorical variables. Also, it must be established whether the categorical variables are independent. These selected categorical variables are LangType, Architecture, and PriLang.

  As suggested by Maxwell [203], to determine if there is a relationship between a categorical variable and a numerical variable, analysis of variance (ANOVA) procedures can be used. To determine if there is a relationship between two categorical variables, the Chi-square test can be used. Test results are presented in Table 7.6. Significant relationships have been identified between selected categorical variables, between LnSize and Architecture, and between LnSize and PriLang. Therefore, we employ Univariate Analysis to allow further discrimination; Table 7.7 presents the test results. The strong correlation between LangType and PriLang and between LangType and Architecture are confirmed by their combined effect on the predicted variable (LnEffort). We cannot include these three categorical variables together in the same model. We keep the one with the highest significant rank based on step 6, that is LangType. Therefore, both Prilang and Architecture are removed. The best model we could obtain so far is the 3-variable model as illustrated in the equation 7.2

$$\text{LnEffort} = f(\text{LnSize, LnDuration, LangType}) \tag{7.2}$$

  This step yields the Extracted PSS data set - EPSS, that is a column reduced subset of the PSS. The EPSS consists of 362 rows (projects) with 4 columns (selected variables, i.e. *effort, size, duration* and LangType).

- Step 8: **Extract and Interpret Equations.**
  Normally, there are transformed variables or categorical variables which have different multiplier (coefficients) for different levels. Therefore we need to extract the equation of identified 3-variable model. The equation read from the final

Variables	$Pr(>F)$	Test Methods	Sign. Code
LnSize vs. LangType	0.117	ANOVA	
LnSize vs. Architecture	9.529e-06	ANOVA	***
LnSize vs. PriLang	0.001230	ANOVA	**
LnDuration vs. LangType	0.565	ANOVA	
LnDuration vs. Architecture	0.8788	ANOVA	
LnDuration vs. PriLang	0.7178	ANOVA	
LangType vs. Architecture	3.305e-08	Chi Square	***
LangType vs. PriLang	1.271e-14	Chi Square	***
Architecture vs. PriLang	0.001034	Chi Square	**

Significant codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Table 7.6: Case Study A: ANOVA and Chi-Square Tests Results

Tests of Between-Subjects Effects
Dependent Variable: LnEffort

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	115.838(a)	37	3.131	3.560	0.000
Intercept	6610.827	1	6610.827	7516.335	0.000
LangType	17.918	1	17.918	20.373	0.000
PriLang	48.222	18	2.679	3.046	0.000
Architecture	5.193	2	2.597	2.952	0.054
LangType * PriLang	0.000	0	0.000	0.000	0.000
LangType * Architecture	0.000	0	0.000	0.000	0.000
PriLang * Architecture	9.956	13	0.766	0.871	0.585
LangType * PriLang * Architecture	0.000	0	0.000	0.000	0.000
Error	284.967	324	0.880		
Total	20320.782	362			
Corrected Total	400.805	361			

a $R^2 = 0.289$ (Adjusted $R^2 = 0.208$)

Table 7.7: Case Study A: Tests of Between-Subjects Effects of Selected Categorical Variables

model's output is

$$\begin{aligned} \text{LnEffort} &= 4.414 + (0.657 \times \text{LnSize}) + \\ &\quad (0.473 \times \text{LnDuration}) - \\ &\quad (0.389 \times \text{LangType}) \end{aligned} \tag{7.3}$$

This can be transformed into the following non-linear equation for effort.

$$\begin{aligned} \text{Effort} &= (82.5992 \times \text{Size}^{0.657}) \times \\ &\quad (\text{Duration}^{0.473} \times e^{-0.389 \times \text{LangType}}) \end{aligned} \tag{7.4}$$

The multipliers for the categorical variable LangType are listed in Table 7.8.

These multipliers indicate what is e to the power (-0.389 $\times$ LangType) when LangType is 1 (3GL), when it is 2 (4GL), etc.?

Category Defn.	Value	LangType Multiplier
3 GL	1	0.219
4 GL	2	-0.327
APG	3	0.000

Table 7.8: Case Study A: Effort Factor Multipliers - Emethod

- Step 9: **Test residuals.** In a well-fitted model, there should be no pattern to the errors (residuals) plotted against the fitted values. The term "fitted value" refers to the project effort predicted by the model, the term "residual" is used to express the difference between the actual effort and the predicted effort for each project. There should be no pattern in the residuals of our final model. Finally, plot a histogram and a Quantile-Quantile (Q-Q) Plot of the residuals, if they are normally distributed, then the model is well fitted. Or in effect, extracted variables are valid.

  Figure 7.21 reveals that there is no pattern to the residuals plotted against the fitted value, and Figure 7.23 confirms that the residuals are normally distributed. Therefore, we conclude that the extracted three variables are valid based on testing residuals.

7.4 Experimental Results

This section lists the comparison results of the models built by four prediction techniques (section 6.4) based on 5 data sets (section 6.3) in Tables 7.11 to 7.14. The specific description of 5 data sets in this case study is presented in Table 7.10. The results analysis and interpretation are detailed in the next section.

Var	Effect	Adj R^2	Significance of Added Var	Statistic Methods	Sig. Code
1-var Model					
LnSize	+	0.4574	$p < 2.2e-16$	Regression	***
2-var Model With *LnSize*					
LnDuration	+	0.5336	$Pr > t = 0.000$	Regression	***
LangType	+	0.5108	$F = 40.285, Pr(> F) = 6.626e-10$	ANOVA	***
3-var Model With *LnSize, LnDuration*					
RLevel	+	0.536	$Pr(> F) =< 2.2e-16$	ANOVA	.
CountApp	+	0.5343	$F = 1.5378, Pr(> F) = 0.2158$	ANOVA	
DevType	+	0.5348	$F = 1.8745, Pr(> F) = 0.1718$	ANOVA	
OrgType	+	0.5395	$F = 5.6177, Pr(> F) = 0.01831$	ANOVA	*
BusiType	−	0.5323	$F = 0.0153, Pr(> F) = 0.9015$	ANOVA	
AppType	−	0.5324	$F = 0.0505, Pr(> F) = 0.8224$	ANOVA	
Architecture	+	0.5404	$F = 6.2856, Pr(> F) = 0.01261$	ANOVA	*
DevPlatform	−	0.5436	$F = 8.8153, Pr(> F) = 0.003188$	ANOVA	**
LangType	+	0.5792	$F = 39.864, Pr(> F) = 8.064e-10$	ANOVA	***
PriLang	+	0.5328	$F = 0.3512, Pr(> F) = 0.5538$	ANOVA	
MDBS	+	0.5526	$F = 16.232, Pr(> F) = 6.845e-05$	ANOVA	***
UMethod	+	0.5347	$F = 1.8621, Pr(> F) = 0.1732$	ANOVA	
RMethod	−	0.5324	$F = 0.0488, Pr(> F) = 0.8253$	ANOVA	
4-var Model With *LnSize, LnDuration,LangType*					
RLevel	+	0.5781	$F = 0.1047, Pr(> F) = 0.7464$	ANOVA	
CountApp	+	0.5809	$F = 2.4731, Pr(> F) = 0.1167$	ANOVA	
DevType	+	0.58	$F = 1.7391, Pr(> F) = 0.1881$	ANOVA	
OrgType	+	0.5878	$F = 8.4837, Pr(> F) = 0.003809$	ANOVA	**
BusiType	−	0.5785	$F = 0.4267, Pr(> F) = 0.5141$	ANOVA	
AppType	−	0.5784	$F = 0.3179, Pr(> F) = 0.5732$	ANOVA	
Architecture	+	0.6018	$F = 21.348, Pr(> F) = 5.356e-06$	ANOVA	***
DevPlatform	−	0.5786	$F = 0.5182, Pr(> F) = 0.4721$	ANOVA	
PriLang	+	0.5906	$F = 10.956, Pr(> F) = 0.001028$	ANOVA	**
MDBS	+	0.5905	$F = 10.856, Pr(> F) = 0.001084$	ANOVA	**
UMethod	+	0.5783	$F = 0.2221, Pr(> F) = 0.6378$	ANOVA	
RMethod	−	0.5809	$F = 2.4684, Pr(> F) = 0.1170$	ANOVA	
5-var Model With *LnSize, LnDuration,LangType,Architecture*					
RLevel	+	0.6011	$F = 0.3249, Pr(> F) = 0.569$	ANOVA	
CountApp	+	0.6058	$F = 4.6263, Pr(> F) = 0.03216$	ANOVA	*
DevType	+	0.6036	$F = 2.6226, Pr(> F) = 0.1062$	ANOVA	
OrgType	+	0.6054	$F = 4.2725, Pr(> F) = 0.03946$	ANOVA	*
BusiType	−	0.6022	$F = 1.3475, Pr(> F) = 0.2465$	ANOVA	
AppType	−	0.6007	$F = 0.0276, Pr(> F) = 0.8682$	ANOVA	
DevPlatform	−	0.6045	$F = 3.4691, Pr(> F) = 0.06335$	ANOVA	.
PriLang	+	0.6122	$F = 0.568, Pr(> F) = 0.001260$	ANOVA	**
MDBS	+	0.6058	$F = 4.5739, Pr(> F) = 0.03314$	ANOVA	*
UMethod	+	0.602	$F = 1.1832, Pr(> F) = 0.2774$	ANOVA	
RMethod	−	0.6011	$F = 0.3569, Pr(> F) = 0.5506$	ANOVA	
6-var model: none significant, no further improvement					

Data:ISBSGR9 362 Projects; Tool:R 2.1.0
Significant codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
F critical values ($p = 0.05$) only record adj $R^2 > |0.5|$

Table 7.9: Case Study A: Building the best 1 to 5-Variable Models in PSS

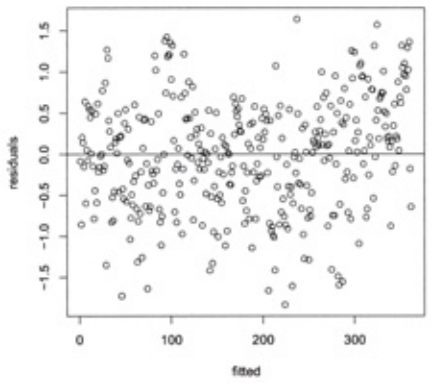

Figure 7.21: Plot Residuals of the Final Model

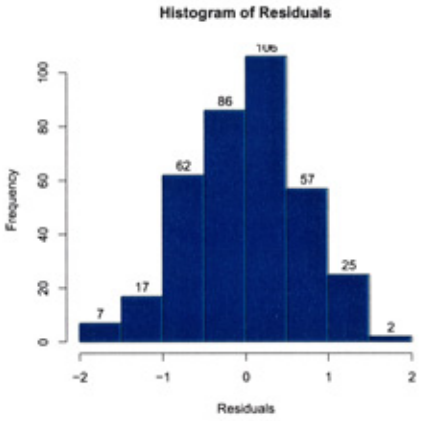

Figure 7.22: Histogram for Residuals of the Final Model

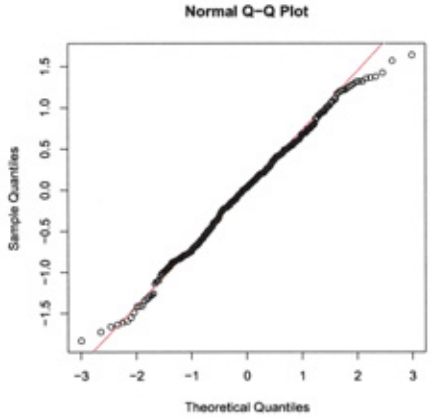

Figure 7.23: Q-Q Plot Residuals of the Final Model

Data Set	Array Size	Predictor Var. Nr.	Predicted Var.	Contain Projects	Contain Variables
PVS	(560 × 16)	15	Effort	560	Effort, Size, Duration, CountApp, Architecture, DevPlatform, LangType, PriLang, MDBS, UMethod, RMethod, RLevel.
PVS_{Size}	(560 × 2)	1	Effort	560	Effort, Size.
PSS	(362 × 16)	15	Effort	362	Effort, Size, Duration, CountApp, Architecture, DevPlatform, LangType, PriLang, MDBS, UMethod, RMethod, RLevel.
EPSS	(362 × 4)	3	Effort	362	Effort, Size, Duration, LangType.
$EPSS_{Size}$	(362 × 2)	1	Effort	362	Effort, Size.

Table 7.10: Case Study A: Description of Data Sets for Comparison

Data Set	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	25	9.46	4.82	0.94	0.49	1.35	0.81	0.50	8785.45
PVS_{Size}	21.43	8.39	3.21	1.19	0.57	1.71	0.98	0.59	8088.32
PSS	29.01	11.05	6.08	0.64	0.42	0.85	0.58	0.43	2362.16
EPSS	30.66	11.88	6.91	0.66	0.44	0.91	0.63	0.47	2404.13
$EPSS_{Size}$	24.03	8.01	4.42	0.82	0.5	1.11	0.72	0.55	2512.78

Table 7.11: Case Study A: Comparison Based on OLS

Data Set	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	23.21	8.04	5.18	0.97	0.48	1.36	0.81	0.50	9128.58
PVS_{Size}	21.61	9.46	3.57	1.21	0.57	1.71	0.96	0.59	8098.57
PSS	28.18	10.77	6.35	0.66	0.42	0.85	0.57	0.44	2392.19
EPSS	30.11	11.33	6.63	0.66	0.43	0.91	0.62	0.46	2410.62
$EPSS_{Size}$	23.76	8.01	4.42	0.84	0.5	1.12	0.7	0.55	2511.9

Table 7.12: Case Study A: Comparison Based on RR

Data Set	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	23.75	9.11	4.46	1.07	0.53	1.64	1.03	0.54	8533.96
PVS_{Size}	20	8.39	2.86	1.27	0.59	1.86	1.07	0.62	8545.74
PSS	26.8	12.98	7.73	0.69	0.42	0.96	0.64	0.47	2410.6
EPSS	23.48	10.5	6.63	0.78	0.48	1.06	0.68	0.51	2423.38
$EPSS_{Size}$	22.1	7.73	2.49	0.85	0.56	1.18	0.77	0.57	2477.5

Table 7.13: Case Study A: Comparison Based on CART)

Data Set	K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	2	13.93	6.96	3.39	5.27	0.83	8.19	3.53	0.80	12879.818
PVS_{Size}	2	13.75	5.36	2.32	5.27	0.85	7.81	3.16	0.82	13827.882
PSS	1	15.75	3.87	1.10	2.36	0.78	4.08	2.30	0.81	4127.00
EPSS	1	15.47	5.52	2.76	2.48	0.81	4.21	2.32	0.81	4345.35
$EPSS_{Size}$	1	17.40	5.80	3.31	2.41	0.79	4.06	2.23	0.79	4173.78

Table 7.14: Case Study A: Comparison Based on KNN

7.5 Analysis and Discussions

Results for each of the five data sets (PVS, PVS_{Size}, PSS, EPSS, and $EPSS_{Size}$) are detailed in Tables 8.10, 7.12, 7.13 and 7.14, for the Least-Square Regression (OLS), Robust Regression (RR), Classification and Regression Trees (CART) and K-Nearest Neighbour (KNN) prediction approaches respectively.

As discussed in section 6.5, the most commonly used criteria (i.e., MRE, MMRE, MdMRE and Pred(l)) and the most recent(i.e., MER, MMER, MdMER, and BMMRE) evaluation criteria are applied for the comparison. We also calculate the standard deviation for information only. The comparisons made in this section aim to:

1. Test whether the proposed preliminary data analysis framework provides sufficient predictor variables (Hypotheses 1a and 1b, section 6.2),

2. Test whether the proposed framework selected variables are significantly improved prediction accuracy (Hypotheses 2a and 2b, section 6.2) and

3. Test whether the effort prediction accuracy is improved for a heterogenous data set from the application of the data analysis framework (Hypothesis 3a section 6.2, section 6.2).

7.5.1 Graphical Results

To graphically illustrate and further compare results yielded by the above five data sets, we plot two bar-charts for each selected prediction technique in Figures 7.24 to 7.31. For each technique, we plot a bar-chart to present Pred(l) values and a bar-chart for remain criteria except the standard deviation, which has different magnitude.

Table 7.15 summarizes the best performed data sets for each technique under each evaluation criterion. For example, according to the average value of Pred(l), where $l = 25\%, 10\%, 5\%$, the models based on the EPSS data set perform the best for OLS and RR methods with the Averg.Pred(l) value 16.48 and 16.02 respectively.

For the Ordinary Least-Square Regression (OLS), Table 7.11 and Figure 7.24 reveal that the prediction model constructed from the EPSS data set outperforms models based on the other four data sets at all three levels of Pred as Pred(25%)=30.66,

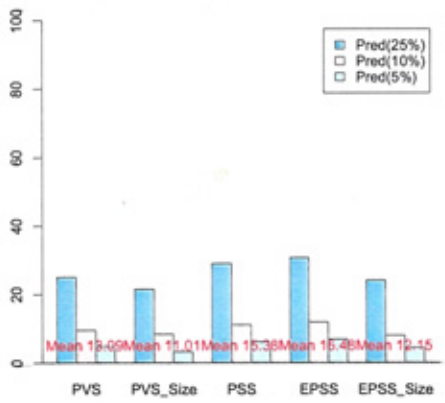

Figure 7.24: Comparison of Pred(l) Based on OLS

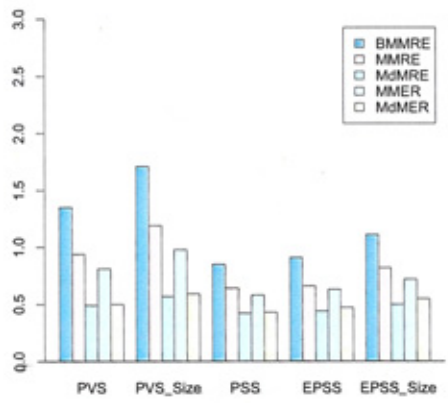

Figure 7.25: Comparison of MMRE etc. Based on OLS

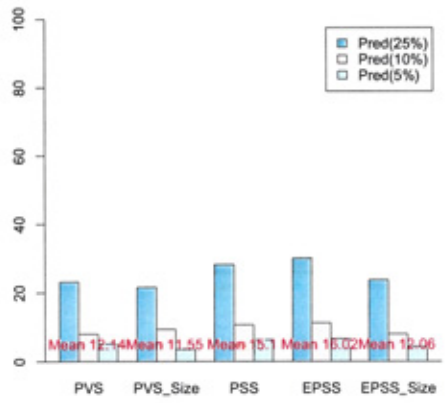

Figure 7.26: Comparison of Pred(l) Based on RR

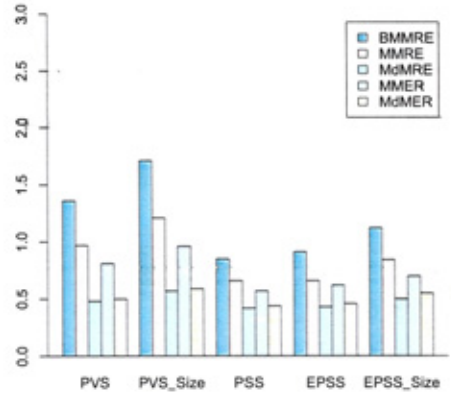

Figure 7.27: Comparison of MMRE etc. Based on RR

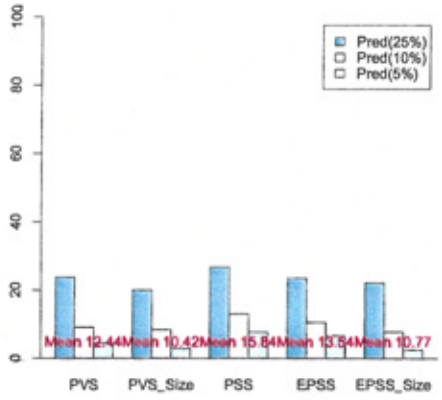

Figure 7.28: Comparison of Pred(l) Based on CART

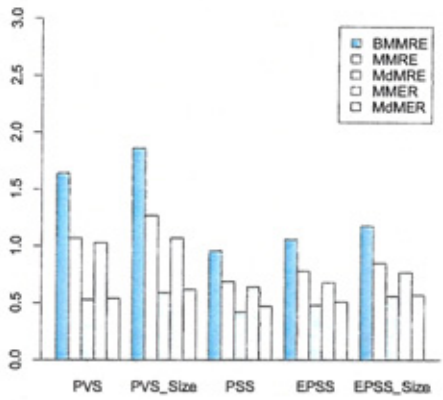

Figure 7.29: Comparison of MMRE etc. Based on CART

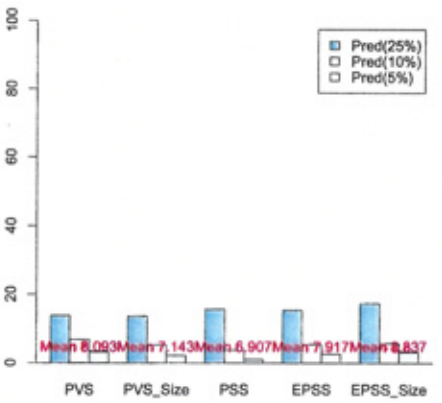

Figure 7.30: Comparison of Pred(l) Based on KNN

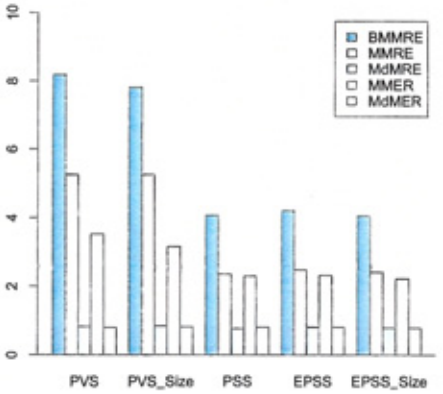

Figure 7.31: Comparison of MMRE etc. Based on KNN

Data Set	Max. Pred(l)	Averg Pred(l)	BMMRE	MMRE	MdMRE	MMER	MdMER	St. Dv.
PVS								
PVS_{Size}								
PSS	C. (26.8)	C. (15.84)	O. (0.85) R. (0.85) C. (0.96)	O. (0.64) R. (0.66) C. (0.69) K. (2.36)	O. (0.42) R. (0.42) C. (0.42) K. (0.78)	O. (0.58) R. (0.57) C. (0.64)	O. (0.43) R. (0.44) C. (0.47)	O. (2362.16) R. (2392.19) C. (2410.6) K. (4127.00)
EPSS	O. (30.66) R. (30.11)	O. (16.48) R. (16.02)		R. (0.66)				
$EPSS_{Size}$	K. (17.40)	K. (8.837)	K.(4.06)			K. (2.23)	K. (0.79)	
O. stands for OLS, R.stands for RR, C. stands for CART, and K. stands for KNN								

Table 7.15: Case Stud A: Best Performed Data Sets Based on Four Prediction Techniques

Pred(10%)=11.88, and Pred(5%)=6.91. Figure 7.25 presents a trend of MRE related (MMRE and MdMRE), MER related (MER and MdMER), and Balanced MRE (BMMRE) metrics. The prediction model based on the EPSS data set has the minimum values of MRE related and MER related criteria. The model based on EPSS has a very similar BMMRE with the model based on the PSS data set. Accordingly, the MMRE and Pred(l) measures suggest that the prediction model based on the EPSS data set performs the best.

For the Robust Regression (RR), Table 7.12, Figures 7.26 and 7.27 present a similar trend with OLS results. The model based on the EPSS data set has the highest average values of Pred(l) ($l = 25\%, 10\%, 5\%$) and the second lowest values of MRE and MER related values. Accordingly, the MMRE and Pred(l) measures imply that the model based on the EPSS data set performs the best.

For the CART, Table 7.13, Figures 7.28 and 7.29 present that the model based on the PSS data set outperforms models based on the other four data sets.

For the KNN, the prediction accuracy performance is inconsistent across metrics. For example, considering the figures in the Table 7.14, Figures 7.30 and 7.31, according to MMRE, MdMRE values and Pred(l), the model based on the data set PSS performs the best, but according to BMMRE, MMER and MdMER, the model based on the $EPSS_{Size}$ data set performs the best.

Based on the above, the following observations can be made in this case study using ISBSG R9 heterogenous data set:

- The performance of the models is inconsistent across metrics except CART. For example, consider the Pred(l) and BMMRE vales of the models built by OLS and

RR, the former of the both techniques support the contention that the model based on the EPSS data set performs the best, while the later of both techniques imply that the model based on the PSS data set performs the best. A similar phenomenon has been observed and discussed by Liu *et al.* [190].

- Apart from KNN and CART, there is a clear tendency that the project effort prediction model constructed from the framework processed data sets (EPSS) performs better than the models based on the raw data set (PVS) and the data sets containing the most significant predictor variable only (PVS_{Size} and $EPSS_{Size}$). For CART, although the model based on the PSS data set outperforms the others, results of the models based on the EPSS and PSS data sets are very close. A further analysis is needed. As described in section 6.6, significance testing of the above comparisons, in particular results between models based on the PSS and EPSS data sets, will be conducted in the following two subsections.

- The model built by KNN performs worse than the models built by other techniques across five data sets. For examples, the model built by KNN has the lowest Averg.Pred(l) value and the highest BMMRE value. The poor performance of the KNN method can be attributed to the equal weights placed on variables in the similarity measure. Greater weighting assigned to more significant predictor variables [279] would change the basis for the similarity measures and analog retrieval.

7.5.2 Comparison Between Data Sets

Although we have made some observations based on figures and graphics in the preceding subsections, to rigorously compare the difference between prediction accuracy between data sets or techniques we need to perform statistical significance testing as designed in section 6.6. Table 7.16 presents the results of Test A, section6.6, that compares difference of the prediction accuracy between the model based on EPSS data set and models based on the other four data sets for each prediction technique. The analysis supports the following contention:

- There is a significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PVS data set across all four techniques, at 0.02 level of significance. That is,

the model based on the EPSS data set outperforms the model based on the PVS data set. Therefore, Hypothesis 1a is true for both parametric methods and non-parametric methods in this case study.

- For all four prediction techniques, there is a significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PVS_{Size} data set, at 0.001 level of significance. That is, the model based on the EPSS data set outperforms the model based on the PVS_{Size} data set. Therefore, Hypothesis 1b is true for this case study.

- For all four prediction techniques, there is no significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PSS data set. However, the EPSS data set consists of much fewer predictor variables (3 variables) than the PSS data set (15 variables), therefore the model based on the EPSS data set is more effective. In addition, for OLS and RR techniques, the model based on the EPSS data set is in favor of higher Max.Pred (l) and Averg. Pred(l) values presented in Table 7.15. Therefore, for all four techniques, the model based on the EPSS data set performs no worse than the model based on the PSS data set. In other words, Hypothesis 2a is true for this case study.

- Except the KNN, there is no significant difference between prediction accuracy based on the MRE values of the model based on the EPSS data set and the model based on the $EPSS_{Size}$ data set. Again, the EPSS data set is in favour of high Averg. Pred(l) values presented in Table 7.15. For OLS, RR and CART, the model based on the EPSS data set perform better than the model based on the $EPSS_{Size}$ data set. For the KNN, the model based on the $EPSS_{Size}$ data set outperforms the model based on the EPSS data set at 0.03 level of significance. Therefore, Hypothesis 2b is not true for the KNN in this case study.

- Accordingly, in view of the above contentions, Hypothesis 3a is true for models built by three prediction techniques: OLS, RR and CART.

7.5.3 Comparison Between Prediction Techniques

According to the Mann-Whitney U-test results presented in Table 7.17, we observe that, for the model based on the EPSS data set there is no significant difference of prediction

Technique	Comparison vs. EPSS	Testing Method	Testing Result	P-value
OLS	PVS	t-test	t = -2.4605	0.01411
	PVS_{Size}	t-test	t = -3.8605	0.0001242
	PSS	Wilcoxon-test	V = 35519	0.09035
	$EPSS_{Size}$	Wilcoxon-test	V = 21936	1
RR	PVS	t-test	t = -2.4387	0.01499
	PVS_{Size}	t-test	t = -3.901	0.0001056
	PSS	Wilcoxon-test	V = 34782	0.1663
	$EPSS_{Size}$	Wilcoxon-test	V = 21829	1
CART	PVS	t-test	t = -2.7515	0.00605
	PVS_{Size}	t-test	t = -3.5565	0.0003981
	PSS	Wilcoxon-test	V = 12360	0.01705
	$EPSS_{Size}$	Wilcoxon-test	V = 21987	0.9999
KNN	PVS	t-test	t = -2.3284	0.02011
	PVS_{Size}	t-test	t = -3.218	0.001341
	PSS	Wilcoxon-test	V = 29065	0.3011
	$EPSS_{Size}$	Wilcoxon-test	V = 33088	0.0254

Table 7.16: Case Study A: Test A Results at 95% Confidence Interval

Comparison Btw Techniques	Testing Method	Testing Result	P-value
OLS vs. RR	Mann-Whitney U-test	W = 65426.5	0.5136
OLS vs. CART	Mann-Whitney U-test	W = 61544.5	0.9213
OLS vs. KNN	Mann-Whitney U-test	W = 38325.5	1
RR vs. CART	Mann-Whitney U-test	W = 61625	0.917
RR vs. KNN	Mann-Whitney U-test	W = 38387.5	1
CART vs. KNN	Mann-Whitney U-test	W = 40519.5	1

Table 7.17: Case Study A: Test B Results at 95% Confidence Interval

accuracy (based on MRE) between various techniques. Therefore, the prediction results based on the EPSS data set are consistent across all four techniques.

7.6 Summary

Using the ISBSG R9 data set, this chapter described the application of the proposed preliminary framework used within four commonly used prediction techniques. For software cost models used each prediction technique, the framework processed data set (EPSS) is compared to other data sets, including the raw data set (PVS), raw data set with the most significant predictor variable (PVS_{Size}), the raw data set without outliers (PSS) and the framework processed data set with the most significant predictor variable ($EPSS_{Size}$).

These comparisons addressed the first two research hypotheses detailed in section 6.2. Significance testing (Test A as detailed in section6.6) was performed for these comparisons. The general question behind the first hypothesis is: Which data

set provides the most accurate prediction: PVS_{Size}, PVS, or EPSS? This evaluates the validity of the framework processed data set EPSS. The general question behind the second hypothesis is: Which data set provides the most accurate prediction: $EPSS_{Size}$, PSS, or EPSS? This evaluates the significance of The framework processed data set EPSS. The significance testing (Test B as detailed in section6.6) also addresses the question: Whether the proposed framework is applicable to the selected commonly used prediction techniques. This evaluates the applicability of the proposed framework.

Overall, the analysis shows as the following. First, for all four prediction techniques Hypotheses 1a and 1b are true in this case study; Second, for three out of four techniques (except KNN), Hypotheses 2a and 2b are true; Third, except the KNN, there does appear to be consistent variation in performance between data sets, the models based on the framework processed data set performs favorably throughout within this case study. In other words, the proposed framework is applicable for OLS, RR and CART, in the multi-organizational data set. Therefore, the Hypothesis 3a is true for these three prediction techniques; Fourth, parametric modeling methods (OLS and RR) produced more accurate results than non-parametric modeling methods (CART, and Analogy-based, i.e.,KNN), within multi-organizational contexts.

Additional results are worth to consider:

The validity of evaluation criteria: except KNN, the BMMRE, MRE based and MER based criteria do show a consistent tendency for each prediction model, however, for KNN, MER based and BMRE criteria show negative correlations with the K value. As illustrated in Tables B.15 to B.19 in Appendix B, with the increasing of K value, Pred(l) values show approximate bell-shape curves and MRE based criteria (MMRE and MdMRE) values show approximate reversed bell-shape curves. Therefore, MER based and BMRE criteria are doubtful within this case study. Further investigation could be made to evaluate the validity of these newly proposed criteria.

KNN does not perform as effectively as the other prediction techniques. This observation is consistent with other research findings, as observed by Briand *et al.* [42, 44], Walkerden and Jeffery [306], etc. The poor performance of the KNN method can be attributed to the equal weights placed on variables in the similarity measure. A further plausible reason for the inaccurate prediction of analogy based techniques (e.g.,KNN) has been argued by Walkerden and Jeffery [306]. They contend that this occurs be-

cause software tools were used to calculate analogues and it is the case that experts are better than tools at selecting analogues.

Chapter 8

Case Study B: Bank63

8.1 Introduction

This second empirical study uses raw data from one bank, which is provided by Dr. Katrina D. Maxwell. This data set is presented in the book "Applied Statistics for Software Managers" [203]. The database contains 25 variables for 63 completed software development applications. We refer to this data set as Bank63 data set.

This chapter outlines the data set and preliminary data analysis using the framework proposed in chapter 5. It explains how the comparative evaluation described in chapter 6 is applied. It presents the results obtained from the application of each extracted subset and the raw data set based on a selection of estimation methods. Finally there is a discussion of the result. A free statistical software package R 2.1.0, which was developed at University of Minnesota [62], was used.

8.2 Data Set Description

After removing the projects and attributes containing missing data, we obtained 61 project data with 25 project attributes as PVS. These attributes include 1 predicted variable - project *effort* and 24 predictor variables - *size, duration, app, har, ifc, source, nlan, telonuse*, and *t01* to *t15*. Within 24 predictor variables, there are 18 numerical variables (*size, duration, nlan*, and 15 different productivity factors: *t01* to *t15*) and 5 categorical variables. These five categorical variables are *app, har, ifc, source*, and *telonuse*. The Data field description see Tables C.1 to C.5 in Appendix C. The detailed data set profile is presented as the outcomes of the step 1 to step 3 of the preliminary analysis framework in the following section.

8.3 Extracting Optimal Subsets

This section details the application of the preliminary data analysis framework to extract optimal subsets of Bank63 data.

- Step 0: **Data Preparation.**

Projects (ID = 13 and ID = 14) containing missing values have been removed. Therefore, PVS starts as an array (61 × 25).

Categories in text are transformed to nominal numbers. The allocated numbers and corresponding categories are detailed in step 3. Table 8.1 illustrates a fragment of the PVS after category transformation.

Numbers	Attributes	Sample 1	Sample 2	Sample i	Sample 61
1	size	562	647	···	3634
2	effort	1062	7871	···	39479
3	duration	14	16	···	33
4	app	408	406	···	401
5	har	1003	1002	···	1005
6	dba	1602	1602	···	1602
7	ifc	2001	2002	···	2002
8	source	7004	7001	···	7001
9	nlan	1	3	···	3
10	telonuse	0	0	···	0
11	t01	5	4	···	2
12	t02	3	3	···	4
13	t03	3	5	···	3
14	t04	2	3	···	3
15	t05	3	3	···	3
16	t06	4	3	···	3
17	t07	4	4	···	4
18	t08	4	5	···	3
19	t09	3	4	···	5
20	t10	3	5	···	5
21	t11	2	4	···	5
22	t12	4	4	···	4
23	t13	3	4	···	4
24	t14	5	4	···	5
25	t15	3	5	···	4

Table 8.1: Case Study B: A Fragment of PVS with Transformed Categorical Variables

- Step 1: **Remove Outliers by Measuring Numerical Variables.**
  Table 8.2 summarizes variables with numerical values. This table shows the the variable name (Var. Name), the Observations (Obs.), the minimum, the maximum, the average (Mean), and the standard deviation (Std. Deviation). There are no missing values for each variable. We observed that some of the cost factors (*t01-t15*) do not use the full range, possibly because these values would never be allowed in banking applications. The summary statistics for *size, effort* and *duration* are of special interest. The *size* of projects in this database ranges from 59 function points to 3,634 function points; the average project *size* is 691 function points. *Effort* ranges from 796 man-hours to 63,694 man-hours, with an average of 8352 man-hours. *Duration* ranges from 4 months to 54 months; the average duration is 17 months. Because the sample size of Bank63 is 61(< 80),

we use the threshold value of standard scores of 2.5. Regarding the characteristic of the software project data (i.e. project *effort, size* and *duration* are always positive), we also measure the 5% trimmed mean and remove the lowest and highest 5% of the observations.

Var. Name	Obs.	Minimum	Maximum	Mean	Std. Deviation
duration	61	4	54	17	11
effort	61	796	63,694	8,351	10,538
nlan	61	1	4	3	1
size	61	59	3,634	691	782
t01	61	1	5	3.11	1.018
t02	61	1	5	3.07	0.704
t03	61	2	5	3.05	0.884
t04	61	2	5	3.15	0.703
t05	61	1	5	3.07	0.704
t06	61	1	4	2.93	0.704
t07	61	2	5	3.31	0.847
t08	61	2	5	3.80	0.963
t09	61	2	5	4.05	0.762
t10	61	2	5	3.62	0.897
t11	61	2	5	3.36	0.984
t12	61	2	5	3.84	0.688
t13	61	1	5	3.10	0.926
t14	61	1	5	3.25	1.011
t15	61	2	5	3.39	0.665
Valid Obs.	61				

Table 8.2: Case Study B: Summary of the Numerical Variables in PVS

- Step 2: **Data Transformation.**

  In this step, we plot the histograms for each numerical variable in PVS to see if the variables are normally distributed. Three ratio scaled numerical variables include *effort, size* and *duration.* Figures C.1, C.3 and C.5 in Appendix C demonstrate that none of these numerical variables are normally distributed. The PVS contains a few projects with a very high effort, or a very big size or a very long duration. It also contains many low effort, small size and short duration projects. This is typical in a software development project data set. To approximate a normal distribution we must transform these variables. Sixteen histogram diagrams were plotted for each numerical (in fact, ordinal) variable including *nlan* and *t01-t15.* FiguresC.7 to C.22 in Appendix C, demonstrate that these ordinal variables are approximately normally distributed in the PVS. From the above diagrams, we learned that in this bank: most applications had high customer participation (t01); most of the applications (97%) have adequate tool resources and equipment (t02); no application had big problems with key personnel availability (t03); no project needed standard development (t04); no application used

an integrated CASE environment (t06); the logical complexity of the software was on the high side (t07); requirements volatility was quite high (t08); no application had zero quality requirements, most of the applications had high to very high quality requirements (t09); some level of attention and planning was needed for the efficiency requirements of all applications (t10); all applications required some kind of user training (t11); for most projects, the analysis skills of the project staff were high. most people had experience with specification (t12); 69% staff members have good or very good staff application knowledge - knowledge of application domain (e.g.supplier and customer); tool experience was on the high side (t14); all staff members have high team skills (t15). The above observations are consistent with Maxwell's [203] description.

We create three new numerical variables for each ratio variable by taking each original numerical's log values, and name new variables as LnEffort, LnSize and LnDuration. We recreate histograms of each transformed variable in Figures C.2, C.4 and C.6 (in Appendix C). This transformation method makes large values smaller and brings the data closer together. These transformed numerical variables will be used in the following steps.

- Step 3: **Remove Outliers by Measuring Categorical Variables.**
  First, we tabulate each categorical variable that has been transformed in step 1 to check how many projects are in each category. Tables C.6 to C.12 and Figures C.23 to C.29 in Appendix C demonstrate the number and the percentage of observations in each category.
  Second, we remove projects that contain any categories appear less than 3 times. There are 6 outliers have been removed so far. the remain of the PVS is called PSS, that is a row reduced subset of PVS.
  Third, we repeat steps 1 to 3 for PVS until there are no more outliers. Table 8.3, Figures 8.1 to 8.6 described three transformed numerical variables in PSS. It is important to understand the range of numerical variables in a data set. Maxwell [203] suggests not using data set to predict new projects with numerical variables out of the above identified range, e.g., not to apply prediction models based on Bank63 data to projects bigger than 2,482 function points, or less than 65 function points. There are not categorical outliers in PSS, we can either boxplot or tabulate categorical variables to confirm it.

- Step 4: **Test Collinearity Between Numerical Predictor Variables.**

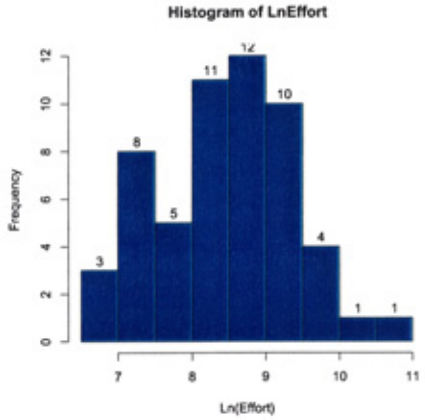

Figure 8.1: Hist LnEffort in PSS

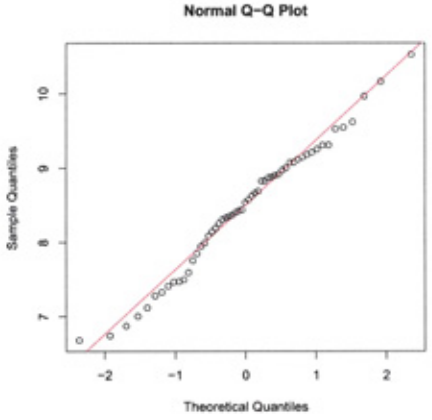

Figure 8.2: Q-Q Plot LnEffort in PSS

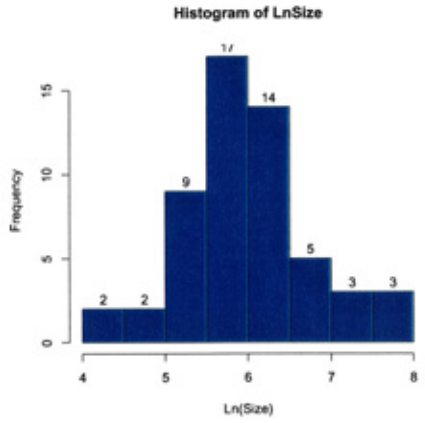

Figure 8.3: Hist LnSize in PSS

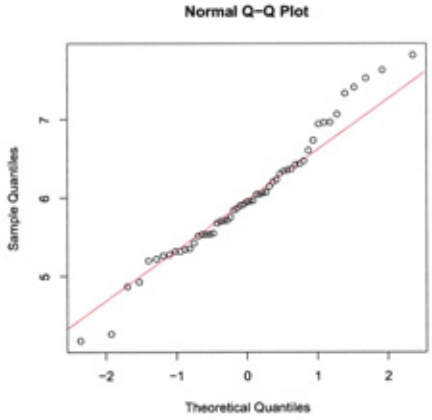

Figure 8.4: Q-Q Plot LnSize in PSS

We produce a 2-dimensional diagram for each pair of transformed numerical variables, and observed there are correlation between LnEffort, LnDuration and LnSize as demonstrate in Figure 8.7. We need to further examine these variables, in particular to identify whether there is a strong relationship between LnSize and LnDuration. Table 8.4 indicates LnSize and LnDuration are not strongly correlated ($rho = 0.51 < 0.75$), and therefore they can be included in a same model to predict project Effort. The results presented in this table also confirmed the preceding observation that both LnSize and LnDuration are correlated to LnEffort, where the correlation coefficients are 0.70 and 0.73 respectively.

- Step 5: **Stepwise Regression Analysis.**

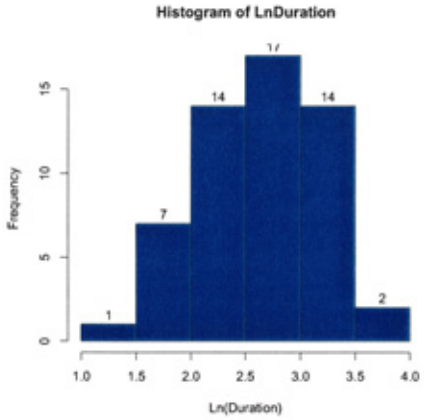

Figure 8.5: Hist LnDuration in PSS

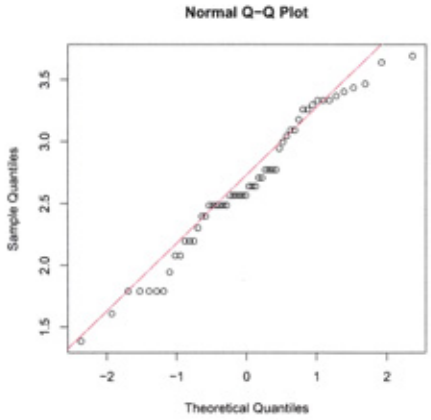

Figure 8.6: Q-Q Plot LnDuration in PSS

Var. Name	Obs.	Minimum	Maximum	Mean	Std. Deviation
duration	55	4	40	16.33	9
effort	55	796	37,286	6,794.96	6,493
nlan	55	1	4	2.51	1
size	55	65	2,482	552.44	510
t01	55	1	5	3.04	0.999
t02	55	1	5	3.04	0.719
t03	55	2	5	3.07	0.920
t04	55	2	5	3.18	0.696
t05	55	2	5	3.11	0.685
t06	55	1	4	2.93	0.716
t07	55	2	5	3.27	0.849
t08	55	2	5	3.80	0.970
t09	55	2	5	4.07	0.766
t10	55	2	5	3.64	0.890
t11	55	2	5	3.35	0.947
t12	55	2	5	3.87	0.668
t13	55	1	5	3.09	0.948
t14	55	1	5	3.20	0.989
t15	55	2	5	3.42	0.658
Valid Obs.	55				

Table 8.3: Case Study B: Summary of the Numerical Variables in PSS

Variables	Correlation Coefficient (rho)
LnEffort and LnDuration	0.73
LnEffort and LnSize	0.70
t09 and t10	0.61
t11 and nlan	0.60
t07 and LnSize	0.54
t12 and t15	0.53
LnSize and LnDuration	0.51
t07 and LnEffort	0.51

Table 8.4: Case Study B: Correlation Analysis of Numerical Variables in PSS

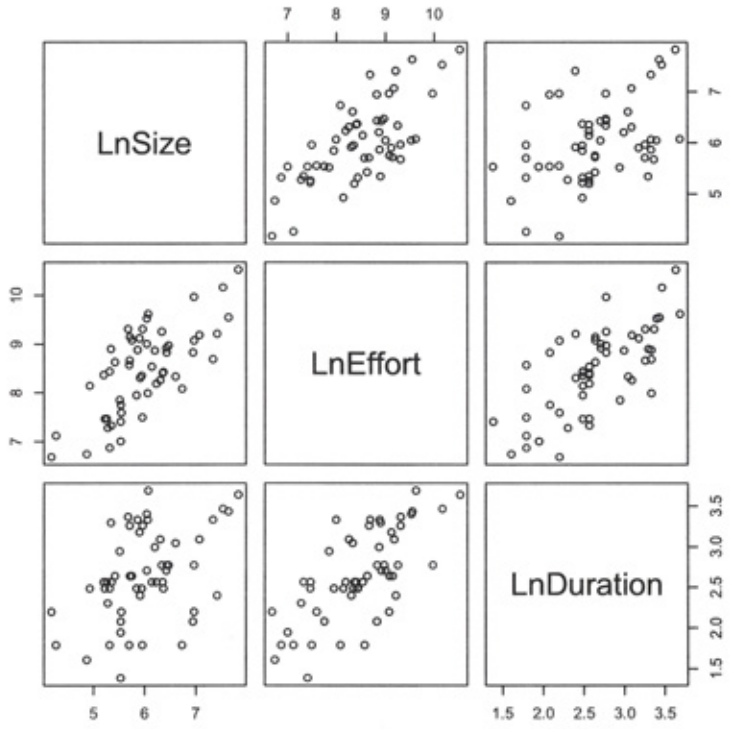

Figure 8.7: Pair Plot Ratio Scaled Vars. in PSS

In this step, we test the correlation between the numerical predictor variables and the predicted variable by running a backward stepwise regression analysis. This procedure allows us to determine the relative importance of each numerical predictor variable's relationship to the predicted variable. Based on the results presented in Table 8.5, there are 11 ordinal variables removed. The remain numerical variables are project LnSize, LnDuration, t09 and t14. The most significant numerical variable is project *size*, the second significant numerical variable is project *duration*, etc. The best model with numerical variables alone explained 78% of the variation in *effort*. We will explore whether any of the categorical variables explain more of the variation in effort in the following step.

- Step 6: **Stepwise ANOVA Analysis to build a k-variable model.**
  The following steps are to be followed.

  - Identify the best one-variable model. There are 24 predictor variables, thus we need to build 24 one-variable models. The best one-variable model will be

Model	R	R^2	Adjusted R^2	
1	0.717[a]	0.514	0.505	
2	0.840[b]	0.706	0.695	
3	0.882[c]	0.777	0.764	
4	0.893[d]	0.798	0.781	
	Est. Coef.	Std. Er.	t value	$Pr(>\|t\|)$
(Intercept)	2.838	0.601	4.726	0.000
LnSize	0.483	0.081	5.941	0.000
LnDuration	0.737	0.114	6.437	0.000
t09	0.296	0.075	3.965	0.000
t14	-0.134	0.060	-2.240	0.030

a Predictors: (Constant), LnSize
b Predictors: (Constant), LnSize, LnDuration
c Predictors: (Constant), LnSize, LnDuration, t09
d Predictors: (Constant), LnSize, LnDuration, t09, t14
Dependent Variable: LnEffort

Table 8.5: Case Study B: Stepwise Regression Summary of PSS

determined by the adjusted correlation coefficient square (R^2) value. This value is calculated for each predictor variable with the response variable (project *effort*). The regression adjusted R^2 value will be calculated when the predictor variable is type ratio or interval (*size* and *duration* in PSS) and the ANOVA adjusted R^2 value will be calculated when the explanatory variable is ordinal or nominal (the remain 22 variables in PSS). The predictor variable that has the greatest adjusted R^2 value explains most of the variation in the predicted variable and is kept in the model, in this case it is (transformed) project *size* with adjusted R^2 value equals to 0.51.

- Identify the best two-variable model by the same procedure. Determine which variable, x_i, in addition to LnSize, explains the most variation in the predicted variable. This means that 23 two-variable models must be built. The most significant variable - LnDuration is selected as the second variable to be added into the two-variable model with LnSize. The adjusted R^2 value of adding LnDuration equals to 0.695. Then we keep both LnSize and LnDuration in the model to build the best 3-var model. This process is repeated until there is no significant variables that can be added to improve the model. Table 8.8 presents the results of this procedure.

- Finally, we obtain the best 4-variable model including LnSize, LnDuration, t09 and t14 predictor variables, which explains 78% of the variance of project *effort*, see equation 8.1. Theses results are consistent with the results ob-

tained in step5. This implies that there are no categorical variables explaining more of the variation in *effort*.

$$\text{LnEffort} = F(\text{LnSize,LnDuration,t09,t14}) \tag{8.1}$$

- Step 7: **Test Collinearity Between Extracted Predictor Variables.**
  No categorical variables were found to be significant, so nothing needs to be checked in this step.

- Step 8: **Extract and Interpret Equations.**
  Normally, there are transformed variables or categorical variables which have different multiplier (coefficients) for different levels. Therefore we need to extract the equation of the identified 4-variable model. The equation read from the final model's output is

$$\begin{aligned} \text{LnEffort} &= 2.838 + (0.483 \times \text{LnSize}) + \\ &\quad (0.737 \times \text{LnDuration}) + \\ &\quad (0.296 \times t09) - (0.134 \times t14) \end{aligned} \tag{8.2}$$

  This can be transformed into the following non-linear equation for effort.

$$\begin{aligned} \text{Effort} &= (17.082 \times \text{Size}^{0.483}) \times \\ &\quad (\text{Duration}^{0.737} \times e^{0.296 \times t09} \times e^{-0.134 \times t14}) \end{aligned} \tag{8.3}$$

  The multipliers for the categorical variables t09 and t14 are listed in Table 8.6 and 8.7. These multipliers indicate what is e to the power $(0.296 \times t09)$ when $t09$ is 1 (the quality requirement is very low), when it is 2 (the quality requirement is low), etc.?

Category Defn.	Value	$t09$ Multiplier
Very low	1	0
Low	2	0
Nominal	3	-0.122
High	4	0.240
Very high	5	0.515

Table 8.6: Case Study B: Effort Factor Multipliers - $t09$ (quality requirements)

Category Defn.	Value	$t14$ Multiplier
Very low	1	0
Low	2	0.442
Nominal	3	0.072
High	4	0.046
Very high	5	-0.105

Table 8.7: Case Study B: Effort Factor Multipliers - $t14$ (staff tool skills)

- Step 9: **Test residuals.** In a well-fitted model, there should be no pattern to the errors (residuals) plotted against the fitted values. The term "fitted value" refers to the project effort predicted by the model, the term "residual" is used to express the difference between the actual effort and the predicted effort for each project. There should be no pattern in the residuals of our final model. Consequently we produce a histogram and a Quantile-Quantile (Q-Q) plot of the residuals. If they are normally distributed, then the model is well fitted. Or in effect, the extracted variables are valid.

  Figure 8.8 reveals that there is no pattern to the residuals plotted against the fitted value, and Figure 8.10 confirms that the residuals are normally distributed. Therefore, we conclude that the extracted four variables are valid based on residual testing.

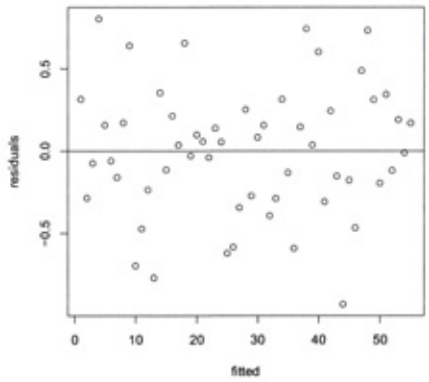

Figure 8.8: Plot Residuals of the Final Model

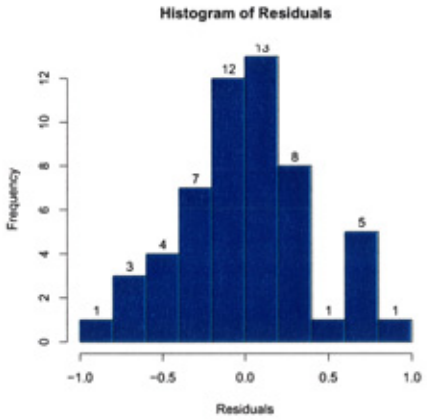

Figure 8.9: Histogram for Residuals of the Final Model

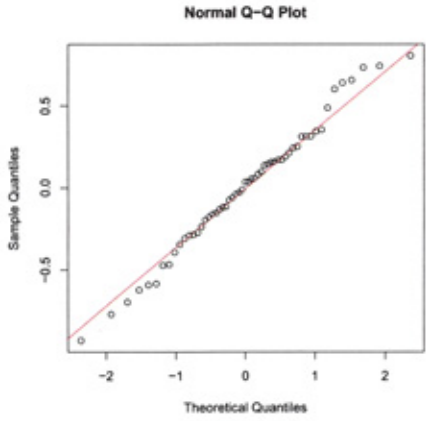

Figure 8.10: Q-Q Plot Residuals of the Final Model

Var	Effect	Adj R^2	Significance	Stat. Method
1-var Model				
LnSize	+	0.51	$7.43e-10$	Regression
2-var Model With *LnSize*				
LnDuration	+	0.695	$1.46e-14$	Regression
ifc	+	0.5532	0.01226	ANOVA
t08	+	0.5805	0.00203	ANOVA
t09	+	0.5546	0.01118	ANOVA
t10	+	0.5389	0.03108	ANOVA
t14	+	0.5653	0.005558	ANOVA
3-var Model With *LnSize, LnDuration*				
t03	+	0.7164	0.031	ANOVA
t09	+	0.7642	0.000186	ANOVA
t10	+	0.7445	0.00164	ANOVA
t11	+	0.7186	0.0247	ANOVA
t14	−	0.7183	0.0256	ANOVA
4-var Model With *LnSize, LnDuration, t09*				
t03	+	0.7771	0.051749	Regression
t14	−	0.7814	0.029592	Regression
5-var Model With *LnSize, LnDuration, t09, t14*				
app	+	0.7818	0.30203	ANOVA
har	+	0.7805	0.376904	ANOVA
ifc	−	0.7811	0.337091	ANOVA
source	−	0.7773	0.766382	ANOVA
nlan	+	0.7898	0.089939	ANOVA
telonuse	+	0.7769	0.999911	ANOVA
t01	+	0.777	0.88917	ANOVA
t02	+	0.7773	0.782756	ANOVA
t03	+	0.7872	0.130413	ANOVA
t04	−	0.782	0.290387	ANOVA
t05	−	0.777	0.865829	ANOVA
t06	−	0.7834	0.232077	ANOVA
t07	+	0.7797	0.43646	ANOVA
t08	+	0.7885	0.107447	ANOVA
t10	+	0.7876	0.1226	ANOVA
t11	+	0.784	0.21191	ANOVA
t12	+	0.7794	0.45974	ANOVA
t13	−	0.781	0.3454539	ANOVA
t15	+	0.7801	0.3996557	ANOVA

no further improvement in 5-var model
Data:Bank63 55 Projects; Tool:R 2.1.0
F critical values ($p = 0.05$) only record adj $R^2 > |0.5|$

Table 8.8: Case Study B: Building the best 1 to 4-Variable Model in PSS

8.4 Experimental Results

This section lists the comparison results of the models built by four prediction techniques (section 6.4) based on 5 data sets (section 6.3) in Tables 8.10 to 8.13. The specific description of 5 data sets in this case study is presented in Table 8.9. The results analysis and interpretation is detailed in the next section.

Data Set	Array Size	Predictor Var. Nr.	Predicted Var.	Contain Projects	Contain Variables
PVS	(61 × 25)	24	Effort	61	size, effort, duration, app, har, dba, ifc, source, nlan, telonuse, t01 - t15.
PVS_{Size}	(61 × 2)	1	Effort	61	Effort, Size.
PSS	(55 × 25)	24	Effort	55	size, effort, duration, app, har, dba, ifc, source, nlan, telonuse, t01 - t15.
EPSS	(55 × 5)	4	Effort	55	Effort, Size, Duration, t09, t14.
$EPSS_{Size}$	(55 × 2)	1	Effort	55	Effort, Size.

Table 8.9: Case Study B: Description of Data Sets for Comparison

Data Set	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	37.7	13.11	4.92	0.49	0.45	0.7	0.55	0.4	6849.86
PVS_{Size}	21.31	6.56	3.28	0.62	0.47	0.84	0.61	0.47	6616.99
PSS	25.45	16.36	7.27	0.54	0.45	0.72	0.54	0.41	5814.22
EPSS	47.27	14.55	3.64	0.38	0.27	0.47	0.37	0.29	3310.49
$EPSS_{Size}$	18.18	9.09	1.82	0.59	0.46	0.8	0.6	0.47	4660.41

Table 8.10: Case Study B: Comparison Based on OLS

Data Set	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	32.79	14.75	3.28	0.51	0.45	0.78	0.62	0.43	7024.45
PVS_{Size}	22.95	9.84	1.64	0.63	0.48	0.85	0.6	0.47	6643.56
PSS	36.36	14.55	7.27	0.44	0.34	0.6	0.47	0.35	5239.23
EPSS	47.27	16.36	5.45	0.39	0.27	0.48	0.37	0.29	3393.25
$EPSS_{Size}$	18.18	9.09	1.82	0.59	0.46	0.8	0.6	0.47	4662.69

Table 8.11: Case Study B: Comparison Based on RR

8.5 Analysis and Discussions

Results for each of the five data sets (PVS, PVS_{Size}, PSS, EPSS, and $EPSS_{Size}$) are detailed in Tables 8.10, 8.11, 8.12, and 8.13, for the Least-Square Regression (OLS),

Data Set	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	32.79	13.11	6.56	0.59	0.46	0.89	0.67	0.39	8123.48
PVS_{Size}	22.95	8.2	4.92	0.7	0.52	1	0.7	0.47	9037.33
PSS	34.55	10.91	7.27	0.52	0.38	0.71	0.53	0.39	4965.93
EPSS	36.36	14.55	9.09	0.51	0.36	0.78	0.61	0.38	5625.85
$EPSS_{Size}$	25.45	3.64	1.82	0.55	0.43	0.78	0.6	0.42	5252.46

Table 8.12: Case Study B: Comparison of Data Sets Based on CART)

Data Set	K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
PVS	5	18.18	5.45	5.45	2.3	0.67	3.21	1.44	0.74	9917.8
PVS_{Size}	4	21.82	9.09	1.82	2.42	0.68	3.24	1.35	0.72	11286.1
PSS	1	21.82	12.73	5.45	1.63	0.74	2.55	1.46	0.77	9866.37
EPSS	4	18.18	10.91	9.09	1.48	0.69	2.1	1.13	0.68	7535.45
$EPSS_{Size}$	4	25.45	14.55	10.91	1.27	0.58	1.75	0.94	0.52	6970.46

Table 8.13: Case Study B: Comparison Based on KNN

Robust Regression (RR), Classification and Regression Trees (CART) and K-Nearest Neighbour (KNN) prediction approaches respectively.

As discussed in section 6.5, the most commonly used criteria (i.e., MRE, MMRE, MdMRE and Pred(l)) and the most recent(i.e., MER, MMER, MdMER, and BMMRE) evaluation criteria are applied for the comparison. The comparisons made in this section aim to:

1. Test whether the proposed preliminary data analysis framework provides sufficient predictor variables (Hypotheses 1a and 1b, section 6.2),
2. Test whether the proposed framework selected variables significantly improve prediction accuracy (Hypotheses 2a and 2b, section 6.2) and
3. Test whether the effort prediction accuracy is improved for a homogenous data set from the application of the data analysis framework (Hypothesis 3b, section 6.2).

8.5.1 Graphical Results

To graphically illustrate and further compare results yielded by the above five data sets, we plot two bar-charts for each selected prediction approach in Figures 8.11 to 8.18.

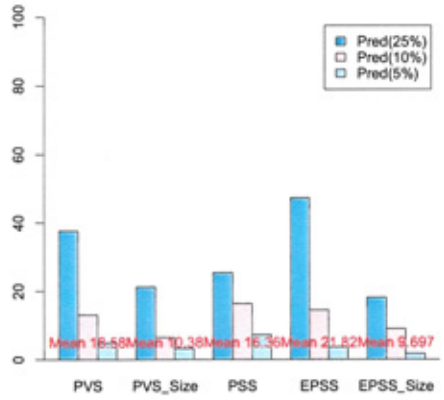

Figure 8.11: Comparison of Pred(l) Based on OLS

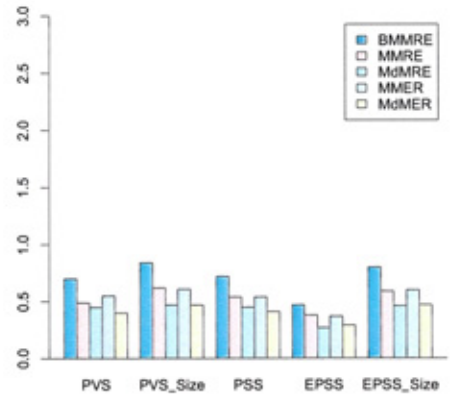

Figure 8.12: Comparison of MMRE etc. Based on OLS

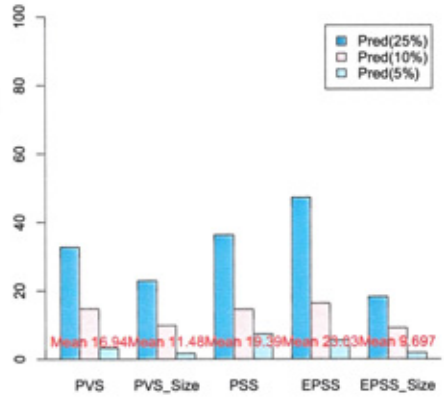

Figure 8.13: Comparison of Pred(l) Based on RR

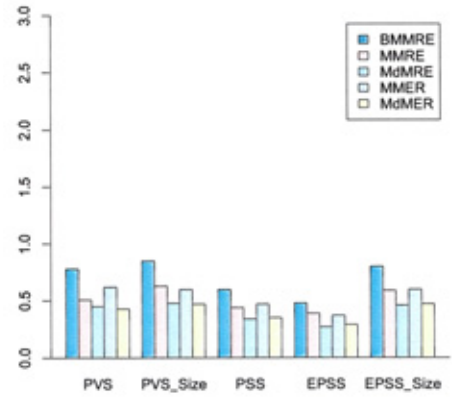

Figure 8.14: Comparison of MMRE etc. Based on RR

Table 8.14 summarizes the best performed data sets for each technique under each of the selected criterion. For example, according to the average value of Pred(l), where $l = 25\%, 10\%, 5\%$, the models based on the EPSS data set perform the best for OLS, RR and CART methods with the Averg.Pred(l) value 21.82, 23.03 and 20.00 respectively.

For the Least-Square Regression (OLS), Table 8.10 and Figure 8.11 reveal that the model based on the EPSS outperforms the models based on the other four data sets

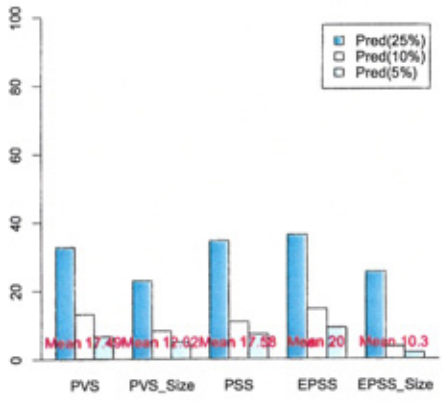

Figure 8.15: Comparison of Pred(l) Based on CART

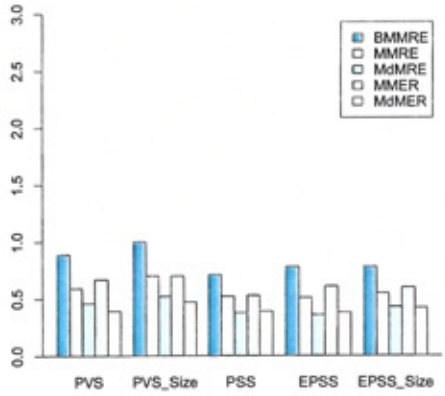

Figure 8.16: Comparison of MMRE etc. Based on CART

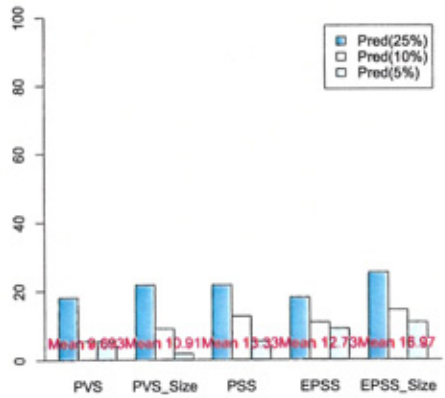

Figure 8.17: Comparison of Pred(l) Based on KNN

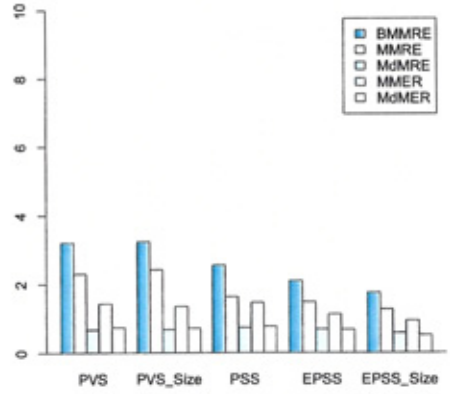

Figure 8.18: Comparison of MMRE etc. Based on KNN

Data Set	Max. Pred(l)	Averg Pred(l)	BMMRE	MMRE	MdMRE	MMER	MdMER	St. Dv.
PVS								
PVS_{Size}								
PSS			C. (0.71)			C. (0.53)		C. (4965.93)
EPSS	O. (47.27) R. (47.27) C. (36.36)	O. (21.82) R. (23.03) C. (20.00)	O. (0.47) R. (0.48)	O. (0.38) R. (0.39) C. (0.51)	O. (0.27) R. (0.27) C. (0.36)	O. (0.37) R. (0.37)	O. (0.29) R. (0.29) C. (0.38)	O. (3310.49) R. (3393.25)
$EPSS_{Size}$	K. (25.45)	K. (12.73)	K.(1.75)	K. (1.27)	K. (0.58)	K. (0.94)	K. (0.52)	K. (6970.64)
O. stands for OLS, R.stands for RR, C. stands for CART, and K. stands for KNN								

Table 8.14: Case Study B: Best Performed Data Sets Based on Four Prediction Techniques

for Pred(25%)=47.27, MMRE = 0.38, MdMRE=0.27 etc. On the whole, the MMRE and Pred(l) measures appear to suggest that the model based on the EPSS data set's overall performance of the analysis of models is the best.

Similarly, for the Robust Regression (RR), Table 8.11, Figure 8.13 and 8.14 present the similar trend with OLS results. the model based on the EPSS data set has the highest average values of Pred(l) ($l = 25\%, 10\%, 5\%$) and the second lowest values of MRE and MER related values. On the whole, the MMRE and Pred(l) measures appear to suggest that the model based on the EPSS data set's overall performance of the analysis of models is the best.

For the CART, the prediction accuracy performance is inconsistent across metrics. For example, consider the figures in the Table 8.12, Figure 8.15 and 8.16, according to MMRE, MdMRE, MdMER values and Pred(l), the model based on the data set EPSS performs the best, but according to BMMRE, and MMER, the model based on the PSS data set performs the best.

For the KNN, the prediction accuracy performance is consistent across metrics. Considering the figures in the Table 8.13, Figures 8.17 and 8.18, the model based on the $EPSS_{Size}$ data set performs the best. This observation is consistent with the findings of the Case Study A in chapter 7.

Based on the above, the following observations can be made in this case study using Bank63 homogenous data set:

- The performance of the models is consistent across metrics except CART. For example, consider the Pred(l) and BMMRE vales of the models built by OLS and

RR, results of both techniques support the contention that the model based on EPSS data set performs the best. While for KNN, the results of both techniques indicate that the model based on $EPSS_{Size}$ data set performs the best.

- Consistent with the ISBSG R9 case study, apart from KNN and CART, there is a clear tendency that the project effort prediction model constructed from the framework processed data sets (EPSS) performs better than the models based on the raw data set (PVS) and the data sets containing the most significant predictor variable only (PVS_{Size} and $EPSS_{Size}$). For CART, although the model based on PSS data set outperforms the others, results of the models based on EPSS and PSS data sets are very close. A further analysis is needed. As described in section 6.6, significance testing of the above comparisons, in particular results between models based on PSS and EPSS data sets, will be conducted in the following two subsections.

- Consistent with the ISBSG R9 case study, the model built by KNN performs worse than the models built by other techniques across five data sets. For examples, the model built by KNN has the lowest Averg.Pred(l) value and the highest BMMRE value. The poor performance of the KNN method can be attributed to the equal weights placed on variables in the similarity measure. Greater weighting assigned to more significant predictor variables [279] would change the basis for the similarity measures and analog retrieval.

8.5.2 Comparison Between Data Sets

Although we have made some observations based on figures and graphics in the preceding subsections, to rigorously compare the difference between prediction accuracy between data sets or techniques we need to perform statistical significance testing as designed in section 6.6. Table 7.16 presents the results of of Test A, section6.6, that compares difference of the prediction accuracy between the model based on EPSS data set and models based on the other four data sets for each prediction technique. The analysis supports the following contentions:

- There is a significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PVS data set across OLS, RR and KNN techniques, at 0.2 level of significance.

That is, the model based on the EPSS data set outperforms the model based on the PVS data set for the above three techniques. Although, there is no significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PVS data set for CART, the model based on the EPSS data set is in favour of the high Max.Pred (l) and Averg. Pred(l) results presented in Table 8.14. Therefore, Hypothesis 1a is true for both parametric methods (OLS and RR) and non-parametric method (CART and KNN) in this case study. This result is consistent with the findings from the Case Study A (based on ISBSG R9 data).

- For all four prediction techniques, there is a significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PVS_{Size} data set, at 0.2 level of significance. That is, the model based on the EPSS data set outperforms the model based on the PVS_{Size} data set. Therefore, Hypothesis 1b is true for this case study. This results is also consistent with the findings from the Case Study A (based on ISBSG R9 data).

- For all four prediction techniques, there is no significant difference between prediction accuracy (based on MRE values) of the model based on the EPSS data set and the model based on the PSS data set. However, the EPSS data set consists of much less predictor variables (4 variables) than PSS data set (24 variables), therefore the model based on the EPSS data set is more effective. In addition, for OLS, RR and CART techniques, the the model based on the EPSS data set is in favor of higher Max.Pred (l) and Averg. Pred(l) values presented in Table 8.14. Therefore, for all four techniques, the model based on the EPSS data set performs no worse than the model based on the PSS data set. In other words, Hypothesis 2a is true for this case study. Again, this results is consistent with the findings from the Case Study A (based on ISBSG R9 data).

- Except the KNN, there is no significant difference between prediction accuracy based on the MRE values of the model based on the EPSS data set and the model based on the $EPSS_{Size}$ data set. Again, the EPSS data set is in favour of higher Averg. Pred(l) values presented in Table 8.14. For OLS, RR and CART, the model based on the EPSS data set perform better than the model based on the $EPSS_{Size}$ data set. For the KNN, the model based on the $EPSS_{Size}$ data set

outperforms the model based on the EPSS data set, at 0.01 level of significance. Therefore, Hypothesis 2b is true for both parametric methods (OLS and RR) and one non-parametric method CART in this case study. This results is still consistent with the findings from ISBSG R9 case study.

- Accordingly, in view of the above contentions, Hypothesis 3b is true for models built by three prediction techniques: OLS, RR and CART.

Technique	Comparison vs. EPSS	Testing Method	Testing Result	P-value
OLS	PVS	t-test	t = -1.6541	0.1009
	PVS_{Size}	t-test	t = -2.3489	0.02102
	PSS	Wilcoxon-test	V = 363	0.9997
	$EPSS_{Size}$	Wilcoxon-test	V = 362	0.99971
RR	PVS	t-test	t = -1.7598	0.08112
	PVS_{Size}	t-test	t = -2.3293	0.02205
	PSS	Wilcoxon-test	V = 641	0.861
	$EPSS_{Size}$	Wilcoxon-test	V = 388	0.9993
CART	PVS	t-test	t = -0.5645	0.5736
	PVS_{Size}	t-test	t = -1.4601	0.1471
	PSS	Wilcoxon-test	V = 66	0.5514
	$EPSS_{Size}$	Wilcoxon-test	V = 318	0.9808
KNN	PVS	t-test	t = -1.2863	0.2019
	PVS_{Size}	t-test	t = -1.3581	0.1784
	PSS	Wilcoxon-test	V = 772	0.495
	$EPSS_{Size}$	Wilcoxon-test	V = 930	0.09071

Table 8.15: Case Study B: Test A Results at 95% Confidence Interval

8.5.3 Comparison Between Prediction Techniques

Comparison Btw Techniques	Testing Method	Testing Result	P-value
OLS vs. RR	Mann-Whitney U-test	W = 1482	0.5735
OLS vs. CART	Mann-Whitney U-test	W = 1334	0.8577
OLS vs. KNN	Mann-Whitney U-test	W = 787	1
RR vs. CART	Mann-Whitney U-test	W = 1359	0.8214
RR vs. KNN	Mann-Whitney U-test	W = 794	1
CART vs. KNN	Mann-Whitney U-test	W = 957	0.9996

Table 8.16: Case Study B: Test B Results at 95% Confidence Interval

According to the Mann-Whitney U-test results presented in Table 8.16, we observe that, for the model based on the EPSS data set there is no significant difference of prediction accuracy (based on MRE) between various techniques. Therefore, the prediction results based on the EPSS data set are consistent across all four techniques.

8.6 Summary

Using the Bank63 data set, this chapter described the application of the proposed preliminary framework used within four commonly used prediction techniques. For software cost models used each prediction technique, the framework processed data set (EPSS) is compared to other data sets, including the raw data set (PVS), raw data set with the most significant predictor variable (PVS_{Size}), the raw data set without outliers (PSS) and the framework processed data set with the most significant predictor variable ($EPSS_{Size}$).

These comparisons addressed the first two research hypotheses detailed in section 6.2. Significance testing (Test A as detailed in section6.6) was performed for these comparisons. The general question behind the first hypothesis is: which data set provides the most accurate prediction, PVS_{Size}, PVS, or EPSS? This evaluates the validity of the framework processed data set EPSS. The general question behind the second hypothesis is: which data set provides the most accurate prediction, $EPSS_{Size}$, PSS, or EPSS? This evaluates the significance of the framework processed data set EPSS. The significance testing (Test B as detailed in section6.6) also addresses the question: whether the proposed framework is applicable to the selected commonly used prediction techniques. This evaluates the applicability of the proposed framework.

Overall, the analysis shows the following. First, for all four prediction techniques, Hypotheses 1a and 1b are true in this case study. Second, for three out of four techniques (except KNN), Hypotheses 2a and 2b are true. Third, except for CART, there does appear to be consistent variation in performance between data sets. The proposed framework is applicable for OLS, RR and CART, in the company specific data set. Fourth, parametric modeling methods (OLS and RR) produced more accurate results than non-parametric modeling methods (CART, and Analogy-based, i.e.,KNN), within compony-specific contexts.

There are additional results that should also be considered.

The validity of evaluation criteria: except KNN, the BMMRE, MRE based and MER based criteria do show a consistent tendency for each prediction model, however, for KNN, MER based and BMRE criteria show negative correlations with the K value. As illustrated in Tables C.13 to C.17 in Appendix C, with the increasing of K value, Pred(l) values show approximate bell-shape curves and MRE based criteria (MMRE

and MdMRE) values show approximate reversed bell-shape curves. Therefore, MER based and BMRE criteria are doubtful within this case study. Further investigation could be made to evaluate the validity of these newly proposed criteria.

KNN does not perform as effectively as the other prediction techniques. This observation is consistent with other research findings, as observed by Briand *et al.* [42, 44], Walkerden and Jeffery [306], etc. The poor performance of the KNN method can be attributed to the equal weights placed on variables in the similarity measure. A further plausible reason for the inaccurate prediction of analogy based techniques (e.g.,KNN) has been argued by Walkerden and Jeffery [306]. They contend that this occurs because software tools were used to calculate analogues and it is the case that experts are better than tools at selecting analogues.

Part IV

Conclusions and Further Work

Chapter 9

Conclusions

9.1 Conclusions

This research has established a preliminary data analysis framework for the construction of effort prediction models based on historical data sets. It has adopted and extended earlier work of other researchers, in particular, Kitchenham [168], and Maxwell [202, 203]. The proposed framework is an enhancement that builds upon Maxwell's analysis of variance procedure [203]. It extends Maxwell's analysis procedure in ways that

- Formally define reduction rules that facilitate:
  - criteria for removing outliers;
  - rules for incorporating inter-correlated variables;
  - clarification of statistical methods that handle differently scaled variables e.g. ratio, ordinal, and nominal.
- Further analyze potential inter-correlated predictor variables, e.g. adopts univariate analysis to identify collinearity between categorical predictor variables.

The proposed framework clearly defines a set of statistical methods that remove extreme project samples, small effect attributes and inter-related project attributes. The framework provides a formal basis for constructing cost prediction models from historical data sets.

To evaluate the accuracy of the estimation, this research provides a comprehensive accuracy evaluation. The predictive accuracy of the framework processed data is empirically evaluated by defining a systematic and repeatable evaluation method. The framework processed data set is compared to other data sets, i.e.

- The raw data set.
- The data set only consists of the most significant predictor variable and the predicted variable from the raw data set.
- The raw data set without outliers.
- The data set only consists of the most significant predictor variable and the predicted variable from the framework processed data set.

These comparisons are repeated for a comprehensive set of commonly used estimation methods: Ordinary Least-Square Regression (OLS), Robust Regression (RR), K-Nearest Neighbour (KNN) and Classification and Regression Trees (CART).

The defined evaluation method was consistently applied to two representative cost data sets from different application domains (ISBSG R9 data set) and from the same application domain (Bank63 data set) respectively. Both of the data sets are in the public domain and are freely accessible. The ISBSG R9 data set is the latest version of the data collection from ISBSG in 2005. The Bank63 data is the same data set released and employed in Maxwell's research.

As in any other experimental or empirical field, replication is the key to establishing the validity and generalizability of results. In the context of software cost estimation, the goal was to determine the stability of the trends observed across data sets and the generalizability of results across applications for commonly used estimation methods. Two questions were empirically investigated and are significant in the field of software cost modeling. First, which subset from the same historical data set will yield more accurate effort prediction results when using commonly used methods? And second, what are the benefits and drawbacks of using organization specific data as compared to multi organization databases?

Most of the findings were consistent in both studies and therefore give evidence to be of general validity of the procedure, as follows.

1. The proposed preliminary framework is valid. i.e., The framework processed data set (EPSS) performed no worse than the raw data set (PVS) and the raw data set with the most significant predictor variable (PVS_{Size}).

2. The predictor variables (i.e., project attributes) extracted by the proposed preliminary framework are statistic significant for application of OLS, RR and CART techniques, at 95% confidence level.

3. The proposed preliminary framework is applicable for effort prediction models based on OLS, RR and CART techniques.

4. The application of the proposed framework improved the effort prediction accuracy of both heterogeneous and homogeneous data sets.

5. The homogenous data set is more effective than the heterogenous data set after being processed by the preliminary analysis framework.

6. The models based on parametric prediction techniques (OLS and RR) performed more accurately than the models based on non-parametric prediction techniques (CART and KNN). This is consistent with [130].

The principal conclusions of this research are as follows. When building a cost prediction model based on historical data sets:

- **Conclusion 1** Removing extreme project data, i.e. outliers improves the accuracy of effort prediction.
  - Within the 20 models developed in Case Study A, compared with the model based on the raw data set, the model based on the data set without outliers improved the prediction accuracy such that (1), Pred(25%) values are increased up to 21% with an average of 16%; (2), MMRE values are decreased up to 55% within an average of 39%.
  - Within the 20 models developed in Case Study B, compared with the model based on the raw data set, the model based on the data set without outliers improved the prediction accuracy such that (1), Pred(25%) values are increased up to 20% with an average of 1%; (2), MMRE values are decreased up to 29% within an average of 11%.
- **Conclusion 2** Removing small effect and inter-correlated predictor variables improves prediction performance.
  - Within the 20 models developed in Case Study A, compared with the model based on the raw data set, the model based on the framework processed data set improved the prediction accuracy such that (1), Pred(25%) values are increased up to 30% with an average of 16%; (2), MMRE values are decreased up to 53% within an average of 35%.
  - Within the 20 models developed in Case Study B, compared with the model based on the raw data set, the model based on the framework processed data set improved the prediction accuracy such that (1), Pred(25%) values are increased up to 44% with an average of 20%; (2), MMRE values are decreased up to 36% within an average of 24%.

- **Conclusion 3** Case studies in this research show only a few predictor variables (3 out of 15 variables for the Case Study A and 4 out of 24 for the Case Study B) in historical data sets contribute the most for effort prediction.
  - In Case Study A, three framework selected variables alone (3/15 = 20% of the total variables) explained 62.22% of the variation in *effort*.
  - In Case Study B, four framework selected variables alone (4/24 = 16.7% of the total variables) explained 78% of the variance of project *effort*.
- **Conclusion 4** Both heterogeneous and homogeneous data sets benefit from the preliminary data analysis framework by providing more accurate effort prediction results.
- **Conclusion 5** The homogenous data set is more effective than the heterogenous data set after the preliminary analysis.
- **Conclusion 6** Overall, the models using parametric prediction techniques (OLS and RR) performed more accurately than the models based on non-parametric prediction techniques (CART and KNN).
  - Within the 20 models developed in Case Study A, the models used parametric prediction techniques based on the framework processed data set outperformed the models used non-parametric prediction techniques based on the same data set such that Pred(25%) values are increased with an average of 26% for the former and only 5% for the latter.
  - Within the 20 models developed in Case Study B, the models used parametric prediction techniques based on the framework processed data set outperformed the models used non-parametric prediction techniques based on the same data set such that Pred(25%) values are increased with an average of 36% for the former and only 5% for the latter.

This research contributes in two areas.

- Academics in the area of empirical software engineering using historical data sets for research. Academics have restricted access to industrial project data, due in part to the reluctance of companies to release such data into the public domain. Data sets are used for research into prediction as well as many other areas in

order to better understand software development phenomena. Results from such research will feed back into industrial practice, further benefitting the software development industry.

- Project or metrics managers, responsible for developing prediction models for software projects. They will have access to the support tool enabling them to make best use of their available data sets, which will in turn allow them to develop more accurate prediction models.

Chapter 10

Limitations and Further Work

10.1 Limitations

This research initiative is the first to undertake a wide-scale and consistent comparison of cost models on relatively large software engineering data sets. Although it overcomes several of the weaknesses of other empirical studies, there are some limitations of this work related to construct, internal, and external validity [155].

From a general perspective, the main limitation of this thesis is its restriction to data-driven cost estimation methods. This, of course, requires to various degrees large amounts of data to apply those methods. Moreover, another well-known limitation of models constructed using historical data is that attributes used to predict software development effort can change over time and/or differ between software development environments [286]. Mohanty [213] makes this point in comparisons between the predictions of a wide variety of models on a single hypothetical software project. Although in this study we used the latest version data set of ISBSG data set (with 70% of the projects being less than six years old), but the other data set Bank63 is relatively old (collected in 1993).

10.1.1 Project Size Measurement

It is widely accepted that among a variety of many factors that affect the cost of software projects, software size is one of the key cost drivers [26, 144]. This assumption has been proved in both case studies in this thesis. The size of a software system was measured in Function Points in the both data sets in this study.

The Function Point (FP) [3] count is a weighted sum of different function types and can further be adjusted for technical processing complexities (adjusted FP). However, the rational for the adjustment is not clear and it is also of concern that using the complexity weights does not necessarily explain more variation in effort compared than non-weighted function points [135]. To avoid this problem, in this study, we used the unadjusted function point count from ISBSG R9 data set. However, for Bank63 data set we only have the so called Experience Function Points, which the rational for the adjustment is still not clear. Great care must be taken when using a model constructed from data from one environment to make predictions about data from another environment.

10.1.2 Data Set Limitations

Data are collected for both studies under the guidance of formal procedures. The data are verified and maintained by highly reputable institution (ISBSG) and experts (K. Maxwell). Despite these assurances, it is worthwhile recognizing that the data collection process was beyond the control of the current study. The reliable measurement for all variables, the sample size, and causes for missing data could not be specifically determined though are understood to be potential problem areas. It should also be emphasized that the presented results are explicitly based upon the data collected.

10.1.3 Research Method Limitations

Without expert opinion and additional insights into the projects' details, this study was limited concerning the application of pure data-driven estimation. For example, the selection of project analogs could have possibly been improved with the input from expert judgement. Several studies [292, 124] emphasize the importance of expert input. Moreover, without the assistance of 'inside knowledge', suitable weights could not be assigned to variables and individual specific projects could not be identified as outliers or 'exceptional' cases. A possible objective method to assign weights would be to incorporate them according to the relative significance of the variables, such as the percentage of the explained variation identified by the proposed framework.

The current study's construct validity is the extent to which the prediction accuracy, is operationalized, measured by magnitude of relative error (MRE). Construct validity is threatened since there are drawbacks associated with the MRE measure [212]. There is an apparent inconsistency in the way the effort estimation methods optimize the MRE. However, the MRE is a widely used measure for evaluating estimation methods and thus was used in this study. As a cross-check, we applied alternative evaluation measures and found no significant changes in the general trends of the results.

Within the ISBSG R9 data set, the project IDs are totally random for the proprietary reason. There is no company-specific data available for evaluating cost models' accuracy based on heterogeneous and homogeneous data. If one or more companies within the ISBSG R9 data set were examined, this could have revealed different results. Therefore, the comparison results between heterogeneous data set and homogeneous data set should be primarily considered in the context of both the entire ISBSG R9

data and Bank63 data set. Thus, the external validity of the research is threatened at the company-specific data level.

10.2 Further Research

From a scientific perspective, several issues remain to be investigated.

First, research needs to be conducted to ascertain the applicability of the preliminary data framework based on further historical data sets. Comprehensive and consistent comparisons of cost estimation methods are still rare. Thus, conclusions of general validity are still difficult to draw. Researchers need to consistently compare estimation methods across many data sets in order to be able to define guidelines about what kind of models to use for a given environment. This thesis makes a big step in this direction.

Second, research needs to be conducted to ascertain the feasibility by integrating missing data techniques such as listwise deletion (LD) [224], mean imputation (MI) [224], similar response pattern imputation (SRPI) and full information maximum likelihood (FIML), sparse data method (SDM) [276] etc., for handling missing values in historical data sets.

Third, an outcome of the research could be the provision of benchmarking results, i.e. allowing future proposed cost estimation methods to be compared against known standards and allowing effective future comparison based on the same historical data set.

In addition, as argued by Stensrud *et al.* [292] and Baik *et al.* [9], resource estimation methods are typically not used in isolation but as a support tool for human estimators. It is therefore important that research in cost estimation takes into account this human factor component. Methods that combine expert knowledge with data driven techniques may allow individual organizations to develop specific resource estimation models while alleviating the stringent requirements for project data, e.g., Bayesian Analysis method. Such methods make use of all information available in organizations and therefore make practical and economic sense.

List of References

[1] D. W. Aha. Case-based learning algorithms. In *Proceedings of the Case Based Reasoning Workshop*, pages 147–158, Washington, D.C., 1991.

[2] F. Akiyama. An example of software system debugging. In *Proceedings of the Information Processing 71*, pages 353–379, 1971.

[3] J. A. Albrecht. Measuring applications development productivity. In *Proceedings of the IBM Applications Development Joint SHARE/GUIDE Symposium*, pages 83–92, Monterey CA, 1979.

[4] J. A. Albrecht and J. E. Gaffney. Software function, source lines of code and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, 9(6):639–648, November 1983.

[5] L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical Software Engineering*, 5(1):35–68, 2000.

[6] L. Angelis and I. Stamelos. Reply to comments by M. Jorgensen, on the paper:'A simulation tool for efficient analogy based cost estimation' by L. Angelis and I. Stamelos, published in Empirical Software Engineering, 5(1):35-68 (2000). *Empirical Software Engineering*, 7(4):377–382, 2002.

[7] K. Ashley and E. Rissland. Compare and contrast, a test of expertise. In *Proceedings of the AAAI-87, the 6th AAAI National Conference*, pages 273–278, Seattle, WA, 1987.

[8] J. Babsiya and C. G. Davis. A hierarchical model for Object-Oriented design quality assessment. *IEEE Transactions on Software Engineering*, 28(1):4–17, January 2002.

[9] J. Baik, B. Boehm, and B. M. Steece. Disaggregating and calibrating the case tool variable in COCOMO II. *IEEE Transactions on Software Engineering*, 28(11):1009–1022, 2002.

[10] J. Bailey and V. Basili. A meta-model for software development resource experiments. In *Proceedings of the 5th International Software Engineering*, pages 107–116, Los Alamitos, California, 1981. IEEE CS Press.

[11] B. Baird. *Managerial Decisions Under Uncertainty.* John Wiley & Sons, 1989.

[12] R. Banker, R. Kauffman, and R. Kumar. An empirical test of object-based output measurement metrics in a Computer Aided Software Engineering (CASE) environment. *Journal of Management Information Systems*, 8(3):127–150, 1992.

[13] L. Baresi, S. Morasca, and P. Paoini. An empirical study on the design effort of Web applications. In *Proceedings of the 3rd International Conference on web Information System Engineering*, pages 345–354. IEEE CS Press, 2002.

[14] V. R. Basili. Quantitative evaluation of software methodology. In *Proceedings of the First Pan Pacific Computer Conference*, 1985.

[15] V. R. Basili. The experimental paradigm in software engineering. In *Proceedings of the Dagstuhl-Workshop*, 1992.

[16] V. R. Basili and D. H. Hutchens. An empirical study of a syntactic complexity family. *IEEE Transaction of Software Engineering*, 9(6):664–672, 1983.

[17] V. R. Basili and H. D. Rombach. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, August 1988.

[18] D. Basmadjian. *The Art of Modelling in Science.* Chapman & Hall Crc, Boca Raton, London, New York, Washington, D.C., 1999.

[19] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1990.

[20] C. A. Behrens. Measuring the productivity of computer systems development activities with Function Points. *IEEE Transactions on Software Engineering*, 9(6):648–652, 1983.

[21] L. A. Belady. Complexity of large systems. In Alan J. Perlis, Frederick G. Sayward, and Mary Hsaw, editors, *Proceedings of the Software Metrics*, pages 225–233. MIT Press, Cambridge, MA, 1981.

[22] H. D. Benington. Production of large computer. In *Proceedings of the Symposium on Advanced Computer Programs for Digital Computers*, Washington, D. C., 1956. Office of Naval Research.

[23] R. Bisio and F. Malabocchia. Cost estimation of software projects through Case Base Reasoning. In *Proceedings of the 1st International Conference on Case-Based Reasoning Research & Development.* Springer-Verlag, 1995.

[24] B. W. Boehm. *Software Engineering Economics.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[25] B. W. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches - a survey. *Annals of Software Engineering*, 10:177–205, 2000.

[26] B. W. Boehm, C. Abts, B. A. Insor, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, and B. Steece. *Software Cost Estimation with COCOMO II.* Prentice Hall PTR, Upper Saddle River, NJ, 2000.

[27] B. W. Boehm, J. R. Brown, J. R. Kasper, and M. J. Merrit. *Characteristics of Software Quality.* North-Holland, Amsterdam, 1978.

[28] B. W. Boehm, S. Chulani, and B. Clark. The rosetta stone: Making COCOMO'81 files work with COCOMO II. Technical report, Center for Software Engineering, 1998.

[29] B. W. Boehm, S. Chulani, and D. Reifer. Calibrating the COCOMO II post architecture model. Technical report, Center for Software Engineering, 1997.

[30] B. W. Boehm, B. Clark, E. Horowitz, R. Madachy, R. Selby, and J. C. Westland. The COCOMO 2.0 software cost estimation model. *American Programmer*, 9(7):2–17, 1996.

[31] B. W. Boehm, B. Clark, E. Horowitz, and C. Westland. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1:57–94, 1995.

[32] B. W. Boehm, T. E. Gray, and T. Seewaldt. Prototyping versus specifying: A multi-project experiment. *IEEE Transactions on Software Engineering*, 10(3):290–303, 1984.

[33] B. W. Boehm, E. Horowitz, R. Madachy, and C. Abts. Future trends, implications in software cost estimation models. *CrossTalk*, pages 4–8, April 2000.

[34] G. Box and G. Tiao. *Bayesian Inference in Statistical Analysis.* Addison Wesley, 1973.

[35] L. Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24:2350–2383, 1996.

[36] L. Breiman, J. H. Friedman, R. A. Olshen, , and C. J. Stone. *Classification and Regression Trees.* Wadsworth Inc, Belmount CA, 1984.

[37] L. Breiman and P. Spector. Submodel selection and evaluation in regression: The X-random case. *International Statistical Review*, 60:291–319, 1992.

[38] L. C. Briand. Measurement and quality modelling of OO systems. In *Proceedings of the Sixth International Symposium on Software Metrics*, Boca Raton, Florida, November 1999.

[39] L. C. Briand, V. R. Basili, and C. Hetmanski. Providing an empirical basis for optimizing the verification and testing phases of software development. In *Proceedings of the IEEE International Symposium Software Reliability Engineering*, October 1992.

[40] L. C. Briand, V. R. Basili, and C. Hetmanski. Developing interpretable models with optimized set reduction for identifying high-risk software components. *IEEE Transactions on Software Engineering*, 19(11):1028–1044, 1993.

[41] L. C. Briand, V. R. Basili, and W. Thomas. A pattern recognition approach for software engineering data analysis. *IEEE Transactions on Software Engineering*, 18(11):931–942, 1992.

[42] L. C. Briand, K. El Emam, and I. Wieczorek. Explaining the cost of European Space and Military projects. In *Proceedings of the International Conference on Software Engineering, ICSE99*, pages 303–312, Los Angeles, USA, May 1999.

[43] L. C. Briand, K. El Eman, and F. Bomarius. COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment. In *Proceedings of the International Conference on Software Engineering*, pages 390–399, 1998.

[44] L. C. Briand, K. El Eman, K. Maxwell, D. Surmann, and I. Wieczorek. An assessment and comparison of common software cost estimation modelling techniques. In *Proceedings of the International Conference on Software Engineering, ICSE99*, pages 313–322, Los Angeles, USA, May 1999.

[45] L. C. Briand, T. Langley, and I. Wieczorek. A replicated assessment and comparison of common software cost modelling techniques. In *Proceedings of the 22nd International Conference on Software Engineering*, pages 377–386, Limerick, Ireland, 04-11 June 2000.

[46] L. C. Briand, S. Morasca, and V. W. Basili. Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22(1):68–86, 1996.

[47] L. C. Briand and A. Porter. An alternative modelling approach for predicting error profiles in ada systems. In *Proceedings of the EUROMETRICS'92, European Conference on Quantitative Evaluation of Software and Systems*, pages 245–254, April 1992.

[48] L. C. Briand and J. Wüst. The impact of design properties on development cost in Object-Oriented systems. In *Proceedings of the 7th IEEE International Software Metrics Symposium (METRICS 2001)*, pages 260–271, London, England, 4-6 April 2001. IEEE Computer Society.

[49] L. C. Briand and J. Wüst. Modeling development effort in Object-Oriented systems using design properties. *IEEE Transactions on Software Engineering*, 27(11):963–986, 2001.

[50] F. P. Brooks. *The Mythical Man-Month, Essays on Software Engineering.* Addison-wesley publishing company, 1975.

[51] M. Brown and C. J. Harris. *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, Hemel Hempstead, 1994.

[52] A. Burr and M. Owen. *Statistical Methods for Software Quality Using Metrics for Process Improvement.* Thompson Computer Press, 1996.

[53] J. Capon. *Elementary Statistics for the Social Sciences.* Wadsworth, Belmont, CA, 1988.

[54] D. Card and B. Scalzo. Measurement for Object-Oriented software projects. In *Proceedings of the 6th International Symposium on Software Metrics*, Boca Raton, Florida, 4-6 November 1999.

[55] D. N. Card and R. L. Galss. *Measuring Software Design Quality.* Prentice Hall, 1990.

[56] M. Cartwright, M. J. Shepperd, and Q. Song. Dealing with missing software project data. In *Proceedings of the 9th IEEE International Metrics Symposium*, Sydney, Australia, 2003. IEEE Computer Society.

[57] D. De Champeaux. *Object-Oriented Development Process and Metrics.* Prentice Hall, Upper Saddle River, N.J., 1997.

[58] S. R. Chidamber and C. F. Kemerer. A metrics suite for Object Oriented design. *IEEE Transactions on Software Engineering*, 20(4):476–498, 1994.

[59] S. Chulani. Incorporating Bayesian analysis to improve the accuracy of COCOMO II and its quality model extension. Qualifying exam report, University of Southern California, February 1998.

[60] S. Chulani, B. W. Boehm, and B. Steece. Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering*, 25(4):573–583, 1999.

[61] S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software Engineering Metrics and Models.* The Benjamin/Cummings Publishing Company, Inc., 1986.

[62] D. Cook and S. Weisberg. *An Introduction to Regression Graphics.* Wiley Series, 1994.

[63] R. Cordero, M. Constamagna, and E. Paschetta. A genetic algorithm approach for the calibration of COCOMO-like models. In *Proceedings of the 12th COCOMO Forum*, 1997.

[64] N. Coulter. Software science and cognitive psychology. *IEEE Transactions on Software Engineering*, 9(2):166–171, 1983.

[65] IBM Object Oriented Technology Council. IBM Object Oriented Metrics. IBM internal technical paper, February 1993.

[66] A. M. E. Cuelenare, M. J. I. M. van Genuchten, and F. J. Heemstra. Calibrating a software cost estimating model: Why and how? *Journal of Information and Software Technology*, 29(10):558–569, 1987.

[67] B. Curits. In search of software complexity. In *Proceedings of the IEEE workshop on quantitative software models*, pages 95–106, 1979.

[68] B. Curits. Measurement and experimentation in software engineering. In *Proceedings of the IEEE*, volume 68, pages 1144–1157, 1980.

[69] B. Curits. Experimental evaluation of software characteristics. In A. J. Perlis, F. G. Sayward, and M. Hsaw, editors, *Software Metrics*, pages 62–75. MIT Press, Cambridge, MA, 1981.

[70] B. Curits. The measurement of software quality and complexity. In A. J. Perlis, F. G. Sayward, and M. Hsaw, editors, *Software Metrics*, pages 203–223. MIT Press, Cambridge, MA, 1981.

[71] N. Dalkey and O. Helmer. An experimental application of the Delphi method to the use of experts. *Management Saence*, 9(3):458, 1963.

[72] M. A. de Almeida, H. Lounis, and W. L. Melo. An investigation on the use of machine learned models for estimating correction costs. In *Proceedings of the IEEE International Conferences on Software Engineering*, 1998.

[73] S. Delany, P. Cunningham, and W. Wilke. The limits of CBR in software project estimation. In L. Gierl and M. Lenz, editors, *Proceedings of the German Workshop on Case-Based-Reasoning*, Berlin, 1998.

[74] T. DeMarco. *Controlling Software Projects: Management, Measurement and Estimation.* Prentice Hall, 1982.

[75] R. DeMillo and J. LiptonR. Software project forecasting. In A. J. Perlis, F. G. Sayward, and M. Hsaw, editors, *Software Metrics*, pages 203–223. MIT Press, Cambridge, MA, 1981.

[76] W. E. Deming. *Out of the Crisis.* MIT Center for Advanced Engineering Studies, Cambridge, Massachusetts, 1986.

[77] M. Denicoff and R. Grafton. Software metrics: A research initiative. In Alan J. Perlis, Frederick G. Sayward, and Hsaw eds. Mary, editors, *Software Metrics*, pages 13–18. MIT Press, Cambridge, MA, 1981.

[78] E. W. Dijkstra. Programming considered as a human activity. In *Proceedings of the 1965 IFIP Congress*, pages 213–217, Amsterdam, The Netherlands, 1965. North Holland Publishing Company.

[79] W. R. Dillon and M. Goldstein. *Multivariate Analysis: Methods and Applications.* John Wiley & Sons, New York, 1984.

[80] C. Ebert. Experiences with criticality predictions in software development. *ACM SIGSoft SEN*, 22:278–293, 1997.

[81] A. Endres. A synopsis of software engineering history: The industrial perspective. In A. Brennecke and R. Keil-Slawik, editors, *the Position Papers for Dagstuhl Seminar 9635 on History of Software Engineering*, pages 20–24. 1996.

[82] W. M. Evangelist. Software complexity metric sensitivity to program restructuring rules. *Journal of Systems and Software*, 3(3):231–243, 1983.

[83] R. Fairley. Recent advances in software estimation techniques. In *Proceedings of the 14th International Conference on Software Engineering*, pages 382–391, Melbourne Australia, 1992.

[84] N. E. Fenton, P. Krause, and M. Neil. Software measurement: Uncertainty and causal modelling. *IEEE Software*, 10(4):116–122, 2002.

[85] N. E. Fenton and M. Neil. Software metrics: Successes, failures and new directions. *The Journal of Systems and Software*, 47(2-3):149–157, 1999.

[86] N. E. Fenton and M. Neil. Bayesian Belief Nets: A causal model for predicting defect rates and resource requirements. *Software Testing and Quality Engineering*, 2(1):48–53, 2000.

[87] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26(8):797–814, 2000.

[88] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* PWS Publishing Company, 1997.

[89] D. V. Ferens and R. B. Gurner. An evaluation of three Function Point models for estimation of software effort. *IEEE National Aerospace and Electronics Conference - NAECON92*, 2:635–642, 1992.

[90] G. R. Finnie and G. E. Wittig. AI tools for software development effort estimation. In *Proceedings of the Software Engineering and Education and Practice Conference*, pages 346–353. IEEE Computer Society Press, 1996.

[91] G. R. Finnie, G. E. Wittig, and J. M. Desharnais. A comparison of software effort estimation techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression models. *Journal of Systems and Software*, 39(3):281–289, 1997.

[92] G. R. Finnie, G. E. Wittig, and J. M. Desharnais. Estimating software development effort with Case-Based Reasoning. In Leake D. and Plaza E., editors, *Proceedings of the International Conference on Case-Based Reasoning*, pages 13–22, 1997.

[93] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003.

[94] K. Funahashi. On the approximate realization of continuous mappings by Neural Networks. *Neural Networks*, 2(3):183–192, 1989.

[95] P. W. Garratt and A. C. Hodgkinson. A neurofuzzy cost estimator. In *Proceedings of the 3rd International Conference of Software Engineering and Applications (SAE)*, pages 401–406, 1999.

[96] D. Giddings. *Research Methods: Introduction to Statistical Analysis with SPSS.* School of Computing, Engineering, and Information Sciences, Northumbria Univeristy, U.K., 2004.

[97] T. Gilb. *Software Metrics.* MA:Winthrop Publishers, Inc., Cambridge, 1976.

[98] T. Gilb and D. Graham. *Software Inspections.* Addison-Wesley, Reading, 1994.

[99] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[100] R. B. Grady. *Practical Software Metrics for Project Management and Process Improvement.* Prentice Hall, 1992.

[101] R. B. Grady and D. Caswell. *Software Metrics: Establishing a Company-wide Program.* Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[102] T. L. Graves and A. Mockus. Identify productivity drivers by modelling work units using partial data. Technical Report BL0113590-990513-09TM, Bell Laboratories, 2000.

[103] F. J. Gravetter and L. B. Wallnau. *Statistics for the Behavioral Science: A First Course for Students of Psychology and Education.* St.Paul: West, 4th edition, 1996.

[104] A. R. Gray and S. G. MacDonell. A comparison of techniques for developing predictive models of software metrics. *Information and Software Technology*, 39(6):425–437, 1997.

[105] A. R. Gray and S. G. MacDonell. Software metrics data analysis - exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, 4(4):297–316, 1999.

[106] The Standish Group. Chaos report. Technical report, The Standish Group, 1995.

[107] J. F. Hair, R. E. Anderson, R. L. Tatham, and W. C. Black. *Multivariate Data Analysis.* Prentice-Hall, Inc., 4th edition, 1995.

[108] J. Hale, A. Parrish, R. Smith, and B. Dixon. Enhancing the COCOMO estimation models. *IEEE Software*, 17(6):45–49, 2000.

[109] T. Hall and N. E. Fenton. Implementing effective software metrics programmes. *IEEE Software*, 14(2):55–66, 1997.

[110] M. Halstead. *Elements of Software Science.* North Holland, 1977.

[111] M. Halstead. Guest editorial on software science. *IEEE Transactions on Software Engineering*, 5(2):74–75, 1979.

[112] P. B. Hamer and G. Frewin. Halstead's software science - a critical examination. In *Proceedings of the 6th International Conference on Software Engineering*, pages 197–206, 1982.

[113] W. Hayes. *Statistics.* Hartcourt Brace College Publishers, 5th edition, 1994.

[114] F. Heestra. Software cost estimation. *Information and Software Technology*, 34(10):627–639, 1992.

[115] F. Heestra and R. J. Kusters. Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems*, 1(4):229–237, 1991.

[116] O. Helmer-Heidelberg. *Social Technology, Basic Books.* New York, 1966.

[117] B. Hendeson-Selles. *Object-Oriented Metrics,Measures of Complexity.* PTR Prentice-Hall, Englewood Cliffs, N.J., 1996.

[118] J. R. Herd, J. N. Postak, W. E. Russell, and K. R. Steward. Software cost estimation study - study results. Final technical report, radc-tr77-220, vol. i, Doty Associates, Inc., Rockville, MD,, 1977.

[119] B. Hetzel. *Making Software Measurement Work: Building an Effective Measurement Program (Qed Software Evaluation).* Wiley, 1993.

[120] J. Hihn and H. Habib-Agahi. Cost estimation of software intensive projects: A survey of current practices. In *Proceedings of the 13th International Conference on Software Engineering*, pages 276–287, 1991.

[121] G. E. Hinton. *Personal Communications.* University of Sydney, February 1993.

[122] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[123] M. Höst and C. Wohlin. An experimental study of individual subjective effort estimation and combinations of the estimates. In *Proceedings of the 20th International Conference on Software Engineering, ICSE98*, pages 332–339, Japan, 1998.

[124] R. T. Hughes. Expert judgment as an estimating methods. *Information and software technology*, 38(2):67–75, 1996.

[125] W. S. Humphrey. *Managing the Software Process.* MA: Addison-Wesley, Reading, 1989.

[126] A. Idri, A. Abran, and T. M. Khosgoftaar. Fuzzy analogy: A new approach for software estimation. In *International Workshop on Software Measurement (IWSM'01)*, Montréal, Québec, Canada, August 28-29 2001.

[127] A. Idri, A. Abran, and L. Kjiri. COCOMO cost model using fuzzy logic. In *Proceedings of the 7th International Conference on Fuzzy Theory & Technology*, Atlantic City, New Jersey, February 27 - March 3 2000.

[128] A. Idri, T. M. Khosgoftaar, and A. Abran. Can neural networks be easily interpreted in software cost estimation? In *Proceedings of the 2002 World Congress on Computational Intelligence (WCCI 2002)*, page 8, Honolulu, Hawaii, May 12-17 2002.

[129] IFPUG. *Counting Practices Manual, Release 4.0.* International Function Point Users Group, Westerville, OH, January 1994.

[130] International Function Point Users Group. *COSMIC - FFP Measurement Manual*, 2.2 edition, 2003.

[131] D. R. Jeffery and G. C. Low. Calibrating estimation tools for software development. *Software Engineering Journal*, pages 215–221, 1990.

[132] D. R. Jeffery, G. C. Low, and M. Barnes. A comparison of Function Point counting techniques. *IEEE Transaction of Software Engineering*, 19(5):529–532, 1993.

[133] R. Jeffery, M. Ruhe, and I. Wieczorek. A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data. *Information & Software Technology*, 42(14):1009–1016, 2000.

[134] R. Jeffery, M. Ruhe, and I. Wieczorek. Using public domain metrics to estimate software development effort. In *Proceeding of the 7th METRICS 2001*, pages 239–247, 2001.

[135] R. Jeffery and J. Stathis. Function point sizing: Structure, validity and applicability. *Empirical Software Engineering*, 1(1):11–30, 1996.

[136] R. W. Jensen. A comparison of the Jensen and COCOMO schedule and cost estimation models. In *Proceedings of the International Society of Parametric Analysis*, pages 96–106, 1984.

[137] R. W. Jensen, G. C. Low, and M. Barnes. An improved macro level software development resource estimation model. In *Proceedings of the 5th ISPA Conference*, pages 88–92, 1983.

[138] R. L. Jenson and J. W. Bartley. Parametric estimation of programming effort: An object-oriented model. *Journal of Systems and Software*, 15(2):107–114, 1991.

[139] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, 1986.

[140] C. Jones. Measuring programming quality and productivity. *IBM Systems Journal*, 17(1):39–63, 1978.

[141] C. Jones. *Programming Productivity.* McGraw Hill, 1986.

[142] C. Jones. *A Short History of Function Points and Feature Points.* Software PRoductivity Research Inc., Burlington, Mass, 1986.

[143] C. Jones. *Applied Software Measurement: Assuring Productivity and Quality.* McGraw Hill, 1991.

[144] C. Jones. *Estimating Software Costs.* McGraw-Hill, New York, 1998.

[145] C. Jones. Sizing up software. *The Scientific American Magazine*, pages 74–79, 1998.

[146] C. Jones. *Software Assessments, Benchmarks, and Best Practices.* Addison Wesley Longman, Inc., 2000.

[147] C. Jones. Software cost estimation in 2002. *CrossTalk, The Journal of Defense Software Engineering*, June, 2002.

[148] M. Jørgensen. Experience with the accuracy of software maintenance task effort prediction models. *IEEE Transactions on Software Engineering*, 21(8):674–681, 1995.

[149] M. Jørgensen. Comments on 'a simulation tool for efficient analogy based cost estimation', by L. Angelis and I. Stamelos, published in Empirical Software Engineering, 5, 35-68 (2000). *Empirical Software Engineering*, 7(4):345–374, 2002.

[150] M. Jørgensen. Realism in assessment of effort estimation uncertainty: It matters how you ask. *IEEE Transactions on Software Engineering*, 30(4):209–217, 2004.

[151] M. Jørgensen. Regression models of software development effort estimation accuracy and bias. *Empirical Software Engineering*, 9(4):297–314, 2004.

[152] M. Jørgensen and K. Moløkken. Combination of software development effort prediction intervals: Why, when and how? In *Proceedings of the 14th International Conference on Software Engineering & Knowledge Engineering (SEKE)*, pages 425–428, 2002.

[153] M. Jørgensen and K. Moløkken. Eliminating over-confidence in software development effort estimates. In *Proceedings of the Product Focused Software Process Improvement, 5th International Conference (PROFES)*, pages 174–184, 2004.

[154] M. Jørgensen and K. Moløkken. How large are software cost overruns? critical comments on the standish groups chaos reports. Accepted for publication in Information and Software Technology, 2005.

[155] C. M. Judd, E. R. Smith, and L. H. Kidder. *Research Methods and Social Relations.* Harcourt Brace Jovanovich College Publishers, USA, 6th edition, 1991.

[156] S. K. Kachigan. *Multivariate Statistical Analysis, A Conceptual Introduction.* Radius Press, New York, 2nd edition, 1991.

[157] G. F. Kadoda, M. Cartwright, L. Chen, and M. J. Shepperd. Experiences using case-based reasoning to predict software project effort. In *Proceedings of the 4th International Conference on Empirical Assessment and Evaluation in Software Engineering*, Keele University, UK, 17-19 April 2000.

[158] N. Karunanithi, D. Whitley, and Y. K.Malaiya. Using neural networks in reliability prediction. *IEEE Software*, 9(4):53–59, 1992.

[159] R. Kauffman and R. Kumar. Modelling estimation expertise in object based icase environments. Technical report, Stern School of Business Report, New York University, 1991.

[160] C. F. Kemerer. An empirical validation of software cost estimation models. *Communication on the ACM*, 30(5):416–429, 1987.

[161] C. F. Kemerer. Metrics for Object-Oriented software: A retrospective. In *Proceedings of the 6th International Symposium on Software Metrics*, Boca Raton, Florida, November 4-6 1999.

[162] H. C. Kennedy, C. Chinniah, P. Bradbeer, and L. Morss. The construction and evaluation of decision trees: A comparison of evolutionary and concept learning methods. In D. Corne and J. L. Shapiro, editors, *Evolutionary Computing. AISB International Workshop.* Springer-Verlag, Manchester, UK, April 7-8 1997.

[163] T. M. Khoshoftaar, R. M. Szabo, and P. J. Guasti. Exploring the behavior of 'Neural Network Software' quality models. *IEEE Software Engineering Journal*, 10(3):89–96, 1995.

[164] C. Kirsopp and M. J. Shepperd. Case and feature subset selection for cbr-based software project effort prediction. In *Proceedings of the 22nd SGAI International Conference on Knowledge Based Systems & Applied Artificial Intelligence*, Cambridge, UK, 2002.

[165] C. Kirsopp and M. J. Shepperd. Making inferences with small numbers of training sets. *IEEE Proceedings - Software Engineering*, 149(5):123–130, 2002.

[166] C. Kirsopp, M. J. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction. In *Proceedings of the GECCO 2002*, pages 1367–1374, 2002.

[167] B. A. Kitchenham. Empirical studies of assumptions that underlie software cost-estimation models. *Information and software technology*, 32(4):211–218, 1992.

[168] B. A. Kitchenham. A procedure for analyzing unbalanced datasets. *IEEE Transactions on Software Engineering*, 24(4):278–301, 1998.

[169] B. A. Kitchenham. A case study of maintenance estimation accuracy. *Journal of Systems and Softwar*, 64(1):57–77, 2002.

[170] B. A. Kitchenham, S. G. MacDonell, L. Pickard, and M. J. Shepperd. What accuracy statistics really measure. *IEEE Proceedings Software*, 148(3):81–85, 2001.

[171] B. A. Kitchenham, S. G. MacDonell, L. M. Pickard, and M. J. Shepperd. Assessing prediction systems. working paper, University of Keele, 2000.

[172] B. A. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. E. Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, 2002.

[173] B. A. Kitchenham and N. R. Taylor. Software cost models. *ICL Technical Journal*, 4(1):73–102, May 1984.

[174] B. A. Kitchenham and N. R. Taylor. Software projects development cost estimation. *Journal of Systems and Software*, 5(4):267–278, 1985.

[175] G. J. Knafl and C. Gonzales. An evaluation of software cost models. Technical report, DePaul University, Chicago, U.S., 1998. http://facweb.cs.depaul.edu/research/TechReports/TR98-005.doc.

[176] D. E. Knuth. Structured programming with goto statements. *Computing Surveys*, 6(4):261–301, 1974.

[177] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 223–228, 1995.

[178] P. A. Kok, B. A. Kitchenham, and J. Kirakowski. The mermaid approach to software cost estimation. In *Proceedings of the ESPRIT'90 Technical Week*, pages 296–314, 1990.

[179] J. L. Kolodner. *Case Based Reasoning.* Morgan Kaufmann, 1993.

[180] J. Kusters, M. Genuchten, and F. Heemstra. Are software cost-estimation models accurate? *Information and Software Technology*, 32(3):187–190, 1990.

[181] J. Lassez, van der K. J. Sheoherd, and C. Lassez. Critical examination of software science. *Journal of Systems and Software*, 2(2):105–112, 1981.

[182] A. L. Lederer and J. Prasad. Nine management guideline for better cost estimating. *Communications of the ACM*, 35(2):51–59, 1992.

[183] A. L. Lederer and J. Prasad. Causes of inaccurate software development cost estimations. *Journal of Systems and Software*, 31(2):125–134, 1995.

[184] P. M. Lee. *Bayesian Statistics - An Introduction.* Oxford University Press Inc., Co-published in the USA, 198 Madison Avenue, New York, 2nd edition, 1997.

[185] S. Lei and M. R. Smith. Evaluation of several nonparametric bootstrap methods to estimate confidence intervals for software metrics. *IEEE Transactions on Software Engineering*, 29(11):996–1004, 2003.

[186] W. Li and S. Henry. Object-oriented metrics that predict maintainability. In *Proceedings of the 6th International Symposium on Software Metrics*, Boca Raton, Florida, November 4-6 1999.

[187] Q. Liu and R. C. Mintram. Preliminary data analysis methods in software estimation. In *Proceedings of the BCS 2004 SQM and INSPIRE Conferences*, pages 273–298, Canterbury, UK, ISBN 1-902505-56-5, April 5-7 2004.

[188] Q. Liu and R. C. Mintram. Preliminary data analysis methods in software estimation. *Software Quality Journal*, 13(1):91–115, 2005.

[189] Q. Liu and R. C. Mintram. The problem of estimating the duration of software development projects. In *Proceedings of the BCS 2005 SQM and INSPIRE Conferences*, pages 293–299, Cheltenham, UK, ISBN:1-902505-67-0, March 21-23 2005.

[190] Q. Liu, R. C. Mintram, and J. Vincent. Evaluation of cost estimation models. In *Proceedings of the International Conference on Computer Science and Information Systems*, Athens, Greece, 2005. Athens Institute For Education And Research.

[191] Q. Liu, J. Vincent, and R. C. Mintram. Issues in quality management of research software. In *Proceedings of the BCS 2003 SQM and INSPIRE Conferences*, pages 151–162, Glasgow, UK, ISBN 1-902505-54-9, April 5-7 2004.

[192] C. Lokan. An empirical study of the correlations between Function Point elements. In *Proceedings of the 6th International METRICS Symposium*, pages 200–206, November 1999.

[193] S. Long. *Regression Models for Categorical and Limited Dependent Variables.* Advanced Quantitative Techniques, Sage Publications, 1997.

[194] M. Lorenz. *Object-Oriented Software Development: A Practical Guide.* PTR Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[195] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics: A Practical Guide.* PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[196] S. MacDonell and M. J. Shepperd. Combining techniques to optimize effort predictions in software project management. *Journal of Systems and Software*, 66(2):91–98, 2003.

[197] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster. An investigation of machine learning based prediction systems. *Journal of Systems Software*, 53(1):23–29, 2000.

[198] C. Mair and M. J. Shepperd. Making software cost data available for meta-analysis. In *Proceedings of the International Conference on Empirical Assessment of Software Engineering*, Edinburgh, May 2004.

[199] M. Mandischer. Genetic optimization and representation of neural networks. In *Proceedings of the 4th Australian Conference on Neural Networks*, pages 122–125, 1993.

[200] R. Marouane and A. Mili. Economics of software project management in Tunisia: Basic Tucomo. *Information and Software Technology*, 31(5):251–257, 1989.

[201] R. Martin. Evaluation of current software costing tools. *ACM SISOFT Software Engineering Notes*, 13(3):49–51, 1988.

[202] K. Maxwell, L. Van Wassenhove, and S. Dutta. A software development productivity of european space, military and industrial applications. *IEEE Transactions on Software Engineering*, 22(10):704–718, 1996.

[203] K. D. Maxwell. *Applied Statistics for Software Managers.* Pearsson Education Inc., UpperSaddle River, New Jersey, 2002.

[204] T. McCabe. A software complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.

[205] E. Mendes, S. Counsell, and N. Mosley. Measurement and effort prediction of Web applications. In *Proceedings of the 2nd ICSE Workshop on Web Engineering*, 2001.

[206] E. Mendes, N. Mosley, and S. Counsell. Web metrics - estimating design and authoring effort. In *IEEE Multimedia, Special Issue on Web Engineering*, pages 50–57, January/March 2001.

[207] E. Mendes, I. Waston, C. Triggs, N. Mosley, and S. Counsell. A comparison of length, complexity & functionality as size measures for predicting web design & authoring effort. In *Proceedings of the IEEE Metrics Symposium*. IEEE-CS Press, 2002.

[208] E. Mendes, I. Waston, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[209] J. Mingers. An empirical comparison of selection measures for decision tree induction. *Machine Learning*, 3, 1989.

[210] Y. Miyazaki and K. Mori. COCOMO evaluation and tailoring. In *Proceedings of the 8th International Conference on Software Engineering*, pages 292–299, 1985.

[211] Y. Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada. Method to estimate parameter values in software prediction models. *Information and Software Technology*, 33(3):239–243, 1991.

[212] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki. Robust regression for developing software estimation models. *Journal of Systems and Software*, 27(1):3–16, 1994.

[213] S. Mohanty. Software cost estimation: Present and future. *Software Practice and Experience*, 11(2):103–121, 1981.

[214] K. Moløkken, M. Jørgensen, and S. S. Tanilkan. A survey on software estimation in the norwegian industry. Technical report, Department of Informatics, University of Oslo, Norway, 2003.

[215] K. Moløkken-Østvold and M. Jørgensen. Expert estimation of Web-development projects: Are software professionals in technical roles more optimistic than those in non-technical roles? *Empirical Software Engineering*, 10(1):7–30, 2005.

[216] P. B. Moranda. Acm computing surveys. *Surveyors forum*, 10(4):503–504, 1978.

[217] J. Moses. Bayesain probability distribution for assessing measurement of subjective software attributes. *Information and Software Technology*, 42(8):533–546, 2000.

[218] J. Moses and J. Clifford. Support for effort estimation in small software companies using bayesian inference. In *Proceedings of the EuroSPI'2000, Practical and Innovation Based Software Process Improvement to Prepare for the New Millennium*, Copenhagen, Denmark, 7-9 November 2000.

[219] J. Moses and M. Farrow. Assessing variation in development effort consistency using a data source with missing data. *Software Quality Journal*, 13(1):71–89, 2005.

[220] T. Mukhopadhyay and S. Kekre. Software effort models for early estimation of process control applications. *IEEE Transactions on Software Engineering*, 18(10):915–923, 1993.

[221] T. Mukhopadhyay and S. S. Vicinanzat. Examining the feasibility of a Case-Based Reasoning model for software effort estimation. *MIS Quarterly*, 16(2):155–171, 1992.

[222] J. D. Musa, A. Iannino, and K. Okumoto. *Software Reliability: Measurement, Prediction, Application.* Mcgraw-Hill Series in Software Engineering and Technology, 1987.

[223] I. Myrtveit and E. Stensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering*, 25(4):510–525, 1999.

[224] I. Myrtveit, E. Stensrud, and U. H. Olsson. Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods. *IEEE Transactions on Software Engineering*, 27(11):999–1013, 2001.

[225] M. Neil and N. Fenton. Applying Bayesian Belief Networks to critical systems assessment. *Safety Critical Systems Club Newsletter*, 8(3):10–13, 1999.

[226] M. Neil, N. Fenton, and L. Nielsen. Building large-scale Bayesian Networks. *The Knowledge Engineering Review*, 15(3):257–284, 2000.

[227] M. Neil, N. Fenton S. Forey, and R. Harris. Using Bayesian Belief Networks to predict the reliability of military vehicles. *IEEE Computing and Control Engineering*, 12(1):11–20, 2001.

[228] E. A. Nelson. *Management Handbook for the Estimation of Computer Programming Costs.* Systems Development Corp., 1966.

[229] P. V. Norden. Useful tools for project management. In Dean B V., editor, *Operations Research in Research and Development.* John Wiley & Sons, New York, 1963.

[230] S. F. Ochoa, M. C. Bastarrica, and G. Parra. Estimating the development effort of web project in chile. In *Proceedings of the 1st Latin American Web Congress (LA-WEB 2003).* IEEE Computer Society, 2003.

[231] S. Oligny, P. Bourque, and A. Abran. An empirical assessment of project duration models in software engineering. In *Proceedings of the 8th European Software Control and Metrics Conference (ESCOM'97)*, page 9, Adrian Cowderoy, Berlin, Germny, 1997.

[232] S. Oligny, P. Bourque, A. Abran, and B. Fournier. Refining empirical models of project duration in software engineering. In *Proceedings IFPUG 1997 Fall Conference*, Scottsdale, AZ, 1997. International Function Point Users Group.

[233] L. M. Otto. The early days of software metrics: Looking back after 20 years. In A. Melton, editor, *Software Measurement: Understanding Software Engineering*, pages 7–25. International Thompson Press, 1995.

[234] G. Oulsname. Cyclomatic numbers do not measure complexity of unstructured programs. *Information Processing Letters*, pages 207–211, December 1979.

[235] R. Park. The central equations of the price software cost model. In *the 4th COCOMO Users Group Meeting*, 1988.

[236] G. N. Parkinson. *Parkinson's Law and Other Studies in Administration.* Houghton-Miffin, Boston, 1957.

[237] S. L. Pfleeger, R. Jeffery, B. Curtis, and B. Kitchenham. Status report on software measurement. *IEEE Software*, 14(2):33–43, March/April 1997.

[238] L. M. Pickard, B. A. Kitchenham, and S. J. Linkman. An investigation of analysis techniques for software data sets. In *Proceedings of the 6th IEEE International Software Metrics Symposium (METRICS 2001),4-6 April*, page 130, Los Alamitos, California, 1999. IEEE Computer Society.

[239] K. Pillai and V. S. Nair Sukumaran. A model for software development effort and cost estimation. *IEEE Transactions on Software Engineering*, 23(8):485–497, 1997.

[240] A. A. Porter and R. W. Selby. Generating hierarchical system descriptions for software error localization. In *Proceedings of the 2nd Workshop Software Testing, Verification and Analysis.* CS Press, Los Alamitos, Calif., 1988.

[241] A. A. Porter and R .W .Selby. Learning from examples: Generation and evaluation of decision trees for software resource analysis. *IEEE Transactions on Software Engineering*, 14(12):1743–1757, 1988.

[242] A. A. Porter and R. W. Selby. Classification tree analysis using the amadeus measurement and empirical analysis system. In *Proc.14th Software Engineering Workshop.* NASA Goddard Space Flight Centre, Greenbelt, Md., 1989.

[243] A. A. Porter and R. W. Selby. Software metrics classification trees help guide the maintenance of large-scale systems. In *Proceedings of the Conference of Software Maintenance.* CS Press, Los Alamitos, California, 1989.

[244] A. A. Porter and R. W. Selby. Empirically guided software development using metric-based classification trees. *IEEE Software*, 7(2):46–54, 1990.

[245] A. A. Porter and R. W. Selby. Evaluating techniques for generating metric-based classification trees. *Journal of Systems Software*, 12(3):209–218, 1990.

[246] R. Premraj, M. J. Shepperd, and M. Cartwright. Meta-data to guide retrieval in CBR for software cost prediction,. In *UK CBR Workshop*, Cambridge, 2003.

[247] R. S. Pressman. *Software Engineering: A Practitioner's Approach.* The McGraw-Hill Companies, Inc., 4th edition, 1997.

[248] M. J. Prietula, P. J. Feltovich, and F. M. Marchak. A heuristic framework for assessing factors influencing knowledge acquisition. In *Proceedings of the 22nd Hawaii International Conference on System Science.* IEEE Computer Society Press, 1989.

[249] M. J. Prietula, S. Vicinanza Steven, and T. Mukhopadhyay. Software-effort estimation with a case-based reasoner. *JETAI: Journal of Experimental and Theoretical Artificial Intelligence*, 8(3-4):341–363, 1996.

[250] L. H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transaction Software Engineering*, 4(4):345–361, 1978.

[251] L. H. Putnam and W. Myers. *Measures for Excellence: Reliable Software on Time, Within Budget.* Yourdon Press, 1992.

[252] L. H. Putnam and R. W. Wolverton. *Quantitative management: Software Cost Estimating.* IEEE Computer Society, 1977.

[253] J. R. Quinland. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[254] D. Reifer. Web development: Estimating quick-to-market software. *IEEE Software*, 17(6):57–64, 2000.

[255] D. Reifer. Let the numbers do the talking. *CrossTalk, The Journal of Defense Software Engineering*, March, 2002.

[256] D. Reifer. Ten deadly risks in internet and intranet software development. *IEEE Software*, 18(2):12–14, 2002.

[257] F. S. Roberts. *Measurement Theory with Applications to Decision Making, Utility, and the Social Sciences. Encyclopedia of Mathematics and its Applications.* Addison Wesley Publishing Company, 1979.

[258] D. Rocacher. *Metrics Definitions for Smalltalk.* ESPRIT Project 1257, MUSE WP9A, 1988.

[259] L. Rosenberg, R. Staplp, and A. Gallo. Applying Object-Oriented metrics. In *Proceedings of the 6th International Symposium on Software Metrics*, Boca Raton, Florida, November 4-6 1999.

[260] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1987.

[261] R. J. Rubey and R. D. Hartwick. Quantitative measurement program quality. In *Proceedings of the ACM, National Computer Conference*, pages 671–677, 1968.

[262] H. Rubin. Software process maturity: Measuring its impact on productivity and quality. *Proceedings of the 15th International Conference on Software Engineering*, pages 468–476, 1993.

[263] H. A. Rubin. Macro and micro-estimation of software development parameters: The estimacs system. In *Proceedings of the SOFTFAIR: A Conference on Software Development Tools, Techniques and Alternatives*, pages 109–118, New York, 1983. IEEE Press.

[264] H. A. Rubin. A comparison of cost estimation tools. In *Proceedings of the 8th International Conference of Software Engineering*, pages 174–181. IEEE Press, 1985.

[265] M. Ruhe, D. R. Ross, and I. Wieczorek. Cost estimation for Web applications. In *ICSE2003*, pages 285–294, 2003.

[266] M. Ruhe, D. R. Ross, and I. Wieczorek. Using web objects for estimating software development effort for Web applications. In *IEEE METRICS 2003*, page 30, 2003.

[267] T. L. Saaty. *The Analytic Hierarchy Process.* New York: McGraw-Hill, 1980.

[268] T. L. Saaty. Highlights and critical points in the theory and application of the analytic hierarchy process. *European Journal Operations Research*, 74(3):426–447, 1994.

[269] B. Samson, D. Ellison, and P. Dugard. Software cost estimation using an albus perceptron (cmac). *Information and Software Technology*, 39(1):55–60, 1997.

[270] C. Sauer and C. Cuthbertson. The sate of IT project management in the UK 2002-2003. The final report from the computer weekly project/programme management survey, Templeton College, University of Oxford, 2003.

[271] R. Selby and A. Porter. Learning from examples: Generation and evaluation of decision tees for software resource analysis. *IEEE Transactions on Software Engineering*, 14(12):1743–1757, 1988.

[272] C. Serluca. An investigation into software effort estimation using a backpropagation neural network. MSc. Thesis, Bournemouth University, 1995.

[273] Y. Shan, R. I. McKay, C. J. Lokan, and D. L. Essam. Software project effort estimation using genetic programming. In *Proceedings of the International Conference on Communications Circuits and Systems*, 2002.

[274] M. Shaw. What makes good research in software engineering? In *Proceedings of the European Joint Conference of Theory and Practice of Software (ETAPS 2002)*, volume 4, pages 1–7, 2002.

[275] V. Y. Shen, S. D. Conte, and H. E. Dunsmore. Software science revisited: A critical analysis of the theory and its empirical support. *IEEE Transactions on Software Engineering*, 9(2):155–165, 1983.

[276] M. Shepperd and M. Cartwright. Predicting with sparse data. *IEEE Transactions on Software Engineering*, 27(11):987–998, 2001.

[277] M. Shepperd, M. Cartwright, and G. Kadoda. On building prediction systems for software engineers. *Empirical Software Engineering*, 5(3):175–182, 2000.

[278] M. Shepperd and G. Kadoda. Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering*, 27(11):1014–1022, 2001.

[279] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12):736–743, 1997.

[280] M. Shepperd, C. Schofield, and B. A. Kitchenham. Effort estimation using analogy. In *Proceedings of the 18th International Conference on Software Engieering ICSE-18*, pages 170–175, 1996.

[281] M. Shin and A. L. Goel. Empirical data modeling in software engineering using radial basis functions. *IEEE Transactions on Software Engineering*, 26(6):567–576, 2000.

[282] B. G. Silverman. The use of analogs in the innovation process: A software engineering protocol analysis. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-I5(1):30–44, 1985.

[283] A. E. Smith and A. K. Mason. Cost estimation predictive modeling: Regression versus neural network. *The Engineering Economist*, 42(2):137–161, 1997.

[284] IEEE Computer Society. *IEEE Standard for a Software Quality Metrics Methodology. IEEE Standard 1061.* IEEE Standard Board, IEEE Standard Office, 445 Hord Lane, P.O.Box 1331, Piscataway, NJ08855-1331, 1993.

[285] I. Sommerville. *Software Engineering.* Addison Wesley, 6th edition, 2001.

[286] K. Srinivasan and D. Fisher. Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21(2):126–137, 1995.

[287] D. St-Pierre, M. Maya, A. Abran, J. Desharnais, and P. Bourque. Full function points: Counting practice manual. Technical report 1997-04, University of Quebec at Montreal, 1997.

[288] R. G. Staudte and S. J. Sheather. *Robust Estimation and Testing, Wiley Series in Probability and Mathematical Statistics.* John Wiley & Sons, 1990.

[289] E. Stensrud. Alternative approaches to effort prediction of erp projects. *Journal of Information & Software Technology*, 43(7):413–423, 2001.

[290] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. An empirical validation of the relationship between the magnitude of relative error and project size. In *Proceedings of the 8th IEEE International Symposium on Software Metrics (METRICS 2002)*, pages 3–12, Los Alamitos, CA, 2002. IEEE Computer Society Press.

[291] E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. A further empirical investigation of the relationship between mre and project size. *Empirical Software Engineering*, 8(2):139–161, 2003.

[292] E. Stensrud and I. Myrtveit. Human performance estimation with analogy and regression models. In *Proceedings of the IEEE International Symposium on Software Metrics (METRICS 1998)*, pages 205–213. IEEE Computer Society Press, 1998.

[293] A. Delurgio Stephen. *Forecasting Principles and Application.* McGraw-Hill, Irwin, 1997.

[294] H. Kan Stephen. *Metrics and Models in Software Quality Engineering.* Peatson Education Inc., 2nd edition, 2003.

[295] M. Stone. Cross-validation choice and assessment of statistic predictions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, B-36(1):111–147, 1974.

[296] K. Strike, K. El Emam, and N. Madhavji. Software cost estimation with incomplete data. *IEEE Transactions on Software Engineering*, 27(10):890–908, 2001.

[297] R. D. Stutzke. Software estimation: A survey. *CrossTalk, The Journal of Defense Software Engineering*, May 1996.

[298] C. R. Symons. *Software Sizing and Estimating: Mark II Function point Analysis.* John Wiley, 1991.

[299] J. Tian. Better reliability assessment and prediction through data clustering. *IEEE Transactions on Software Engineering*, 28(10):997–1007, 2002.

[300] J. Tou and R. Gonzalez. *Pattern Recognition Principles.* MA: Addison-Wesley, Reading, 1974.

[301] S. Treble and N. Douglas. *Sizing and Estimating Software in Practice, Making MKII Function Point Work.* McGraw-Hill book company, 1995.

[302] A. R. Venkatachalam and McConnell Hall. Software cost estimation using artificial neural networks. In *Proceedings of the 1993 International Joint Conference on Neural Networks*, pages 987–990, 1993.

[303] S. S. Vicinanza, T. Mukhopadhyay, and M. J. Prietula. Software-effort estimation: An exploratory study of expert performance. *Information Systems Research*, 2(4):243–262, 1991.

[304] S. S. Vicinanza, M. J. Prietula, and T. Mukhopadhyay. Case-based reasoning in software effort estimation. In *Proceedings of the 11th International Conference on Information Systems*, pages 149–158, 1990.

[305] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hobn, and Y. Liu. Productivity factors and programming environments. In *Proceedings of the International Conference on Software Engineering*, pages 143–152, 1984.

[306] F. Walkerden and R. Jeffery. An empirical study of Analogy-based software effort estimation. *Empirical Software Engineering*, 4(2):135–158, 1999.

[307] C. E. Walston and C. P. Felix. A method of programming measurement and estimation. *IBM Systems Journal*, 16(1):54–73, 1977.

[308] L. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, Engelwood Cliffs, 1994.

[309] R. D. H. Warburton. A method of programming measurement and estimation. *IEEE Transactions on Software Engineering*, 9(5):562–569, 1983.

[310] I. Watson. *Applying Case-Based Reasoning: Techniques for Enterpriser Systems.* Morgan Kaufmann Publishers, 1997.

[311] G. Wittig and G. Finnie. Estimating software development effort with connectionist models. *Information and Software Technology*, 39(7):469–476, 1997.

[312] R. W. Wolverton. The cost of developing large-scale software. *IEEE Transactions on Computer*, 23(6):615–636, June 1974.

[313] C. D. Wrigley and A. S. Dexter. Software development estimation models: A review and critique. In *Proceedings of the ASAC Conference*, pages 125–138. University of Toronto, 1987.

[314] E. N. Yourdon. *Techniques of Program Structure and Design.* Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[315] L. A. Zadeh. Fuzzy set. *Information and Control*, 8(3):338–353, 1965.

[316] L. A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–84, 1994.

[317] M. Zelkowitz, R. Yeh, R. Hamlet, J. Gannon, and V. Basili. Software engineering practices in the us and japan. *IEEE Computer*, pages 57–66, 1984.

[318] H. Zuse. *Software Complexity: Measures and Methods.* De Gruyter, Berlin, 1991.

[319] H. Zuse. *A Framework of Software Measurement.* Walter de Gruyter & Co., Berlin, 1997.

Part V

APPENDIX

Appendix A

Abbreviations and Acronyms

Abbreviation	Stands for
AHP	Analytic Hierarchy Process
ANGEL	Analogy-Based Estimation and response tools
ANN	Artificial Neural Networks
ANOVA	Analysis of Variance
BBN	Bayesian Belief Network
BMMRE	The Balanced MMRE
CART	Classification and Regression Trees
CBR	Case-Based Reasoning
CMM	Capability Maturity Model
COBRA	Cost estimation, Benchmarking and Risk Analysis
COCOMO	the Constructive Cost Model
CWADEE	Chilean Web Application Development Effort Estimation
DE	Differential Evolution
DT	Decision Trees
FFP	Full Function Point
FIML	Full Information Maximum Likelihood
FPA	Function Point Analysis
FP	Function Point
GA	Genetic Algorithms
GGGP	Grammar Guided Genetic Programming
GP	Genetic Programming
IFPUG	the International Function Point Users Group
ISBSG	International Benchmarking Standards Group
KNN	K-Nearest Neighbour
LBRS	Least-squares of Balanced Relative Errors
LD	Listwise Deletion
LIRS	Least-squares of Inverted Balanced Relative Errors
LMS	Least Median of Squares
LOC	Lines of Code
LSD	Logarithmic Standard Deviation
MACS	Management and Computer Services
MBI	Manpower Buildup Index
MBP	Manpower Buildup Parameter
MCAR	Missing Completely At Random
MDT	Missing Data Techniques
MdMRE	The Median of MRE
MI	Mean Imputation
ML	Machine Learning
MLP	Multi-Layer Perceptron
MMRE	Mean Magnitude of the Relative Error
MRE	Magnitude of Relative Error
NATO	North Atlantic Treaty Organization
OLS	Ordinary Least-Square Regression
OO	Object-Oriented
OOTC	Object Oriented Technology Council
OSR	Optimized Set Reduction
PCA	Principal Components Analysis
PI	Productivity Index
PP	Productivity Parameter
QA	Quality Assurance
RE	Relative Error
RI	Rule Induction
RL	Reinforcement Learning
RSD	Relative Standard Deviation
RR	Robust Regression
SCF	Schedule Compression Factor
SD	Standard Deviation
SDM	Software Development Metrics
SE	Software Engineering
SLIM	Software LIfe-cycle Model
SLOC	Thousand Lines of Code
SPR	Software Productivity Research
SRPI	Similar Response Pattern Imputation
SWR	StepWise Regression
TCA	Technical Complexity Adjustment
TCF	Technical Complexity Factor
TUCOMO	Tunisian Cost Model
UFC	Unadjusted Function-point Counts
UML	Unified Modelling Language
USDP	Unified Software Development Process
WBS	Work Breakdown Structure

Table A.1: Abbreviations and Acronyms

Appendix B

ISBSG R9 Data and Experiment Details

Abbreviation	Stands For	Type	Description
Size	Functional Size (Function Points)	Ratio	The unadjusted function point count (before any adjustment by a Value Adjustment Factor if used). This may be reported in different units depending on the FSM Method.
Effort	Summary Work Effort	Ratio	Provides the total effort in hours recorded against the project.
Duration	Project Elapsed Time	Ratio	Total elapsed time for the project in calendar months.
CoutApp	Count Approach	Nominal	A description of the technique used to size the project. For most projects in the ISBSG repository this is the Functional Size Measurement Method (FSM Method) used to measure the functional size (e.g. IFPUG, MARK II, NESMA, COSMIC-FFP etc.).
DevType	Development Type	Nominal	This field describes whether the development was a new development, enhancement or re-development.
OrgType	Organization Type	Nominal	This identifies the type of organization that submitted the project. (e.g.: Banking, Manufacturing, Retail).
BusiType	Business Type	Nominal	This identifies the type of business area being addressed by the project where this is different to the organization type. (e.g.: Manufacturing, Personnel, Finance).
AppType	Application Type	Nominal	This identifies the type of application being addressed by the project. (e.g.: information system, transaction/production system, process control.)
Architecture	Architecture	Nominal	A derived attribute for the project to indicate if the application is Stand alone, Multi-tier, Client server, or Multi-tier with web public interface.
DevPlatform	Development Platform	Nominal	Defines the primary development platform, (as determined by the operating system used). Each project is classified as: PC, Mid Range, Main Frame or Multi platform.
LangType	Language Type	Nominal	Defines the language type used for the project: e.g. 3GL, 4GL, Application Generator etc.
PriLang	Primary Programming Language	Nominal	The primary language used for the development: JAVA, C++, PL/1, Natural, Cobol etc.
MDBS	The Primary DBMS Used	Nominal	Where known, this is the primary technology database used to build or enhance the software (i.e. that used for most of the build effort), otherwise (if known) it is whether the project used a DBMS.
UMethod	Used Methodology	Nominal	States whether a development methodology was used by the development team to build the software.
RMethod	How Methodology Acquired	Nominal	Describes whether the development methodology was purchased or developed in-house, or a combination of these.
Rlevel	Resource Level	Nominal	Data is collected about the people whose time is included in the work effort data reported. Four levels are identified in the data collection instrument. 1 = development team effort (e.g., project team, project management, project administration) 2 = development team support (e.g., database administration, data administration, quality assurance, data security, standards support, audit & control, technical support) 3 = computer operations involvement (e.g., software support, hardware support, information centre support, computer operators, network administration) 4 = end users or clients (e.g., user liaisons, user training time, application users and/or clients).

Table B.1: Case Study A: Project Attributes Description

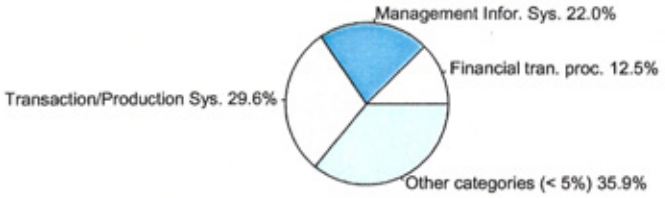

Figure B.1: Case Study A: Pie Chart of Main Categories of App-Type

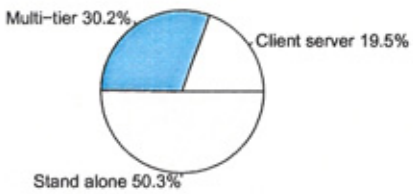

Figure B.2: Case Study A: Pie Chart of Architecture

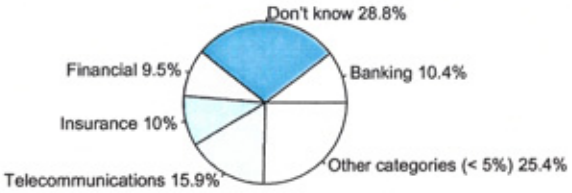

Figure B.3: Case Study A: Pie Chart of Main Categories of Busi-Type

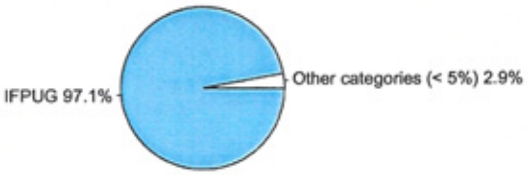

Figure B.4: Case Study A: Pie Chart of Main Categories of CountAPP

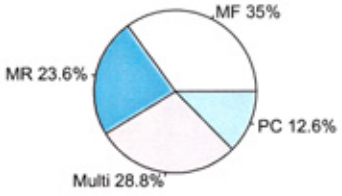

Figure B.5: Case Study A: Pie Chart of Main Categories of Dev-Platform

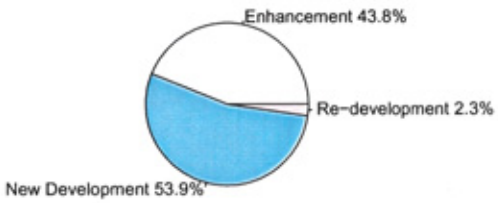

Figure B.6: Case Study A: Pie Chart of Main Categories of Dev-Type

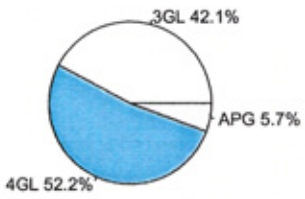

Figure B.7: Case Study A: Pie Chart of Main Categories of Lang-Type

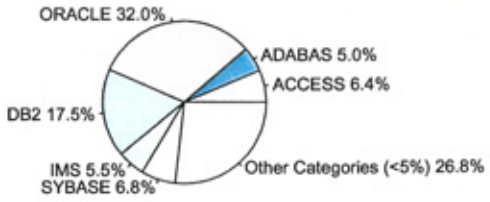

Figure B.8: Case Study A: Pie Chart of Main Categories of MDBS

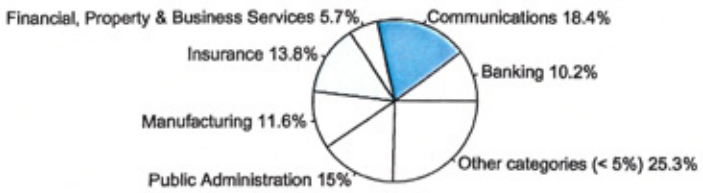

Figure B.9: Case Study A: Pie Chart of Main Categories of OrgType

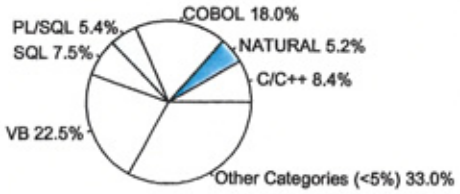

Figure B.10: Case Study A: Pie Chart of Main Categories of PriLang

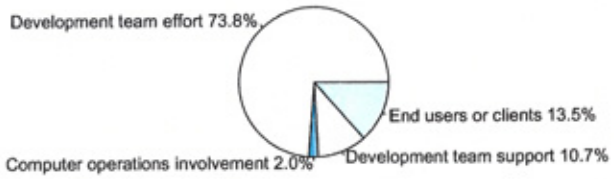

Figure B.11: Case Study A: Pie Chart of Main Categories of RLevel

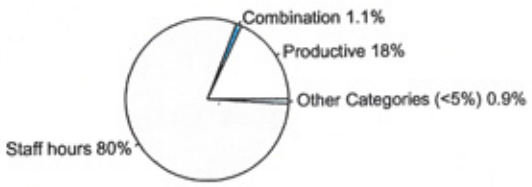

Figure B.12: Case Study A: Pie Chart of Main Categories of RMethod

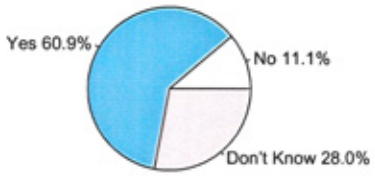

Figure B.13: Case Study A: Pie Chart of Main Categories of UMethod

Category Nr.	Application Type	Frequency	Percent
1	Administrative Support System	1	0.2
2	Advertising/Mailing Campaign	1	0.2
3	Business enabling service	1	0.2
4	Card Administration INCAS & Fault Assurance CAPRA	1	0.2
5	Case Management Study	1	0.2
6	Corporate Taxation	1	0.2
7	Customer billing/relationship management	10	1.8
8	Data Warehouse	1	0.2
9	Decision Support System	11	2.0
10	Document management	3	0.5
11	Electronic Data Interchange	16	2.9
12	Equipment Management	1	0.2
13	Executive Information System	11	2.0
14	Fault Tolerance	1	0.2
15	Financial transaction process/accounting	70	12.5
16	Imbedded software for machine control	3	0.5
17	information systems (Web)	4	0.7
18	Inventory Control	1	0.2
19	Job, case, incident, project management	4	0.7
20	Logistic or supply planning & control	6	1.1
21	Management Information System	126	22.5
22	Network Management	13	2.3
23	Not specified	1	0.2
24	Office Information System	19	3.4
25	other **	35	6.3
26	personal system	1	0.2
27	Process Control	11	2.0
28	purchasing	1	0.2
29	Real-time System	7	1.3
30	RECONCILIATION	1	0.2
31	Scientific	1	0.2
32	Security management	1	0.2
33	Stock control & order processing	11	2.0
34	System conversion	1	0.2
35	Technical Information System	1	0.2
36	tools or system	2	0.4
37	Trading	7	1.3
38	Transaction/Production System	167	29.8
39	Type 1 Function Point Counting Tool	1	0.2
40	Vehicle Systems Software	1	0.2
41	Workflow support & management	4	0.7
	Total	560	100.0

Table B.2: Case Study A: Frequency of AppType

Category Nr.	Architecture	Frequency	Percent
1	Client server	109	19.5
2	Multi-tier	169	30.2
3	Stand alone	282	50.3
	Total	560	100.0

Table B.3: Case Study A: Frequency of Architecture

Category Nr.	BusiType	Frequency	Percent
1	Accounting	9	1.6
2	Actuarial System	1	0.2
3	Administration	1	0.2
4	Architectural	1	0.2
5	Banking	58	10.4
6	Blood Bank	1	0.2
7	Case Management	1	0.2
8	Chartered flight ope	1	0.2
9	Claims Processing	1	0.2
10	Dispute Resolution	1	0.2
11	Distribution & Trans	2	0.4
12	don't know	161	28.8
13	Energy generation	1	0.2
14	Engineering	17	3.0
15	Financial	53	9.5
16	Fine Enforcement	8	1.4
17	Generate & Distribut	1	0.2
18	Government	1	0.2
19	Insurance	56	10.0
20	Inventory	5	0.9
21	Legal	6	1.1
22	Logistics	3	0.5
23	Manufacturing	22	3.9
24	Mining Production In	1	0.2
25	Network Management	1	0.2
26	Pension Funds Manage	8	1.4
27	Personnel	12	2.1
28	Procurement	1	0.2
29	Project management	1	0.2
30	Provide **	3	0.5
31	Public **	2	0.4
32	Purchasing	1	0.2
33	regulation	4	0.7
34	Regulatory agency	3	0.5
35	Research & Development	4	0.7
36	Sales & Marketing	12	2.1
37	Social Services	5	0.9
38	Telecommunications	89	15.9
39	TESTING	1	0.2
40	Valuation	1	0.2
	Total	560	100.0

Table B.4: Case Study A: Frequency of BusiType

Category Nr.	CountApp	Frequency	Percent
1	IFPUG	544	97.1
2	Albrecht	1	0.2
3	Feature Point	1	0.2
4	Mark II	6	1.1
5	NESMA	8	1.4
	Total	560	100.0

Table B.5: Case Study A: Frequency of CountApp

Category Nr.	DevPlatform	Frequency	Percent
1	MF	196	35.0
2	MR	132	23.6
3	Multi	161	28.8
4	PC	71	12.7
	Total	560	100.0

Table B.6: Case Study A: Frequency of DevPlatform

Category Nr.	DevType	Frequency	Percent
1	Enhancement	245	43.8
2	New Development	302	53.9
3	Re-development	13	2.3
	Total	560	100.0

Table B.7: Case Study A: Frequency of DevType

Category Nr.	LangType	Frequency	Percent
1	3GL	236	42.1
2	4GL	292	52.1
3	APG	32	5.7
	Total	560	100.0

Table B.8: Case Study A: Frequency of LangType

Category Nr.	MDBS	Frequency	Percent
1	ACCESS	36	6.4
2	ADABAS	28	5.0
3	CA-IDMS	1	0.2
4	ORACLE	179	32.0
5	DATACOM	4	0.7
6	DB2	98	17.5
7	DL/1	20	3.6
8	Domino	3	0.5
9	ENSCRIBE	1	0.2
10	Exchange	6	1.1
11	Flat File	1	0.2
12	FOXPRO	1	0.2
13	SQL	25	4.5
14	HYDRA	1	0.2
15	IBM	2	0.4
16	IDMS	12	2.1
17	IMS	31	5.5
18	INFORMIX	2	0.4
19	INGRES	3	0.5
20	ISAM	17	3.0
21	LOTUS NOTES	3	0.5
22	M204	4	0.7
23	Notes	4	0.7
24	OBJECTSTOR	6	1.1
25	Other	14	2.5
26	SYBASE	38	6.8
27	VSAM	3	0.5
28	RDB	9	1.6
29	RMS	4	0.7
30	PICK	2	0.4
31	Proprietery	1	0.2
32	Raima	1	0.2
	Total	560	100.0

Table B.9: Case Study A: Frequency of MDBS

Category Nr.	OrgType	Frequency	Percent
1	Aerospace / Automotive	11	2.0
2	Agriculture, Forestry, Fishing, Hunting	4	0.7
3	Any organization which counts function points	57	10.2
4	Banking	1	0.2
5	Business Services	3	0.5
6	Chemicals	6	1.1
7	Communications	103	18.4
8	Community Services	6	1.1
9	Computer Consultants	1	0.2
10	Computers & Software	5	0.9
11	Construction	8	1.4
12	Consultancy	1	0.2
13	Consumer Goods	2	0.4
14	Coronial Services	1	0.2
15	Defence	1	0.2
16	Developing global software solutions for Citibank	1	0.2
17	Distribution	1	0.2
18	Electricity, Gas, Water	25	4.5
19	Electronics	2	0.4
20	Energy	1	0.2
21	Financial, Property & Business Services	32	5.7
22	Food Processing	1	0.2
23	Government	10	1.8
24	Insurance	77	13.8
25	Manufacturing	65	11.6
26	Media	1	0.2
27	Medical and Health Care	9	1.6
28	Mining	5	0.9
29	Occupational Health and Safety	1	0.2
30	Other	1	0.2
31	Professional Services	5	0.9
32	Public Administration	84	15.0
33	Real Estate & Property Services	2	0.4
34	Recreation & Personnel Services	3	0.5
35	Transport & Storage	18	3.2
36	Wholesale & Retail Trade	6	1.1
	Total	560	100.0

Table B.10: Case Study A: Frequency of OrgType

Category Nr.	PriLang	Frequency	Percent
1	4GL	21	3.8
2	Access	23	4.1
3	BASIC	3	0.5
4	C/C++	47	8.4
5	COBOL	101	18.0
6	CSP	2	0.4
7	DELPHI	1	0.2
8	Develope	7	1.3
9	Domino S	3	0.5
10	EASYTRIE	1	0.2
11	HPS	10	1.8
12	IDEAL	4	0.7
13	IEF	1	0.2
14	JAVA	9	1.6
15	LEX	1	0.2
16	LOTUS NO	3	0.5
17	NATURAL	29	5.2
18	NOTES SC	7	1.3
19	OPO(Orac	1	0.2
20	ORACLE	27	4.8
21	OTHER 3G	2	0.4
22	OTHER Ap	9	1.6
23	OutlookV	4	0.7
24	PL/SQL	30	5.4
25	PowerBui	21	3.8
26	ProC	6	1.1
27	RALLY	2	0.4
28	SAP ABAP	2	0.4
29	SAS	2	0.4
30	Shell Sc	1	0.2
31	SQL	42	7.5
32	TELON	12	2.1
33	VB	126	22.5
	Total	560	100.0

Table B.11: Case Study A: Frequency of PriLang

Category Nr.	RLevel	Frequency	Percent
1	Development team effort	413	73.8
2	Development team support	60	10.7
3	Computer operations involvement	11	2.0
4	End users or clients	76	13.5
	Total	560	100.0

Table B.12: Case Study A: Frequency of RLevel

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	10.36	4.82	2.50	5.40	0.86	10.20	5.44	0.87	14213.08
2	13.75	5.36	2.32	5.27	0.85	7.81	3.16	0.82	13827.88
3	11.61	4.29	2.32	5.22	0.88	7.61	3.01	0.82	12814.60
4	12.14	3.93	1.79	5.10	0.85	7.12	2.64	0.79	12236.47
5	10.54	4.46	1.79	4.88	0.85	6.77	2.50	0.80	11807.89
6	13.21	4.82	2.32	4.90	0.84	6.81	2.51	0.80	11968.13
7	13.75	5.18	2.32	4.84	0.84	6.73	2.49	0.80	11933.39
8	12.50	4.82	1.43	4.85	0.86	6.67	2.43	0.79	11780.52
9	11.07	4.11	2.14	4.91	0.87	6.75	2.45	0.79	11694.74
10	12.86	5.36	3.21	4.78	0.87	6.60	2.43	0.79	11545.84

Table B.16: Case Study A: Cost Models based on KNN (K=1 to 10) in PVS_{Size}

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	15.75	3.87	1.10	2.36	0.78	4.08	2.30	0.81	4127.00
2	11.88	4.42	2.21	2.52	0.78	3.90	1.97	0.79	3853.99
3	13.26	5.25	1.66	2.40	0.79	3.47	1.66	0.78	3738.44
4	12.71	4.14	1.93	2.52	0.79	3.51	1.58	0.80	3713.35
5	10.77	4.70	2.21	2.52	0.79	3.47	1.55	0.80	3709.20
6	11.60	5.80	2.49	2.51	0.80	3.45	1.54	0.78	3695.22
7	11.33	4.70	3.04	2.50	0.79	3.42	1.51	0.79	3677.62
8	11.33	4.42	2.21	2.47	0.80	3.41	1.53	0.79	3631.09
9	11.88	3.59	1.66	2.53	0.80	3.46	1.53	0.79	3670.59
10	11.33	4.14	2.76	2.51	0.80	3.46	1.55	0.78	3658.37

Table B.17: Case Study A: Cost Models based on KNN (K=1 to 10) in PSS

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	15.47	5.52	2.76	2.48	0.81	4.21	2.32	0.81	4345.35
2	11.60	5.52	3.04	2.47	0.80	3.78	1.90	0.78	3895.40
3	13.81	4.14	2.21	2.52	0.79	3.64	1.72	0.80	3870.19
4	11.33	2.21	1.10	2.47	0.80	3.50	1.62	0.79	3768.43
5	10.22	3.59	1.93	2.49	0.80	3.47	1.58	0.79	3722.29
6	10.77	4.42	2.21	2.51	0.79	3.50	1.58	0.78	3724.21
7	11.88	4.97	2.49	2.54	0.80	3.50	1.55	0.78	3714.81
8	12.43	5.80	2.21	2.54	0.78	3.49	1.53	0.77	3669.63
9	12.71	5.52	1.66	2.58	0.79	3.49	1.50	0.77	3665.83
10	12.43	5.52	2.21	2.57	0.79	3.50	1.52	0.77	3672.41

Table B.18: Case Study A: Cost Models based on KNN (K=1 to 10) in EPSS

Category Nr.	RMethod	Frequency	Percent
1	Productive	101	18.0
2	All figures	1	0.2
3	Combination	6	1.1
4	Most recorded	1	0.2
5	Only total hours	1	0.2
6	Other Method	1	0.2
7	Staff hours	448	80.0
8	Weekly recorded	1	0.2
	Total	560	100.0

Table B.13: Case Study A: Frequency of RMethod

Category Nr.	UMethod	Frequency	Percent
1	Don't Know	157	28.0
2	No	62	11.1
3	Yes	341	60.9
	Total	560	100.0

Table B.14: Case Study A: Frequency of UMethod

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	12.32	4.82	2.68	5.19	0.85	9.55	4.99	0.87	14006.84
2	13.93	6.96	3.39	5.27	0.83	8.19	3.53	0.80	12879.81
3	15.00	5.18	2.14	5.39	0.83	7.68	2.89	0.79	12998.31
4	13.75	5.36	3.39	5.14	0.86	7.28	2.74	0.80	12336.09
5	12.32	3.93	2.68	4.96	0.85	6.90	2.54	0.77	11841.01
6	12.86	4.46	1.96	4.98	0.86	6.88	2.51	0.79	12033.99
7	12.86	4.46	1.61	5.01	0.86	6.94	2.54	0.79	11981.19
8	11.79	4.11	1.43	4.89	0.86	6.84	2.56	0.79	11805.95
9	12.50	5.00	2.14	4.90	0.86	6.82	2.53	0.78	11716.11
10	13.75	4.82	2.86	4.77	0.86	6.61	2.44	0.77	11558.05

Table B.15: Case Study A: Cost Models based on KNN (K=1 to 10) in PVS

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	17.40	5.80	3.31	2.41	0.79	4.06	2.23	0.79	4173.78
2	11.60	4.14	1.66	2.44	0.81	3.72	1.87	0.78	3895.31
3	12.71	4.97	3.04	2.41	0.81	3.50	1.68	0.79	3819.68
4	12.15	3.59	1.10	2.39	0.80	3.40	1.60	0.77	3780.82
5	12.71	4.14	2.21	2.44	0.79	3.42	1.57	0.77	3711.12
6	13.54	4.14	1.93	2.48	0.79	3.43	1.54	0.79	3716.82
7	12.15	5.25	1.38	2.51	0.79	3.46	1.54	0.78	3697.40
8	12.43	6.08	2.76	2.52	0.80	3.45	1.52	0.77	3674.23
9	12.43	5.52	3.59	2.54	0.79	3.46	1.51	0.78	3667.90
10	12.71	6.08	2.21	2.54	0.80	3.48	1.53	0.78	3668.25

Table B.19: Case Study A: Cost Models based on KNN (K=1 to 10) in $EPSS_{Size}$

Appendix C

Bank63 Data and Experiment Details

Variable	Full Name	Type	Definition
id	identification number	nominal	each completed project has a unique identification number.
size	application size	ratio	function points measured using the experience method.
effort	effort	ratio	work carried out by the software supplier from specification until delivery, measured in hours
duration	duration	ratio	duration of project from specification until delivery, measured in months.
syear	start year	interval	1985 - 1993
telonuse	Telon use	nominal	0=No, 1= Yes
app	application type	nominal	401= customer service (CustServ) 402=management information system (MIS) 406=transaction processing (TransPro) 407=Production control, logistics order processing (ProdCont) 408=Information/on-line service (InfServ)
har	hardware platform	nominal	1001=networked (Network) 1002=Mainframe (Maifrm) 1003=Personal computer (PC) 1004=Mini computer (Mini) 1005=Multi-platform (Multi)
dba	DBMS architecture	nominal	1602=relational (Relatnl) 1604=Other (Other) 1605=Sequential (Sequentl)
ifc	user interface	nominal	2001=Graphical user interface (GUI) 2002=Text user interface (TextUI)
source	where developed	nominal	7001=In-house (Inhouse) 7004=Outsourced (outsrced)
lan1	language used	nominal	Up to four languages were used per application They could be used in any order; thus, lan1 is not necessarily the most important language. Too many codes to list here; however, codes
lan2	language used	nominal	of special interest include:
lan3	language used	nominal	2671= COBOL
lan4	language used	nominal	2660= Telon
t01	customer participation	ordinal	how actively customer took part in development work: 1=Very low; none 2=Low; passive; client defined or approved ¡30% of all functions 3=Nominal; Client defined and approved 30-70% of all functions 4=High; active; client defined and approved all of most important functions, and over 70% of others 5=Very high; Client participated very actively thus most functions were slightly volatile and changes had to be made

Table C.1: Case Study B: Project Attributes Description 1

Variable	Full Name	Type	Definition
t02	development environment adequacy	ordinal	Performance level of tool resources and equipment during project: 1=very low; continuous shortcomings in devices, building of test environment and testing required special arrangements 2=Low;shared equipment/machine resources; delays in some work stages (e.g. compiling and testing) 3=Nominal; enough during development work; a workstation for everybody 4=High; enough to deal with capacity peaks (efficiency, storage, response time) 5=Very high; dedicated, over-dimensioned development environments in practice only for this project
t03	staff availability	ordinal	Availability of software personnel during project: 1=Very low; big problems with key personnel availability; lots of simultaneous customer and maintenance responsibilities; special know-how required 2=Low; personnel involved in some other simultaneous projects and/or maintenance responsibilities 3=Nominal; key members involved in only one other project 4=High; project members involved almost full-time 5=Very high; qualified personnel available when needed; full-time participation
t04	standards use	ordinal	Level and use of standards 1=Very low; standards developed during project 2=Low; some standards, but not familiar ones; more must be developed for some tasks 3=Nominal; generally known standards applied in environment before; some tailoring needed 4=high; detailed standards applied in same environment for some time 5=Very high; stable and detailed standards; already familiar to team; use controlled
t05	methods use	ordinal	Level and use of methods 1=Very low; no modern design methods; mostly meetings; used by individuals 2=Low; use beginning; traditional concepts employed (structural analysis and design, top-down design, etc.) 3=Nominal; generally known methods used 4=High; methods integrated in detail and most activities covered; support existed; used by everyone 5=Very high; methods used during entire lifestyle; methods tailored for specific needs of project; methods supported for individual projects

Table C.2: Case Study B: Project Attributes Description 2

Variable	Full Name	Type	Definition
t06	tools use	ordinal	Level and use of tools 1=Very low; minimal tools; editors, compilers, and testing tools 2=Low; basic tools: interpreters, editors, compilers, debuggers, databases, and libraries 3=Nominal; development environment, database management systems (DBMS), and support for most phases 4=High; modern tools like CASE, project planners, application generators, and standardized interfaces between phases 5=Very high; integrated CASE environment over entire lifestyle; all tools support each other
t07	software's logical complexity	ordinal	Computing, I/O needs, and user interface requirement: 1=Very low; only routines; no need for user interface; simple databases 2=Low; functionally clear; no algorithmic tasks; database solution clear 3=Nominal; functionally typical; normal, standard database; no algorithms 4=High; processing more demanding; database large and complex; new requirements for user interfaces 5=Very high; functionally and technically difficult solution; user interface very complex; distributed database
t08	requirements volatility	ordinal	Volatility of customer/ user requirements during project: 1=Very low; no new features; standard components; conversions only 2=Low; some changes to specifications, some new or adapted functions; some minor changes in data contents 3=Nominal; more changes to specifications but project members could handle them; impact minor (¡15% new or modified functions) 4=High; some major changes affecting total architecture and requiring rework; 15-30% of functions new or modified 5=Very high; new requirements added continuously; lots of rework; more than 30% new or modified functions compared to original requirements

Table C.3: Case Study B: Project Attributes Description 3

Variable	Full Name	Type	Definition
t09	Quality requirements	ordinal	Quality goals of software: 1=Very low; no quality requirements; "quick-and -dirty" allowed 2=Low; basic requirements satisfied (documentation, implementation testing, system testing, and module testing); no statistical controls or reviews 3=Nominal; proper documentation of critical features; design- and implementation-tested modules/job flows tested; walkthroughs; maintenance work planned 4=High; formal reviews and inspections between all phases; attention to documentation usability, and maintenance 5=Very high; quantified quality requirements 100% satisfaction of technical and functional goals; maintenance work minimal
t10	efficiency requirements	ordinal	Efficiency goals of software: 1=Very low; no efficiency requirements needing attention or planning 2=Low; efficiency goals easy to reach; requirements below average 3=Nominal; capacity level of software stable and predictable; response time, transaction load, and turnaround time typical 4=High; specific peaks in capacity, response time, transaction processing, and turnaround time reached by specific design and implementation techniques 5=Very high; efficiency essential; strict efficiency goals needing continuous attention and specific skills
t11	installation requirements	ordinal	training needs for users and variants of platform 1=Very low; no training needs; ¡10 users 2=Low; some training; about 10 users; creation of basic data only minor 3=Nominal; typical training for several organizations; ¡1000 users; extra software for conversions; possible parallel runs; several platforms 5=Very high; ¿1000 users; long expected lifetime; several use organizations; several different platforms
t12	staff analysis skills	ordinal	Analysis skills of project staff at kick-off: 1=Very low; no experience in requirements analysis or similar project 2=Low; ¡30% of project staff with analysis and design experience in similar projects 3=Nominal; 30-70% of project staff with analysis experience; one experienced member 4=High; most members of staff with experience in specifications and analysis; analysis professional in charge 5=Very high; project staff composed of first-class professionals; members have strong vision and experience with requirements analysis

Table C.4: Case Study B: Project Attributes Description 4

Variable	Full Name	Type	Definition
t13	staff application knowledge	ordinal	Knowledge of application domain in project team (supplier and customer): 1=Very low; team application experience ¡6 months on average 2=Low; application experience low; some members have experience; 6-12months on average 3=Nominal; applications experience good; 1-3 years on average 4=High; application experience good both at supplier and customer sites; 3-6years on average; business dynamics known 5=Very high; both supplier and customer know application area well, including the business; ¿6 years' average experience
t14	staff tool skills	ordinal	Experience level of project team (supplier and customer) with development and documentation tools at project kick-off: 1=Very low; team has no experience in necessary tools; team's average experience ¡6 months on average 2=Low; tools experience less than average; some members have experience with some tools; 6-12months on average 3=Nominal; tools experience good in about half the team; some members know development and documentation tools well; 1-3 years on average 4=High; most team members know tools well; some members can help others; 3-6 years on average 5=Very high; team knows all tools well; support available for specific needs of project; ¿6 years' average experience
t15	staff team skills	ordinal	Ability of project team to work effectively according to best project practices: 1=Very low; scattered team; minimal project and management skills 2=Low; some members with previous experience on similar projects; not united as a group 3=Nominal; most members with experience on similar projects; commitment on project goals good; no motivation to utilize real team spirit 4=High; group very active and knows how to exploit team effectiveness 5=Very high; very anticipatory team; team can solve in an innovative way most personal and team problems; superior spirit

Table C.5: Case Study B: Project Attributes Description 5

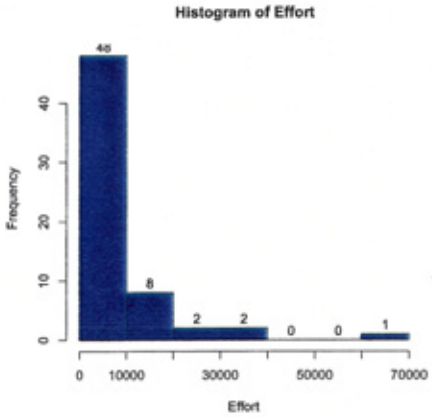

Figure C.1: Hist Effort in PVS

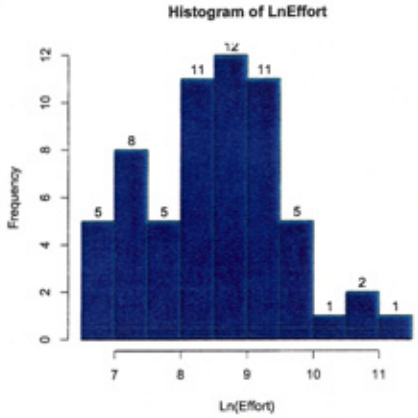

Figure C.2: Hist LnEffort in PVS

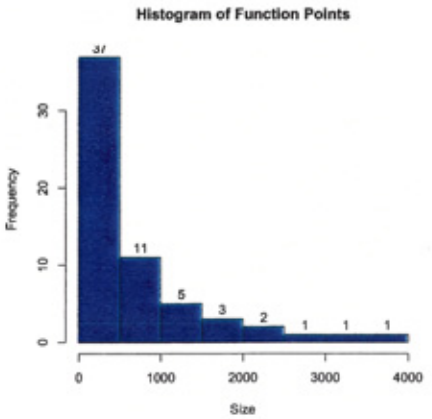

Figure C.3: Hist Size in PVS

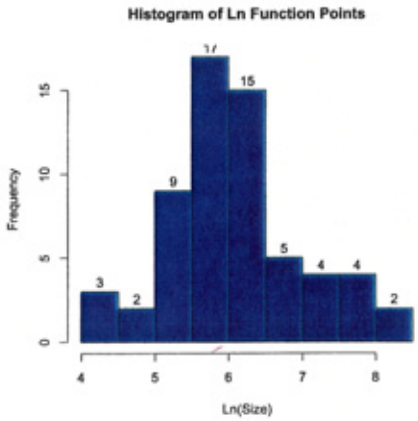

Figure C.4: Hist LnSize in PVS

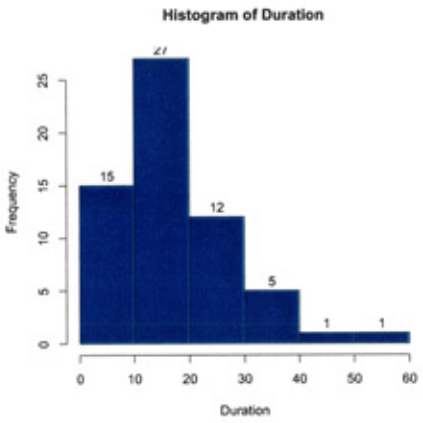

Figure C.5: Hist Duration in PVS

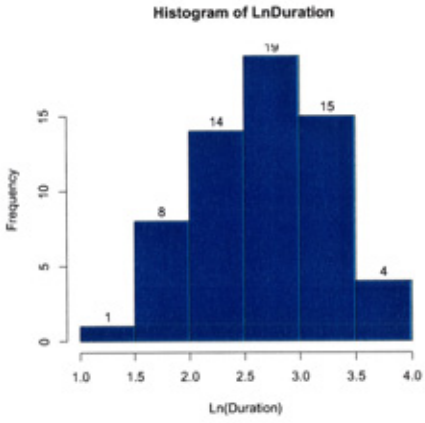

Figure C.6: Hist LnDuration in PVS

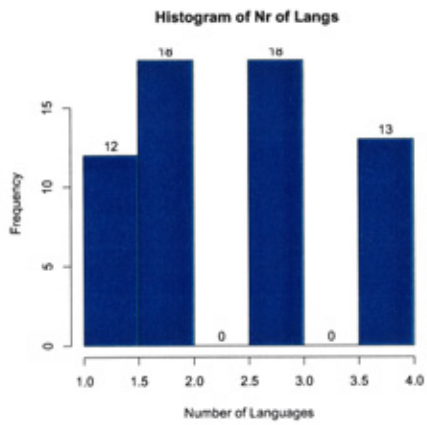

Figure C.7: Hist nlan in PVS

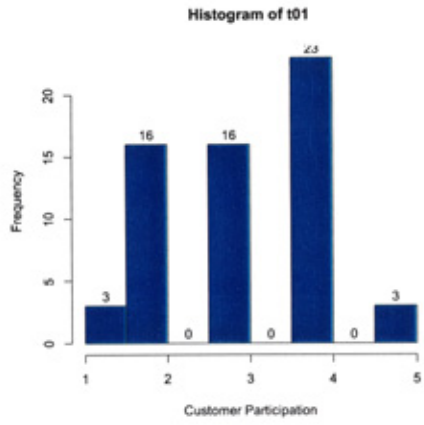

Figure C.8: Hist t01 in PVS

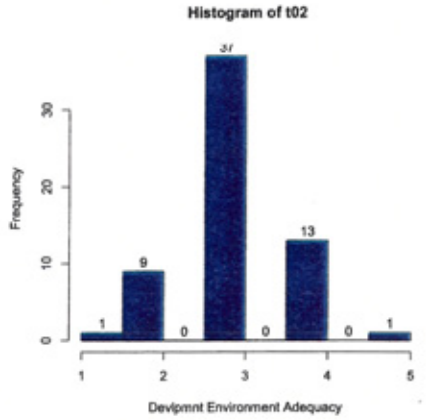

Figure C.9: Hist t02 in PVS

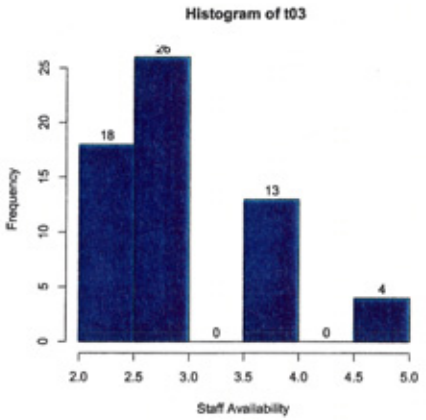

Figure C.10: Hist t03 in PVS

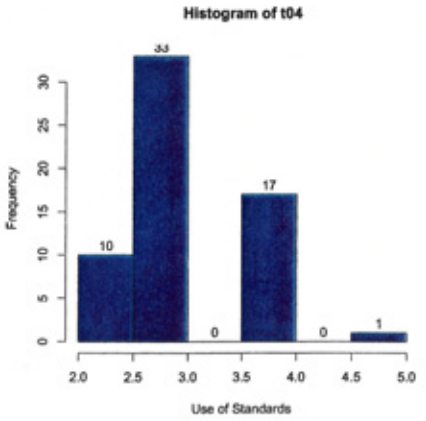

Figure C.11: Hist t04 in PVS

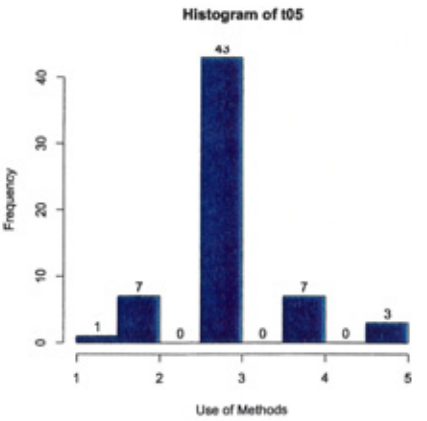

Figure C.12: Hist t05 in PVS

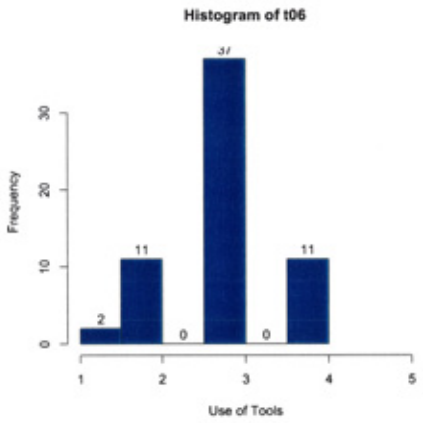

Figure C.13: Hist t06 in PVS

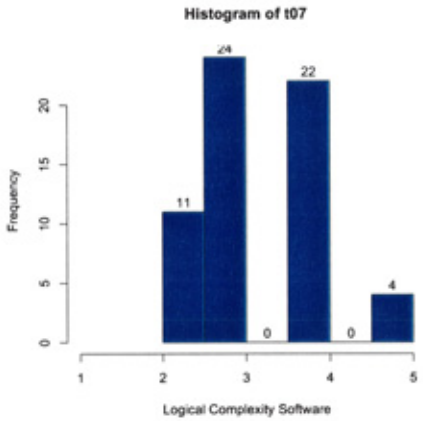

Figure C.14: Hist t07 in PVS

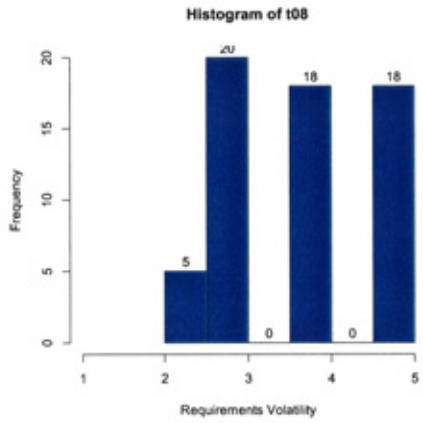

Figure C.15: Hist t08 in PVS

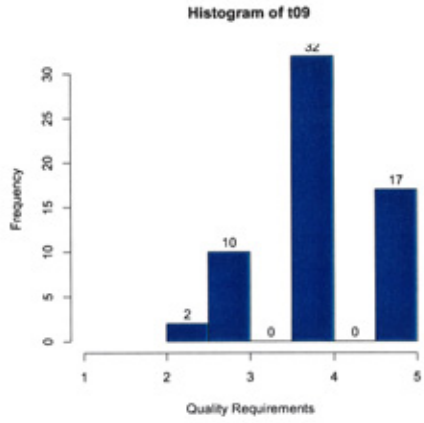

Figure C.16: Hist t09 in PVS

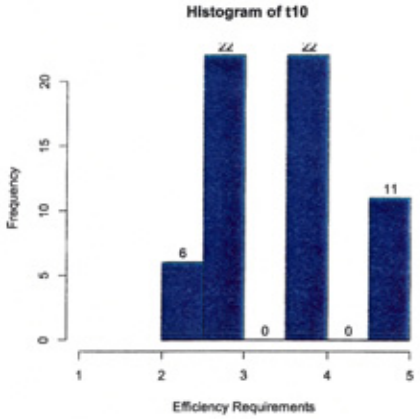

Figure C.17: Hist t10 in PVS

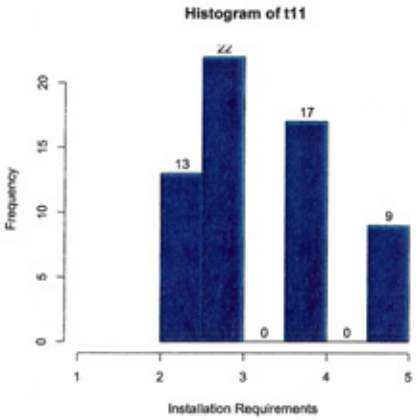

Figure C.18: Hist t11 in PVS

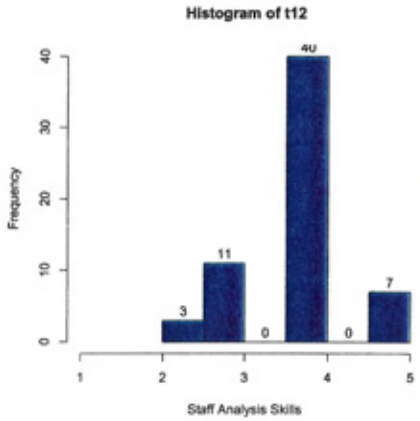

Figure C.19: Hist t12 in PVS

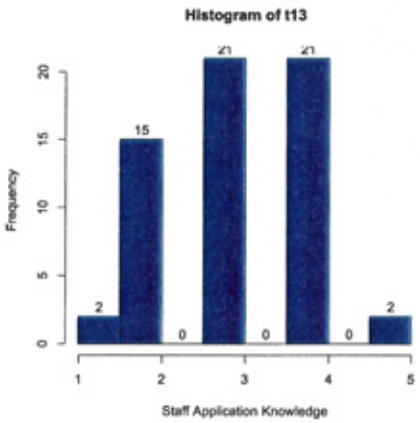

Figure C.20: Hist t13 in PVS

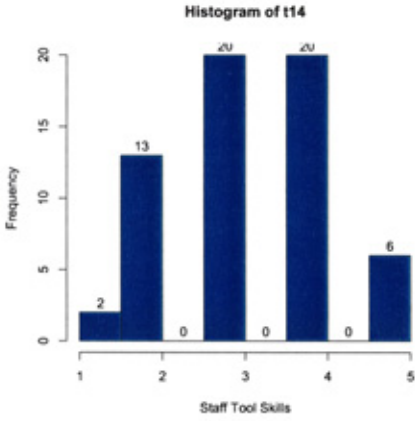

Figure C.21: Hist t14 in PVS

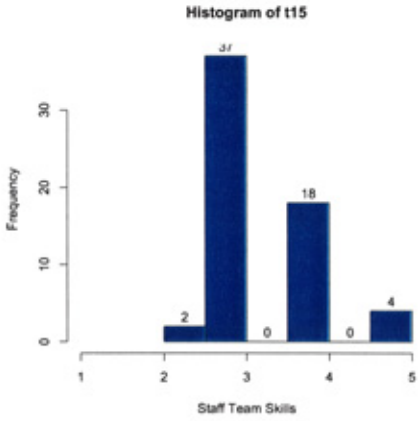

Figure C.22: Hist t15 in PVS

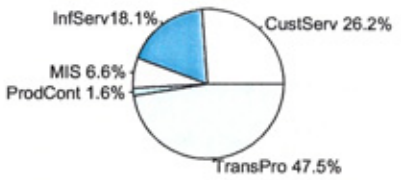

Figure C.23: Case Study B: Pie Chart of App in PVS

Figure C.24: Case Study B: Pie Chart of Dba in PVS

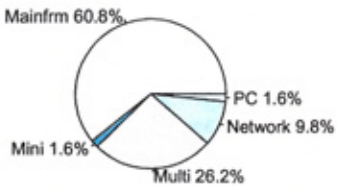

Figure C.25: Case Study B: Pie Chart of Har in PVS

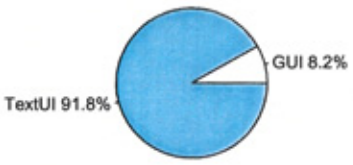

Figure C.26: Case Study B: Pie Chart of Ifc in PVS

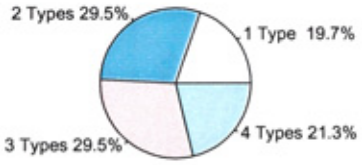

Figure C.27: Case Study B: Pie Chart of Nlan in PVS

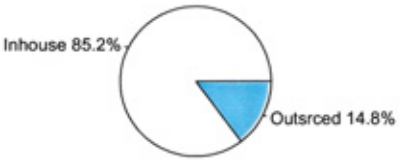

Figure C.28: Case Study B: Pie Chart of Source in PVS

Figure C.29: Case Study B: Pie Chart of Telonuse in PVS

Category Nr.	App	Frequency	Percent
1	CustServ	16	26.2
2	InfServ	11	18.1
3	MIS	4	6.6
4	ProdCont	1	1.6
5	TransPro	29	47.5
	Total	61	100.0

Table C.6: Case Study B: Frequency of App in PVS

Category Nr.	Dba	Frequency	Percent
1	Other	1	1.6
2	Relatnl	59	96.8
3	Sequentl	1	1.6
	Total	61	100.0

Table C.7: Case Study B: Frequency of Dba in PVS

Category Nr.	Har	Frequency	Percent
1	Mainfrm	37	60.8
2	Mini	1	1.6
3	Multi	16	26.2
4	Network	6	9.8
5	PC	1	1.6
	Total	61	100.0

Table C.8: Case Study B: Frequency of Har in PVS

Category Nr.	Ifc	Frequency	Percent
1	GUI	5	8.2
2	TextUI	56	91.8
	Total	61	100.0

Table C.9: Case Study B: Frequency of Ifc in PVS

Category Nr.	Nlan	Frequency	Percent
1	1 Type	12	19.7
2	2 Types	18	29.5
3	3 Types	18	29.5
4	4 Types	13	21.3
	Total	61	100.0

Table C.10: Case Study B: Frequency of Nlan in PVS

Category Nr.	Source	Frequency	Percent
1	Inhouse	52	85.2
2	Outsrced	9	14.8
	Total	61	100.0

Table C.11: Case Study B: Frequency of Source in PVS

Category Nr.	Telonuse	Frequency	Percent
1	No	46	75.4
2	Yes	15	24.6
	Total	61	100.0

Table C.12: Case Study B: Frequency of Telonuse in PVS

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	14.55	9.09	7.27	3.57	0.7	4.81	1.81	0.84	17904.92
2	16.36	3.64	1.82	2.87	0.68	4.06	1.74	0.67	12664.69
3	16.36	9.09	5.45	2.59	0.72	3.71	1.67	0.7	11080.6
4	14.55	5.45	3.64	2.41	0.63	3.25	1.38	0.71	10471.08
5	18.18	5.45	5.45	2.3	0.67	3.21	1.44	0.74	9917.8
6	14.55	1.82	0	2.25	0.7	2.98	1.27	0.67	9543.14
7	16.36	3.64	3.64	2.09	0.71	2.84	1.28	0.74	9143.27
8	18.18	5.45	0	1.96	0.67	2.72	1.29	0.7	8806.84
9	16.36	5.45	1.82	1.93	0.67	2.67	1.26	0.59	8643.65
10	18.18	1.82	1.82	1.87	0.66	2.61	1.25	0.54	8519.93

Table C.13: Case Study B: Cost Models based on KNN (K=1 to 10) in PVS

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	10.91	7.27	1.82	3.29	0.78	4.83	2.13	0.93	15877.24
2	16.36	5.45	0	2.88	0.71	3.99	1.66	0.72	12697.11
3	9.09	7.27	1.82	2.71	0.77	3.9	1.75	0.78	12532.14
4	21.82	9.09	1.82	2.42	0.68	3.24	1.35	0.72	11286.1
5	18.18	7.27	3.64	2.33	0.67	2.95	1.15	0.69	10775.85
6	18.18	5.45	3.64	2.35	0.61	2.94	1.1	0.71	10324.17
7	12.73	5.45	0	2.25	0.64	2.86	1.13	0.68	9713.08
8	14.55	7.27	0	2.18	0.63	2.81	1.15	0.67	9506.1
9	12.73	3.64	1.82	2.26	0.58	2.9	1.17	0.68	9606.78
10	10.91	5.45	5.45	2.28	0.67	2.93	1.18	0.67	9631.95

Table C.14: Case Study B: Cost Models based on KNN (K=1 to 10) in PVS_{Size}

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	21.82	12.73	5.45	1.63	0.74	2.55	1.46	0.77	9866.37
2	12.73	7.27	1.82	1.5	0.66	2.23	1.26	0.73	8356.71
3	18.18	7.27	7.27	1.43	0.65	2.15	1.24	0.73	7865.82
4	16.36	7.27	5.45	1.38	0.7	1.99	1.13	0.71	7685.5
5	20	1.82	0	1.34	0.66	1.97	1.15	0.62	7608.84
6	18.18	5.45	3.64	1.36	0.69	1.89	1.05	0.66	7165.28
7	16.36	3.64	0	1.33	0.68	1.79	0.96	0.63	6864.1
8	21.82	9.09	3.64	1.3	0.66	1.79	1	0.63	6937.68
9	20	7.27	5.45	1.24	0.61	1.68	0.92	0.61	6742.79
10	25.45	9.09	1.82	1.24	0.62	1.65	0.89	0.6	6706

Table C.15: Case Study B: Cost Models based on KNN (K=1 to 10) in PSS

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	12.73	3.64	3.64	1.62	0.78	2.39	1.34	0.78	8124.42
2	12.73	5.45	0	1.69	0.68	2.39	1.24	0.68	8168.33
3	16.36	5.45	3.64	1.58	0.67	2.26	1.22	0.72	7717.34
4	18.18	10.91	9.09	1.48	0.69	2.1	1.13	0.68	7535.45
5	14.55	1.82	1.82	1.5	0.65	2	1.01	0.7	7401.02
6	16.36	1.82	0	1.46	0.67	1.95	1	0.67	7361.73
7	10.91	3.64	3.64	1.47	0.68	1.95	1.01	0.67	7364.96
8	16.36	3.64	3.64	1.46	0.66	1.92	0.98	0.66	7214.11
9	14.55	5.45	5.45	1.41	0.69	1.87	0.97	0.68	7227.09
10	14.55	9.09	1.82	1.38	0.67	1.84	0.97	0.64	7150.65

Table C.16: Case Study B: Cost Models based on KNN (K=1 to 10) in EPSS

K	Pred25	Pred10	Pred5	MMRE	MdMRE	BMMRE	MMER	MdMER	St. Dv.
1	23.64	5.45	3.64	1.26	0.61	2.3	1.55	0.7	8939.08
2	21.82	16.36	9.09	1.53	0.6	2.16	1.11	0.65	7924.35
3	25.45	12.73	5.45	1.37	0.58	1.88	0.99	0.65	7322.64
4	25.45	14.55	10.91	1.27	0.58	1.75	0.94	0.52	6970.46
5	23.64	9.09	3.64	1.27	0.59	1.71	0.9	0.52	6692 447
6	21.82	9.09	5.45	1.3	0.56	1.7	0.86	0.52	6824.96
7	25.45	7.27	1.82	1.29	0.63	1.68	0.85	0.47	6671.74
8	29.09	7.27	1.82	1.25	0.62	1.61	0.82	0.48	6669.13
9	25.45	7.27	5.45	1.19	0.64	1.56	0.83	0.45	6575.72
10	25.45	5.45	1.82	1.19	0.61	1.59	0.86	0.5	6668.83

Table C.17: Case Study B: Cost Models based on KNN (K=1 to 10) in EPSS_{Size}