SOFTWARE ENGINEERING MODELING WITH META-ANALYSIS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

IN

ELECTRONICS SYSTEMS ENGINEERING

UNIVERSITY OF REGINA

BY

Sean Kuang Chin Lim

Regina, Saskatchewan

August 2001

0-612-71348-2

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

CERTIFICATION OF THESIS WORK

We, the undersigned, certify that Sean Kuang Chin Lim, candidate for the Degree of Master of Applied Science, has presented a thesis on ***Software Engineering Modeling with Meta-Analysis,*** that the thesis is acceptable in form and content, and that the student demonstrated a satisfactory knowledge of the field covered by the thesis in an oral examination held on October 23, 2001.

External Examiner: [illegible]
Dr. E. Ahmed, Department of Mathematics & Statistics

Internal Examiners: [illegible]
Dr. L. Benedicenti, Supervisor

[illegible]

[illegible]

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

PERMISSION TO USE POSTGRADUATE THESIS

TITLE OF THESIS:	Software Engineering Modeling with Meta-Analysis
NAME OF AUTHOR:	Sean Kuang Chin Lim
DEGREE:	Master of Applied Science

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from the University of Regina, I agree that the Libraries of this University shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the professor or professors who supervised my thesis work, or in their absence, by the Head of the Department or the Dean of the Faculty in which my thesis work was done. It is understood that with the exception of UMI Dissertations Publishing (UMI) that any copying, publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Regina in any scholarly use which may be made of my material in my thesis.

SIGNATURE: [illegible]

DATE: 23. 10. 01

Abstract

Data collection is one of the most demanding tasks in software engineering. Without the aid of a solid theoretical basis to validate the measurement procedure, this software engineering data is rarely useful. Researchers and practitioners develop numerous ways to validate the collected data and transform the data into sophisticated information to understand the software systems development.

In order to predict the characteristics of software development accurately, software engineers seek to build a general estimation model for software project, which will represent the characteristics of software resources, processes and products. This thesis illustrates a framework for using meta-analysis as a statistical analysis technique to combine the knowledge acquired from different software development experiments and studies. An example is presented to show the results of data analysis is more accurate by using meta-analysis than traditional statistical analysis.

Acknowledgments

I would like to send special and heartfelt thanks to my parents for being the special people they are. I love you very much and thank you for teaching me, sharing your love and ideas with me, and raising me to be a happy person.

I would like to express my sincerest gratitude to my supervisor, Dr. Benedicenti, who gave me valuable advice, countless encouragement. I am very thankful for the generous financial support originating from the Faculty of Graduate Studies and Research (two Graduate Scholarships) and the Natural Sciences and Engineering Research Council of Canada. I am grateful to my internal committee member, Dr. Raman Paranjape and Dr. Christine Chan, for their constructive comments and suggestions.

Extra special thanks go to those whom I know I can count on without fail and who have always gone the extra mile for me. And to all my dear friends who gave me unconditional love, encouragement, affection and much needed support during a very difficult time in my life, I give you my undying love in return.

Thank you, I love you all.

Contents

List of Figures

Chapter 1

Introduction

Software projects are very difficult to control as they involve unpredictable time- and resource-consuming activities. With the increasing usage of automation in data collection of software development activities, the obtained information is still insufficient to precisely estimate a software project. Therefore, software project managers are urged to seek accurate information and guidelines to make decisions, plan activities, and allocate resources for various software activities that constitute software development.

Many researchers have developed models and equations to predict important factors such as the cost of any given programming project. However, due to the differences in data collection, in the types of projects and in environmental factors among software development sites, these models and equations are hardly transferable and are only valid within the organization where they were developed. In addition, the rapidly changing nature of software development causes difficulties in building empirical models with high prediction accuracy. As a result, software development costs continue to increase and practitioners keep expressing their concerns over their inability to accurately predict software development characteristics [36].

The complexity of software development arises from both technical issues and organizational issues. One of the major factors contributing to this problem is an inadequate understanding of the real behavior of the software development processes. Various software metrics and statistical models have been developed with a correspondingly large amount of literature, but only few of them seem to have enough consistency and accuracy to provide a combined perspective on software development. For instance, most of the published individual experimental studies form the basis of statistical models that often produce inconsistent analytical results.

Many factors may have contributed to overrun problems with software projects. However, researchers and practitioners are unable to combine these factors into a single general estimation model. These general estimation models for software projects should accurately represent the fundamental characteristics of software systems development processes and products.

There is still some lack of understanding of which kind of statistical analysis to perform on software engineering data. There is a need to further investigate the best approach to the statistical analysis of software engineering data on the level of the individual studies and to investigate the problem of generalizing the conclusions from the individual studies [46].

The characteristics of many existing Software Metric Models are too specific for an organization or an individual project. The model parameters must be calibrated for the particular software development environment in which the model is to be used regardless of which model is chosen. A generic model, which is independent of an organization's characteristics, is needed [46].

In this thesis, meta-analysis is selected as a statistical tool to combine the knowledge acquired from different software development experiments and studies. Meta-analysis utilizes statistical procedures to combine two or more empirical studies by relating one parameter to another with similar characteristics. This method combines the outcomes of many different studies and investigates the effect of parameters on an across-study basis.

This thesis is organized as follows: Chapter 2 describes the previous work done in software metrics from the points of view of both research and practice. This chapter addresses a few issues about software estimation models. Chapter 3 contains the architecture framework on how meta-analysis can be used as a statistical tool for software data sets. A catalog will be introduced, which consists of data sets from the literature, and has been used as the principal data source for this thesis.

Chapter 4 contains the theoretical results of meta-analysis and the information on how to use meta-analysis to build a general model for software production. The procedures of applying meta-analysis for point estimates, correlation estimates, regression estimates and surface estimates are discussed in this chapter. Chapter 5 provides an example of a real software case selected from data sets in the catalog. This example is analyzed by using meta-analysis and compared to the results analyzed by using other fundamental statistical tools.

A summary of the current research, the theoretical exploration works and the experimental results on this thesis is included in Chapter 6. Conclusions are made according to the proposed meta-analysis method used to approach the software estimation model problem. Main contributions and possible topics for future research are also discussed.

Chapter 2

Previous Work

In seeking to improve the software development process, researchers and practitioners are finding out how much is involved in its measurement. Without adequate quantitative support, they cannot readily comprehend the complexities of software structure, testability, quality and reliability. Well-defined software measurement frameworks emerge with rigorously defined measures and meaningful data is needed to approach effective software production [9].

Chapter 2 contains the following sections: Section 2.1 discusses the essential role of software engineering and the previous works that contributed to the discipline. Section 2.2 introduces the existing defined software measurement frameworks. This section first presents the definition of software attributes and software entities, followed by the representational theory of measurement. Scales of measurement and the validation of software measurement used to ensure software engineers collect meaningful software measures also introduced in this section. Section 2.2 also discusses the existing software experimental design, measuring equipments and analyzing tools. The existing software estimation models are presented in section 2.3. Section 2.4 discusses the problems of current software estimation methods. Finally, the advantages of using meta-analysis over traditional reviews in combining the outcomes of many different studies are explained in section 2.5.

2.1 Software Engineering

There are a number of possible definitions of software engineering. According to Fenton and Pfleeger, software engineering describes "the collection of engineering approach techniques to the construction and the support of software products" [36]. As software development expenses are the major component of computer system costs, the future prosperity of an industrialized software economy depends on effective software engineering [9].

Software engineering concentrates on developing the software in a controlled and scientific way while computer science provides the theoretical foundations for building software. By using an engineering approach, each software development activity can be understood and controlled. Therefore, the software product is specified, designed, built and maintained so that few surprises arise [36].

Insufficient knowledge of the maintenance process and the cause and effect relationships between software practices and maintenance outcomes cause many software maintenance problems. Kemerer and Slaughter have designed an approach for direct research that enlarges the scope of the empirical data available on software evolution so that evolution patterns could be linked to a number of organizational and software engineering factors [12].

Researchers like Murphy et al. describe and criticize an evaluation that they deployed to determine whether a new software development technique is useful and usable [15]. Their software engineering experiences are essential to other researchers who attempt to assess new programming techniques in an early stage of software development.

Software engineers and managers seek to understand the software development process and determine necessary changes for productivity, quality and cycle time improvement. Software measurement is used to enable software engineers to understand the behavior of software processes and products [36].

2.2 Software Measurement

Fenton and Pfleeger define 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" [36]. Measurement thus creates quantitative descriptions of key processes and products in any scientific field. Based on these quantitative descriptions, software engineers can select better techniques and tools to control and improve their software processes, products and resources.

2.2.1 Attributes and Entities

Since measurement captures information about attributes of entities, software engineers can understand the characteristics of the entities. According to Fenton and

Pfleeger, an entity is an object (such as a person) or an event (such as the testing phase of a software project) in the real world [36]. The entity is described by identifying its characteristics that are important to distinguish one entity from another.

Fenton and Pfleeger also mention that an attribute is a feature or property of an entity [36]. Typical attributes include the height (of an object) or the elapsed time (of the testing phase). When entities are described by using attributes, the attributes are defined as numbers or symbols. For instance, price is designated as a number of dollars and height is defined in term of inches or centimeters.

Another example, module coupling, is defined as the measure of the strength of association established by a connection from one module to another [53]. This measurement is a more complex software attribute in object-oriented systems. There is no standard terminology and formalism for expressing measures yet measures are not fully operationally defined. As a result, it is very difficult to understand how different coupling measures relate to one another. Briand and his colleagues have provided a framework for the comparison, evaluation and definition of coupling measures in object-oriented systems [26]. Their framework is intended to be exhaustive, integrating new ideas with existing frameworks in the literature.

All entities of interest in software can be classified in three categories: processes, products and resources [36]. Anything one wishes to measure is an identifiable attribute of these entities. The attributes are either internal or external. Internal attributes can be the lines of code, the number of operands or the number of decision points. At the same time, external attributes can be the reliability of products, the stability of processes or the productivity of resources. Although software engineers are most interested in measuring external attributes, they cannot measure the attributes directly. Hence, software engineers are generally forced to use indirect measures of internal attributes.

Software development processes vary from company to company and even a single corporation may use several different development models. These differences limit the possibility of using metric sets across different projects. Boegh and his colleagues propose the SQUID (Software Quality In Development) approach to define product quality requirements by establishing targets for external quality characteristics [18].

SQUID helps in establishing relationships between internal and external quality characteristics and using experience from similar software development projects from the past. This model analyzes the actual values of the external attributes achieved by the selected past projects and compares these with the new target values of products. Hence, SQUID can be aimed to encourage software developers to collect quality data and analyze the relation between properties of both the intermediate products and the development process with final product behavior.

2.2.2 The Representational Theory of Measurement

There are rules in any measurement activity to be followed for consistency and to provide a basis for interpreting data. In order to distinguish real-world objects or entities, one must describe the characteristics of the entities. Fenton and Pfleeger describe that a measure maps entities from the real empirical world to the mathematical world, so that one can easily understand an entity's attributes and relationships to other entities. This approach is called "representational theory of measurement [36]. However, there are some difficulties in how to interpret their mathematical behaviors and judge what they mean in the real world.

In order to meet the representational theory of measurement, the measures should represent attributes of the entities observed and the manipulation of the data should preserve relationships among the entities observed. According to Fenton and Pfleeger, empirical relations require a consensus of opinion about the relationships in the real world [36]. Many empirical relations are unary and they are defined on the basis of individual entities. Hence, the representational theory of measurement preserves the relationships of mapping entities and their attributes in the real world to the defined mathematical world.

For instance, the direct measurement of an attribute of an entity involves no other attribute or entity. This direct measurement of an attribute assigns a representation or a mapping, M, from an empirical relation system to some numerical relation system. The purpose of performing the mapping is to manipulate data in the numerical systems and use the results to draw conclusions about the attribute in the empirical system. According to Fenton and Pfleeger, the relationship between these domain and range relationships is called the "representation condition" [36].

2.2.3 Scales of Measurement

Researchers, such as Barbara Kitchenham, Norman Fenton, and Shari Lawrence Pfleeger, have worked extensively in applying measurement theory to proposed software metrics. There are many ways of assigning numbers that satisfy the representation conditions. The nature of different assignments determines the scale type of an attribute.

The notion of a measurement scale helps software engineers to understand which analyses are appropriate.

An admissible transformation occurs when mapping from one acceptable measure to another. According to Fenton and Pfleeger, the measurement mapping, M, together with the empirical and numerical relation systems, is referred as a measurement scale [36].

These measurement scales are classified as one of five major types: nominal, ordinal, interval, ratio and absolute. One relational system is said to be richer than another if all relations in the second are contained in the first. The richer the empirical relation system, the more restrictive the set of representations, and the more sophisticated the scale of measurement. By using this concept, the scale types listed above are shown in increasing levels of richness.

For example, a nominal scale puts items into categories. The empirical relation system consists only of different classes. There is no notion of ordering among the classes. An ordinal scale ranks items in an order. Any mapping that preserves a monotonic function is acceptable. An interval scale defines a distance from one point to another so that there are equal intervals between the consecutive numbers. There is no absolute zero point in an interval scale and thus the ratio does not make sense.

The scale with the most information and flexibility besides the absolute scale is the ratio scale. A ratio scale incorporates an absolute zero, preserves ratios and permits the most sophisticated analysis. All basic arithmetic can be meaningfully applied to the classes in the range of the mapping.

As the scales of measurement carry more information, the defining classes of admissible transformations [36] become increasingly restrictive. The absolute scale is the most restrictive of all; there is only one possible measurement mapping namely the actual count. All arithmetic analysis of the resulting count for the absolute scale is meaningful.

The importance of measurement scales to software measurement depends on the type of calculations one can do with each scale. For instance, software engineers cannot compute a meaningful mean and standard deviation for a nominal scale. Such calculations require an interval or ratio scale. Hence, not all the measurement mapping is the same and the differences among the mappings can restrict the kind of analysis one can perform.

2.2.4 Validation

The measures are validated so software engineers can be sure that the measurements they use are actually measuring what they claim to measure. Fenton and Pfleeger describe that "a measure is valid if it satisfies the representation condition, where this measure captures the behavior perceived in the empirical world into the mathematical world" [36]. Software engineers must consider whether they are measuring something with a direct measure or what entity and attribute are being addressed.

Kitchenham, Pfleeger and Fenton proposed a framework for validating software measurement [8]. They defined a measurement structure model that identifies the elementary component of measures and the measurement process. They also consider five other models involved in software measurement: Unit definition models, instrumentation models, attribute relationship models, measurement protocols and entity population models.

Several attempts have been made to list a set of rules for validation. Schneidewind recommends an empirical validation process in which a software metric is validated by showing that it is associated with some other measure of interest [34]. Weyuker restricts her discussion to complexity metrics and suggests that validation can be performed, by identifying a set of desirable properties that measures should possess and by determining whether or not a prospective metric exhibits those properties [14].

However, each of these frameworks has been criticized and there is not yet a standard accepted way of validating a measure. Kitchenham, Pfleeger and Fenton have proposed a general framework for validating software measurements based on the representational theory of measurement and the statistical rules. [8]

A measurement makes sense only when it is associated with one or more models. An essential model describes the domain and the range of a measure mapping, the entity and the attribute being measured, the set of possible resulting measures, and the relationships among several measures. For example, productivity is equal to size produced per unit of effort.

The representational theory of measurement and validation should not distract one from the considerable difficulty of measuring software in the field. Software engineers often try to relate measures of software projects with human and organizational behaviors, which do not necessarily follow physical laws. However, practitioners and customers only rely on empirical evidence of a measurement's utility regardless of its scientific grounding. Therefore, researchers should work closely with practitioners to understand the valid uses and interpretations of a software measure based on its measurement theoretic attributes [21].

2.2.5 Meaningfulness

By understanding the scale types, software engineers will be able to determine whether statements about a measurement make sense. Although metrics in mathematics implies the concept of distance, in this thesis, the terms "measure" and "metric" will be used interchangeably according to the current literature reviews [36]. Measures map attributes to real numbers and the mapping is attempting to manipulate the real numbers in familiar ways, such as averaging, adding, subtracting and performing sophisticated statistical analysis. This statistical analysis is constrained by the scale types. The knowledge of scale types tells software engineers about limitations on the kind of mathematical manipulations that can be performed.

Those calculations can only be performed if they are permissible for a given scale. The calculations also reflect the type of attribute and the mapping of measure to an attribute. Fenton and Pfleeger state, "a statement involving measurement is meaningful if its truth value is invariant of transformations of allowable scales." [36]

Many attributes of interest in software engineering are not directly measurable. These situations force software engineers to use vectors of measures, with rules for combining the vector elements into a larger indirect measure. Scale types are defined for these indirect measures in a similar way to direct measures. Hence, software engineers can determine whether statements and operations of indirect measures are meaningful.

2.2.6 Data Collection and Measuring Tools

Once software engineers know what data to collect and where in the process to collect it, they must decide how to collect the data as well. The mechanics of data collection must be considered in concert with the data quality. If the right data are not collected, the data analysis will be meaningless and the objectives of the measurement program cannot be met.

By using measuring tools and standard forms, software engineers will be able to avoid difference in interpretation of counting rules. There are many types of measuring tools available, but not all of which are directed primarily at metrics program. The choice of measurement tools depends on how a software engineer will employ the measurements.

Since not all the data collection can be automated, standard forms are critical to good data. These standard forms should be easy to use and validate. Any new form should be tried on a pilot project to determine the clarity, correctness and completeness of form. A corrected form can then be made as a standard one for a wider audience [18].

It is not always necessary to use sophisticated tools for measurement collection and storage, especially on projects just that begin to use metrics. A metrics database acts as a common culture focus within an organization and a key source of information for monitoring and prediction. This database holds all that projects considered important as an organizational history and legacy to future projects. For instance, a team in the United Kingdom, led by Kitchenham, has developed a tool that builds a repository for the collected data and helps practitioners to choose appropriate software metrics [46].

2.2.7 Analyzing tools

When software engineers are sampling from data, it is essential that the characteristics of the sample can be applied to a larger population. It is important to choose appropriate analysis techniques for analyzing large data sets. For example, many practitioners use spreadsheet software or database management systems to store and analyze data.

Many organizations employ simple analysis techniques such as scatter charts and histograms to provide them useful information about what is happening on a project or in a product. Others prefer to use statistical analysis, such as regression analysis, correlation analysis, box plots, measures of central tendency and dispersion.

2.2.8 Experimental design

As part of the effort to create the scientific bases for software engineering research, researchers and practitioners focus on defining mathematically sound software measures, development of a solid mathematical framework for data collection and assessment, and a sound statistical analysis process. However, only few studies have targeted the problem of experimental design. For example, Mendonca and Basili have developed an approach by combining Goal Question Metric paradigm (GQM) and Attribute Focusing (AF) for use when a large number of diverse measurements are already being collected by a software organization [30].

The Goal Question Metric paradigm (GQM) is a useful approach for deciding what to measure. GQM paradigm guides software engineers in deriving and applying measurement [40], [50], [51]. Since managers and practitioners have little time to measure everything, the GQM approach allows them to choose those measures that relate to the most important goals or the most pressing problems.

The GQM approach creates a hierarchy of goals that to be addressed, questions that should be answered in order to know if the goals have been met, and measurement that must be made in order to provide an organization with a largely accurate analysis of its software production. This technique considers the perspective of people who need the information and the context in which the measurement will be used.

Process maturity must also be considered when software engineers decide what to measure. If an entity is not visible in a software development process, then it cannot be associated with the types of measurements that can be made. GQM and process maturity must work hand in hand. By using GQM to decide what to measure and then assessing the visibility of the entity, software engineers can measure an increasingly richer set of attributes.

Many misunderstandings and misapplications of software measurement would be avoided if people thought carefully about the above framework. This framework highlights the relationships among apparently diverse software measurement activities.

For example, Offen and Jeffery propose the M^3P framework (Model, Measure and Manage Paradigm), an extension of the well-known Quality Improvement Paradigm (QIP) and GQM [43]. According to Offen and Jeffrey, M^3P framework is an appropriate measure selection technology links the underlying empirical and numerical measurement models. This M^3P framework can be bootstrapped from existing material, which describes the success factors of general measurement program.

2.3 Software Estimation

Measurement is concerned with the assessment of attribute of some entity that already exists and the prediction of an attribute of some future entity. Software engineers need to be able to predict attributes such as the cost and effort of processes, as well as the reliability and maintainability of products. Hence, Software engineers need an effective prediction, which requires an estimation system or a model supplemented with a set of forecasting procedures for determining the model parameters and applying the results.

Managers want to be able to predict the costs of software projects during early phases of the software development lifecycle. Numerous models for software cost and effort estimations have been proposed and used. Examples include Boehm's COCOMO model [9], Putnam's SLIM model [25] and Albrecht's Function Points model [4], [5]. These cost estimation models can be ranging from highly theoretical ones, such as Putnam's, to empirical ones, such as the estimation method developed by Walston and Felix [11] and Boehm's COCOMO model [10].

Many software engineers are at a lack of understanding of the kind of statistical analysis to perform on their software development data. Hence, one of the basic goals of software engineering is the establishment of useful models and equations to predict the cost of any given software project. For example, Briand et al. argue that linear parametric statistics are sufficient to analyze most of the data coming from software development processes. [27]

Although many models have been proposed, the differences in the collected data, the types of projects and the environmental factors among software development site cause these models being not transformable. These models are only valid within the organization where they were developed. These results seem reasonable when software engineers consider that a model developed at a certain environment will only be able to capture the impact of factors, which have a variable effect within that environment.

Bailey and Basili found that the main difficulty seems to be in determining which environmental attributes really capture the reason for the differences in productivity among the projects. The use of too few of these attributes will mean less of the variation can possibly be explained, while the use of too many makes the analysis statistically meaningless. Hence, Bailey and Basili present a model-generation process, which permits the development of a resource estimation model for any particular organization [20].

Meta-analysis is an effective tool for the integration of the knowledge across different studies by combining the knowledge in each individual study to general applicable knowledge across those studies. Pickard et al. argued that the combination of the knowledge across different studies is best done using meta-analysis, rather than incorporating all the data together into a single dataset [23].

Brooks suggested that software engineering could benefit in generalizing results of empirical studies by using meta-analysis, a research method commonly used in psychology [1]. Hochman et al. show that meta-analysis can also minimize the errors in experimental research [41]. In order for these studies to be appropriate, the underlying phenomenon must be the same in all the studies. In other words, the data from the primary level studies must be homogeneous.

2.4 Dealing with the Problems of Current Estimation Methods

Without a proper demonstration of their validity, software measures and prediction systems are neither widely used nor respected. The commonly accepted ideas and approaches to validating software measures only endure little relation to the rigorous requirements for measurement validation in other disciplines. Hence, a formal validation requirement must first be addressed before practitioners and researchers can tackle any informal notion such as usefulness and practicality.

The lack of common software metrics that are well defined for data collection, data interpretation and measurement tools, causes the problems of current estimation methods. There is a need for generalizing the conclusions about the relations between different software measures derived from each of the individual statistical studies.

By using meta-analysis to generalize across the correlation data, its relative case of calculation and the flexibility in its use can help to solve the software estimation problems. The sample data can be any type of correlation value, such as Pearson's product-moment correlation coefficient and Spearman's rank correlation coefficient. All the data should be of the same type of correlation coefficient for a proper generalization. Sohn develops a statistical meta-model, which compares the classification performances of several algorithms in terms of data characteristics [47]. Her empirical model is expected to aid in the decision-making processes of finding the best classification tool in the sense of providing the minimum classification error among alternatives.

Meta-analysis uses the statistical procedures to integrate research findings across studies. Generally, meta-analysis has important advantages over traditional views. Glass had first applied meta-analysis in 1976 [16]. Meta-analysis has then been warmly embraced in those fields in which experiments are widely used, such as medicine, education and psychology. However, meta-analysis is extremely underutilized in the field of software development. It has been applied in only few of studies in recent years, such as Alavi and Joachimsthaler [28], Benbasat and Lim [17], Hwang and Wu [31], McLeod [38], Montazemi and Wang [7] and Pettingell, Marshall and Remington [22].

Researchers usually bypass meta-analysis in favor of more traditional qualitative reviews, due to the unfamiliarity with or misconceptions about it. For example, Dennis, Numamaker and Vogel concluded that meta-analysis is not suitable for summarizing the findings of group support systems research because of the large variables involved [6]. However, Benbasat and Lim and McLeod were able to meta-analyze the same literature later [17], [38].

In another example, Vessey also asserted that meta-analysis is not suitable for resolving the controversy in the graphics literature without first analyzing task type as a moderator variable. However, Hwang and Wu, and Montazemi and Wong were able to show the moderating effect of task complexity in two separate meta-analyses [7], [31].

2.5 Advantages over Traditional Reviews

According to Baroudi and Orlokowski, conclusions drawn from traditional literature reviews are typically based on the results of significance tests reported in individual studies [56]. Facing a large number of non-significant findings, the researchers and practitioners can only conclude that the data are inconclusive and call for more research on a topic for which hundreds of experiments may have already been conducted. The development of meta-analysis was motivated by the failure of traditional, narrative reviews to provide definite answers to the research questions examined by researchers and practitioners [16].

An essential advantage of meta-analysis is that every study contributes to the understanding of the phenomenon under analysis through its effect size, whether its findings are significant or not. Meta-analysis also gives a quantitative measure of a relationship that is found either significant or non-significant. Cook argued that meta-analysis can enhance the general validity of conclusions through its impact on statistical conclusion validity, internal validity, construct validity and external validity [49].

According to Cook, meta-analysis enhances conclusion validity by using highly reliable mean differences as the units of analysis. These highly reliable mean differences may outweigh any loss in statistical power due to the sample of studies being smaller than that of individual studies. Because the reliability of a sample value affects statistical power, Cohen shows that the reduction in errors in a meta-analysis enhances its statistical conclusion validity [19].

Internal validity can be enhanced as the sources of differences in individual studies may cancel out or be tested empirically. Since many realizations of a treatment and effect are tested in meta-analysis, construct validity of causes and effects can also be enhanced. Meta-analysis allows for testing of casual connections in a wider range of persons, settings and times than is possible in individual studies, thus external validity can be enhanced.

Chapter 3

Architectural Framework

The current objective of software engineering research is to develop a general model for software production. In order to achieve the objective, a catalog that collects datasets from previous published studies is built in this thesis. The catalog reorganizes the datasets across different software development studies and acts as a central repository. Meta-analysis statistical tool can then be applied to this central repository to find the relationships between the parameters with similar characteristics in the datasets.

Chapter 3 contains the following sections: Section 3.1 introduces the measures that software engineers use to describe the attributes of entities in software production. In Section 3.2, the criteria of the architectural framework that is used to build a general model for software production are described. The steps for data collection process before meta-analysis are detailed in Section 3.3. Once the datasets have been organized into a catalog, steps of validation in both mathematical and empirical ways are discussed in Section 3.4. In Section 3.5, the steps of applying meta-analysis to datasets are explained.

3.1 Introduction To The Architectural Framework

The attributes of entities in software production can be measured according to these three categories: resources, products and processes as shown in Figure 3.1. For examples, the entities of products include software programs, software documentation and software designs and the entities of processes are time, quality skill and etc.

Software engineers have adopted measurement theories to provide a common base to the measurement process that characterizes their attributes. Lines Of Code (LOC), Returns of Investment (ROI) and Cyclomatic Complexity (CC) [36] are some of the measure examples that are used to describe the attributes of entities.

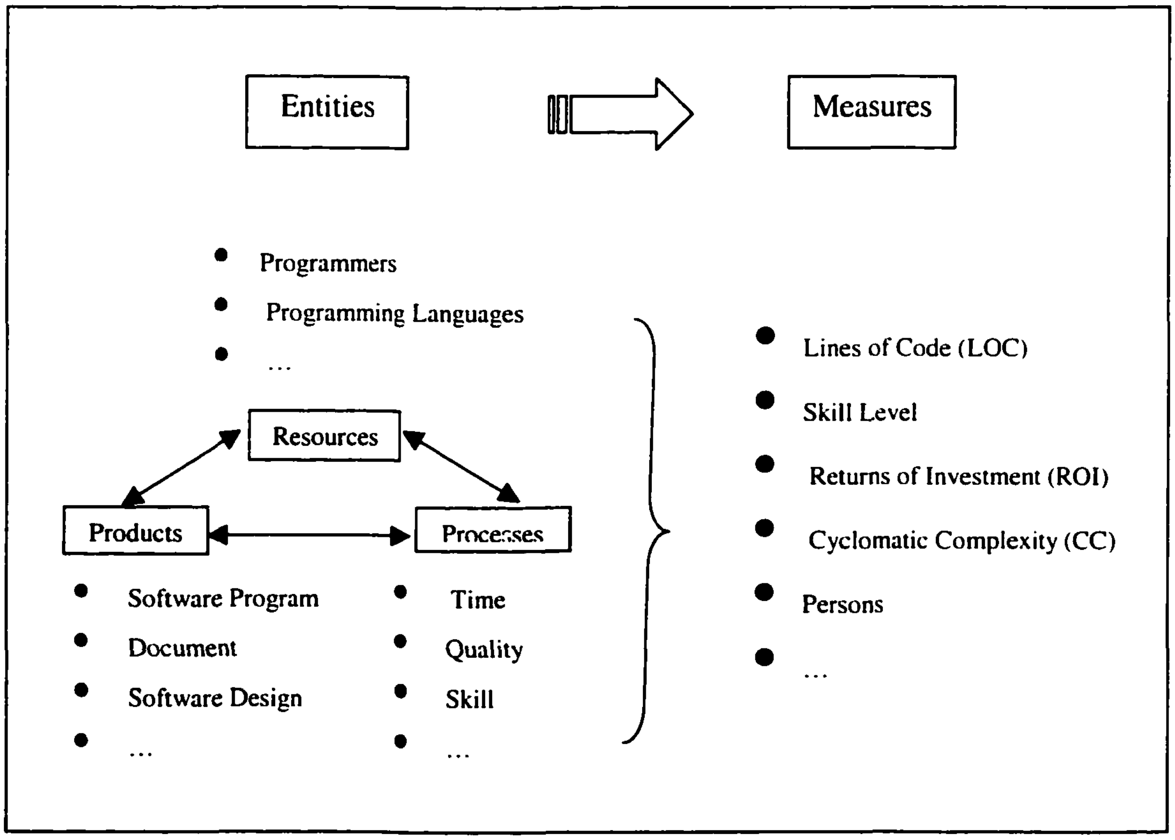
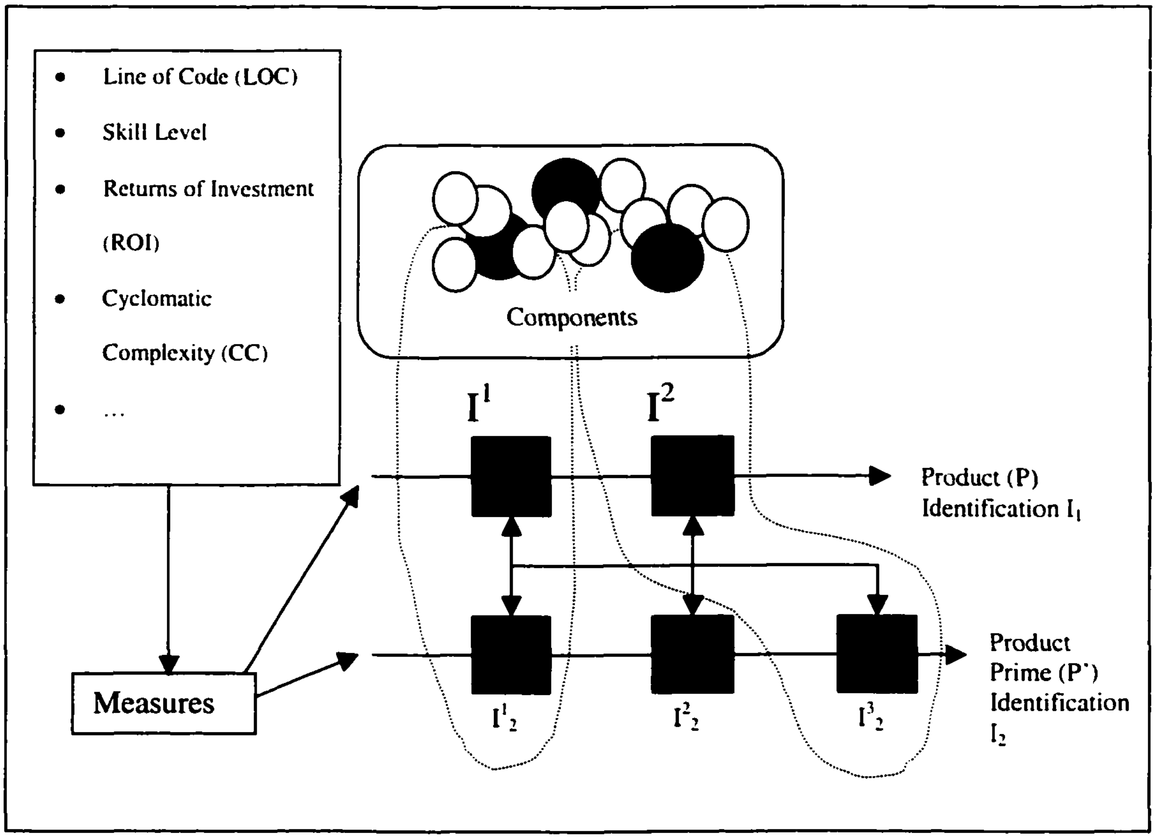


Figure 3.1: Measures Used to Describe The Entities in Software Production.

In modern industrial software production, each software product is made of software components. Software components include software libraries, user interface programs, word processor templates, databases and their access clients, etc. For example, Product 1 (P1), as illustrated in Figure 3.2, can include a database (e.g., Oracle) and some libraries.



Line of Code (LOC)
Skill Level
Returns of Investment (ROI)
Cyclomatic Complexity (CC)
...
Measures
Components
I^1
I^2
Product (P) Identification I_1
I^1_2
I^2_2
I^3_2
Product Prime (P') Identification I_2

Figure 3.2: Software Components used in Software Production.

Different software products can utilize the same software components. For example, in Figure 3.2, there are two different software production lines, namely Product (P) and Product Prime (P'). Since P1 and P2 are on the same production line, they are different versions of identical products. For example, product P1 can only be run on Microsoft Windows 3.11 and product P2 can be run on Microsoft Windows 98. Product

P2 is the "integrated" version of product P1 because product P2 has extra features that product P1 does not have.

In order to understand the characteristics of the products, resources and processes, measures are used to describe their attributes of entities. Software engineers collect these measures and manipulate them into useful information, which describes the characteristics of software production. To build a dataset, a software engineer will measure process, product, and resources employed during the construction of one or more similar products. The dataset can also be called "identification", because it represents one specific instance of the process.

Different software development teams produce distinct software products in many diverse ways. Software engineers measure the same attributes in very different ways. As a result, different measures are used to measure the same things. The implications of one measure may lead to a management decision opposite to the implication of another. Researchers and practitioners confuse which measure will be appropriate for an attribute. In order to construct a general model for software production, there are few requirements needed to establish the architecture framework.

3.2 Requirements of Architectural framework

The following are some essential requirements that enable software researchers and practitioners to generate general software models. Each of the criteria is carefully considered before meta-analysis is applied to the datasets.

3.2.1 Homogeneous Datasets

Each software project has its own unique nature that causes the engineering data to be different from one to another. As a result, software engineering data collected from different studies are unlikely to be homogeneous. The major consequence of a non-homogeneous dataset is that the large variances could make any statistical results insignificant. In order for the meta-analysis studies to be appropriate, the data from the previously published studies must be homogeneous [16].

3.2.2 Assumptions on Datasets

When software engineers do not have a population model of the entity they are measuring, they can use statistics to create one. For example, a set of modules has the average length of 180 lines with a standard deviation of 20 lines. If software engineers understand their population model, they can use statistics that represent that population to decide whether particular modules are normal or abnormal with respect to the population norms. A population model can usually be derived from the following considerations:

- Conditions or states that affect the entity - For example, language for programs, education for programmers, tool support for tasks.
- The goals of the measurement program – For example, investigating the effect of process changes on productivity and quality, or identifying entities with abnormal values.

The population modeling activity raises a few measurement issues. When software engineers derive statistics from measures on a set of entities, the statistics of central tendency and dispersion are constrained by the scale type being used. For instance, standard deviations can be used for ratio and interval scale measures, but medians and percentiles should be used for ordinal scale measures. The distribution of a set of dataset values also affects the choice of statistics. For example, one would prefer to use medians and percentiles for interval and ratio scale measures in datasets that are severely skewed. Even if each of the experimental datasets has normal distribution, one cannot ensure the combination of these datasets having normal distribution. In this thesis, the datasets in the combined studies are assumed to have normal distribution.

3.2.3 Existing Traditional Statistical Analysis Results on Datasets

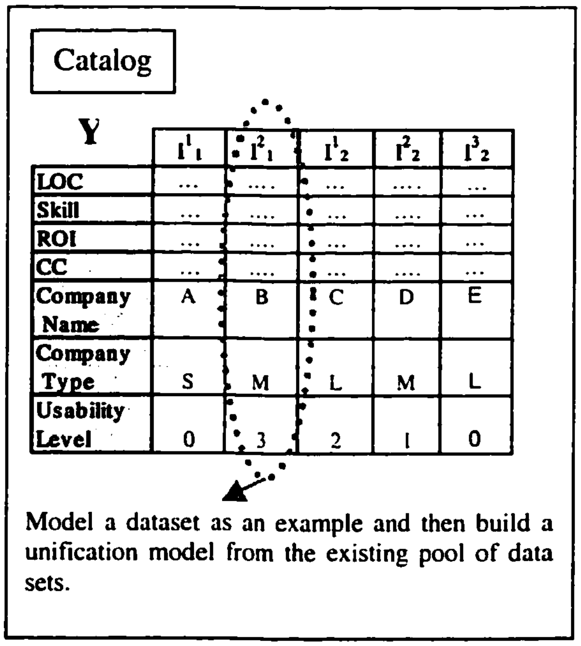
By applying meta-analysis across the correlation of datasets, the generalization of the relations between different software measures derived in the individual statistical studies can be established. For example, the sample data can be any type of correlation value, such as Pearson's product moment correlation coefficient and Spearman's rank correlation coefficient. Two of the measures from a number of projects, which have already been analyzed individually using standard correlation analyses between many attributes of interest relationships, can be taken as inputs to a meta-analytical study.

3.3 Data Collection

In order to develop a general model for software production, a central repository for all the valid and usable data is required before employing the datasets in the meta-analysis. In this thesis, a catalog is built to collect a pool of datasets for meta-analysis.

3.3.1Building a Catalog

The first step in meta-analysis is data collection. This is achieved by building a catalog, which collects previously published studies (identifications) and summarizes their findings. The catalog collects the pool of datasets from existing published studies, reorganizes and develops them into a multi-dimensional matrix. Figure 3.3 illustrates that the catalog models a dataset as an example and builds a unification model from this existing pool of datasets.


Catalog
Model a dataset as an example and then build a unification model from the existing pool of data sets.

Y	I^1_1	I^2_1	I^1_2	I^2_2	I^3_2
LOC	...	....	...	....	...
Skill	...	....	...	....	...
ROI	...	....	...	....	...
CC	...	....	...	....	...
Company Name	A	B	C	D	E
Company Type	S	M	L	M	L
Usability Level	0	3	2	1	0

Figure 3.3: Catalog Which Collects The Pool of Datasets.

The measures used to describe the attributes from the datasets are identified and they are validated for mathematical and empirical meaningfulness [36]. Studies that do not meet the representational theory of measurement [36] are not considered. The result is a comprehensive and reliable data source. This catalog then becomes the central repository for all the valid and usable data to be employed in the meta-analysis.

3.4 Validation

When lacking software validation, software engineers are uncertain on how to choose any appropriate measure that actually captures the attribute information they seek. In addition, a validation must also take into account the purpose of measurement. According to Fenton and Pfleeger, "validating a software measure is a process of ensuring that the measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied" [36]. Fenton and Melton et al. suggest that a valid metric must obey the representational theory of measurement, so that the intuitive understanding of an attribute is preserved when it is mapped to a numerical relation system [2], [33].

Practitioners and researchers have proposed many validation frameworks for software measurement, but it is not clear that which approaches actually lead to a widely accepted view of software measurement validity. Chidamber and Kemerer [44] refer to Weyuker's properties to discuss their measures for object-oriented designs, while Zuse [55] claims that at least two of Weyuker's properties are inconsistent.

New measures are being justified according to doubtful criteria and some commonly used measures may not be valid according to any widely accepted criteria.

Zuse criticizes Weyuker's complexity measures properties as contradictory because her property 5 implies a ratio scale and property 6 explicitly excludes a ratio scale. Both the properties are regarded as inadmissible because they imply a scale type. Kitchenham et al. also regard Weyuker's property 9 as inadmissible because it involves the relation "<" and thereby excludes nominal scale units [8].

Since there is no standard scientific principle existing in the software measurement community yet, some software measurement researchers choose a validation method by reference to a specific author's viewpoint. Unless the software measurement community can agree on a valid, consistent and comprehensive theory of measurement validation, both practitioners and researchers may be in a potentially calamitous situation.

The software measurement validation framework is used to determine whether a measurement unit being used is an appropriate means of measuring the attribute and any model underlying a measuring instrument is valid. This validation framework ensures that the measuring instrument is properly calibrated and an acceptable measurement protocol is adopted. The following are some of the procedures for software measurement validity. There are two basic methods of testing the validity of software measurement: theoretical validation and empirical validation.

3.4.1 Theoretical Measurement Validation

Theoretical validation confirms that the measurement does not violate any necessary property of the elements of measurement. For example, a scale type determines the admissible transformations. Therefore, a compound unit must be constructed by transformations permitted by the scale type of its components.

A measurable attribute must allow different entities to be distinguished from one another. Weyuker's first property states that "two entities must exist for which the measure results in different values". Since a valid measure must obey the representational theory of measurement, it must preserve the intuitive notions about the attribute and the way in which it distinguishes different entities. Weyuker's eighth property states that if one has two same programs except that one has given them different labels, their complexity measures must be the same.

A measurement protocol is associated with obtaining a measurement value and the use to which the measure will be put. When software engineers have decided to measure a specific attribute on a certain entity by using a particular measurement unit for a specific purpose, a measurement protocol can be defined. Measurement protocols must be unambiguous, self-consistent and prevent problems such as double counting.

3.4.2 Empirical measurement validation

Empirical validation verifies that any measured values of attributes are consistent with the values predicted by models involving the attribute. For instance, the measured distance from an origin to a point is the square root of the sum of the x and y coordinate values. To corroborate a measure, software engineers can perform experiments to see whether people agree that an attribute exists or whether a mapping to a value captures their understanding of the attribute. Attribute validity must be considered both for directly measurable attributes and for indirectly measurable attributes that are derived from other attributes.

3.4.2.1 Direct Measures validation

For direct measures, software engineers can query a random selection of individuals to classify a set of entities according to a set of possible categories for nominal scale measures or to rank the set of entities with respect to the attribute of interest for ordinal, interval or ratio scale measures. Software engineers can then use measures of association identified by Siegel and Castellan [45] to access how consistently a group of people categorize or order a set of entities.

For example, measures of association for ordinal attributes vary from –1 (perfect inverse association) to 1 (perfect association) with 0 indicating no association. When software engineers consider the consistency among subjects, the measures of association need not be perfect but should not be significantly different from 1. If the attribute is valid, software engineers then investigate the validity of the measurement unit.

If a measurement unit is accepted as a valid mean of measuring an attribute, an alternative unit will be valid if it is an admissible transformation from the original unit. For example, if Lines Of Code (LOC) is accepted as a valid measure of program length, then Thousand Lines Of Code (KLOC) is also valid. Many direct measures at least conform to the representational theory of measurement. Such measures include LOC as a measure of a program length, cyclomatic number as a structure measure through a program, effort in person hours as a measure of the human resources need for a task and many others.

3.4.2.2 Indirect Measures validation

Indirect measures are based on attribute relationship models. If a measure has been derived from a proven underlying model, the use of the measure can be justified by reference to that model. Otherwise, software engineers can use standard statistical techniques such as regression to corroborate that an indirect measure is identical to a direct measure.

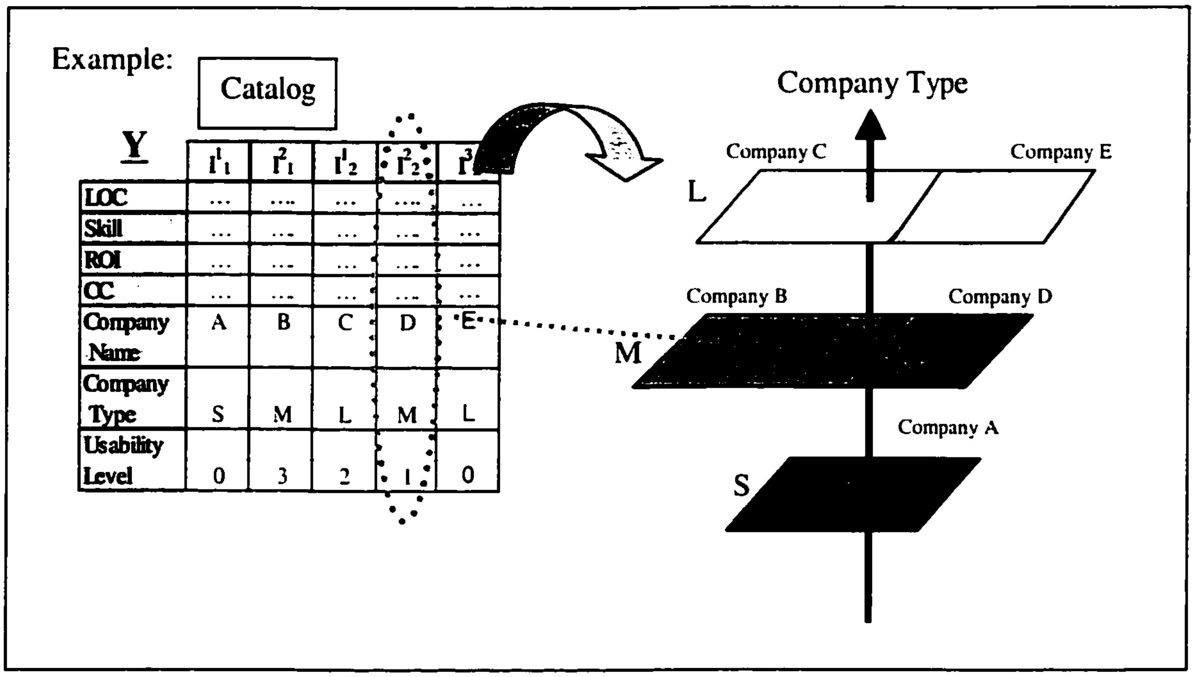
To determine if one measure can be used to predict the value of another, software engineers need to seek whether or not there is a relationship between the two attributes being measured. For example, medical researchers have observed a relationship between smoking and lung cancer and software researchers have identified relationships between structural complexity measures and fault rates. Basic statistical techniques such as correlation analysis can be used to investigate relationships between attributes.

3.5 Steps of Meta-Analysis

After the data collection process, a catalog that collects previously published studies and summarizes their findings is built. The measures used to describe the attributes in the datasets, are identified and validated for mathematical and empirical meaningfulness as described in the previous section.

As a result, the catalog becomes the central repository for all the valid and usable data to be employed in the meta-analysis. Studies that do not meet the representational theory of measurement [36] are eliminated. The result is a comprehensive reliable data source. The following sections are the steps of meta-analysis that employed to the validated datasets.

3.5.1 Transformation of Datasets



Y	I^1_1	I^2_1	I^1_2	I^2_2	I^3
LOC	...	....	...	....	...
Skill	...	...	...	...	...
ROI	...	...	...	...	...
CC	...	...	...	...	...
Company Name	A	B	C	D	E
Company Type	S	M	L	M	L
Usability Level	0	3	2	1	0

Figure 3.4: Dimensional View of Datasets and Regression Relationships of Measures.

The datasets collected in the catalog can be represented using different dimensional views. For example, figure 3.4 demonstrates the transformation of datasets from the catalog into a three-dimensional-plane diagram, by selecting Company Type as its y-axis. The datasets for company A to E are plotted into a surface plane correspondingly. Since company C and E have the same company size (Large, L), they are plotted into the same plane, and are not necessary separate.

3.5.2 Linear Regressions and Meta-Analysis

Each dataset on the same plane can be analyzed, for example to compute a linear regression. Figure 3.5 also shows a linear regression analysis example projected from the Company-C-E plane. This diagram has both the linear regression lines for company C and company E. By using a meta-analysis statistical method, a new linear regression line which describes the relationships of LOC and Skill for both of the companies, is generated.

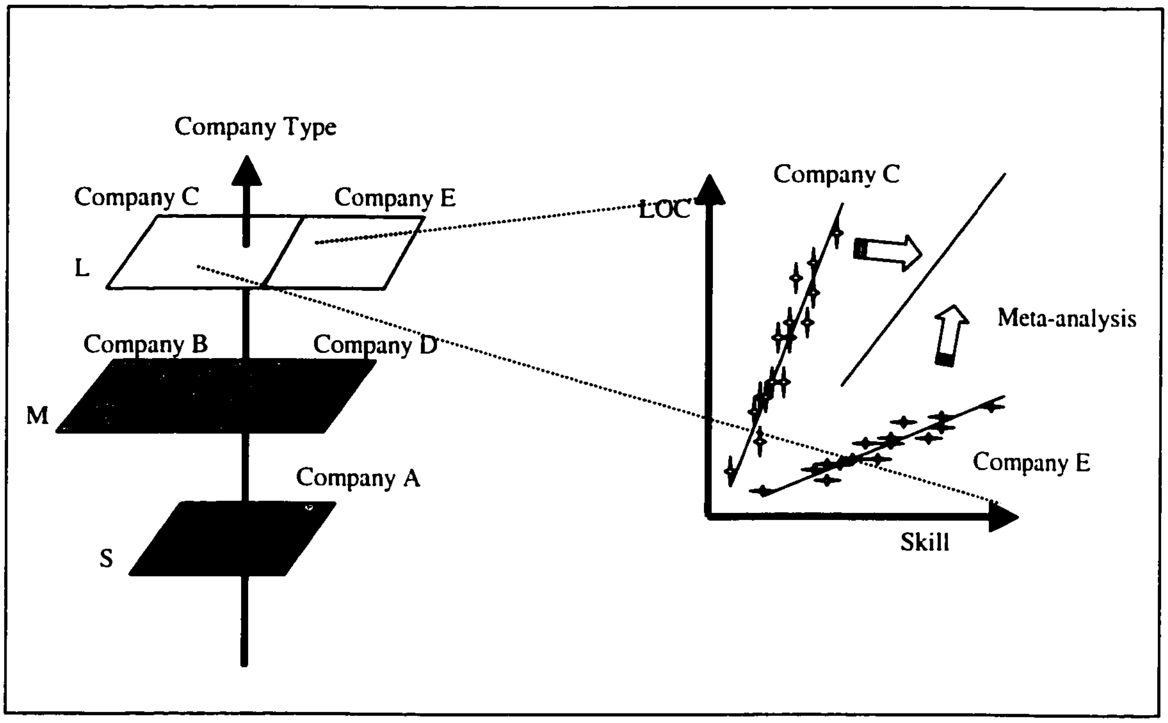


Figure 3.5 Regression Relationships Among Measures.

From the linear regression line, one can understand the common characteristics of the relationships between, for example, LOC and skill level for company size Large (L). After the meta-analysis process is applied to each company size, a linear regression line representing each company size is generated.

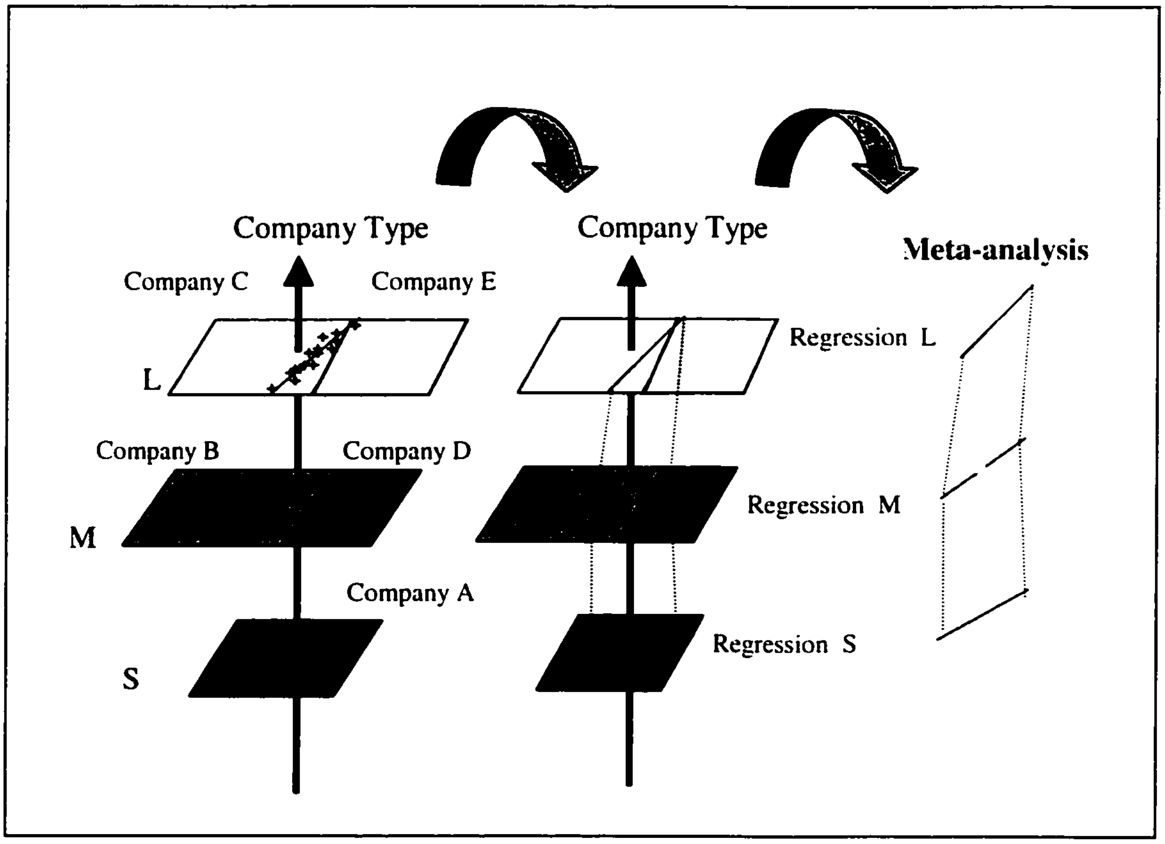


Figure 3.6: General Regression Line Across Different Datasets.

If the catalog has three different company sizes, such as Large (L), Medium (M) and Small (S), there will be three common linear regression lines. Figure 3.6 simplifies the meta-analysis process, which is used to obtain a general linear regression line across different datasets for three company sizes. From the general linear regression line, one can understand the common characteristics of the relationships between LOC and skill level for different company sizes. Hence, this general linear regression line can be used to estimate the relationships between LOC and skill level in a future software development project.

Chapter 4

Approach

As is mentioned in Chapter 3, there are steps to approach the meta-analysis. This chapter presents the theoretical result of meta-analysis and the space representation of these datasets. Section 4.1 describes the steps in detail with examples to approach the architecture framework of a general model. Goal Question Metrics (GQM) is used to set the goal of a general model. The measures of attributes are validated empirically and mathematically for meaningfulness according to the representational theory of measurement [36]. Finally, multiple features space representation of datasets is depicted in this section. Section 4.2 characterizes the meta-analysis for point estimates and correlation estimates, an example for each estimate is given following the procedures of meta-analysis. Finally, meta-analytical procedures for the estimation of regression lines on surfaces are presented and explained.

4.1 The Approach to build the architecture framework

Meta-analysis can be expanded into a five-stage conceptual framework: formulating the problem, collecting the data, evaluating the data, synthesizing the data and presenting the result. During the process of building catalog, a problem of software development can be characterized. Figure 4.1 summarizes the meta-analysis conceptual framework.

Once software engineers formulate a problem, they decide which of the gathered data is valid and usable. Goal Question Metrics (GQM) helps in forming the solutions and characterizing the problem according to the business objectives and the achievement of the goal. Based on the GQM results, software engineers identify and actualize the measures of attributes. They eliminate the studies that do not meet standards and possibly remove outliers from extreme cases.

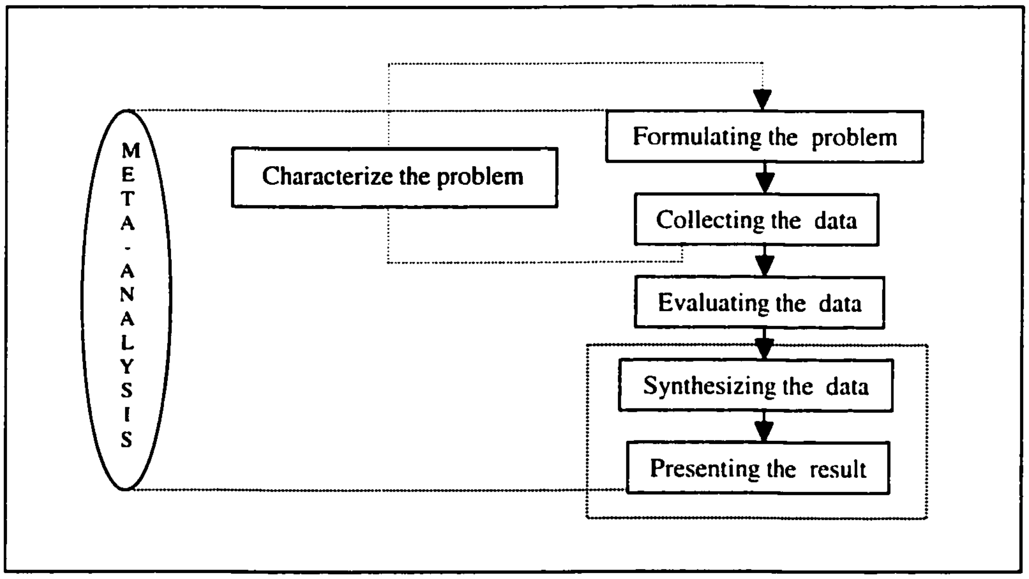


Figure 4.1: Summary of Meta-analysis Conceptual Framework.

Software engineers justify the validity and the applicability of the attributes if they are theoretically and mathematically meaningful. They perform Hypothesis Test (H0 & H1), select a level of confidence (α) and finally perform Statistical Tests (Parametric Tests and Non-Parametric Tests).

Software engineers then synthesize the validated datasets and accomplish an estimator for a measure by using meta-analysis. They explore a relationship and confirm a theory for the measures and parameters. Finally, they present this analytical result that explains the characteristics of software production in term of specific factors [36]. For example, software engineers can understand the constitution of a project based on a general software production model in term of the number of programmers, the experiences of the programmers and the programming language being used for the project.

4.1.1 Goal Question Metrics

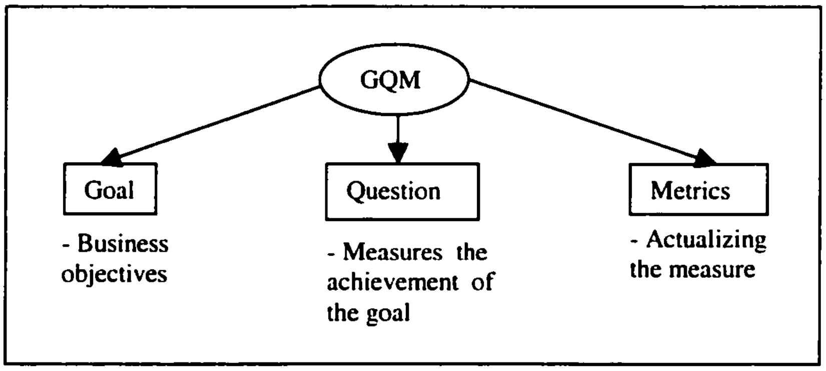

GQM
Goal
- Business objectives
Question
- Measures the achievement of the goal
Metrics
- Actualizing the measure

Figure 4.2: The Structure of GQM Framework.

Many metrics programs measure what is convenient or easy to measure, rather than measure what is needed. Consequently, the resulting data is not useful to software engineers. By using GQM, the overall goals of an organization will be first expressed to ensure the scope of metrics and goals is up to the organization needs [36], [50].

Then, GQM guides the organization to generate questions whose answers software engineers must know in order to determine if the organization goals are being met. Finally, software engineers analyze each question in terms of what measurements they need in order to answer to each question. Figure 4.2 depicts the structure of GQM framework.

4.1.2 Validation of the measures

The measures reflect the behavior of entities in the real world. If the validation of measures cannot be done, no one can be sure about the decisions made according to those measures and if they will have any expected effects. Validation makes sure that measures are defined properly and are consistent with the real world behavior of entities.

The example provided in this thesis first identifies the measures for two attributes and proves their meaningfulness according to the representational theory of measurement. Some rules are defined to collect the experimental data. The samples of the experiment are categorized into different groups based on the interests of classification. These experimental procedures avoid the analysis of the experiment from being biased. This example also identifies and generalizes the sample to a population. A statistical tool is chosen to find the proof. The following sections are the procedures used to validate the measures.

4.1.2.1 Example of Identifying A Measure for Two Attributes

As each product of software development is a physical entity, measuring the size of software products should be easily measurable, straightforward, relatively simple and consistent with the measurement theory principle. The examples provided in this thesis are the attributes of software complexity and software productivity.

Software Complexity = Size (Program Length) = *Lines of Code (LOC)*

Software Productivity = Size (Program Length) / Effort = $\frac{LOC}{PersonHours}$

4.1.2.2 Prove that the attributes are mathematically meaningful

Software engineers ensure that their measures satisfy the representational theory of measurement. The representation condition preserves the empirical and numerical relations when a measure of attributes is mapped into numbers and empirical relations are mapped into mathematical relations. The following section describes the steps to validate the measures of attributes.

4.1.2.2.1 Identify the measure scales and prove their meaningfulness

It is a common mistake to assume that LOC is an absolute scale measure of length, because it is obtained directly by counting the program lines of code. The length of program cannot be an absolute scale measure since there are many different ways to measure it. The program length can be measured in a variety of ways, including the number of characters contained in the program, number of bytes, LOC, thousands of LOC (KLOC) and more [36].

Besides, an empty piece of code is the notion of a zero-length object. Therefore, the length of software code is measurable on a ratio scale. The Absolute Scale type has the defining classes of increasingly restrictive admissible transformation. Hence, LOC is an absolute scale measure of the attribute "number of LOC" of a program.

Validation:

Suppose M is the measure of the program length in lines of code, while M' captures length as a number of characters. The measure can then be transferred one to the other by computing $M' = \alpha M$, where α is the average number of characters per line of code.

LOC = α number of characters *where* $\alpha = 38$,

M(18 LOC) > M(8 LOC)

and M'(18 α number of characters) > M'(8 α number of characters)

i.e. M'(684 number of characters) > M'(304 number of characters)

The "Person Hours" measurement can be represented by the "Person Months" "person week", "person day" and more. This measurement has the ratio transformation of the form $M' = \alpha M$ where α is positive scalar (constant).

Validation:

PersonHours = α person day *where* $\alpha = 1/24$,

M(3 Person Hours) > M(2 Person Hours)

and M'(3 α person day) > M'(2 α person day)

i.e. M'(1/8 person day) > M'(1/12 person day)

When αM is substituted for M' in the above examples, the true value is preserved under the admissible transformation. The measures can be mapped from one into another by multiplying a suitable positive constant, α. Since the definition of meaningfulness requires the true value to be preserved, the measure "LOC" and "Person Hours" are in a mathematically meaningful ratio scale.

The mathematical mapping preserves the notions of length in the relations that describes the programs. If the length of a program is measured by counting the numbers of LOC and the count preserves the relationships, the LOC measure can be defined as a valid measure of program length.

Validation:

Concatenate two programs P1 and P2 to get a program P1:P2 whose length is the combined length of P1 and P2. Any measure m of length has always to satisfy the condition:

$$m(P1:P2) = m(P1) + m(P2)$$

4.1.2.2.2 The Indirect measurement on the attributes

Productivity is an external resource attribute, since it depends on the underlying development process. For the software development process, the measure of output is usually computed as the number of LOC produced as the final product, while the input measure is the number of person hours used to specify, design, code and test the software.

The scale types are used as the ability to determine which representation is the most suitable for measuring an attribute of interest. One relation system is said to be richer than another is if all relations in the second are contained in the first. The richer the empirical relation system, the more restrictive the set of representations and the more sophisticated the scale of measurement. Therefore, every meaningful statistic of a ratio scale type is also meaningful for an absolute scale type.

Since the measure "number of LOC" is in absolute scale, it inherits the properties of a ratio scale as well. Thus, the measure of the attribute "Software complexity" in "LOC" (the size) is mathematically meaningful as a ratio scale. The scale type for an indirect measure will generally be no stronger than the weakest of the direct scale type. If the measure M contains a mixture of ratio, interval and nominal scale types, the scale type for M will be the nominal scale (the weakest).

In this example, the measure of attribute "software productivity" is an indirect measurement composed of the two measures "LOC" (size) and "Person Hours" (effort). Even if the measure "LOC" is assumed as the absolute scale, the indirect measure of attribute "software productivity" will be considered as a ratio scale because the weakest scale type in this mixture is ratio scale.

In addition, the "software productivity" (SP) measurement which is expressed by $SP = LOC/PersonHours$ should meet the rescaling (admissible transformation) of SP to the form of $SP' = \alpha SP$ (for $\alpha > 0$). Suppose that the measurement of each n sub-attributes with measure M_i.

Let M be an indirect measure involving components $M_1, M_2, \ldots, M_n$.

$M = f(M_1, M_2, \ldots, M_n)$

M' is a rescaling of M if there are rescalings $M'_1, M'_2, \ldots, M'_n$ of $M_1, M_2, \ldots, M_n$ respectively, such that

$M' = f(M'_1, M'_2, \ldots, M'_n)$

Validation:

Since

$$SP = \frac{LOC}{PersonHours}$$

$$\alpha SP = \frac{LOC'}{PersonHours'} = \alpha\left(\frac{LOC}{PersonHours}\right) = \frac{\alpha LOC}{PersonHours}$$

Therefore, *SP* has "rescaling".

Since LOC and PersonHours are ratio scale measures, every rescaling of LOC must be of the form α_1LOC for some α_1 and every rescaling of PersonHours must be of the form α_2LOC for some α_2. Therefore, every rescaling of PersonHours has the form

$$\frac{\alpha_1 * LOC_1}{\alpha_2 * PersonHours} = \frac{\alpha_1}{\alpha_2}\left(\frac{LOC}{PersonHours}\right) = \frac{\alpha_1}{\alpha_2}(SP) = \alpha SP \qquad where\ \alpha = \frac{\alpha_1}{\alpha_2}$$

4.1.2.2.3 Validation Criteria of the measures

The data collection process for a measure must be valid and applicable according to the representational theory of measurement, otherwise the datasets are useless. In this example, the measure LOC will be taken by counting the number of single non-commented and executable LOC. For example, a single LOC with two executable statements will be counted as two LOCs.

The version of the programming language and the operating systems platform used in the experiment will be specified clearly to reduce the possibility of errors. If there are two operating system platforms, the experiment will be classified into two groups. If there is more than one programming language involved during the data collection process, the programs should be classified according to their language group.

To analyze and model productivity, one measures a resource attribute such as personnel either as a team or individuals active during particular processes. Since software productivity is viewed as a resource attribute, it can be captured as an indirect measure of product attribute measures and a process measure. To create a good experimental design, the factors such as programmer experience, application type or development environment should be controlled so that they will not confuse the results.

The experience or intelligence of a developer may affect the quality of the design or code. The size, structure and communication patterns of the development team are important. Classification methods and analysis tools will also cause the result [36].

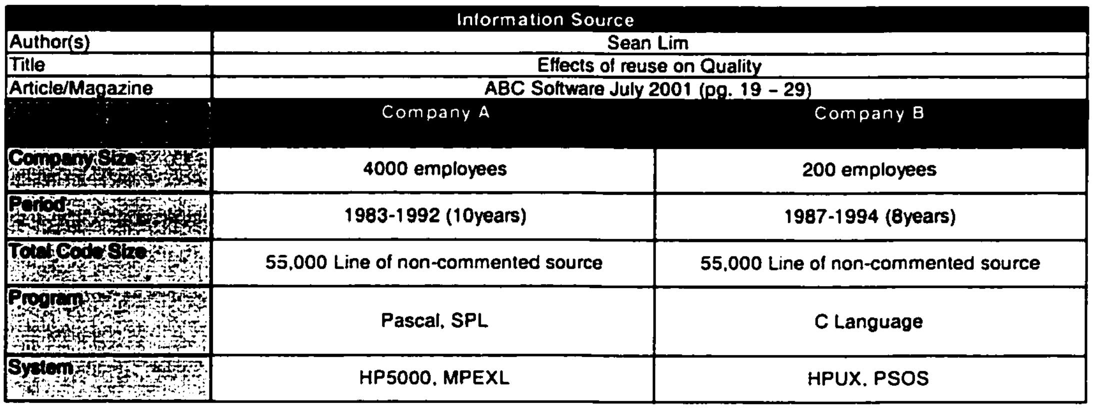
For example, programmers are experienced or not, languages are block structured or not, object oriented or not and techniques can be rated as manual or automated, tools can require special training or experience. All these attributes of resources will help one to understand on how to use tools and methods in more effective ways.

4.1.3 Space Representation of Data Sets

The datasets collected in the catalog consist of various published papers and hence the datasets collected are in different data formats. Generally, they are raw datasets and significant measures datasets.

4.1.3.1 Measures and Parameters

Software engineers select different measures to describe the attributes of entities, such as Lines Of Code (LOC). Besides the measures selected in the example, there are various parameters contributed to the datasets. For instance, the company size, the programming language and the computer systems of both companies are the parameters. Each parameter contributes to the controlled experiment environment in a company and hence controls the results of the projects. The combination of measures and parameters characterize a software production model.



	Information Source	
Author(s)	Sean Lim	
Title	Effects of reuse on Quality	
Article/Magazine	ABC Software July 2001 (pg. 19 - 29)	
	Company A	Company B
Company Size	4000 employees	200 employees
Period	1983-1992 (10years)	1987-1994 (8years)
Total Code Size	55,000 Line of non-commented source	55,000 Line of non-commented source
Program	Pascal, SPL	C Language
System	HP5000, MPEXL	HPUX, PSOS

Figure 4.3: A Sample of Datasets Collected in The Catalog.

Figure 4.3 shows a fraction of a dataset collected from the catalog. For example, although both “Company A” and “Company B” have the same amount of “total code size”, “Company A” with 4000 employees is the parameter, which differentiates it from “Company B” with 200 employees. Since each of the datasets can be composed of various parameters, the datasets can be presented in a multi-dimensional space.

4.1.3.2 Space Representation

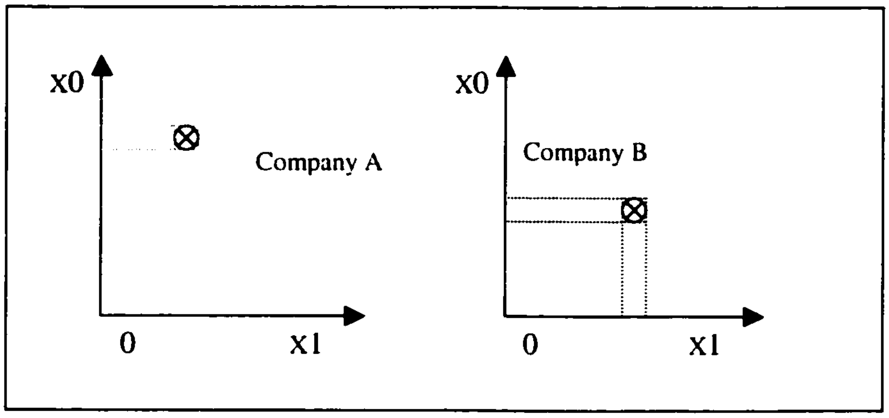


Figure 4.4: Two-dimensional Space Representational Diagram.

Each of the significant measures may be contributed by various parameters. When the data sets are collected from the significant measures, they can be represented in a two-dimensional diagram as illustrated in Figure 4.4. The co-ordinate points for "Company A" and "Company B" are formed by two parameters, namely X0 and X1.

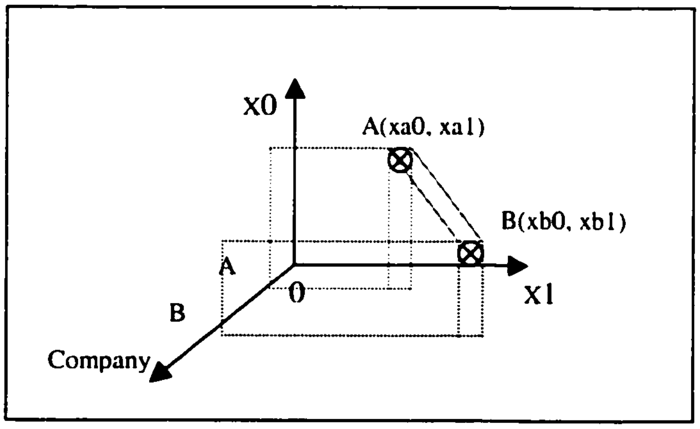


Figure 4.5: Three-dimensional Space Representational Diagram.

Since the co-ordinate points are composed of more than one parameter, software engineers can extend the view from two-dimension to multiple-dimension. For example, "Company" is one of the parameters used to determine the characteristics of the points. Figure 4.5 demonstrates a three-dimensional diagram for the significant measures with three parameters (X0, X1 and "Company").

4.1.3.3 Example of Meta-Analysis in Space Representation

As mentioned before, a significant measure can be represented in a multiple-dimension view. Figure 4.6 shows a process to obtain the meta-analysis results across the datasets collected from various companies with different parameters. In this example, Line of Code (LOC) and Person-Month are used to measure the productivity of software project in both companies. The datasets collected for projects running in "Company A" and "Company B" are plotted in Figure 4.6a with a two-dimensional view diagram.

"Company A" has collected the measurement data of two projects. The linear regression lines for each of the projects are plotted in figure 4.6a. Meta-analysis is used to find the general linear regression line, "Meta-Analysis 1", for "Company A". The same meta-analysis process is used to get the general linear regression line, "Meta-Analysis 2", for "Company B" as illustrated in Figure 4.6b and Figure 4.6c. Meta-analysis in this example is used to estimate the regression line.

The linear regression lines "MA1" and "MA2" form a surface in a three-dimensional diagram. By using linear interpolation methods, each point from the linear regression line "MA1" is linked to "MA2". Figure 4.6d shows a surface formed by the regression lines.

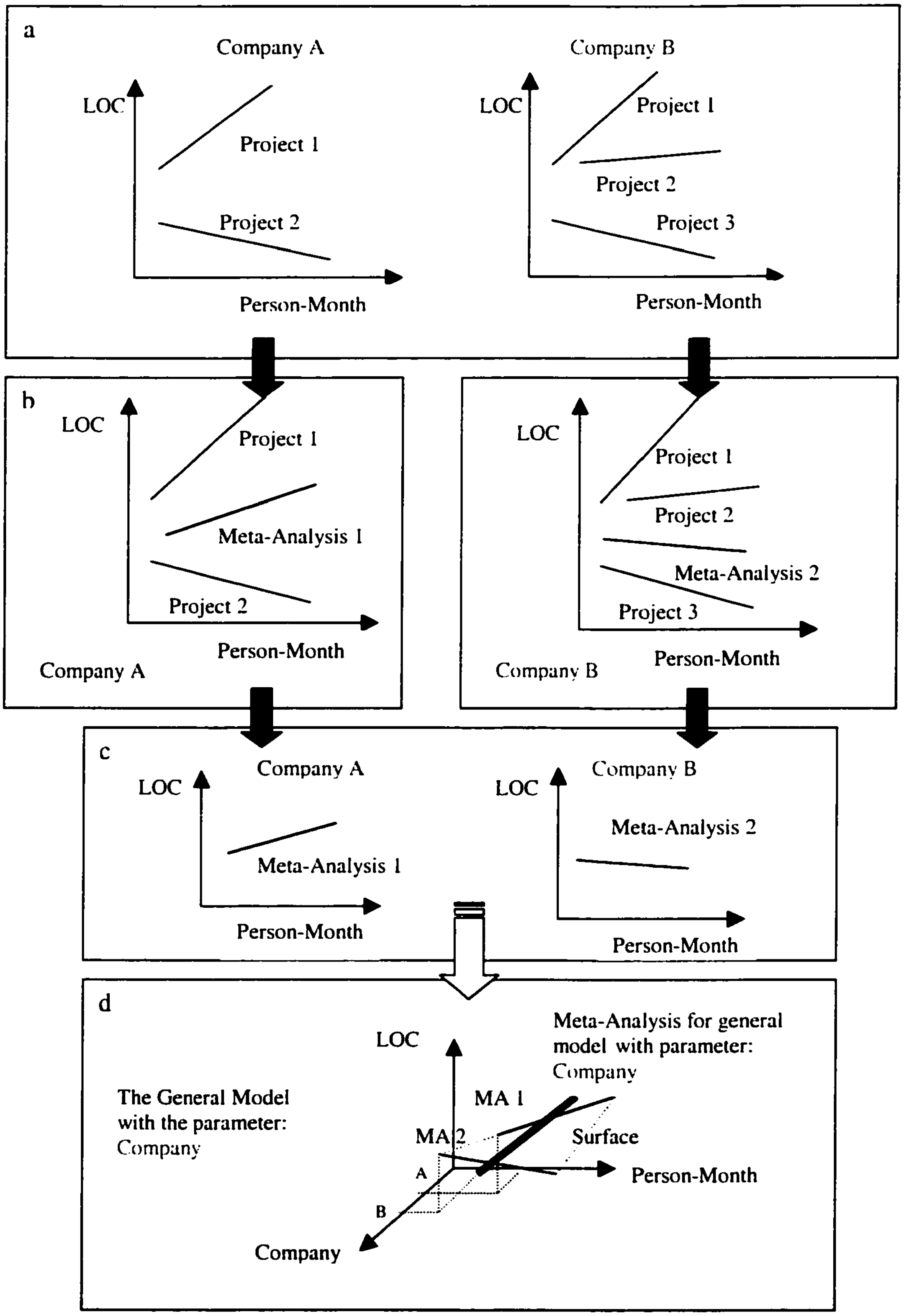


Figure 4.6: Process to Obtain The Regression Line Across Datasets With Different Parameters.

Once the linear regression lines for both the companies are found, a general linear regression line that describes the productivity characteristics of both companies in LOC and Person-Month can be generated by using meta-analysis.

By choosing one of the common parameters from the datasets, the general linear regression line is established as shown in Figure 4.6d. "Company" is chosen as the parameter in this example. This linear regression line can be used to estimate the productivity of future projects in other companies. The meta-analysis operation in this example is used for the surface estimates.

4.2 Meta-Analysis for Estimates

Sampling error can be corrected for, but the accuracy of the correction depends on the total sample size that the meta-analysis is based on. The correction for sampling error becomes perfect as total sample size approaches infinity. If the number of sample studies is small, there will be a sampling error in the final meta-analysis results. Hence, software engineers implicitly assume that the meta-analysis is based on a very large number of studies [36]. The number of points in each sample will also affect the meta-analysis result.

If there is variation in actual correlations, that variation must be caused by some aspects of studies that vary from one study to the next. This variation is defined as a moderator variable. Software engineers consider the measures obtained from the productivity with their interval of confidence and they average the measures by taking account of the weight of moderator. The operation of this statistical analysis is called meta-analysis.

4.2.1 Meta-Analysis for Point Estimates

Software engineers use a set of points to estimate a new point by using statistical tools. For example, software engineers run different projects with the same environment and they obtain different productivity results. There could be some controlled parameters, some unidentified factors or errors which have contributed to the results. When software engineers use the datasets to estimate a point, they take into account these unidentified factors or errors by considering their interval of confidence and mean.

A measure of central tendency is some typical value or average to represent or describe the entire data set. Four types of averages often used as measures of central tendency are the arithmetic mean, the median, the mode and the midrange. By formally examining the central tendency of data sets, software engineers can summarize, understand and convey the information contained in a set of data.

4.2.1.1 Example for Point Estimates

"Company A" developed 10 projects in 5 years with the same parameters such as programming languages and computer hardware. Software engineers measure the duration of projects as their productivity. Since each of the project teams has various skills level of software programmers, skill level will be the moderator effect for meta-analysis.

Software engineers average the duration of each project and take into account the weight of the moderator contributed to each project. Based on the averaged results, software engineers are able to estimate the measure for a new project.

4.2.2 Meta-Analysis for Correlation Estimates

Correlation analysis is used to measure the strength of the association between two variables and measure the degree of relationship or covariation that exists between them. Based on the correlation relations of two parameters, software engineers estimate a new correlation value. There are effects of sampling errors on the correlation study. At the level of a single study, sampling errors are random events. At the level of meta-analysis, sampling errors can be estimated and corrected for.

When software engineers find the mean of the correlation value, the sampling errors are averaged. The sampling error in the mean correlation is the average of the sampling errors in the individual correlation. If all studies were conducted perfectly, the distribution of study correlations could be used directly to estimate the distribution of actual corrections.

If the total sample size is large, there is very little sampling error in the average correlations. The variance of correlations across studies is the average squared deviation of the study correlations from its mean. Squaring the deviation eliminates the sign of the sampling error and hence eliminates the tendency for errors to cancel themselves out in summation.

To eliminate the effect of sampling errors from meta-analysis, the distribution of population correlations can be derived from the distribution of observed correlations. Software engineers replace the mean and standard deviation of the observed sample correlations by the mean and standard deviation of population correlations.

Software engineers consider the measures obtained from the productivity with the mean and standard deviation of population correlations. They average the measures and take into consideration the weight of the moderator. The operation of averaging correlations across studies with the correction of these errors is called meta-analysis for correlations estimates.

4.2.2.1 Example for Correlation Estimates

Software engineers collect a dataset of correlations between two measures, LOC and PersonHour. Based on the correlations coefficient of these measures, software engineers establish a general equation which represents the relationships of the measures. The general equation can be used to estimate the next value of correlations between the measures.

4.2.3 Meta-Analysis for Regression Line Estimates

The purpose of using regression analysis is for modeling and prediction. The development of a statistical model can be used to explain variability and to predict values of the dependent variable (Y-variable). By using regression lines obtained from previous studies, software engineers can estimate a general regression line by using meta-analysis.

Effect size is an indicator of the strength of the relationship under study. It may be treated as a parameter which takes the values zero when the null hypothesis is true and some other non-zero values when the null hypothesis is false. The two most popular effect size measures are Cohen's "d" and Pearson's product moment "r".

Cohen's "d" effect size are standardized by standard deviations, effect sizes from different studies are comparable and therefore can be combined to integrate findings across studies. Pearson's "r" measures the strength of the linear relationship between two variables. Although "d" and "r" are transformable statistically, they serve different proposes. "d" is used for experimental studies and "r" is used for non-experimental studies.

4.2.3.1 Example for Regression Estimates

Software engineers average the interval of confidences, sampling errors and the variances of the regression lines. They compute the variance of regressions and weigh each by its sample size. By selecting an appropriate type of effect size, software engineers compute the variance expected solely on the sampling error and the mean distant of the regression lines with sample size weighted.

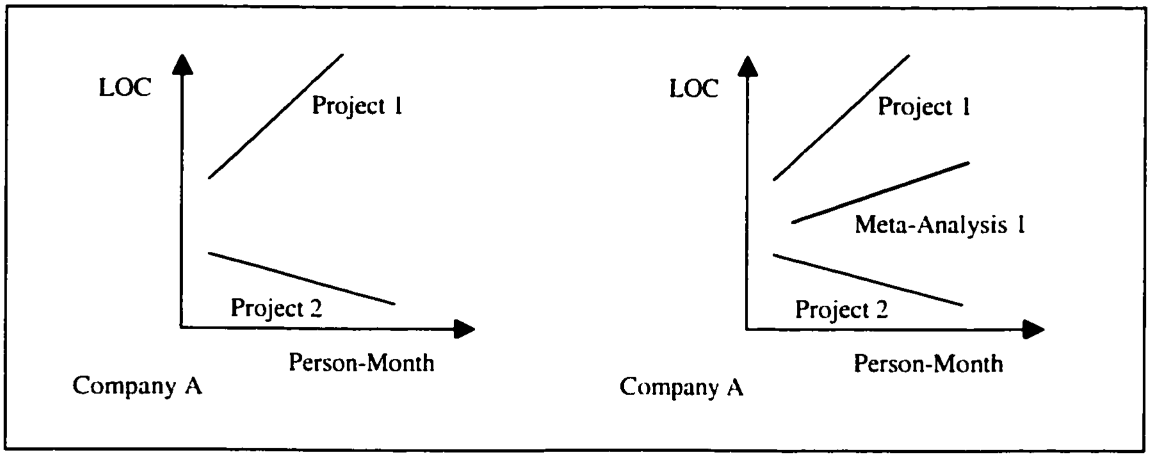


Figure 4.7: Using Meta-analysis to Estimate A Regression Line Through Other Regression Lines.

They then compute the sum of squared variances to minimize the distance from each regression line and reduce errors. Figure 4.7 shows the operation of meta-analysis by combining the information obtained from two projects. This meta-analysis operation estimates a new general regression line for company A. The general linear regression line captures the information of two projects and software engineers can use this general linear regression line to estimate the future software development project.

4.2.4 Meta-Analysis for Surface Estimates

When software engineers collect datasets of regression lines with various parameters, they can determine the moderator effect from the datasets and estimate a new regression line. The operation of averaging regression lines across studies with the correction of the errors is called meta-analysis for surface estimates [36].

4.2.4.1 Example for Surface Estimates

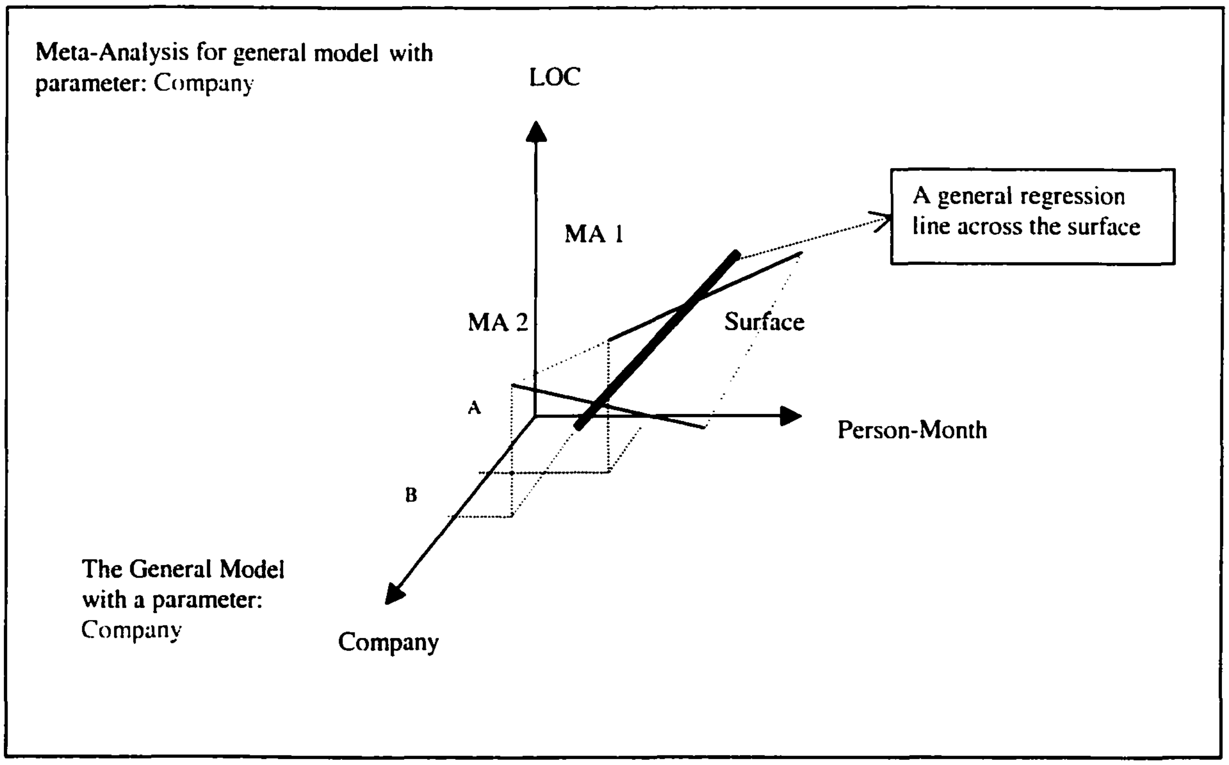

Meta-Analysis for general model with parameter: Company
LOC
MA 1
MA 2
A general regression line across the surface
Surface
A
Person-Month
B
The General Model with a parameter: Company
Company

Figure 4.8: Using Meta-analysis to Estimate A Regression Line Through A Surface.

Figure 4.8 shows the space representational view of few datasets of correlation relations between two measures, LOC and PersonHour, which are collected from various projects in different companies. A regression line is obtained from each dataset. There are numbers of regression lines presented by taking a three-dimensional space representation view on the datasets. The linkages of each regression line form a surface. Software engineers average the sampling errors and variances of the regression lines and estimate a new general regression line across the surface. By using this new general regression line, software engineers can estimate a new regression line.

Chapter 5

Validation

The architectural framework that can be used to build a general model for software production was described in Chapter 3 and Chapter 4. In this chapter, a validation example of the framework is provided. The chapter is organized into 2 sections. Section 5.1 describes the implementation of the architecture framework and section 5.2 delivers the validation results of the example.

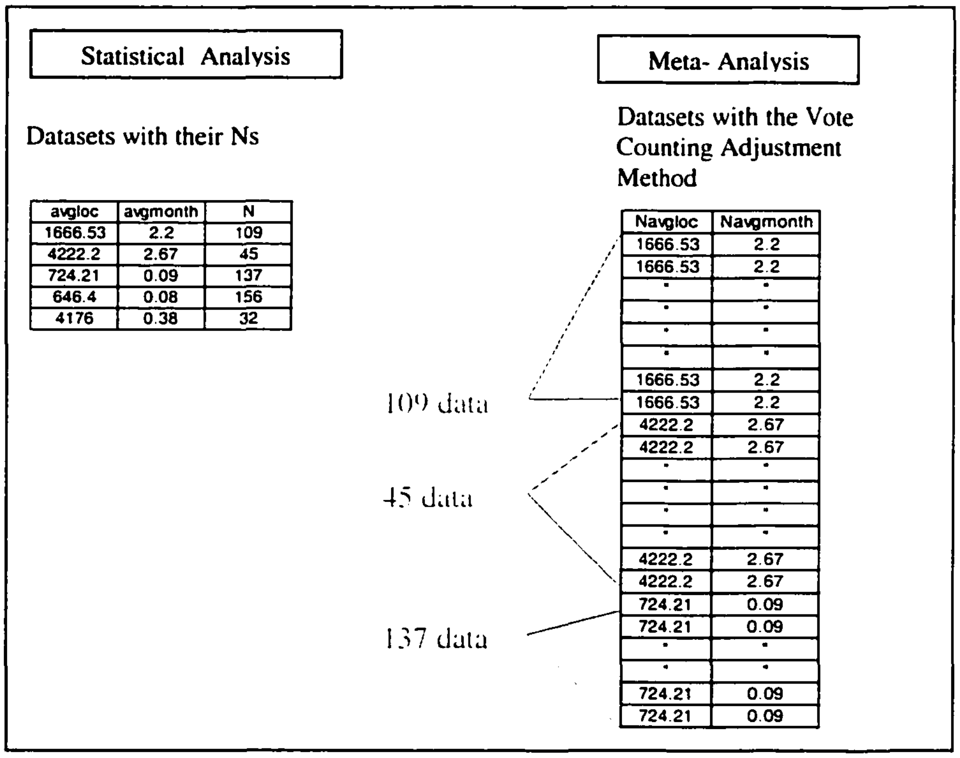
5.1 Implementation

Author	Data Set	N	LOC	Period (Month)
Kemerer [3]	Financial Sales Reporting System	109	181,652	240
Kemerer [3]	Manifest Shipping System	45	189,999	120
Project1	Data Set 1	137	99,217	12
Project2	Data Set 2	156	100,839	12
Cartwright [5]	System	32	133,632	12

Figure 5.1: Datasets Selected From The Catalog.

Based on the framework described in the previous chapters, five datasets are selected from the catalog built considering the principles described in Chapter 3 to demonstrate an example of meta-analysis. The datasets are summarized in Figure 5.1 and the process of using sample results to draw conclusions about characteristics of the population is described in this section.

The average of LOC (avgloc) is computed by dividing the total LOCs by the number of programs (N) in each dataset while average duration of the projects (avgmonth) is calculated by dividing the number of months taken to accomplish the projects by the number of programs (N).



Statistical Analysis

Datasets with their Ns

avgloc	avgmonth	N
1666.53	2.2	109
4222.2	2.67	45
724.21	0.09	137
646.4	0.08	156
4176	0.38	32

Meta- Analysis

Datasets with the Vote Counting Adjustment Method

Navgloc	Navgmonth
1666.53	2.2
1666.53	2.2
•	•
•	•
•	•
•	•
1666.53	2.2
1666.53	2.2
4222.2	2.67
4222.2	2.67
•	•
•	•
•	•
•	•
4222.2	2.67
4222.2	2.67
724.21	0.09
724.21	0.09
•	•
•	•
724.21	0.09
724.21	0.09

Figure 5.2: Datasets Used for Traditional Statistical Analysis And Meta-analysis.

For example, the first dataset collected from Kemerer and Slaughter [3], the Financial Sales Reporting System, has a project size of 181,652 LOCs. The project took 20 years and developed 109 programs (N). Therefore, the average of LOC (avgvloc) is

1666.53 and the average duration of the project (avgmonth) is 2.2. Figure 5.2 illustrates the processed datasets that are used for both traditional statistical analysis and Meta-analysis.

Since the variance of the parameters of each individual dataset is discarded, the result of a linear regression operation on the five individual parameters is not correct. The Vote Counting Adjustment Method (VCAM) [36], [23] is used to correct the result by taking the total sample sizes (Ns) into account. This approach is considered a meta-analysis process. For example, instead of using a single point for statistical analysis, VCAM uses 109 data points for the first dataset in meta-analysis.

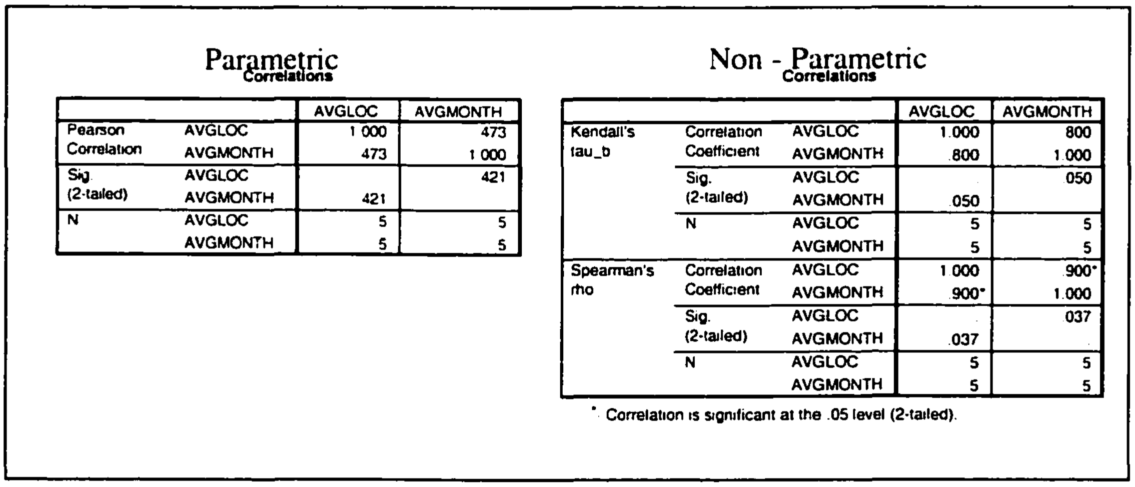
SPSS 7.5 for Windows is the statistical software package used to analyze the datasets for this chapter. The statistical test looks for the relationships between the variables, "avgloc" and "avgmonth". By assuming the datasets have a normal distribution, the parametric correlation test is used.

5.2 Validation Results

By using the processed datasets as shown in Figure 5.1 through traditional statistical analysis and meta-analysis, parametric test and non-parametric test are performed. The comparison on these test results between traditional statistical analysis and meta-analysis is discussed in this section.

The statistical techniques applied to the data yield the probability that the sample represents the general population. The result of hypothesis test provides the statistical significance of the test. A Type II error occurs while software engineers are accepting the null hypothesis when it is false.

Asterisks (*) in Figure 5.2 and Figure 5.3, indicate when a particular correlation is significant at the 0.05 level (*) or the 0.01 level (**). Once correlations are computed, one may wish to consider a corrected significance level to minimise the chances of making a Type I error. A correlation coefficient would not be significant unless its p-value is less than the corrected significance level.



Parametric

Correlations

		AVGLOC	AVGMONTH
Pearson Correlation	AVGLOC	1 000	473
	AVGMONTH	473	1 000
Sig. (2-tailed)	AVGLOC		421
	AVGMONTH	421	
N	AVGLOC	5	5
	AVGMONTH	5	5

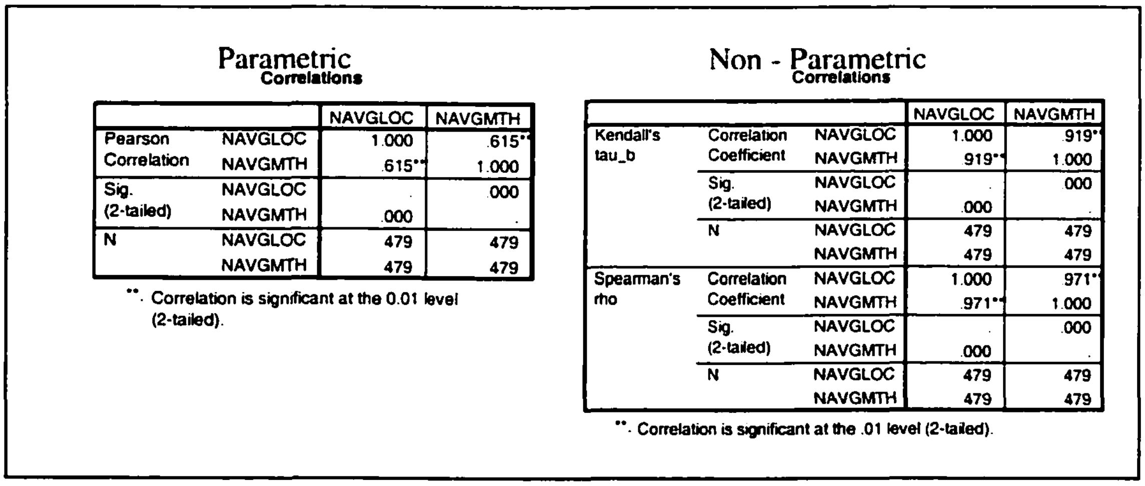
Non - Parametric

Correlations

			AVGLOC	AVGMONTH
Kendall's tau_b	Correlation Coefficient	AVGLOC	1.000	800
		AVGMONTH	.800	1.000
	Sig. (2-tailed)	AVGLOC	.	050
		AVGMONTH	.050	
	N	AVGLOC	5	5
		AVGMONTH	5	5
Spearman's rho	Correlation Coefficient	AVGLOC	1 000	.900*
		AVGMONTH	.900*	1.000
	Sig. (2-tailed)	AVGLOC	.	037
		AVGMONTH	.037	.
	N	AVGLOC	5	5
		AVGMONTH	5	5

*. Correlation is significant at the .05 level (2-tailed).

Figure 5.3: Traditional Statistical Analysis Result of Datasets From The Catalog.



Parametric

Correlations

		NAVGLOC	NAVGMTH
Pearson Correlation	NAVGLOC	1.000	.615**
	NAVGMTH	.615**	1.000
Sig. (2-tailed)	NAVGLOC	.	.000
	NAVGMTH	.000	.
N	NAVGLOC	479	479
	NAVGMTH	479	479

**. Correlation is significant at the 0.01 level (2-tailed).

Non - Parametric

Correlations

			NAVGLOC	NAVGMTH
Kendall's tau_b	Correlation Coefficient	NAVGLOC	1.000	.919**
		NAVGMTH	.919**	1.000
	Sig. (2-tailed)	NAVGLOC	.	.000
		NAVGMTH	.000	.
	N	NAVGLOC	479	479
		NAVGMTH	479	479
Spearman's rho	Correlation Coefficient	NAVGLOC	1.000	.971**
		NAVGMTH	.971**	1.000
	Sig. (2-tailed)	NAVGLOC	.	.000
		NAVGMTH	.000	.
	N	NAVGLOC	479	479
		NAVGMTH	479	479

**. Correlation is significant at the .01 level (2-tailed).

Figure 5.4: Meta-analysis Result of Datasets From The Catalog.

The Pearson's correlation coefficients for "avgloc" and "avgmonth" are 0.473 while the p-values for "avgloc" and "avgmonth" are 0.421 as shown in Figure 5.3. This parametric correlation test result for the datasets has a low Pearson's correlation coefficient and is not significant.

Following the parametric correlation test, the non-parametric correlation test is introduced to analyze these datasets. Kendall's correlations test and Spearman's correlation test are used in this example. The Spearman's correlation coefficients (rho) for "avgloc" and "avgmonth" are 0.900 while the p values for "avgloc" and "avgmonth" are 0.370. The result of the non-parametric correlation test shows a high correlation coefficient and is significant at 0.05 level.

For meta-analysis, VCAM is used to reduce the analysis errors contributed to the datasets. The Pearson's correlation coefficients for "navgloc" and "navgmonth" are 0.615 while the p-values for "navgloc" and "navgmonth" are 0.000 as shown in Figure 5.4. The Parametric Correlation test for the meta-analysis produced datasets has a high correlation coefficient and is significant at 0.01 level.

The non-parametric correlation test for the datasets has a higher correlation coefficient and is highly significant as illustrated in figure 5.4. The Kendall's correlation coefficients (tau_b) for "navgloc" and "navgmonth" are 0.919 and the Spearman's correlation coefficients (rho) for "navgloc" and "navgmonth" are 0.971. Both of the p-values for "navgloc" and "navgmonth" are 0.000. The result of the non-parametric correlation test shows a high correlation coefficient and is significant at 0.01 level.

Figure 5.5 shows the graph of statistical analysis that considers the 5 data points (1 per dataset) and the graph of meta-analysis that considers all 479 data points from the datasets. For example, the correlation points of "navgloc" and "navgmonth" with values of 1666.53 and 2.2 respectively, are plotted 109 times. Since the correlation points are plotted on the same location, there is no difference visually for the graphs.

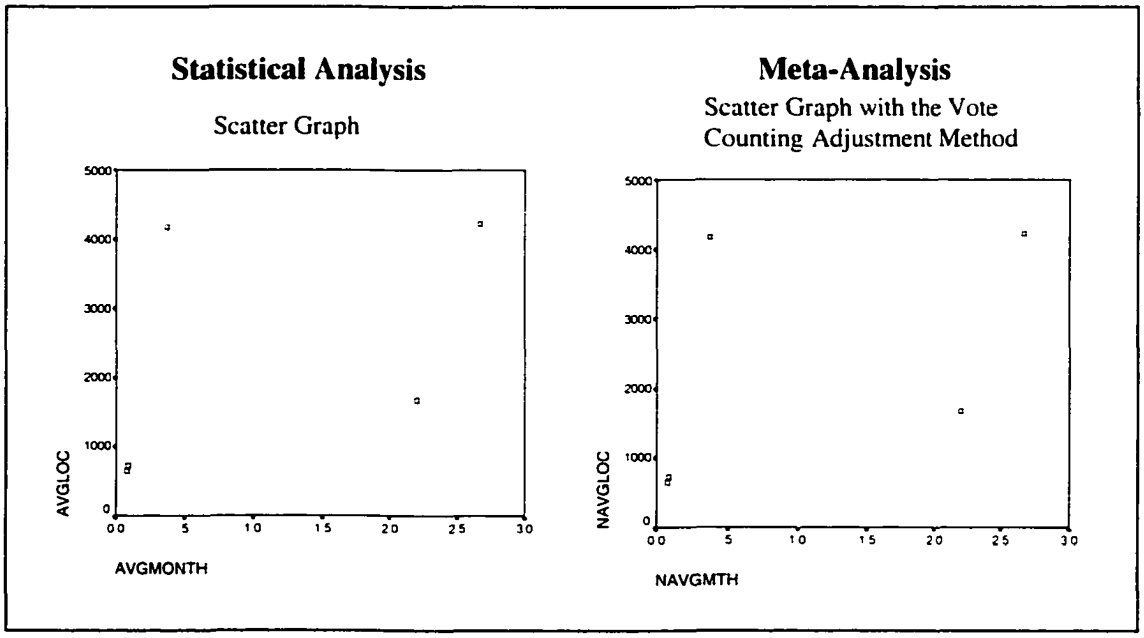


Figure 5.5: Scatter Graphs For Statistical Analysis And Meta-analysis.

Figure 5.6 shows that the correlation coefficients analyzed from the datasets by using meta-analysis are higher than the correlation coefficients analyzed by traditional statistical analysis. The correlation coefficients in meta-analysis are significant while the correlation coefficients in traditional statistical analysis are not significant. Moreover, all 2-tailed significance tests in the meta-analysis are significant at 0.001 levels. These significant values indicate that the meta-analysis statistical method is potentially able to reduce the errors.

	Statistical Analysis	Meta - Analysis
N	5	479
Pearson Correlation	0.473	0.615**
Sig. (2-tailed)	0.421	0.000
Kendall's tau b Correlation	0.8	0.919**
Sig. (2-tailed)	0.05	0.000
Spearman's rho Correlation	0.900*	0.971**
Sig. (2-tailed)	0.037	0.000

* Correlation is significant at the 0.05 level (2-tailed)

** Correlation is significant at the 0.001 level (2-tailed)

Figure 5.6: Comparisons of The Analysis.

5.3 Summary of Validation Results

The meta-analysis results are over confident since the VCAM does not take into account the standard deviation and the interval of confidence for the datasets during the re-sampling process. Hence, the generated re-sampling datasets have the same mean and the same standard deviation (i.e. 0). This type of meta-analysis, then, seems better suited to confirm the validity of the null hypothesis.

Chapter 6

Conclusions

By using meta-analysis, software engineers can use the results of a number of previous studies on a subject of interest and determine a model for the empirical relationships connecting these studies. This thesis illustrates a framework for using meta-analysis as a statistical analysis technique to combine the knowledge acquired from different software development experiments and studies. A catalog has been built by reorganizing datasets across different software development studies, while the meta-analysis statistical tool is used to find the relationships between the attributes in the datasets. This thesis shows that by using meta-analysis approach, the data analysis results are more effective than using statistical analysis.

Chapter 6 contains the following sections: The summary of meta-analysis architectural framework is presented in Section 6.1. Section 6.2 describes the main contribution of this thesis. Finally, future directions of this thesis are discussed in Section 6.3.

6.1 Summary of Meta-Analysis Architectural Framework

This thesis proposes a meta-analysis architectural framework for the following estimates:

- Meta-Analysis for Point Estimates

  By using meta-analysis, software engineers are able to use a set of points to estimate a new point by using statistical tools. Software engineers takes into account the unidentified factors or errors by considering their interval of confidence and mean. By formally examining the central tendencies of data sets, software engineers can summarize, understand and convey the information contained in a set of data. Based on this information, software engineers manage to estimate a new point.

- Meta-Analysis for Correlation Estimates

  There are the effects of sampling errors on the correlation study and these sampling errors can be estimated and corrected for at the level of meta-analysis. Software engineers consider the measures obtained from the productivity with the mean and standard deviation of population correlations. They average the measures and take into consideration the weight of the moderator. Based on the correlation relations of two parameters, software engineers estimate a new correlation value.

- Meta-Analysis for Regression Line Estimates

  Software engineers compute the variance of regressions and weigh each by selecting an appropriate type of effect size. They then compute the sum of squared variances to minimize the distance from each regression line and reduce errors. By using regression lines obtained from previous studies, software engineers can estimate a general regression line by using meta-analysis. Software engineers use this general linear regression line to estimate the future software development project.

- Meta-Analysis for Surface Estimates

  Software engineers collect datasets of regression lines with various parameters, they determine the moderator effect from the datasets and estimate a regression line. The linkages of each regression line form a surface. Software engineers average the sampling errors and variances of the regression lines and estimate a new general regression line across the surface. By using this new general regression line, software engineers can estimate a new regression line.

6.2 Main Contributions

The main contributions of this thesis are as follows:

- A catalog that collects previously published studies and summarizes their findings is built. The catalog becomes the central repository for all the valid and usable data to be employed in the meta-analysis.
- The datasets collected in the catalog are transformed into different dimensional views. This thesis uses the catalog to model a dataset as an example and builds a unification model from these existing pool of datasets by using meta-analysis.

- A comparison result of using meta-analysis method and statistical method is demonstrated in Chapter 5. The result shows that meta-analysis works more effectively than statistical analysis in drawing conclusions about characteristics of the population.

6.3 Future Directions

The measures are validated so software engineers can be sure that the measurements they use are actually measuring what they claim to measure. The current software measures validation frameworks are not a standard accepted way of validating a measure. Software engineers can improve the performance of software measures validation through Monte-Carlo simulation.

The possibility of combining information across many projects by using meta-analysis can be very beneficial in software engineering if software engineers manage to overcome the limitations of applying this method in practice. Software engineers can extend to consider more parameters into accounts while applying meta-analysis. As a result, it will lead to improve the accuracy of software projects estimation and the forecasting quality of software product.

Since the meta-analysis in this paper does not consider the standard deviation and the interval of confidence for the individual datasets, the result obtained is often over confident. Besides meta-analysis, other non-parametric statistical methods like “Bootstrap” can also be used to perform the task of combining knowledge across various studies.

Bibliography

1. A. Brooks, "Meta-Analysis: A Silver Bullet for Meta Analyst", *Empirical Software Engineering*, Vol.2, No.4, 1997, pp. 333-338.
2. A. Melton, D. Gustafson, J. Bieman and A. Baker, "A Mathematical Perspective for Software Measures Research", *Journal of Software Engineering*, Vol. 5 No. 5, 1990, pp. 246-254.
3. A. Waheed, D.T. Rover and J.K. Hollingsworth, "Modeling and Evaluating Design Alternatives for an On-Line Instrumentation System: A Case Study", *IEEE Transactions on Software Engineering,* Vol. 24 No. 6, June 1998, pp. 451-470.
4. A.J. Albrecht and J. Gaffrey, "Jr. Software Function, Source Lines of Codes and Development Effort Prediction: A Software Science Validation", *IEEE Transactions on Software Engineering*, Vol. 9, No.6, November 1993, pp. 639-648.
5. A.J. Albrecht, "Measuring Application Development Productivity", In Proceedings of the IBM Applications Development Symposium, GUIDE/SHARE (Monterey, Calif., Oct. 14-17) IBM, 1979, pp. 83-92.
6. A.R. Dennis, J.F. Nunamaker, Jr., and D.R. Vogel, "A Comparison of Laboratory and Field Research in the Study of Electronic Meeting Systems", *Journal of Management Information Systems*, Vol. 7 No. 3, pp. 107-135.

7. A.R. Montazemi and S. Wang, "The Effects of Models of Information Presentation on Decision-Making: A Review and Meta-Analysis", *Journal of Management Information Systems*, Vol. 5, No. 3, Winter 1989, pp. 101-127.

8. B. Kitchenham, S.L. Pfleeger and N. Fenton, "Towards a Framework for Software Measurement Validation", *IEEE Transactions on Software Engineering*, Vol. 21 No. 12, December 1995, pp. 929-943.

9. B.W. Boehm, Software Engineering Economics, Prentice-Hall Englewood Cliffs, N.J., 1981.

10. B.W. Boehm, W. Barry and D.R. Jeffery, "Interorganational Comparison of Programming Productivity", Department of Information Systems, University of New South Wales, March 1979.

11. C. Walston and C. Felix, "A Method of Programming Measurement and Estimation", *IBM Systems Journal*, Vol. 16 No. 1, 1977.

12. C.F. Kemerer and S. Slaughter, "An Empirical Approach to Studying Software Evolution", *IEEE transactions on Software Engineering*, Vol. 25 No. 4, July/August 1999, pp. 493-509.

13. C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models", *Communication ACM*, vol. 30 no. 5, May 1987, pp. 416-429.

14. E. Weyuker, "Evaluating Software Complexity Measures", *IEEE Transactions on Software Engineering*, Vol. 14 No. 9, September 1988, pp. 1357-1365.

15. G.C. Murphy, R.J. Walker and E.L.A. Baniassad, "Evaluating Emerging Software Development Technologies: Lessons Learned from Assessing Aspect-Oriented

Programming", *IEEE Transactions on Software Engineering*, Vol. 25 No. 4, July/August 1999, pp. 438-455.

16. G.V. Glass, "Primary, Secondary and Meta-Analysis of Research", *Educational Researcher*, Vol. 5, No. 10, 1976, pp. 3-8.
17. I. Benbasat and L. Lim, "The Effects of Group, Task, Context and Technology Variables on the Usefulness of Group Support Systems: A Meta-Analysis of Experimental Studies", *Small Group Research*, Vol. 24, No. 4, November 1993, pp. 430-462.
18. J. Boegh, S. Depanfilis, B. Kitchenham and A. Pasquini, "A Method for Software Quality Planning, Control, and Evaluation", *IEEE Software*, March/April 1999, pp. 69-77.
19. J. Cohen, Statistical Power Analysis for the Behavioral Sciences, 2nd ed., Hillsdale, N.J., Lawrence Erlbaum Associates, 1988.
20. J.W. Bailey and V.R. Basili, "A Meta-Model for Software Development Resource Expenditures", *IEEE Transactions on Software Engineering*, 1981, pp. 107-115.
21. K. Kautz, "Making Sense of Measurement for Small Organization", *IEEE Software*, March/April 1997, pp. 14-20.
22. K. Pettingell, T. Marshall and W.A. Remington, "Review of the Influence of User Involvement on System Success", Proceedings of the Ninth International Conference on Information Systems, Minneapolis, MN, 1988, pp. 227-236.
23. L. M. Pickard, B.A. Kitchenham and P.W. Jones, "Combining Empirical Results in Software Engineering", *Information and Software Technology*, Vol. 40, 1998, pp.811-821.

24. L. Putnam and A. Fitzsimmons, "Estimating Software Costs", *Datamation*, 25, September/November 1979, pp. 10-12.

25. L. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", *IEEE Transactions on Software Engineering*, Vol.4 No. 4, 1978, pp. 345-361.

26. L.C. Briand, J.W. Daly and J.K. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", *IEEE Transactions on Software Engineering*, Vol. 25 No. 1, January/February 1999, pp. 91-121.

27. L.C. Briand, K.E. Eman and S. Morasca, "On The Application of Measurement Theory in Software Engineering", *Theoretical and Empirical Validation of Software Product Measures*, ISERN Technical Report, 1995.

28. M. Alavi and E.A. Joachimsthaler, "Revisiting DSS Implementation Research: A meta-analysis of the Literature and Suggestions for Researchers," *MIS Quarterly*, Vol. 16, No.1, March, pp. 95-116.

29. M. Cartwright and M. Shepperd, "An Empirical Investigation of an Object-Oriented Software System", *IEEE Transactions on Software Engineering*, Vol. 26 No. 8, August 2000, pp. 786 - 796.

30. M.G. Mendonca and V.R. Basili. "Validating of an Approach for Improving Existing Measurement Frameworks", *IEEE Transactions on Software Engineering*, Vol. 26, No. 6, June 2000.

31. M.I. Hwang and B.J. Wu, "The effectiveness of Computer Graphics for Decision Support: A Meta-Analytic Integration of Research Findings", *The DATA BASE for Advances in Information Systems*, Vol.21, No.3, Fall, 1990, pp. 11-20.

32. M.S. Krishnan and M.I. Kellner, "Measuring Process Consistency: Implications for Reducing Software Defects", *IEEE transactions on Software Engineering*, Vol. 25 No. 6, November/December 1994, pp. 800-815.

33. N. Fenton, "Software Measurement: A Necessary Scientific Basis", *IEEE Transactions on Software Engineering*, Vol. 20 No. 3, March 1994, pp. 199-206.

34. N. Schneidewind, "Methodology for Validating Software Metrics", *IEEE Transactions on Software Engineering*, Vol. 18 No. 2, December 1995, pp. 929-943.

35. N.E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models", *IEEE Transactions on Software Engineering*, Vol. 25 No. 5, September/October 1999, pp. 675 - 689.

36. N.E. Fenton and S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing Company, second edition, 1997.

37. N.Y. Lee and Charles R. Litecky, "An Empirical Study of Software Reuse with Special Attention to Ada", *IEEE Transactions on Software Engineering*, Vol. 23 No. 9, September 1997, pp. 537-549.

38. P.L. McLeod, "An Assessment of the Experiental Literature on Electronic Support of Group Work: Results of A Meta-Analysis", *Human-Computer Interaction*, Vol. 7, 1992, pp.257-280.

39. Q. Hu, "Evaluating Alternative Software Production Functions", *IEEE Transactions on Software Engineering*, Vol. 23 No. 6, June 1997, pp. 379–387.

40. R. Bache and M. Neil, "Introducing Metrics into Industry: A Perspective on GQM", Software Quality, Assurance and Measurement: A Worldwide

Perspective, N. E. Fenton et al., eds., Int'l Thompson Computer Press, London, 1995.

41. R. Hochman, T. Khoshgoftaar, E. Allen and J. Hudepohl, "Evolutionary Neural Networks: A Robust Approach to Software Reliability Problems", Proceedings of the 8th International Symposium on Software Reliability Engineering, New Mexico, November 1997, pp. 13-26.
42. R.D. Banker and C.F. Kemerer, "Scale Economies in New Software Development", *IEEE Transactions on Software Engineering*, Vol. 15 No.10 October 1989, pp. 1199-1205.
43. R.J. Offen and R. Jeffery, "Establishing Software Measurement Programs", *IEEE Software* March/April 1997, pp. 45-53.
44. S. Chidamber and C. Kemerer, "A metrics suite for object oriented design", IEEE Transactions on Software Engineering, Vol. 20, No.6, June, 1994, pp. 476-493.
45. S. Siegel and N. Castellan, Jr., *Nonparametric Statistics for the Behaviour Sciences*, 2nd edition, McGraw Hill Book Company, New York, 1988, pp. 224-312.
46. S.L. Pfleeger, R. Jeffery, B. Curtis and B. Kitchenham, "Status Report on Software Measurement", *IEEE Software*, March/April 1997, pp. 33-43.
47. S.Y. Sohn, " Meta Analysis of Classification Algorithms for Pattern Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No.11, November 1999, pp. 1137-1144.
48. T. Hall and N. Fenton, "Implementing Effective Software Metrics Programs", *IEEE Software*, March/April 1997, pp. 55-64.

49. T.D. Cook, "Meta-Analysis: Its Potential for Casual Description and Casual Explanation with Program Evaluation", *Social Prevention and the Social Sciences: Theoretical Controversies*, Research Problems and evaluation Strategies, Berlin, 1991, pp.245-285.

50. V.R. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data", *IEEE Transactions on Software Engineering*, Vol. 10 No. 3, 1984, pp. 728-738.

51. V.R. Basili and H.D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments", *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, June 1988, pp. 758-773.

52. V.R. Basili, L.C. Briand and W.L. Melo, "A validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE Transactions on Software Engineering*, Vol. 22 No. 10, October 2000, pp. 751 – 761.

53. W. Stevens, G. Myers and L. Constantine, "Structured Design", *IBM Systems Journal*, Vol. 13 No. 20, 1974, pp. 115-139.

54. W.C. Lim, "Effects of reuse on Quality, Productivity and Economics", *IEEE Software*, September 1994, pp. 23-30.

55. H. Zuse, "Properties of software measures", *Software Quality Journal*, Vol. 4 No. 1, 1992, pp. 225-260.

56. J.J. Baroudi and W.J. Orlikowski "The Problem of Statistical Power in MIS Research", *MIS Quarterly*, Vol. 13 No. 1, March 1989, pp. 86-106.

Appendix A

ACRONYMS

AVGLOC: Average of LOC

AVGMONTH: Averaged Duration of the Projects

GQM: Goal Question Metrics paradigm

H0 & H1: Hypothesis Test

LOC: Line of Codes

M^3P: Model, Measure, Manage Paradigm

N: Number of Programs

Ns: Total Sample Sizes

SQUID: Software Quality In Development

VCAM: Vote Counting Adjustment Method