# A NEW METHODOLOGY FOR QUANTIFYING THE IMPACT OF NON-FUNCTIONAL REQUIREMENTS ON SOFTWARE EFFORT ESTIMATION

Rolan Abdukalykov

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Software Engineering)

at Concordia University

Montréal, Québec, Canada

August 2011

**CONCORDIA UNIVERSITY**

School of Graduate Studies

This is to certify that the thesis prepared

By: Rolan Abdukalykov

Entitled: A New Methodology for Quantifying the Impact of Non-Functional Requirements on Software Effort Estimation

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science (Software Engineering)**

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair
Dr. Gregory Butler

______________________________ Examiner
Dr. Ching Y. Suen

______________________________ Examiner
Dr. René Witte

______________________________ Supervisor
Dr. Olga Ormandjieva

______________________________ Supervisor
Dr. Mohamad Kassab

Approved by ______________________________
Chair of Department or Graduate Program Director

______________________________
Dr. Robin A. L. Drew, Dean
Faculty of Engineering and Computer Science

Date ______________________________

## Abstract

A New Methodology for Quantifying the Impact of Non-Functional Requirements on Software Effort Estimation

Rolan Abdukalykov

The effort estimation techniques used in the software industry often tend to ignore the impact of Non-functional Requirements (NFR) on effort and reuse standard effort estimation models without local calibration. Moreover, the effort estimation models are calibrated using data of previous projects that may belong to problem domains different from the project which is being estimated. The approach described in this thesis suggests a novel effort estimation methodology that can be used in the early stages of software development projects. The proposed methodology initially clusters the historical data from the previous projects into different problem domains and generates domain specific effort estimation models, each incorporating the impact of NFRs on effort by sets of objectively measured nominal features. The complexity of these models is reduced using a feature subset selection algorithm. In this thesis, our approach is discussed in detail, and the results of our experiments using different supervised machine learning algorithms are presented. The results show that our approach performs well by increasing the correlation coefficient and decreasing the error rate of the generated effort estimation models and achieving more accurate effort estimates for the new projects.

## Acknowledgments

I would like to thank my two supervisors for their excellent supervision, support, mentoring, encouragement and help: Dr. Olga Ormandjieva and Dr. Mohamad Kassab from the department of Computer Science and Software Engineering at Concordia University in Montreal.

Also, I would like to thank Ishrar Hussain and members of READ group for their help, useful feedback and support during the execution of our graduate projects at the department of Computer Science and Software Engineering as part of READ group.

In addition, I would like to thank SAP and its former and current employees including but not limited to: Dr. N. Mahe, Dr. J. Hendricks, R. Senoune, Dr. Y. Belala, M. Kostal, M. Codipietro, A. Gauthier, A. Sciortino, B. How-Choong, C. Nikolakakis, D. Hebert, F. Nguyen, H. S. Chandre, H. Braune, I. Thore, J. N. Robitaille, J. Vichhi, J. Laplante, K. Komitski, K. K. Avadhanula, M. Parent, M. Seccafien, N. Lipari, P. Nguyen, S. Falardeau, T. Gast, W. Wang, X. Liang, Y. Grenier, and Y. Zhao for their feedback, collaboration, and in kind support.

In addition, I would like to thank my examiners, Dr. C. Y. Suen and Dr. R. Witte, for their precious time to review my thesis and give me helpful advice.

Furthermore, I would like to thank Dr. A. Sen and Dr. N. Bouguila from Concordia University in Montreal and Dr. M. A. Hall from the University of Waikato in Hamilton, New Zealand for their valuable feedback.

Finally, I would like to thank my parents, grand-parents and brother for their encouragement and support while I was doing my masters degree.

# Table of Contents

## List of Figures

# List of Tables

# Chapter 1: Introduction

## 1.1. Problem Statement and motivation

The success of planning and management of software project largely depends on the estimation of size and effort. A good estimation of these variables available right from the start in a project gives the project manager confidence about any future course of action, since many of the decisions made during development depend on, or are influenced by, the initial estimations. Hence, effort estimation is one of the most crucial steps of planning and management of a software project.

The work presented in [HKO08] showed that the functional requirements (FRs) and NFRs can automatically and effectively be extracted from software requirements document using natural language processing techniques, and a recent work [HKO10, HOK09] have shown that the functional size of the software can be computed objectively from any form of unrestricted textual representation of FRs. The work described in this thesis uses the previous work as foundation and builds on it a comprehensive methodology to estimate software development effort during the early phases of requirements development using the functional size of the software as primary variable.

Although, effort estimation in practice is largely performed by subjective evaluations, there has been numerous works in this field attempting to build parametric models for estimating effort. All these models are calibrated with historical data from past projects, so that the effort of the new software projects can be estimated. However, while some tend to ignore the impacts of different NFRs [Put81], others [Boe81] include them partially requiring subjective judgment by human experts.

Ignoring NFRs and introducing subjective evaluations can often result in a large magnitude of error in effort estimation [AHKO11]. In contrast, the work described in the thesis proposes a methodology that can objectively quantify the impact of NFRs on effort estimation. The impact of four high-level classes of NFRs chosen from the NFR ontology, which is described in [Kas09a], is taken into consideration to encompass all possible classes of NFRs.

## 1.2. Research objectives

To resolve the open problems mentioned above, we need to develop a new effort estimation methodology that

- Reduces the human estimator's bias.
- Generates effort estimation models based on the historical data of previous projects with similar problem domains to estimate software development effort during the requirements specification phase.
- Shows how to assess the impacts of different NFRs and different problem domains on the estimation of the software development effort.
- Makes effort estimation model robust by dynamically reducing the feature space.
- Assists with data collection process from historical projects.

## 1.3. Major Contributions

The work completed as part of the thesis has the following contributions:

- It provides a new effort estimation methodology based on the historical data of previous projects to estimate the software development effort during the requirements specification phase.
- The methodology shows how to assess the impacts of different NFRs and different problem domains on the estimation of the software development effort.
- The methodology allows making effort estimation models robust by dynamically reducing the feature space using both statistical and semantic techniques.
- It proposes a questionnaire that assists in the data collection process from historical projects.

## 1.4. Research Methodology

The research methodology used in this thesis consists of the following major phases:

- Inception
- Method Selection
- Design and Documentation of the Methodology
- Design and Documentation of the Questionnaire
- Implementation of the Methodology
- Design of Experiment
- Experiment Execution and Results Analysis

The summary of the research methodology is shown in Figure 1-1.

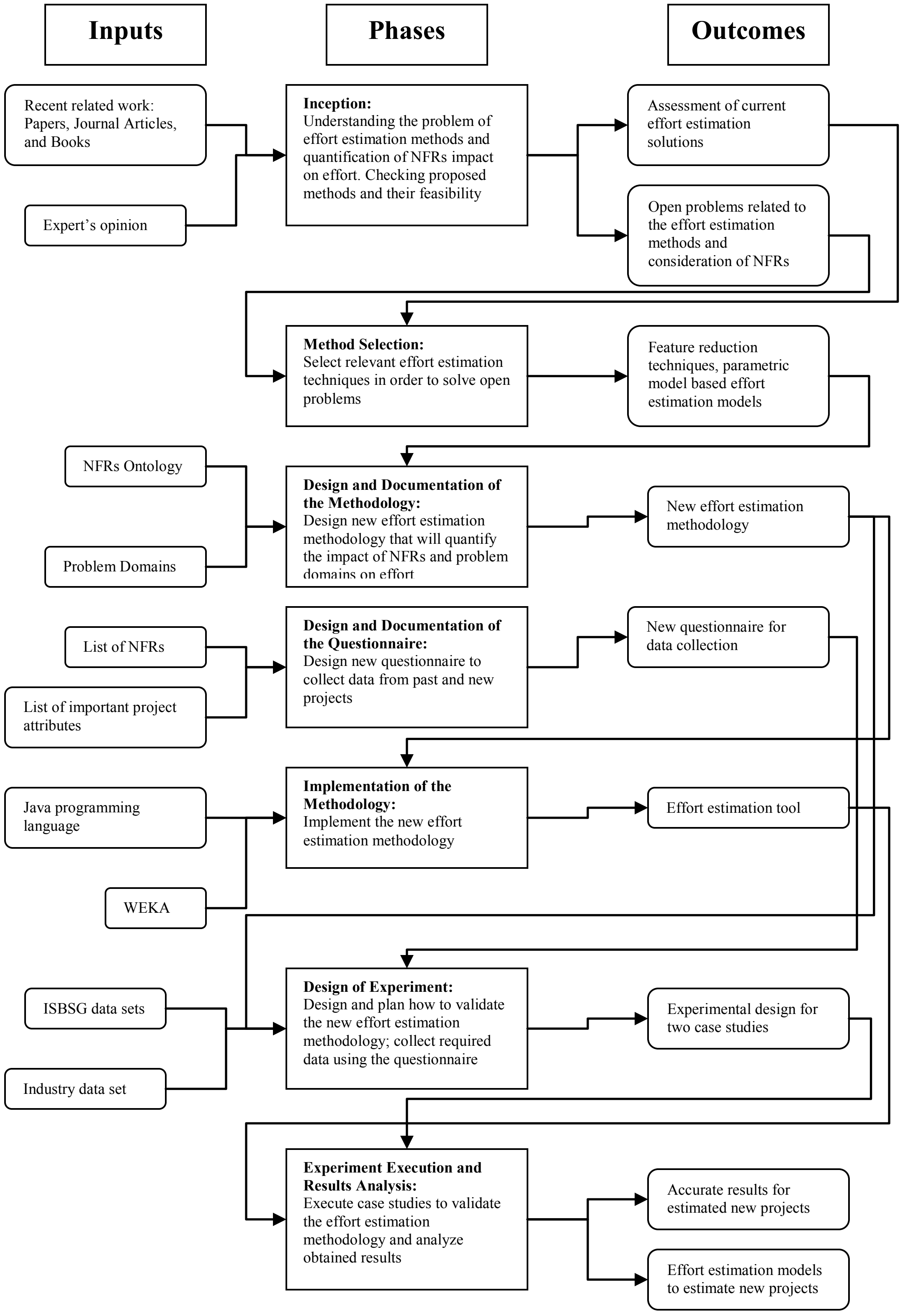

Inputs
Phases
Outcomes
Recent related work: Papers, Journal Articles, and Books
Expert's opinion
Inception: Understanding the problem of effort estimation methods and quantification of NFRs impact on effort. Checking proposed methods and their feasibility
Assessment of current effort estimation solutions
Open problems related to the effort estimation methods and consideration of NFRs
Method Selection: Select relevant effort estimation techniques in order to solve open problems
Feature reduction techniques, parametric model based effort estimation models
NFRs Ontology
Problem Domains
Design and Documentation of the Methodology: Design new effort estimation methodology that will quantify the impact of NFRs and problem domains on effort
New effort estimation methodology
List of NFRs
List of important project attributes
Design and Documentation of the Questionnaire: Design new questionnaire to collect data from past and new projects
New questionnaire for data collection
Java programming language
WEKA
Implementation of the Methodology: Implement the new effort estimation methodology
Effort estimation tool
ISBSG data sets
Industry data set
Design of Experiment: Design and plan how to validate the new effort estimation methodology; collect required data using the questionnaire
Experimental design for two case studies
Experiment Execution and Results Analysis: Execute case studies to validate the effort estimation methodology and analyze obtained results
Accurate results for estimated new projects
Effort estimation models to estimate new projects

Figure 1-1: Summary of Research methodology

### 1.4.1. Inception

The goal of this phase is to understand the current problems of the effort estimation methods and quantification of NFRs impact on effort by surveying the related work and engaging in discussions, workshops and brain storming sessions with experts in this field. As a result, we have identified the available effort estimation techniques and their corresponding open problems.

### 1.4.2. Method Selection

This phase focuses on the selection of effort estimation methods and techniques that are suitable for us to design a new effort estimation methodology. For example, as part of this phase we have identified the feature subset selection techniques and parametric model based effort estimation models in order to resolve open problems identified earlier in previous phases.

### 1.4.3. Design and Documentation of the Methodology

The goal of this phase is to design and to document afterwards the new effort estimation methodology based on the results of the previous phases. The NFR Ontology developed by Mohamad Kassab is used to take into consideration the impact of NFRs on effort [Kas09b]. The effort estimation methodology designed in this phase is used as input for the subsequent phases.

### 1.4.4. Design and Documentation of the Questionnaire

In this phase, we have designed a special questionnaire to collect important historical and new information about projects. The questionnaire is designed taking into account the list of NFRs that are derived from the NFR Ontology and the list of important project attributes such as Complexity of the Product and Experience.

#### 1.4.5. Implementation of the Methodology

The goal of this phase is to implement the new effort estimation methodology as a usable software tool. It will be used to validate the methodology in subsequent phases and to prove our concept. The implementation is done using the Java programming language and WEKA tool.

#### 1.4.6. Design of Experiment

The goal of this phase is to design and to plan validation of the new effort estimation methodology. It also focuses on collection of required data using the previously designed questionnaire. As part of this phase, several data sets coming from ISBSG and Industry are considered for the methodology validation. The validation is planned to take place using 2 case studies. The selection criteria for the best effort estimation method are selected.

#### 1.4.7. Experiment Execution and Results Analysis

In this phase, we execute case studies to validate the effort estimation methodology and analyze obtained results. The effort estimation models are obtained as part of validation. We conclude based on analysis of obtained results that our methodology performs well and achieves accurate results.

### 1.5. Thesis Outline

The structure of the thesis is following:

- Chapter 2 provides the necessary background needed to understand the remaining chapters.
- Chapter 3 surveys the related work and lists the open problems in the chosen area of research.

- Chapter 4 discusses the methodology proposed in this thesis.
- Chapter 5 describes how the methodology is implemented.
- Chapter 6 illustrates the methodology on case studies.
- Chapter 7 presents the conclusions and the future work.

# Chapter 2: Background

In this chapter, we present and explain the important concepts used in the current thesis. The concepts include feature selection and related techniques, software size measurement and its different perspectives such as length, complexity, functional size, COSMIC Method, effort estimation, and NFRs. The feature selection concept described in section 2.1 is used to reduce the complexity of the effort estimation models generated by our methodology. The WEKA tool described in section 2.1.3 provides the feature subset reduction algorithm which we use to reduce the set of features in one of the steps of our methodology. The functional size concept described in section 2.2 is one of the important input parameters of effort estimation models generated by our methodology. The NFRs and its subset including performance, security, usability, and maintainability described in section 2.4 are important concepts used in our methodology. The impacts of these four NFRs on effort are considered during the generation of the effort estimation models from the historical data and during the estimation of new project effort by our methodology.

## 2.1. Feature Selection

In the context of software engineering, a feature is a variable that describes information about certain projects. For example, Number of Developers is a feature that measures the number of software developers in a software project. Features can have different types such as numerical or nominal. Numerical feature is a feature with possible values containing numerical values. On the other hand, nominal feature (often used interchangeably with the categorical term) is a feature with possible values containing qualitative values [Sta01]. For example, Project Complexity can be measured using 3 values: Low, Medium, and High.

According to [Ste46], there are 4 different scale types: nominal, ordinal, interval and ratio/absolute. The nominal scale is used to describe attributes based on the class label such as Male and Female. The ordinal scale is used to describe attributes using nominal scale and subjective information about ordering the entities in terms of the attribute being measured, such as maintainability or usability [IAK02, Ste46]. The interval scale allows describing attributes using nominal and ordinal scales with extra information about assigned measurement units such as Celsius or Fahrenheit degrees which do not have the notion of an absolute zero (that is, a zero representing a lack of the attribute). The ratio of the interval differences such as [(20C – 10C)/(5C-0C)] is meaningful [Sch10, Ste46]. The ratio or absolute scale types allow quantifying the well-understood attributes in terms of units of measurement for which there exists an absolute zero, such as measuring the length of code in LOC (lines of code). The ratio of the measurement values on the ratio and absolute scale types is meaningful [Sch10, Ste46].

Nominal features are mostly measured on the nominal or ordinal scale [IAK02]. On the other hand, numerical features are mostly measured on the ratio or absolute scale [IAK02].

The number of features collected from previously completed projects can be quite high. Therefore, it is necessary to reduce the complexity of the feature space by using feature selection. The feature subset selection helps to reduce the redundancy in the feature subset [Hal00]. It is widely used in statistics and pattern recognition [Hal00]. There are various ways to perform feature subset selection. For example, Filter and Wrapper methods are one of common feature reduction techniques [Hal00].

### 2.1.1. Filter

Filter methods reduce a feature set with the help of heuristics such as correlation, standard deviation, and entropy. The entropy is a measure of uncertainty in the system. Filter methods are not based on an induction algorithm. The induction algorithm is a learning algorithm that builds knowledge by analyzing the data and represents it in a special form such as decision tree or probabilistic summary [Hal00]. The examples of the induction algorithms are C4.5, naïve Bayes, and IB1. Filter methods perform usually faster than Wrapper methods [Hal00]. The examples of Wrapper methods based algorithms are forward and backward selection and Hill climbing [ANC08].

The correlation based feature selection (CFS) is a filter method that uses correlation as a heuristic criterion to evaluate the merit of a feature subset [Hal00]. The method evaluates each subset of features for its merit and selects the best feature subset. It evaluates the correlation among each pair of features and the correlation between the features and the class (e.g. effort estimate). The higher the correlation between the features and the class attribute and the lower the correlation among each pair of features, the higher the merit of the subset. The following formula is used to calculate the merit of the feature subset [Hal00]:

$$Ms = \frac{n * \overline{r_{cf}}}{\sqrt{n + n * (n-1) * \overline{r_{ff}}}}$$

***Ms*** describes the “merit” of the feature subset that has n number of features. $\overline{r_{cf}}$ represents the mean of the correlation between features and the class and $\overline{r_{ff}}$ is the mean of the correlation among each pair of features. The correlation between features and the class and the correlation among each pair of features can be calculated using three different feature selector methods such as relief, minimum description length (MDL), and symmetric uncertainty [Hal00]. In the symmetric uncertainty feature selector, the information gain [Qui86] is used to evaluate the degree to which each feature contributes when it is added to the feature subset. The Relief feature selector uses feature weighting algorithm to measure feature interactions [KR92]. The MDL feature selector tries to reduce the complexity of the feature subset to make sure it is predictive of the data and contains a minimal number of features [Hal00].

### 2.1.2. Wrapper

Wrapper methods are based on a machine learning algorithm that reduces the feature space by evaluating each subset of features for its merit every time a new feature is added [ANC08, Hal00]. The initial feature subset is an empty set. The stopping condition of the method is all features have been considered or there is no more improvement after adding features to the subset [ANC08].

Wrapper methods often provide better results, because they are adjusted to the interaction of the induction algorithm and the corresponding training data [Hal00]. On the other hand, the performance of Wrapper methods often degrades on large data sets due to multiple calls to the induction algorithm and re-initialization when a new induction algorithm is applied [Hal00]. Also, the results of cross-validation of a small data set often vary [Hal00].

### 2.1.3. WEKA

There are many statistical software tools in the market that can help one to perform feature selection or regression analysis. For example, we have used Waikato Environment for Knowledge Analysis (WEKA) tool in our experiments. WEKA is one of the well-known powerful tools containing multiple learning algorithms designed for feature reduction, statistical analysis and data mining [HFH09]. It is developed using the Java programming language by the research group at the University of Waikato in New Zealand. It has received multiple contributions from around the world in terms of algorithms from researchers working in this area.

## 2.2. Software Size Measurement

Software size is an attribute that describes a particular view on a software product such as length, complexity and functionality without executing the actual system, which may not even exist in the beginning of a project [FP97, Kas09b]. It is not easy to perform software size measurement [FP97]. The obtained measure needs to be used for the purpose it was measured. For example, if we measured the size while being interested in the length of code, then it would not make sense to use it to analyze the complexity.

### 2.2.1. Length

The length can provide physical size measurement of the software code, specification or design document [FP97]. The length of code is measured using line of code (LOC) or SLOC measure. There are different guidelines available on measuring the length of code [Kas09b].

### 2.2.2. Complexity

The software size measured with a perspective of complexity can refer to the computational complexity or algorithmic complexity [FP97]. The computational complexity categorizes the problem based on its inherent difficulty [FP97]. It is modelled using mathematical models with decision problems representing the computational tasks [Kas09b]. The algorithmic complexity measures the complexity of the algorithm proposed to solve the problem [FP97]. It is used to analyze the resources required for a particular solution to the problem. The algorithmic complexity is measured using big-Oh notation, a mathematical measure which identifies the highest order of the function f(n) where n is the input of the function [FP97]. In other words, the efficiency of an algorithm with function f(n) is measured as O(f(n)) [FP97]. For instance, O(f(n)) = n*log(n).

### 2.2.3. Functional Size

Functional size is one of the important attributes of functional user requirements by [ISO1414307]. We refer to it as Size in this thesis. The size is often used during the project planning process to estimate various attributes of a project, such as the effort to build a software product [Kas09b].

The functional size measurement focuses on the functional aspect of the final software product instead of technical or implementation details, such as programming languages and development platforms used to develop the product [Kas09b].

The functional size measurement can be done using multiple methods, including Functional Point Analysis, IFPUG, NESMA and COSMIC.

Allan Albrecht has originally proposed functional size measurement in 1979 in his "Function Point Analysis" (FPA) method which later inspired the creation of the IFPUG method [COS09]. The NESMA method was inspired from the IFPUG and can be considered as a simplified version of the IFPUG [COS09].

#### 2.2.3.1. COSMIC Method

COSMIC is a functional size measurement method used to objectively measure Size [COS09]. The method follows all requirements of the ISO 14143 related to the functional size measurement [Kas09b]. It has become an International Standard ISO/IEC 19761:2003 and has been used widely in the academia and industry [COS09].

The COSMIC method measures functional size from the perspective of an end-user looking from the outside of the system or a component [COS09]. In order to measure the functional size, the method models the functional user requirements as “COSMIC Generic Software Model” (see Figure 2-1).

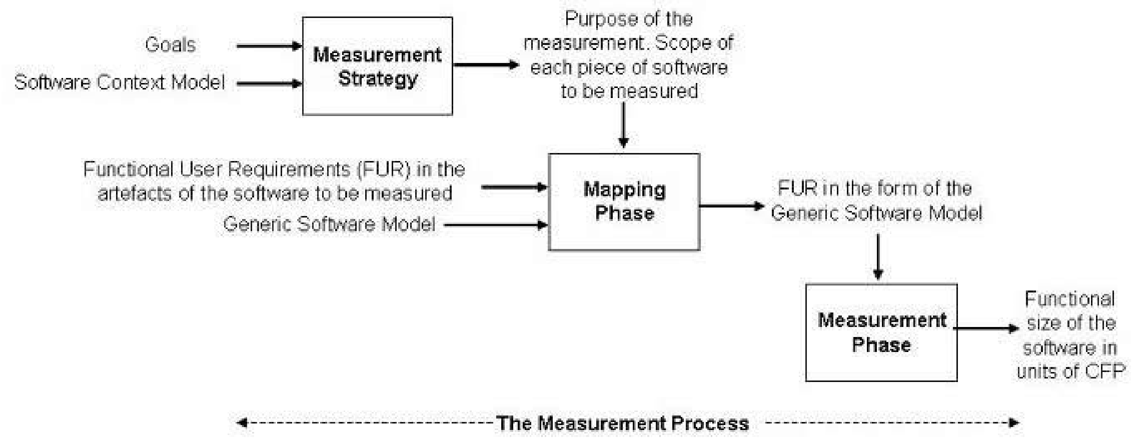

Goals
Software Context Model
Measurement Strategy
Purpose of the measurement. Scope of each piece of software to be measured
Functional User Requirements (FUR) in the artefacts of the software to be measured
Generic Software Model
Mapping Phase
FUR in the form of the Generic Software Model
Measurement Phase
Functional size of the software in units of CFP
The Measurement Process

Figure 2-1: Functional Size measurement using COSMIC method [COS09]

The functional process is an elementary unit of a set of FRs invoked by a single or multiple events. The event can be started directly or indirectly by a user often called as an actor in the COSMIC method [ADOSS03]. The functional processes contain sub-processes which are data movements [COS09]. The data movement describes process of a transfer of attributes belonging to a single data group. It is measured using 1 COSMIC

Function Point (CFP). Entry, Exit, Write and Read are types of data movements. The functional processes start with a data movement Entry (Figure 2-2) which describes movement of data group from the user to the functional process [ADOSS03].

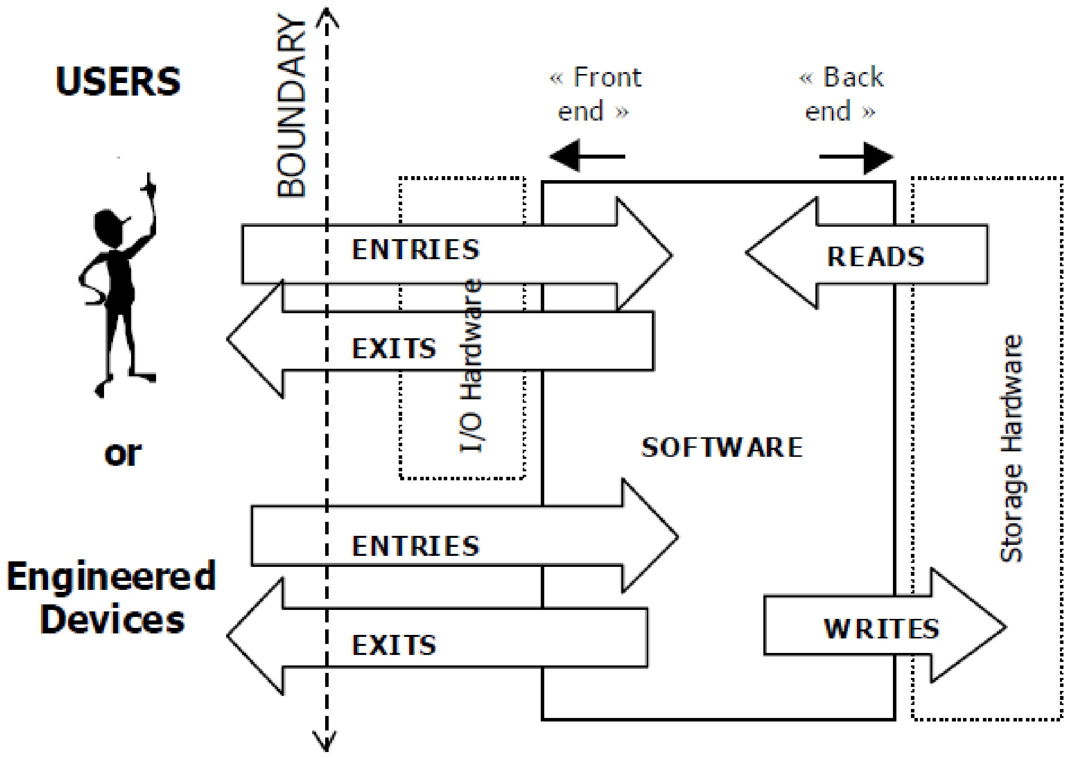


Figure 2-2: Generic Flow of Data Attributes through Software from a Functional Perspective [ADOSS03]

The Exit data movement transfers data group from the functional process to the user located outside of the software boundary. The Write data movement transfers the data group from the functional process to the persistent storage (e.g. database). The Read data movement transfers the data group from the persistent storage to the functional process. The functional size is calculated by summation of all data movements taking place in each functional process for each data group [COS09].

The minimum functional size of software measured using COSMIC can be 2 CFPs (1 Entry data movement and 1 Exit data movement) and the maximum functional size is not limited [COS09].

## 2.3. Effort Estimation

The effort is a measure of how many units it will take to complete a certain task or activity, such as developing a software application, module or method. Units can be person days, person hours, person months, story points, or use case points [Mcc06]. The cost is not the same as effort. The cost refers to the expenses associated with completing the required task or activity. For example, it might take 100 person days to build a software application and 100,000 Canadian dollars will be the associated cost of this activity, provided that each day we spend 1000 Canadian dollars to build the software application.

There are many different ways to estimate effort and cost [BC00, Mcc06]. The complexity and accuracy of these methods vary accordingly [BC00, Mcc06]. Functional size is often used as one of the main attributes to estimate effort [FP97, GHL09, PWL05]. The effort estimation techniques are usually grouped into several major groups: expert based, model based/algorithmic, regression-based, and learning oriented (neural and case based) [BC00]. We discuss expert based techniques in section 3.1.1.1. The model based/algorithmic and regression-based techniques are discussed in section 3.1.1.2. The section 3.1.1.3 describes learning oriented (neural and case based) techniques.

## 2.4. NFR

The NFR has many different definitions in the literature and the industry. Some of the international organizations did not agree yet on an official definition. For example, in the IEEE 830-1998 standard, the term is not defined and the list of categories of the NFR is given (e.g. functionality, external interfaces, performance) [IEEE83098]. [Kas09b] defines it as "Umbrella term to cover all those requirements which are not explicitly defined as functional". NFRs can be classified either using intramodel dependency view or intermodal dependency view.

In the intramodel dependency view, NFRs are refined into a hierarchy consisting of a root NFR category and multiple children refinements such as *decomposition* and *operationalization* [Kas09b]. The decomposition is a special procedure where a NFR is described using children sub-NFRs [Kas09b]. For instance, the security NFR can be decomposed into smaller sub-NFRs in order to better address it. The operationalization is a special procedure where an NFR is refined into operations, functions, data representations and architecture design decisions that are necessary to address the NFR adequately.

The NFR type is a type of refinement where the NFR can be categorized into one of 5 major sub-classes: Design Implementation Constraint, Economic Constraint, Operating Constraint, Political Cultural Constraint, and Quality Requirement. The Design Implementation Constraints are restrictions that must be met in order to satisfy certain commitments (e.g. technical, business) [LW03] and cover Hardware Design Implementation, Physical, Regulations, and Environmental areas [Kas09b]. The Economic Constraints are restrictions impacting the development cost of the software development project. The Operating Constraints describe various restrictions applicable to the operation of the software development and software being developed. For example, resource availability, systems accessibility during the maintenance, and level of the skill set are operating constraints. The Political Cultural Constraints cover legal and policy related aspects of the software development. The Quality Requirements are a set of requirements related to the Quality aspect of the software [Kas09b]. The Quality attribute is a set of characteristics demonstrating how well the requirement meets the user needs. The quality requirements include NFR requirements such as performance, usability, maintainability, security, accessibility, and reliability. We consider a subset of NFRs during case studies shown in the thesis. The subset includes performance, usability, maintainability, and security.

In [Kas09b] the NFR is viewed as an interdependent entity linked with other types of attributes of the project and software such as FRs, Product, and Process.

### 2.4.1. NFR Ontology

Ontology is a concept which helps to specify and describe various objects and concepts in a formal way. It is often viewed as a domain containing terms and associations among them [Kas09b]. Mohamad Kassab developed a comprehensive ontology to describe and model NFR formally [Kas09b]. There is a variety of ontology languages to create ontologies. For example, OWL is a popular web ontology language based on description logics [BHS03, Kas09b].

Performance, usability, maintainability, and security are of the major classes of the NFR Ontology developed by Kassab [Kas09b].

### 2.4.2. Performance

Performance is a NFR that describes the response time required by a system to perform a certain task during a certain period of time [TEMPLATE09, Kas09b]. For example, "The system shall provide user with a list of cancelled flights within 1 second after the initial request" is a performance NFR.

### 2.4.3. Usability

Usability is a NFR that describes how easy it is for users to learn, use, and interact with the system to achieve their goals [ISO912601, Kas09b]. For example, "The UI of the system shall be attractive to an end-user. It shall allow an end-user to decrease the learning time" is a set of usability NFRs.

### 2.4.4. Maintainability

Maintainability is a NFR that describes how easy it is to maintain, adapt, understand the implementation of the system to correct issues and to introduce new changes including both functional and technological [ISO912601, Kas09b]. For example, "The software developer maintaining the software application code shall be able to understand the code easily using sufficient number of comments" is a maintainability NFR.

### 2.4.5. Security

Security is a NFR that measures how well it can protect itself against unauthorized attacks, usage and continuity [ISO912601, Fir03, and Kas09b]. For example, "The system should provide its functionalities with high confidentiality when it is required" is a security NFR [Kas09b].

In the next chapter we survey the related work and list the open problems in the chosen area of research.

# Chapter 3: Related Work and Open Problems

The success of a software project depends on many factors, including accurate software effort estimation [AHKO11]. In the recent years, there have been different approaches, techniques and models developed to estimate effort [AHKO11]. The majority of software estimation models include functional size due to strong relationship between functional size and effort [FP97, GHL09, PWL05]. Hence, accurate functional size measurement is very important and helps to reduce the uncertainty of software effort estimation [AHKO11]. However, there are additional features such as environmental factors, technical factors and operating constraints that affect the relationship between effort and functional size [AHKO11, GHL09].

There have been various studies done in the field of software effort estimation attempting to build parametric models for estimating effort. All these models are calibrated with historical data from past projects, so that the effort of the new software projects can be estimated. Multiple studies were done to identify a set of NFRs that influence the relationship between effort and functional size [AHKO11]. Some effort estimation models tend to ignore the impacts of different NFRs [Put81]. On the other hand, others [Boe81] include them partially, requiring subjective judgment by human experts. A large magnitude of error in effort estimation can be introduced due to improper consideration of NFRs and introduction of subjective evaluations [AHKO11].

In [BC00], Chulani *et al.* classify effort estimation techniques and methodologies into 6 different categories such as expert based, model based/algorithmic, regression-based, and learning oriented (neural and case based).

In this chapter, we discuss related work, effort estimation models and functional size estimation methods which consider NFRs and open problems in this area.

## 3.1. Related Work

### 3.1.1. Effort Estimation Methodology

Pfleeger *et al.* have emphasized the importance of the identification of uncertainty and its impact on the quality of estimation. The authors provide a checklist for selecting appropriate size estimation technique taking into account various associated risks [PWL05].

The bottom-up and top-down approaches should be used when sufficient information about project and its staff is available [PWL05]. Pfleeger *et al.* recommend using them after more information about architecture and design is available. The top-down approach is considered to be faster to produce effort estimate in comparison to other methods. The main source of uncertainty associated with effort estimation models include: system definition, system development and estimation process [PWL05].

#### 3.1.1.1. Expert based

Pfleeger *et al.* consider the expert judgment as a complementary effort estimation model in addition to other models, due to its high level of uncertainty [PWL05]. An expert based estimation can generate results quite fast, but relies on the experience, judgment and expertise of a human estimator. In other words, this method often is prone to subjectivity and errors [AS00].

McConnell [Mcc06] recommends always finding some attributes of a project to count, compute if counting is not possible, and use judgment as a last resort. According to him, the judgment often tends to have certain degree of bias, intentionally or unintentionally, which could potentially skew the effort estimation. In addition, it is recommended to use current project data, historical data from similar projects performed at the current organization, or data coming from projects executed at outside organizations to calibrate effort estimates. However, industry data should be used when it is impossible to obtain historical or current project data as it may not be quite relevant for the current organization or project. The approach suggested by McConnell is similar to estimation by analogy where one could try to estimate software effort by using similar projects. According to the author, linear models could be sufficient to model effort estimation of the project if the variance of collected data is small [Mcc06].

Jorgensen has suggested the usage of regression models as a complement to human based effort estimation, because according to him the regression models currently cannot fully replace the human judgment to identify uncertainty in estimation. The author mentions 4 different approaches to identify estimation accuracy: usage of effort prediction intervals, usage of previously estimated task's effort estimation accuracy, usage of regression-analysis model of estimation accuracy (studied by Jorgensen), and usage of human judgment to identify effort prediction intervals. The study was conducted on a data set collected from a single middle-size development company in Norway. It was found that the estimation by software developers instead of project managers, estimation of somebody else's work, and time-to-delivery priority instead of cost or quality priority were main factors of inaccurate effort estimation. Also, Jorgensen noticed project managers tend to underestimate especially when they believed their previous task's effort was inaccurate. However, project managers had overall more accurate estimates than developers. The author suggested more detailed study needed to understand how various factors affect the effort estimation accuracy [Jor04].

It is known that the software industry has problems with estimating effort accurately and there have been multiple surveys done by researchers, commercial companies and other individuals regarding software cost estimation and software development project budget overruns. It was reported many of these studies tend to emphasize more failed attempts of software estimation in software industry [MJ03]. Molkken *et al.* studied dozen of reviews in order to identify reasons and extent to which software projects deviate in terms of cost and effort from the original estimate, effort estimation techniques used by software industry and their accuracy, and acceptable level of effort estimation accuracy. The authors have estimated effort and cost overrun to be between 30% and 40% for surveyed projects instead of 89% as reported by Standish group report. In addition, it was noticed that majority of surveyed companies (85%) used expert-based judgment and analogy based estimation techniques [MJ03]. Model based effort estimation models provided less accurate results which could be due to lower percentage of projects among surveyed companies using the model based effort estimation models and different nature of projects in comparison to projects using expert based judgment technique. Model based models included COCOMO, Use-Case-based estimation, and Functions Points Analysis (FPA). Surveyed companies considered acceptable level of effort estimation accuracy to be +/- 20% of original estimated effort. Moreover, respondent companies were aware of effort estimation as a major issue. Authors could not find indication among surveyed companies regarding their intention to improve used effort estimation models and techniques. The reasons behind inaccuracy in effort estimation among surveyed companies could have been related to over-optimistic effort estimations, which could have been caused by pressure from the project management or customers to reduce accurate effort estimates and requirements changes. However, the reasons for cost and effort overrun could be linked to more than several reasons and be complex in nature [MJ03].

#### 3.1.1.2. Effort estimation methodologies using parametric models

Algorithmic estimation methods can produce more precise effort estimation than other techniques [AS00], but they are recommended to be recalibrated and derived again for the organization based on the past project historical data of the organization where it is planned to be used [PWL05]. However, the process of the data collection from the past project historical data is not explicitly discussed in [PWL05].

In addition, algorithmic methods should not be used if they were derived using projects whose size significantly differs in several orders of magnitude from the current project. These methods should consider the uncertainty coming from the size attribute, one of its largest input parameters [PWL05].

##### *3.1.1.2.1. Feature reduction*

The software requirements specification (SRS) often contains irrelevant information about software product to be built. Grimstad *et al.* have studied whether irrelevant information could impact software effort estimate. They have conducted two experiments and have found that irrelevant information in both experiments have increased the software estimate performed by humans. It was also noticed by the authors in one of the experiments that the confidence level of the estimator has increased when irrelevant information was introduced, but the accuracy and reliability of the effort estimate has decreased [GJ07]. However, they were not able to identify how and to what degree irrelevant information could impact software estimate. The main challenge is also to identify which specific information in SRS could be classified as irrelevant to software estimation as this could be subjective in nature as well [GJ07]. The irrelevant or redundant information can become an input feature to the effort estimation model. The recent study has confirmed the need to perform feature reduction (or so called column pruning) before performing effort estimation in order to increase the quality of the effort estimation process [Men10].

Mark A. Hall has performed a study in the area of correlation based feature selection (CFS) algorithm, where he discovered that CFS, a Filters based approach, performs well in reduction of the feature subset [Hal00]. The CFS algorithm relies on the analysis of feature-to-feature interaction and feature-to-class interaction. It was shown that CFS reduced the feature subset and removed redundant features. The performance of the CFS was as good as Wrapper methods [Hal00].

###### *3.1.1.2.2. Putnam's SLIM Method*

Putnam's Software Life-cycle Model (SLIM) was inspired from Rayleigh's Model in 1970s [BC00]. The model estimates effort using productivity and software size. The productivity is calibrated using the past projects, but the data collection process from the past projects is not defined by SLIM method. The software size is based on Source Lines of Code (SLOC), which means the user needs to have either implementation of the software code or measure software size using Function Points and perform the conversion to SLOC. The model does not consider the impact of NFRs on effort and problem domain of the past projects during the calibration [BC00]. The feature reduction is not available in this model.

##### *3.1.1.2.3. Albrecht and Gaffney Method*

Albrecht and Gaffney suggest estimation of function points of a software system to be developed in its early stage. Using this estimation they propose to derive source lines of code from the function points, which will be used later to estimate the required effort for the development of the software system. The work was done in 1983 when there was not enough data and advancement in the area of correlation of function points with total effort [AG83]. It is one of first attempts to quantify the size and to derive effort more objectively. However, the conversion of the SLOC to FPs is often criticised due to the mix of implementation details (SLOC) with the FRs size (FPs) and increased rate of error [GHL09]. The model does not consider the problem domain and the impact of NFRs on effort [AG83]. The feature reduction is not available in this model.

##### *3.1.1.2.4. COCOMO II Method*

In [Boe00] Boehm indicates how projects could estimate effort based on the typical cost and productivity attributes of the previous projects or history-based software cost analysis methods using COCOMO II. The model was an improved successor version of the COCOMO model developed in the 1980s [BAB+00]. The main input parameter of the COCOMO II model is the measure of size expressed in source lines of code (SLOC), function points (FPs) or application points (APs). The exponent of the size attribute in the model contains only project level cost drivers [BAB+00]. The author shows, using an example of a project, how a project manager could analyze different cost trade offs in total effort and cost of the project by working with various factors such as staffing, platform, and language.  The analysis is done using effort multipliers, which are used to adjust the total available effort of a project based on the historical data on the previous projects and the size of the current project [Boe00]. The COCOMO II model has three different versions of its model depending on the development phase in which it is applied. The model has more cost drivers in comparison to the initial COCOMO model. Chulani *et al.* show that COCOMO II provided effort estimation within 30% accuracy 52% of the time which is still large error [CBS99]. A study done in [LPH02] on a small set of 19 projects has achieved 31 % MRE.

According to the author, COCOMO II can be used, provided the organization where a project is being developed has collected accurately the data needed for the COCOMO II model. Otherwise, the model could be used only to provide relative guidance and cannot be used as a precise model for effort estimation [Boe00]. The COCOMO II model does not directly take into account the impact of NFRs on effort, but the model includes impact of NFR on effort only partially requiring subjective judgement from human experts [Boe81].

###### ***3.1.1.2.5. SEER-SEM***

There is a range of commercial tools available on the market that perform automated effort estimation based on size and other project attributes such as project staff, experience and development environment. SEER for Software Estimation is one of such tools developed by Galorath. The software has functionality to track project effort and estimate its effort and cost. The size is measured using the IFPUG method. The tool also uses historical data, ISBSG benchmark data and project attributes to perform effort estimation and compare it to the historical data [Gal08]. SEER-SEM supports estimation at various phases of the project like COCOMO II. It is partially based on the model of R. Jensen developed in 1979. The effective size is measured in the following way taking into account not only the new size of the software to be developed, but also the existing software size, redesigned size, reimplementation size and retested size:

$Se = NewSize + ExistingSize \times (0.4 \times Redesign + 0.25 \times Reimpl + 0.35 \times Retest)$

Next, the effort is calculated using the formula similar to other parametric models:

$K = D^{0.4}(Se/Cte)^{1.2}$

D = staffing complexity (rate at which staff added)

Cte = effective development technology constant taking into account efficiency or productivity of development, people, process, and product parameters [FMG05].

The SEER-SEM is a proprietary tool that requires license purchase and it does not consider NFR impact on effort during the effort estimation process. SEER-SEM is reported to be using the Jensen model partially, but actual mathematical models used in the effort estimation are not shown as they are considered intellectual property of Galorath [Gal08]. However, a user can simulate and analyze trade offs between project effort, cost, schedule and staffing by using input parameters of the tool. Users can also enter the data from past projects to allow the tool to perform analogy based size and effort estimation [Gal08]. However, it is not clear whether the SEER-SEM model allows performing feature reduction in order to reduce the complexity of the effort estimation model.

###### *3.1.1.2.6. Select Estimator*

The Select Estimator effort estimation software tool is based on ObjectMetrix Model developed in 1998. It is recommended to be used for distributed large scale development and incremental life-cycle projects. The estimation is performed by splitting a project into project elements (objects and/or components), which are assigned a predefined activity profile with a predefined effort estimate. Next, a set of qualifiers is applied on the effort estimate to adjust the effort based on the project scope, technology and staff parameters [BC00]. The problem with this approach is that these parameters are not calibrated to be relevant for the current problem domain or organization. The NFRs such as usability, performance and security are not considered in the effort estimation process. However, there is a parameter designed to consider software reuse [BC00].

###### *3.1.1.2.7. Checkpoint*

Checkpoint is a commercial tool that uses primarily software size, measured using Function Points (IFPUG), to determine the effort. It allows the effort estimation to be done at task level, activity level, phase level, or project level. It does not take into account the problem domain and impact of NFRs on effort [BC00]. It is not known whether it performs feature reduction before performing effort estimation.

###### *3.1.1.2.8. PRICE-S*

PRICE-S is a commercial tool originally developed for the US Department of Defence and NASA. The estimation algorithms used in the model are not known very well, but Park has discussed some of the algorithms used in the model [BC00]. The model consists of three submodels, among which the Sizing submodel measures the software size using Function Points, SLOC or Predictive Object Points (POP). The PRICE-S considers also the development process, programming language and organizational productivity during the effort estimation [Pri11]. It does not take into account the impact of NFRs on effort, but it looks like the problem domain is considered [Pri11].

###### *3.1.1.2.9. ESTIMACS*

The ESTIMACS was developed by Howard Rubin in the 1970s and became part of a commercial product later on [BC00]. The model considers various parameters, such as the function size measured using function points, customer complexity, target system complexity, and developer knowledge. The estimation is performed after the estimator answers a set of predefined questions [BC00]. The model allows estimators to perform staffing and cost estimations. The impacts of NFR and problem domain are not taken into account. Also, the feature reduction is not performed [BC00].

###### *3.1.1.2.10. Parthasarathy Method*

Parthasarathy recommended to calculate effort by using the following formula: Effort = Application Size * Productivity + Project Management Effort + Configuration Management Effort [Par07]. Project Management Effort and Configuration Management Effort are optional according to him and added as needed by the estimator. The Application Size would be measured by IFPUG FPs, Object Points, SLOC, or UCP. Productivity would be derived based on the technology platform and development environment calibrated by the organization's historical data. He also suggests revising effort estimates during the project execution as project scope, software design, skill level of the team, and productivity change. In addition, it is useful for the team to improve the effort estimation model by increasing its usage in the development, evaluating and increasing its accuracy, and defining a clear process definition for the estimation. The author classifies the effort estimation models into heuristic and parametric models. The examples of heuristic models include expertise-based, analogy-based, bottom-up, top-down and algorithmic (based on regression or observed data-pattern). Whereas, parametric models include SLIM, SEER-SEM by Galorath based on Jensen Model, SELECT Estimator based on ObjectMetrix Model, COCOMO II, COSMIC-FFP, FP – Albrecht, etc. [Par07].

#### *3.1.1.2.11. Formal and Quantitative Methodology to Measure Effort [Kas09b]*

In [Kas09b], the impact of NFR is assessed using a formal and quantitative methodology. Also, NFRs Ontology is developed to model and organize NFRs. Kassab also developed effort estimation technique that allows adjusting effort estimate taking into account the impact of NFR on effort. In his technique, the linear regression model is generated to estimate effort and includes functional size measured using COSMIC CFPs. The technique uses the value of the impact of NFR to increase the effort value of a relevant requirement or to decrease the effort value of a relevant requirement or leave it unchanged. The adjusted effort per each requirement is then summed to get the overall effort estimate. However, the main assumption is that the estimator using the methodology uses the personal experience in implementing certain NFR in order to evaluate the impact of NFR on effort [Kas09b].

#### *3.1.1.2.12. Mendes Method*

It has been shown by Mendes *et al.* that the data from a single company could be useful to obtain more accurate results than data collected from different companies. Often companies tend to use data gathered from multiple companies due to absence of data from its own past projects. Forward stepwise regression (SWR) and case-based reasoning (CBR) were used to estimate effort using collected data. If a Web development company does not have data from its own past projects, then it was suggested mean or median-based estimation could be used to estimate effort of its new project [MMFG07]. The authors noticed the size and effort of cross-company projects were smaller than for single-company projects. However, they found the differences in sizes of cross-company and single-company project data did not affect results of their study [MMFG07]. The authors of the study did not consider an important factor of problem domain of the company projects, because there could be different problem domains addressed by projects of a single company. Also, the impact of NFR on effort was not taken into account during the effort estimation using this model.

##### *3.1.1.2.13. Kultur Method*

In [KKB09], the effort estimation is done by considering the application domain of projects and the distribution of effort. The application domain is considered to be an important factor that improved the effort estimation accuracy using this method. However, the projects are not grouped according to the problem domain. The functional size measured using Function Points is another important attribute of the model generated using this method [KKB09]. COSMIC and IFPUG are one of the several methodologies used to count Function Points. The model is built using the regression analysis, but the authors plan to try out machine learning algorithms as well. Feature reduction is not performed in this method. In addition, the impact of NFRs on effort is not considered in the estimation of the effort.

##### *3.1.1.2.14. Martin Method*

In [MPYT05], Martin *et al.* use fuzzy logic to perform effort estimation and compare it to the regression based effort estimation. The fuzzy rules involve McCabe's cyclomatic complexity, SLOC, and Dhama coupling measures. The effort estimation model generation involves these features. The model cannot be used in the early stages of software development, because it needs prediction of the SLOC. Also, the impact of NFR on the effort and problem domain of the software projects is not considered. In addition, the feature reduction is not used in the effort estimation model.

#### 3.1.1.3. Effort estimation methodologies using learning based models (Analogy, Neural Networks)

In [SSK96], the authors promote effort estimation by analogy using a software tool called ANGEL. The higher computational cost of effort estimation by analogy is considered to be one of the main disadvantages. Also, the authors have used brute force feature reduction technique which slows down the effort estimation process.

Angelis and Stamelos considered ways of calibrating the estimation method for the organizations planning to use estimation model based on analogy and proposed approaches to produce estimates within a certain interval range. Analogy based estimation model is one of the estimation techniques where the effort of a new project is estimated by using historical data from the previous projects [AS00].

The analogy based effort estimation model is not recommended when estimation analysts lack experience or sufficient past project data or there is no past project similar to the current one [PWL05].

The previous completed projects often differ from the new project being estimated. The difference between these projects can be analyzed using several important attributes, such as a “distance” metric between projects. The distance metric, Euclidian distance, is calculated using the values of certain common attributes of the new and old projects. The shorter the distance, the closer the projects are. The effort of the new project is calculated as the mean of the efforts of the close projects [AS00]. The analogy based effort estimation techniques rely on historical data set to estimate the new project’s effort. However, the generation of the effort estimate is based mostly on the distance between the new project and old projects [AS00]. The problem domain is not considered directly in determining which projects from the historical database are relevant for the effort estimation. Moreover, the authors have acknowledged that estimation of the distance between nominal features is not clarified yet and needs to be studied further [AS00].

Azzeh *et al.* propose ways of improving analogy software effort estimation using a fuzzy feature subset selection algorithm and comparing it to other existing analogy software effort estimation methods. Some of the benefits of using the subset of features available in the project include an increase of accuracy and performance of the effort estimation model and a reduction of training time for the model [ANC08]. The full list of features may contain certain redundancy or irrelevant features, which may lead to inaccurate results. There are two main approaches to find the feature subset: exhaustive searching algorithms (e.g. Wrapper) or statistical approach (Filter). Azzeh *et al.* propose an exhaustive search algorithm that uses fuzzy c-means clustering and fuzzy logic to identify the best feature subset. The Fuzzy feature subset selection (FFSS) algorithm proposed by authors performed better than exhaustive search and forward selection. Authors recommended using FFSS, Forward and Backward selection algorithms when better computation time is needed. On the other hand, FFSS would not be the best choice to use if the data set is large, because it is an exhaustive algorithm [ANC08]. It has been noted that further research is necessary to identify whether fuzzy logic is a reliable method of feature reduction when using analogy software effort estimation (ASEE).

Braga *et al.* have studied an application of genetic algorithm (GA) on feature selection and parameters optimization for support vector regression (SVR) used for the effort estimation. The GA method in the software domain is a concept where the best features that contribute towards optimal arrival at solution (i.e. effort) are kept and others are removed. A data set from NASA and another data set, Desharnais, were used to validate the approach. The authors showed positive results in the application of their GA-based approach on the effort estimation in both data sets. MMRE was lower and PRED(25) was higher for GA-based approach for both data sets. The training was done in Desharnais data set, but the considerable difference was identified in NASA data set, which was used for testing only [BOM08]. The authors did not consider the impact of NFRs on effort in the process of effort estimation and calibration of the effort estimation models using the historical data set.

Burgess *et al.* compare various machine learning (ML) based effort estimation models, artificial neural network (ANN) models, case based reasoning (CBR) model, and genetic programming (GP) based effort estimation model. The authors intend to prove that GP can produce better result in the effort estimation than other techniques. The comparison of these techniques is based on accuracy, explanatory value and ease of configuration criteria. The authors consider accuracy by itself is not a sufficient criterion for the selection of the best suitable technique. The study used the Desharnais data set as well, but the authors did not exclude any outliers like some other studies did. The results of the study showed there is a need for an additional study of the GP and different measures such as MMRE, AMSE, and Pred(25) used to evaluate the effort estimation models. The authors have noticed that GP has performed well only in Correlation and MMRE measures, but it was not the best among other measures such as AMSE, Pred (25), a BMMRE. On the other hand, in GP one could see easier how an effort estimate is derived by using the algebraic expression generated by the use of the model. However, authors found that CBR could be useful for the software project manager in better understanding of the effort estimate. In terms of ease of configuration, GP and ANN require certain level of knowledge and expertise in order to set it up with different parameters, which increase the complexity of the effort estimation model [BL01]. One of the main disadvantages of GA is the configuration of GA's free parameters and absence of guarantee that the found solution is the most optimal one [CCT01]. In contrast, linear regression models require less effort for setup and configuration. The accuracy and ease of configuration of the GP and ANN tend to be inverse proportional [BL01].

Shan *et al.* have studied Grammar Guided Genetic Programming (GGGP) in comparison with linear regression models. GGGP is an evolutionary method calibrated by the data from past projects in order to obtain the effort estimate for the current project. The method used by authors does not require closure requirement associated with GP. Also, GGGP reduces the search space due to the usage of grammars that supply background information and evolve during the GP search process. The training and validation were done on the ISBSG data set. The obtained results showed that all compared methods had high MMRE, but GP had relatively better values for other error measures in comparison to other models. However, the authors did not focus on the impact of non-numerical values (e.g. usage of 4 GLs) on the final effort estimate and considered to study them in their future work. They also did not consider the impact of NFRs on effort. Moreover, they plan to provide more background knowledge in their developed grammars and handle missing values of project attributes as part of their follow-up study [SMLE02].

Park *et al.* have studied neural network models for effort estimation, taking into account different software development attributes. The authors have identified that the neural network they built using 6 input variables, such as Function Point Size and Staff Experience, had better MRE compared to expert-based and regression effort estimation models including only Function Point Size. The training and validation set consisted of

174 projects from a Korean IT company [PB05]. This study did not include the impact of NFRs on effort, but at least it attempted to show there could be additional features impacting effort in addition to Functional Size [PB05].

Shukla states that certain parametric/algorithmic models (CARTX, back-propagation-trained neural network (BPNN), quick-propagation trained neural network (QPNN)) can not capture the complex relationship between project attributes and effort estimate. Shukla studied neuro-genetic models of software development effort, calibrated using historical data. The training of this neuro-genetic model is performed using a genetic algorithm (GA), and two data sets from COCOMO and Kemerer were used for both validation and training. It was found that the genetic algorithm neural network (GANN) had better accuracy than CARTX and QPNN. The author mentioned at the same time that lack of historical data often leads to improperly calibrated effort models used in effort estimation [Shu00].

There have also been attempts to better understand how software development cost is calculated using Neural Networks by mapping the neural network to a fuzzy rule-based system as attempted by Idri *et al.* The Neural Networks used in the cost estimation have often been considered hard to understand, because they often do not show how the effort estimation model is derived [IKA02]. The authors use the method of Benitez *et al.* to map if-then rules from the ANN to fuzzy rules. The accuracy was not the main concern of their study. The ANN used in the study was Backpropagation three-layer Perceptron with sigmoid function. The training and testing of the approach were both done on the COCOMO’81 set. In the study, mapped fuzzy rules correspond to the COCOMO’81 cost drivers. As a result of the study, the authors identified fuzzy rules from ANN and were able to describe, based on several examples, how certain fuzzy rules could affect the obtained effort estimate. Nevertheless, there were certain problems with obtained fuzzy rules' consequence part, which sometimes could contain values invalid for a given cost driver. This makes it hard for the user to interpret the impact of a fuzzy rule on the effort estimation. Hence, authors concluded they need to conduct more research into finding other approaches to obtain more understandable fuzzy rules from the ANN [IKA02].

Stamelos *et al.* recommended Bayesian Belief Networks (BBN) as a supplemental method in addition to expert judgment based methods in order to take uncertainty into account. The authors studied how BBN can be used to estimate productivity at the early stage of a software project. As a result, they have discovered it is necessary to create different BBNs tailored for a problem domain and development environment. Moreover, it was suggested that BBN can be useful to companies which lack historical data [SADS03].

The Bayesian probabilistic effort estimation model can be used to produce an effort estimate containing a probability range, which allows being aware what is the uncertainty associated with the estimate. Pendharkar *et al.* compare the Bayesian probabilistic effort estimation model with nonparametric neural network based and regression tree based effort estimation models. According to their study, the Bayesian effort estimation model can change the outputted probability of the effort estimation model as more information is given as input. Also, they have suggested that Bayesian effort estimation model could allow input of additional information not part of the Bayesian model [PSR05].

Huang *et al.* designed a neuro-fuzzy tool to simulate thinking of software estimators performing software estimation [HHRC04]. The authors have validated the tool using data from COCOMO and industry projects. The fuzzy rules are specified by the human software estimators for the tool, which uses them to perform the actual estimation. However, the examples of fuzzy rules are not specified in the given study and there could be subjectivity introduced by the human estimator. A neuro-fuzzy bank is used to calibrate parameters of contributing factors, but it is not clear whether the calibration is done by taking into account the problem domain [HHRC04].

Bayesian Network models were compared with simple effort estimation models (mean and median-based), Manual Stepwise Regression (MSWR), and CBR in the study done by Mendes et al [MM08]. The study has shown that the MSWR method was the best effort estimation model for web based projects among compared techniques, while simple effort estimation models can provide better estimation results than more complex models like BNs [MM08]. It is known that BNs support the inclusion of uncertainty in effort estimation. The study used data from the Tukutuku Benchmarking project, which contains data from 195 Web based projects. The authors recommended considering simple effort estimation models as an additional estimation model of Web based projects. In addition, they have mentioned usage of two training and/or validation sets' combinations give more precise ground for comparison of effort estimation models. Also, it was suggested that more detailed comparison of various BNs is needed in future to benchmark them better in comparison with other effort estimation models. Moreover, the study has shown the increase in total number of web pages leads to increase of total effort of web application development [MM08].

### 3.1.2. NFR Impact on Effort

In [ASM01], authors attempt to generate effort estimation models using categorical features by converting the categorical values to numerical ones using the CATREG tool and by building an effort estimation model using statistical regression. It was determined there are several important project and environment features, such as Development Type, Language Type, and Development Platform that affect the relationship between effort and functional size (Table 3-1) [ASM01]. The work was done using the ISBSG data set. The authors suggested using the generated effort estimation model for organizations without previous historical database [ASM01]. However, this may pose a problem if the problem domain of the new project to be estimated differs from the problem domain of projects from which the effort estimation model was generated.

Table 3-1: Features Affecting Productivity by L. Angelis [ASM01, AHKO11].

| **Data set** | ISBSG release 6 |
|---|---|
| **Features** | 1. Development Type<br>2. Development Platform<br>3. Language Type<br>4. Used Methodology<br>5. Organization Type<br>6. Business Area Type<br>7. Application Type |
| **Base of Size Measurement** | IFPUG Function Point |

In [FTAS08], the authors recommend the collection of Psychometrics to improve empirical studies. Angelis *et al.* consider human factors as additional crucial factors that may affect the relationship between effort and functional size [ASM01, AHKO11].

Linguistic values, such as low, high, very low, are often used as values for various project and cost attributes. These values present a certain problem during effort estimation by analogy. Idri *et al.* attempt to resolve this problem by using Fuzzy Analogy, consisting of the following steps: identification of cases, retrieval of similar cases, and cases identification. The study conducted by the authors identified Fuzzy Analogy to be more accurate in terms of accuracy in comparison with Fuzzy intermediate COCOMO'81, Classical intermediate COCOMO'81, and Classical analogy. However, the Fuzzy Analogy approach did not satisfy learning and uncertainty criteria specified by Soft Computing mentioned by Zadeh [IAK02]. In addition, the work done by Idri *et al.* did not consider the feature reduction of the feature subset in order to reduce the complexity and improve performance of the effort estimation model.

In [MF00], Maxwell *et al.* study a data set collected from Finland based companies in order to better understand features affecting productivity and effort estimation [AHKO11]. Authors have reported that Efficiency Requirement and User Interface are some of the features that impact productivity (Table 3-2). On the other hand, it was found that the DBMS Architecture does not affect productivity [AHKO11].

Table 3-2: Features Affecting Productivity by Pekka Forselius (adapted from [MF00, AHKO11]).

| | |
|---|---|
| **Data set** | Experience Database (206 business software projects collected from 26 companies). |
| **Variables considered in Database Productivity Analysis** | Application Programming Language, Application Type (MIS etc), Hardware Platform, User Interface, Development Model, DBMS Architecture, DB Centralization, Software Centralization, DBMS Tools, Case Cools, Operating System, Company where project was developed, Business Sector (Banking, Insurance etc), Customer Participation, Staff Availability, Standard Use, Method Use, Tool Use, Software Logical Complexity, Requirement Volatility, Quality |

| | |
|---|---|
| | Requirement, Efficiency Requirement, Installation Requirement, Staff's Analysis Skills, Staff's Tools Skills, Staff's Team Skills, Staff's Application Knowledge |
| **Base of Size Measurement** | Experience 2.0 Function Point Method |

Liebchen and his colleague study factors affecting productivity [LS05]. They have found that the Degree of Technical Innovation, Team Complexity, and Project Management experience are some of the important factors for software productivity (Table 3-3). Also, the authors suggest that software productivity varies based on the industry sector.

Table 3-3: Factors Affecting Productivity by Martin Shepperd [LS05, AHKO11].

| **Data Set** | 25,000 closed projects of a large multinational company |
|---|---|
| **Attributes Influencing Software Productivity** | 1. The Degree of Technical Innovation, Business Innovation, Application Innovation,<br>2. Team Complexity<br>3. Client Complexity<br>4. Degree of Concurrency<br>5. Development Team Degree of Experience With Tools, Information Technology, Hardware, or With Adopted Methodology<br>6. The Project Management Experience |
| **Base of Size Measurement** | Function Point |

In [MP08], the authors report effort estimation being impacted by the requirement changes and ambiguity, unavailability of templates and problems with coordination between the software project and product development [AHKO11].

In [LWHS01], Lokan *et al.* determined that the productivity of projects collected in the ISBSG data set is influenced by the Programming Language used for the development of the product, the development Team Size, the Type of Organization and Application Type [AHKO11].

The study of the China Software Benchmarking Standard Group by Yang *et al.* has found that the Software Development Effort (Tale 3-4) is influenced by the Development Size, Software Size, Team Size and Development Life Cycle [AHKO11, YHLWB08]. However, the software size measurement is done using LOC instead of Function Points of CFPs.

Table 3-4: Factors Affecting Phase Distribution for Software Development Effort [AHKO11, YHLWB08]

| **Data Set** | China Software Benchmarking Standard Group |
|---|---|
| **Factors** | 1. Development Life Cycle<br>2. Development Size<br>3. Software Size<br>4. Team Size |
| **Base for Size Measurement** | LOC |

Based on the recent work done in this area, it can be observed that the factors and features reported previously match concepts of the main NonFunctionalRequirement concept in the NFRs Ontology [AHKO11, Kas09b].

## 3.2. Open Problems

After reviewing the related work as shown previously, we can summarize the open problems as shown in Table 3-5.

Table 3-5: Linking Open Problems to their corresponding sections in the related work

| **Open Problem** | **Link to the Related Work** |
|---|---|
| [OP1] The estimation of effort have tendency to include human estimator's subjectivity leading often biased results. | Section 3.1.1.1 |
| [OP2] The impact of NFR on effort is often not considered in the effort estimation. | Sections 3.1.1.2, 3.1.1.2.3, 3.1.1.2.4, 3.1.1.2.7, 3.1.1.2.8, and 3.1.1.2.9 |
| [OP3] The quantification of the impact of | Sections 3.1.1.2, 3.1.1.2.4, and 3.1.2 |

| | |
|---|---|
| NFR on effort is not performed objectively. | |
| [OP4] The effort estimation techniques using historical database do not clearly distinguish clustering of projects by problem domain to achieve better precision. | Sections 3.1.1.2, 3.1.1.2.6, 3.1.1.2.7, and 3.1.1.2.9 |
| [OP5] The effort estimation methodologies do not always allow the user to perform feature reduction techniques. | Sections 3.1.1.2, 3.1.1.2.1, 3.1.1.2.2, 3.1.1.2.3, and 3.1.1.2.9 |
| [OP6] Data collection from historical projects is not clearly defined. | Section 3.1.1.2 |

The above open problems are addressed by the work described in this thesis. The next chapter presents the effort estimation methodology that describes the proposed solution for the above mentioned problems.

# Chapter 4: Effort Estimation Methodology

Our effort estimation methodology takes into account the problem domain of the software project to be estimated, by generating effort estimation models specific to a certain problem domain. Also, it incorporates nominal features, such as the impact of the NFRs on effort. The complexity of the models is reduced using a feature subset selection algorithm. The effectiveness of the methodology is validated using case studies presented in the experimental work chapter.

The methodology has two parts: i) generation of an effort estimation model from historical data; and ii) application of the model on the new project(s) (Figure 4-1). The separation of the methodology into these parts allows organizations to easily apply them together or separately, based on the required purpose. In the first part, we cluster projects by problem domain, gather historical data, split the feature subset into nominal and numerical feature groups, reduce the nominal feature subset using either a statistical or semantic method, generate the effort estimation model from the feature subset, consisting of the reduced nominal feature subset and the original numerical subset. In the second part, we identify the new project's problem domain, gather its objectively measurable features, estimate the new project's subjective features, and estimate the new project's effort using the generated effort estimation model. The number of steps in each part of the methodology is designed to be as minimal as possible in order to facilitate the learning process for organizations and users that would like to adopt our methodology.

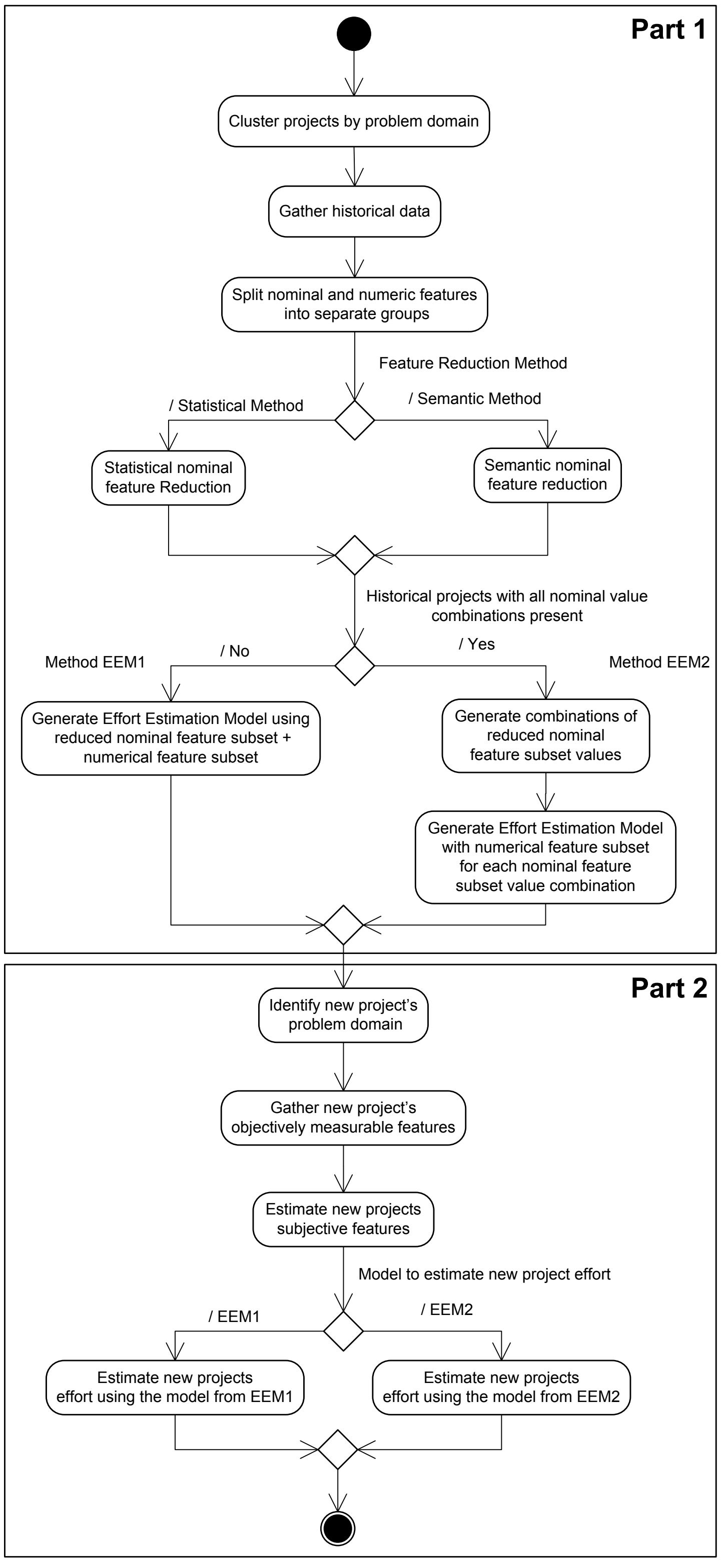

Part 1
Cluster projects by problem domain
Gather historical data
Split nominal and numeric features into separate groups
Feature Reduction Method
/ Statistical Method
/ Semantic Method
Statistical nominal feature Reduction
Semantic nominal feature reduction
Historical projects with all nominal value combinations present
/ No
/ Yes
Method EEM1
Method EEM2
Generate Effort Estimation Model using reduced nominal feature subset + numerical feature subset
Generate combinations of reduced nominal feature subset values
Generate Effort Estimation Model with numerical feature subset for each nominal feature subset value combination
Part 2
Identify new project's problem domain
Gather new project's objectively measurable features
Estimate new projects subjective features
Model to estimate new project effort
/ EEM1
/ EEM2
Estimate new projects effort using the model from EEM1
Estimate new projects effort using the model from EEM2

Figure 4-1: Effort Estimation Methodology Steps

The nominal feature reduction step of the methodology proposed here can be considered as a point of extension where the estimator can decide to plug-in the desired method to reduce the number of nominal features (e.g. impact of NFR on effort, the type of architecture style used in the project, project difficulty). In this work, we use the feature subset reduction algorithm CFS of WEKA developed previously by Mark A. Hall in order to reduce the set of nominal features. However, one can use the wrappers method or another filter method to reduce the feature subset.

Also, it was observed that effort estimation models perform better when the nominal feature values are converted into numerical ones. For example, the impact of performance on effort values can be mapped as following {-2=Very Low, -1=Low, 0=Nominal, 1=High, 2=Very High} where Very Low means the performance requirements reduce the effort significantly, Low means the performance requirements reduce the effort slightly, Nominal means the performance requirements do not have any impact on the effort, High means the performance requirements increase the effort slightly, Very High means the performance requirements increase the effort significantly.

Furthermore, we have found that separation of original features into nominal and numerical groups, reducing separately the subset of nominal features and then combining the reduced subset with the original numerical subset produces better result with the linear effort estimation model.

We have designed a special questionnaire to collect important historical information about projects. The questionnaire can also be used for the new project to be estimated. It makes it easy and efficient to gather the impact of NFR on effort, project complexity, and the average experience of the project team members.

## 4.1. Generate Effort Estimation Model from Historical Data

### 4.1.1. Cluster Projects by Problem Domains

Problem domains dictate the use of different architectural design patterns and, thus, play a significant role in predicting the complexity of the software to be developed. Our work identifies how this variation in the problem domains translates into changes in development effort, by first clustering the historical dataset into problem domain categories before calibrating our effort estimation model. This step addresses open problem [OP4] mentioned in section 3.2.

Software development industries categorize the problem domains for organization of their product inventories. Thus, the classification of problem domains varies from organization to organization based on their internal needs. For example, Microsoft Corporation [Mic11] prescribes 40 different classes of problem domains for software products. We, therefore, allow the decomposition of problem domains into open categories that can be customized to have an organization-specific classification. We set the following attributes to describe a problem domain class:

| | |
|---|---|
| **id:** | INT |
| **name:** | STRING |
| **application_type:** | {"desktop", "web", "plug-in", "real-time", "developer", "publisher", "embedded", "business", "utility", "game", "academic", "communication", "system", "portable", "graphics", "multimedia", "driver", "framework", "research", "prototype", "component", "other"} |
| **deployment_type:** | {"private", "public-open", "public-closed"} |

where id allows us identify each problem domain uniquely, and application_type and deployment_type allows a higher level classification of the problem domain to provide additional nominal features during model calibration [AHKO11].

Each instance in our historical dataset that represents a software project is tagged with a problem domain id, indicating the problem domain that the software belongs to. Thus, when calibrating our effort estimation model, we first choose a target problem domain based on the software project that is to be estimated. Our system then automatically selects from the historical database the instances that belong to the chosen problem domain and calibrates the effort estimation model based on those instances only.

### 4.1.2. Gather historical data

In this step, we gather past projects' data, which includes effort and other important variables such as Size, NFR Impact, etc. We classify projects into the corresponding problem domain classes. We use objective guidelines on assigning different NFRs nominal values (e.g. analyze how much LOC, effort spent per nominal NFR value).

Size can be measured in function points, COSMIC CFPs, or any other accepted unit of measure, as long as historical projects and the project to be estimated are consistent in the method of Size counting. Our approach is to use COSMIC CFPs in the methodology validation.

The impact of NFR on effort, average experience of project members, and project complexity are some of important features that can be gathered effectively using the questionnaire we have designed. Therefore, the open problems [OP1] and [OP2] mentioned in section 3.2 are being addressed accordingly. The main prerequisite is that the questionnaire needs to be filled in by a team member of the project or a person who has access to the historical data.

If there are missing values, then a placeholder mark (e.g. question mark - ?) can be used that is understandable by project members and software tool to be used for regression analysis.

### 4.1.3. Split the feature subset

Split the feature subset of historical projects into 2 groups: nominal and numerical. Nominal features could include variables with categorical values such as Low, Nominal, and Very High. For example, the impact of NFR on effort and the project complexity are one of the types of nominal features. Numerical features include variables with numerical values. For example, Size and Number of Developers are numerical features.

### 4.1.4. Feature subset reduction

For each domain class, in order to reduce complexity and increase precision of effort estimation models, perform one of the below mentioned techniques. The goal here is to eliminate NFRs and nominal features that do not affect the effort variable or is found to be redundant.

The feature subset reduction can be done either using a statistical method or semantic method. The statistical method relies on statistical principles to find redundant features. On the other hand, the semantic method is based on the analysis of the semantic meaning of each feature and its contribution towards the class feature (e.g. effort).

If the projects in the historical database have no nominal features, then this step can be skipped and we can proceed to the next step.

#### 4.1.4.1. Statistical feature reduction

The statistical feature reduction uses either the Filter or Wrappers method to reduce the number of features in the feature subset.

1. Assign each nominal feature a numerical value, such as Low = -1, Medium = 0, High = 1, if possible.

2. Run the feature subset selection analysis to find out what nominal features have more impact on Effort. The Filters or Wrappers method can be used for this step. For example, WEKA provides the Correlation-based Feature Subset Selection algorithm for feature selection using CFS Filter approach.

3. Select the best feature subset of nominal features and proceed to the next step, where the best feature subset replaces the original set of nominal features.

#### 4.1.4.2. Semantic feature reduction

The semantic feature reduction method uses previous knowledge about features to identify redundancy among them. For example, one can use an available study such as [SX07] and to analyze which combination of NFRs is redundant and eliminate the redundant features. We can expand our semantic feature reduction technique further as more studies will be done in this area,

### 4.1.5. Generate effort estimation model

The effort estimation model can be generated in two different ways, which can be practical for organizations that would like to use our methodology. If there are enough historical projects to cover all combinations of nominal features, then the method EEM2 can be used (see section 4.1.5.2). Otherwise, the method EEM1 can be used (see section 4.1.5.1).

#### 4.1.5.1. Method EEM1

The result of the reduced nominal feature subset is combined with numerical features (Figure 4-2). For example, let us consider the data set that originally contains nominal features, such as ProjectComplexity, PerformanceImpact, UsabilityImpact, SecurityImpact, MaintainabilityImpact, and numerical features, such as NumberOfDevelopers, AverageExperience, Size, and Effort. The reduced nominal feature subset contains PerformanceImpact and SecurityImpact after the feature subset reduction step performed previously in our methodology (Section 4.1.4). As a result, the combined numerical and nominal feature set contains now: NumberOfDevelopers, AverageExperience, Size, Effort, PerformanceImpact, and SecurityImpact. The effort estimation model is generated from this combined subset. The generation of the model can be done using available tools such as WEKA.

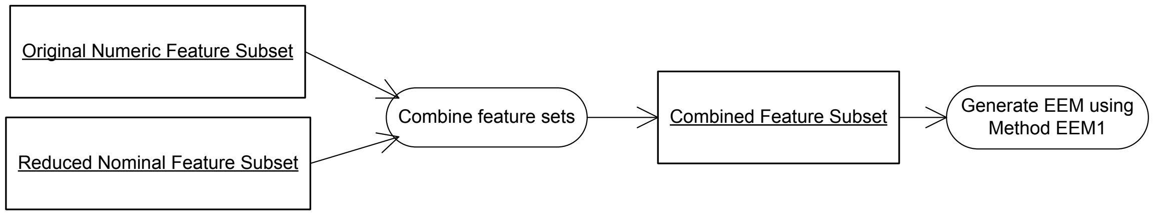

Original Numeric Feature Subset
Reduced Nominal Feature Subset
Combine feature sets
Combined Feature Subset
Generate EEM using Method EEM1

Figure 4-2: Generation of EEM using the Method EEM1

#### 4.1.5.2. Method EEM2

The method EEM2 generates a separate effort estimation model for each combination of nominal feature values based on the historical projects data.

For example, let us have N nominal features and M numerical features.

Nominal_Feature1 has 3 values {Low, Medium, High}.

Nominal_Feature2 has 5 values {Very Low, Low, Nominal, High, Very High}.

…

Nominal_FeatureN has 5 values {Very Low, Low, Nominal, High, Very High}.

| Model # | Nominal_Feature1 | Nominal_Feature2 | … | Nominal_FeatureN |
|---|---|---|---|---|
| 1 | Low | Very Low | … | Very Low |
| 2 | Medium | Very Low | … | Very Low |
| 3 | High | Very Low | … | Very Low |
| 4 | Low | Low | … | Very Low |

| | | | | |
|---|---|---|---|---|
| 5 | Medium | Low | … | Very Low |
| … | … | … | … | … |

Each generated effort estimation model contains only numerical features. Therefore, a new project to be estimated will be mapped into the corresponding effort estimation model by finding out to which combination it matches. For example, if a new project has Nominal_Feature1 = Low, Nominal_Feature2 = Very Low, Nominal_FeatureN = Very Low, then the model #1 will be used to estimate the effort of the new project.

## 4.2. Apply generated Effort Estimation Models on new projects

### 4.2.1. Identify the new project's problem domain

Find out to what problem domain the new project belongs to. The schema described earlier in section 2.1.1 for the classification of the project problem domain can be reused for this purpose as well. This step addresses open problem [OP5] mentioned in section 3.2.

### 4.2.2. Gather the new project's objectively measurable features

Gather objectively measurable features of a new project. For example, Size, Number of Developers, Average Experience can be objectively measured and quantified for the new project.

If a new project has more features than the historical data set, then omit the extra features. These features can be added later on when the new project becomes part of the historical data set and the effort estimation models will be recalibrated.

If the new project has fewer features than the historical data set, then the missing feature values can be marked using a special mark understandable by estimator and the software to be used for regression analysis (e.g. question mark - ?).

### 4.2.3. Estimate the new project's subjective features

Estimate features that can not be predicted or measured objectively (e.g. impact of NFR on effort) using the methodology based on historical projects' data and objective features of a new project such as Size, Problem Domain, Average Experience.

The subjective features can be estimated by extracting a model from the objectively measured features of historical projects, while keeping out the other nominal features. The generated model is then used with new project's objectively measured features as input to estimate the subjective features such as nominal features.

The estimation model can be generated using available tools in the following way:

- Use WEKA tool's regression algorithms directly.
- Use WEKA tool's CFS algorithm to reduce the feature subset and then apply the WEKA regression algorithm (e.g. Linear Regression) on the reduced feature subset.
- Use another algorithm of choice of a statistical software tool such as MATLAB [Mat11] or SAS Analytics [Sas11].

We have found that estimation model tend to achieve higher accuracy when the nominal feature values are converted first to numerical ones before using CFS or regression algorithms. The estimation model can be specified to generate values in a range to give more flexibility for the estimator. The step of this methodology helps to reduce the subjectivity of the estimation of nominal features, which can be incorrectly assessed by human estimators. Therefore, the open problems [OP1], [OP2] and [OP3] mentioned in section 3.2 are being addressed accordingly.

### 4.2.4. Estimate new project's effort

The effort of a new project can be estimated using effort estimation models generated with the help of Method EEM1 or Method EEM2.

If the Method EEM1 is used, then it is necessary to plug in values of objectively measured features and values of subjective features into the generated effort estimation model. If the method EEM2 is used, then:

1. Map nominal feature values into the corresponding effort estimation model of the particular problem domain generated previously, in order to identify the correct effort estimation model.

2. Plug values of numerical features of the current project into the effort estimation model to estimate the effort of the new project.

The effort estimation model derived using Method EEM1 and Method EEM2 can be specified to generate values in a range to give more flexibility for the estimator. For example, the effort of a new project can be between 100 and 120 person days.

## 4.3. Questionnaire

The questionnaire designed in this thesis allows collecting important information about new and historical projects. It simplifies and helps the process of gathering information about the impacts of NFRs on effort, project complexity, and average experience of project member. The questionnaire addresses the open problem [OP6] mentioned in section 3.2. The questionnaire template is given in Appendix A.

The background section of the questionnaire contains a sample text to explain the purpose of the questionnaire for respondents. It could be replaced by organization specific background information. The intended audience section describes who should be filling in the questionnaire. The questionnaire can be filled in by the project manager or scrum master who has good knowledge and overview of the project, because certain studies have found project managers/scrum masters have provided overall better estimations [Jor04].

Questions are split into 2 groups: general questions and NFRs questions. The goal of the general questions is to find out the average experience of the development team with the development environment and development languages, number of development locations, and the impact of complexity of the product, risk management strategy, and delivery deadline on the effort. The general questions section contains 7 questions.

**Question 1: How many years of experience the development team have with the development environment and languages (e.g. Java, Eclipse SDK, or any other relevant technology for the problem domain or organization) at the time of execution of the project?** The purpose of question 1 is to understand the skill set level of the development team, by finding out information about the experience of a project member with the development environment and languages. Also, the questions about experience can be extended to include questions about other topics related to experience such as experience in a specific business area.

**Question 2: How many development locations (geographical) were involved in the development of the project? (e. g. 1, 2, 3, etc.)** The purpose of question 2 is to understand how distributed the development project was. Often, the increased number of development locations can create more complexity in the project management due to the delay of information propagation and decision implementation at all development locations.

**Question 3: Rate the impact on the project effort of the following factor: complexity of the product (related to Functional Requirements).** The purpose of question 3 is to understand how complex the product and project are in terms of the difficulty to implement FRs.

**Question 4: Rate the impact on the project effort of the following factor: the delivery deadline (schedule) (e.g., delivery of the project in accelerated format [more effort spent in the beginning of the project] or in stretched-out format [effort is stretched over the long period of time]).** The purpose of question 4 is to understand the pace of delivery of the project by finding out how effort is distributed over the course of the project. The question is inspired from a cost multiplier in the COCOMO II method [Boe00].

**Question 5: Rate the impact on the project effort of the following factor: the risk management done as part of your project.** The purpose of question 5 is to learn how well the risk management is done during the project execution. The efficient risk analysis, mitigation and management help to quickly find out and resolve upcoming risks, minimizing the impact on project budget, schedule or product quality.

**Question 6: Rate the impact on the project effort of the following factor: the team cohesion for the project you were part of (e. g. difficulty in synchronizing project stakeholders (users, customers, developers, others) due to differences of their objectives).** The purpose of question 6 is to find out how well the communication worked among project members and stakeholders and how well the team worked together to complete the project. For example, there could be conflicts between team members or difficulty to synchronize with stakeholders regarding the project status and decisions.

**Question 7: If you have any additional comments regarding your answers to general questions mentioned above, please provide it here.** The purpose of question 7 is to gather any additional feedback in a free text format to allow respondents to convey additional information.

The NFR questions section of the questionnaire contains questions regarding the impact of various NFRs on the effort on a scale of 5 (increases effort significantly, increases effort, no impact on effort, decreases effort, decreases effort significantly) as recommended in [Kas09b]. If the NFR does not apply to the project or product, then there is a Not Applicable (N/A) option which could be selected by the respondent. The definition of each NFR is given in the appendix section.

**Questions 8 to 19.** The questions 8 to 19 focus on assessing the degree of impact of each NFR on effort using the scale of 5.

**Question 20: If you have any additional comments regarding your answers to questions related to non-functional requirements mentioned above, please provide it here.** The purpose of question 20 is to gather any additional feedback in a free text format to allow respondents to convey additional information regarding NFRs.

The next chapter describes the implementation details of the methodology presented in this thesis.

# Chapter 5: Implementation of the Effort Estimation Methodology

The prototype implementation of the effort estimation methodology proposed in this thesis is done using the Java programming language and WEKA tool. However, an organization may choose to use another preferred programming language or statistical tool to implement our methodology. This chapter discusses the prototype implementation of our methodology and possible extension points.

## 5.1. Architecture

The architecture of the methodology prototype implementation follows a layered architecture style with 3 layers: User Interface (UI), Business Object (BO), and Technical Services (TS). The UI layer is responsible for gathering and visualization of information from and to the end user, respectively. The BO layer is responsible for the actual effort estimation model generation and estimation of effort for new projects. The TS layer is responsible for performing operations related to reading and writing the results of the effort estimation to the database (DB) or file system (Figure 5-1).

The current prototype implementation architecture is not yet web-based, but in the future we plan to integrate the effort estimation methodology into the LASR system developed by Ishrar Hussain in the READ research group at Concordia University [Hus11]. We will preserve the same 3 layered architecture style.

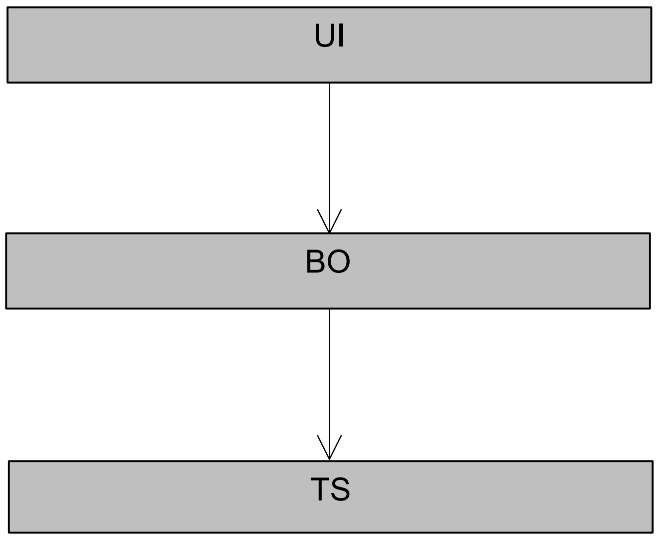


Figure 5-1: Architecture of the effort estimation methodology prototype implementation

## 5.2. Detailed Design

The section discusses detailed design of the effort estimation methodology implementation. We describe in detail the UI, BO, and TS layers of our implementation.

### 5.2.1. UI Layer

The UI layer contains 4 main entities: MainEEMUI, NewProjectEffortEstimatorUI, ProblemDomainManagerUI and EffortEstimationModelGeneratorUI. The MainEEMUI is responsible for presenting the end user with choices. The EffortEstimationModelGeneratorUI allows users to generate the effort estimation model from the historical projects. The NewProjectEffortEstimatorUI is responsible for performing estimation of effort for new projects (Figure 5-2). The ProblemDomainManagerUI allows users to manage problem domains.

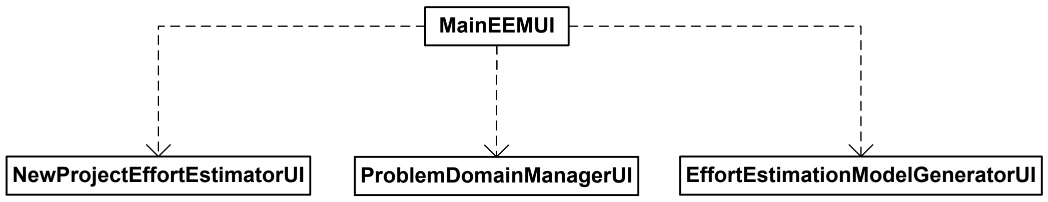
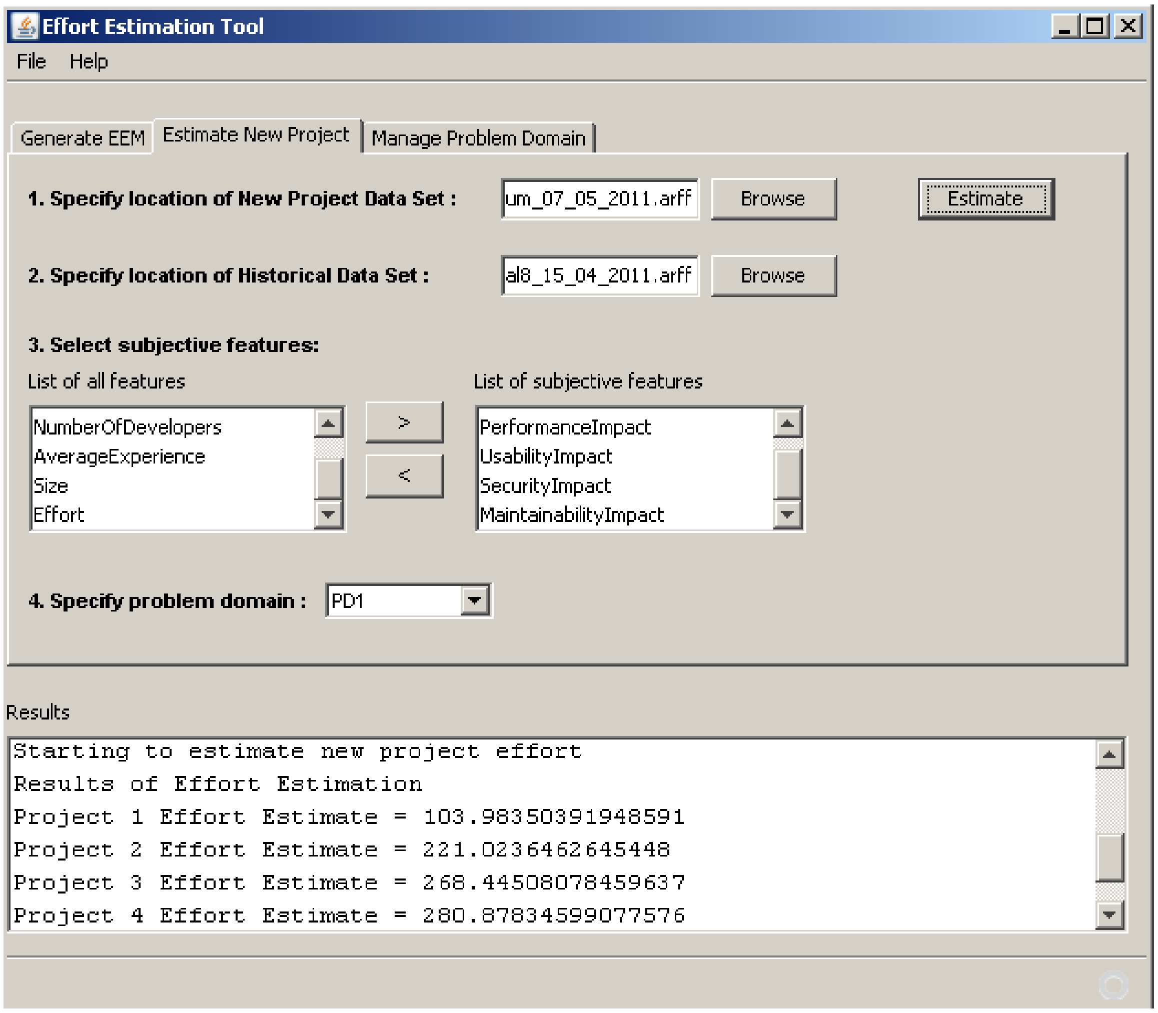


Figure 5-2: UI layer elements

The user interface implementation of the prototype is presented in Figure 5-3.



1. Specify location of New Project Data Set : um_07_05_2011.arff

2. Specify location of Historical Data Set : al8_15_04_2011.arff

3. Select subjective features:

| List of all features | List of subjective features |
|---|---|
| NumberOfDevelopers | PerformanceImpact |
| AverageExperience | UsabilityImpact |
| Size | SecurityImpact |
| Effort | MaintainabilityImpact |

4. Specify problem domain : PD1

```
Starting to estimate new project effort
Results of Effort Estimation
Project 1 Effort Estimate = 103.98350391948591
Project 2 Effort Estimate = 221.0236462645448
Project 3 Effort Estimate = 268.44508078459637
Project 4 Effort Estimate = 280.87834599077576
```

Figure 5-3: The implementation of the prototype – Estimate New Project mode

### 5.2.2. BO Layer

The BO layer contains 5 main interfaces with corresponding implementations: INewProjectEffortEstimatorBO, IEffortEstimationModelGeneratorBO, IProblemDomainManagerBO, IEstimationModelGeneratorBO, IApplicationTypeManagerBO, IDeploymentTypeManagerBO (Figure 5-4).

The INewProjectEffortEstimatorBO allows performing effort estimation of new projects using the generated effort estimation model. The IEffortEstimationModelGeneratorBO is responsible for generating effort estimation models and feature subset selection, which can be considered as a point of extension where the customer can plugin the desired method for semantic feature selection or connect to the desired statistical feature selection tool. The IEstimationModelGeneratorBO provides generic services, such as performing linear regression or performing least median square regression used during the estimation of new projects or generation of effort estimations models. The IProblemDomainManagerBO is responsible for the management of problems domains to support their creation, deletion and editing. The IApplicationTypeManagerBO allows managing application types. The IDeploymentTypeManagerBO is responsible for managing deployment types. The customers can provide their own implementation of the above mentioned interfaces in order to extend or modify the behaviour of the effort estimation tool.

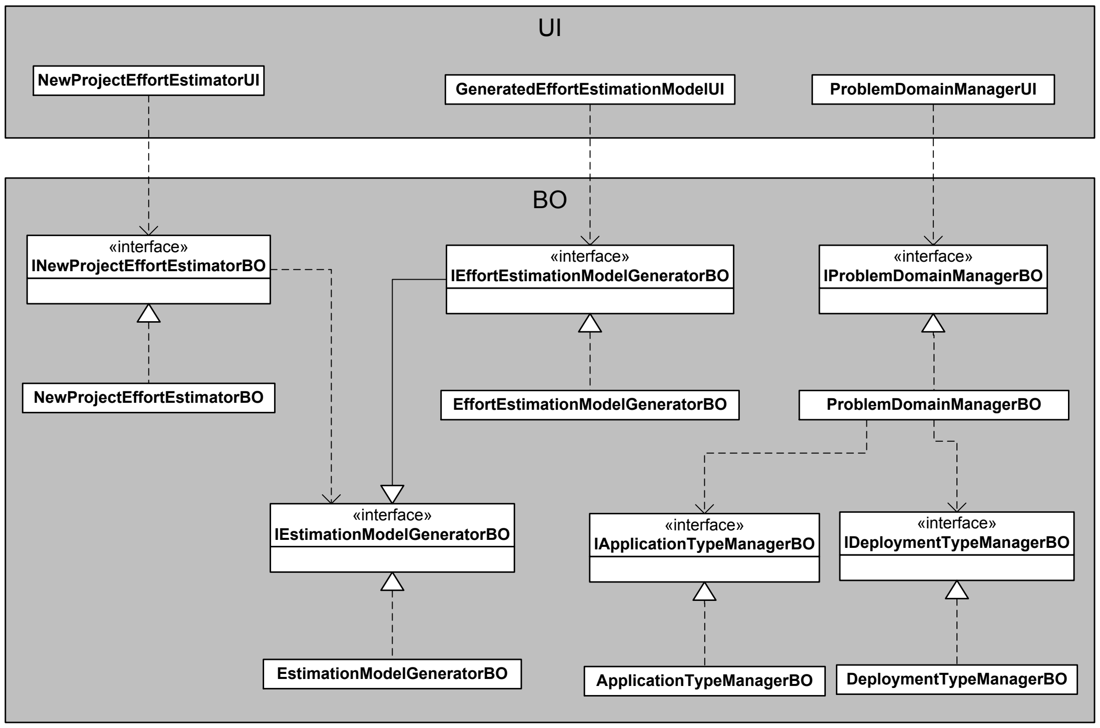

UI
NewProjectEffortEstimatorUI
GeneratedEffortEstimationModelUI
ProblemDomainManagerUI
BO
«interface»
INewProjectEffortEstimatorBO
«interface»
IEffortEstimationModelGeneratorBO
«interface»
IProblemDomainManagerBO
NewProjectEffortEstimatorBO
EffortEstimationModelGeneratorBO
ProblemDomainManagerBO
«interface»
IEstimationModelGeneratorBO
«interface»
IApplicationTypeManagerBO
«interface»
IDeploymentTypeManagerBO
EstimationModelGeneratorBO
ApplicationTypeManagerBO
DeploymentTypeManagerBO

Figure 5-4: UI and BO layer elements

### 5.2.3. TS Layer

The TS layer has IEffortEstimationTS with its corresponding implementation responsible for reading data sets containing new project data and historical projects. The IEffortEstimationTS is also responsible for saving any intermediate data sets or models containing effort estimation, which is necessary during the effort estimation process. The current prototype implementation uses a OS level file system, but in future we plan to use a database management system (DBMS) with the LASR system (Figure 5-5).

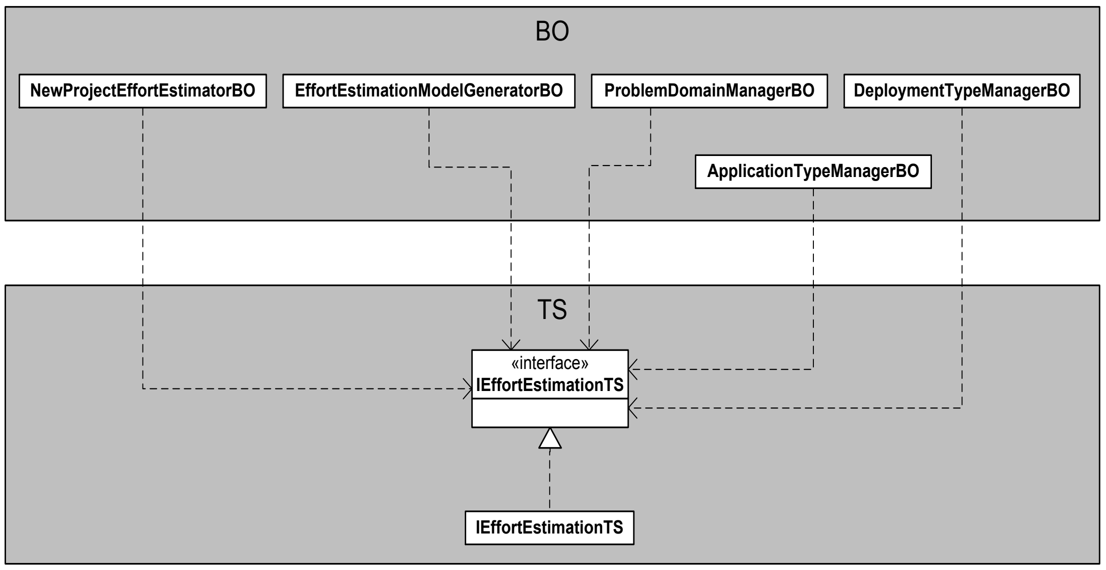


Figure 5-5: BO and TS layer elements

The next chapter presents the design of the case studies illustrating the methodology and discusses the results of the experimental work.

# Chapter 6: Experimental Work

## 6.1. Design

The validation of the methodology is done using two case studies. In the first case study, the data set DS1 consisting of industrial projects completed at a large international software development company is used. In the second case study, the data set DS2 from ISBSG is used. The data set is split into 3 groups according to the problem domain.

The data sets use COSMIC as a method of measurement for software size. The automation of certain steps in the methodology is achieved using the WEKA tool. However, an alternative statistical tool can be used by the estimator to execute the steps of the methodology. The details of the data sets are presented in Table 6-1.

Table 6-1: Data sets used in the validation of the methodology

| **Data Set** | **Source** | **Number of Projects** | **Problem Domain** | **Used in Case Study** |
|---|---|---|---|---|
| DS1 | Industry | 20 | PD1 | 1, 2 |
| DS2 | ISBSG | 151 | PD2 | 2 |
| DS2 | ISBSG | 18 | PD3 | 2 |
| DS2 | ISBSG | 64 | PD4 | 2 |

In the first case study, different experiments are performed to compare the approach of our methodology against a traditional approach where the effort estimation methodology is generated directly out of the data set.

In this case study, 16 projects are used to generate the effort estimation model and the effort of 4 new projects is estimated by human experts and by our methodology. The dataset of the first case study represents projects from the same problem domain PD1. The PD1 problem domain refers to application type = web and deployment type =

private. The private deployment type means the applications are only accessible within an organization only and not usable by users outside of the organization. In trial 1, the human expert is asked to estimate the new project's effort based on personal experience and knowledge. In the trial 2, we build a regular linear regression model out of the projects of the DS1 data set coming from the software industry as private projects completed at a large software company. The best model among all trials, excluding trials 1 and 2, is selected to estimate the effort of new projects. The result of this estimation is compared to the trials 1 and 2 results. The goal of trials [3-7] is to gradually observe the improvements in the generated effort estimation model by comparing the correlation coefficient and error rates. In the trials [3-7], we use WEKA based algorithms, which include ANN based algorithms and model based algorithms (e.g. linear regression) to derive the effort estimation model. The comparison of the trials' effort estimation model generation parts is based on the best performing algorithm of the trial. The best performing algorithm is selected based on the correlation coefficient, relative absolute error and relative squared error. Also, in trials [3-8], we perform 10-fold cross-validation during the generation phase.

In trial 3, we perform the feature subset selection on all features, including nominal and numerical ones. Next, we run the regression analysis on the reduced feature subset to derive the effort estimation model from the historical dataset. The generated effort estimation model is afterwards used to estimate new project's effort, but the subjective features of the new project, such as the impact of NFR on effort, are estimated by human experts. In trial 4, we perform similar activities as in the trial 3, but we also convert the nominal feature values into numerical ones (Table 6-2).

Table 6-2: Design of Experiment of case study 1

| Step | Trial | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Separate nominal and numerical features | | | | | | | X | X |
| Convert nominal feature values to numerical values | | | | X | | X | | X |
| Perform Feature Subset Selection on nominal features | | | | | | | X | X |
| Perform Feature Subset Selection on all features (numerical and nominal) | | | X | X | | | | |
| Run regression on the selected feature subset | | | X | X | | | X | X |
| Run regression on all features in the set | | | | | X | X | | |
| Run regression on all features in the set using LinearRegression | | X | | | | | | |
| Estimate new projects' subjective features using expert knowledge | | | X | X | X | X | | |
| Estimate new project's subjective features using our approach | | | | | | | X | X |
| Estimate new project's effort using human expert only | X | | | | | | | |

In trial 5, we directly run the regression analysis using all features in the set to generate the effort estimation model from the historical dataset. The generated model is used afterwards to estimate the new project's effort, but the subjective features of the new project, such as the impact of NFR on effort, are estimated by human experts. In trial 6, we perform similar activities as in trial 5, but we also convert the nominal feature values into numerical ones.

In trial 7, we separate the nominal and numerical features into two separate groups. Next, we perform the feature subset selection on nominal features only to reduce the feature set complexity. Afterwards, we take the reduced nominal features set and combine it with the numerical features to generate the effort estimation model from the historical dataset. The generated effort estimation model is used to estimate the new project's effort. However, the subjective features of the new project, such as the impact of NFR on effort, are estimated using our approach.  In trial 8, we perform similar activities as in trial 7, but we also convert the nominal feature values into numerical ones. When we arrive to trial 8, it represents our approach and its results are compared to the results of trials [3 – 7] to select the best performing trial.

In the second case study, the DS1 and DS2 data sets are used to compare our methodology to a regular approach using four trials. The DS2 data set comes from the ISBSG organization. The goal of this case study is to show that our approach performs better than a regular approach that does not take into account the impacts of different NFRs and different problem domains on the estimation of software development effort and does not reduce the feature space. The main difference between case studies 1 and 2 is that in the second case study, we have four problem domains as part of the experimental work and we perform more comparisons of the experimental work based on these problem domains.

The DS2 data set is ISBSG COSMIC Industry Data Release 1 of the International Software Benchmarking Standards Group (ISBSG). The ISBSG collects and distributes international datasets in the field of IT to improve management of the projects [ISB11]. The original data set contains 354 projects estimated using COSMIC method. The original companies or organizations that completed these projects are not disclosed by ISBSG due to privacy reasons, but they are coming from around the world. The projects are rated using the special quality mechanism of the ISBSG, which rates projects based on the four level quality rank containing rating A, B, C and D. We have selected only projects of the highest quality ratings (i.e. A and B), which correspond to projects that were assessed to have sound data and no factors (rating A) or some minor (rating B) factors that might impact the data integrity. We have selected 233 projects out of the original ISBSG data set to perform our case study. The selection mechanism was based on the quality rating, problem domains and availability of sufficient number of features to perform our empirical evaluation.

In the first trial of case study 2, we estimate the projects of problem domain PD2 using our methodology as indicated using keyword "Yes" in the Table 6-3. The problem domain PD2 refers to application type = business and deployment type = public closed.

Table 6-3: Outline of Case Study 2

| **New Projects' Problem Domain** | **Problem Domain of the Historical Data Set used for Effort Estimation Model Generation** | **Trial in the Case Study 2** | **Our Methodology** |
|---|---|---|---|
| PD2 | PD2 | 1 | Yes |
| PD1 | PD1, PD2, PD3, PD4 | 2 | No |
| PD2 | PD1, PD2, PD3, PD4 | 2 | No |
| PD1 | PD2 | 3 | No |
| PD1 | PD3 | 3 | No |
| PD2 | PD1 | 3 | No |
| PD2 | PD3 | 3 | No |
| PD2 | PD1, PD2, PD3, PD4 | 4 | Yes |

In the second trial, we estimate the effort of new projects of the problem domain PD1 and PD2 without using our approach as indicated using keyword "No" in the Table 6-3. In fact, the new projects are estimated using the linear regression based effort estimation model generated directly from the data set containing projects of problem domains PD1, PD2, PD3, and PD4, where PD3 problem domain refers to application type = utility and deployment type = public open and PD4 problem domain refers to application type = other and deployment type = public open. In the third trial, we estimate new projects of problem domains PD1 and PD2 without using our approach as indicated using keyword "No" in the Table 6-3, but this time new projects from problem domain PD1 are estimated using an effort estimation model generated from the problem domains PD2 and PD3 and new projects from the problem domain PD2 are estimated using effort estimation model generated from the problem domains PD1 and PD3. In the fourth trial, we estimate the projects of problem domain PD2 using our methodology as indicated using keyword "Yes" in the Table 6-3. However, the effort estimation model is generated using our methodology from the data set containing projects of problem domains PD1, PD2, PD3, and PD4. The goal of this trial is to validate how the problem domain affects the accuracy of the effort estimation performed using our methodology.

There are other possible scenarios of experiments that can be executed to validate additional cases as summarized in the Table 6-4. We only validate those scenarios that are most likely to occur in real life situation. For example, when our methodology is not used an estimator will most likely use the whole problem domain set consisting of all problems domains to generate regular linear regression estimation model instead of picking two or three problem domains. In order to observe how the impact of problem domain affects the estimation accuracy, in case study 2 we validate several scenarios when our methodology is not used and the historical data set problem domain is not the same as the new project's problem domain. One of the main steps of our methodology is to use the estimation model generated from the historical data set that matches the problem domain of the new project. Nevertheless, we validate a scenario when our methodology is used and the new project problem domain is not the same as the problem domain of the generated estimation model. In future, we would like to continue experiments with our methodology to validate additional scenarios indicated in the Table 6-4.

Table 6-4: Outline of scenarios involving problem domains during estimation of effort

| **New Projects' Problem Domain** | **Problem Domain of the Historical Data Set used for Effort Estimation Model Generation** | **Related Experiment** | **Our Methodology** |
|---|---|---|---|
| PD2 | PD2 | Case Study 2 | Yes |
| PD1 | PD1, PD2, PD3, PD4 | Case Study 2 | No |
| PD2 | PD1, PD2, PD3, PD4 | Case Study 2 | No |
| PD1 | PD2 | Case Study 2 | No |
| PD1 | PD3 | Case Study 2 | No |
| PD2 | PD1 | Case Study 2 | No |
| PD2 | PD3 | Case Study 2 | No |
| PD2 | PD2 | Not Covered | No |
| PD1 | PD1, PD2, PD3, PD4 | Not Covered | Yes |
| PD2 | PD1, PD2, PD3, PD4 | Case Study 2 | Yes |
| PD1 | PD2 | Not Covered | Yes |
| PD1 | PD3 | Not Covered | Yes |
| PD2 | PD1 | Not Covered | Yes |
| PD2 | PD3 | Not Covered | Yes |
| PD1 | PD1, PD2, PD3 | Not Covered | No |
| PD2 | PD1, PD2, PD3 | Not Covered | No |
| PD1 | PD1, PD2, PD4 | Not Covered | No |
| PD2 | PD1, PD2, PD4 | Not Covered | No |

The error rates used for the comparison of the generated effort estimation models are relative absolute error and root relative squared error. Relative absolute error is calculated as following [Gep11a]:

$$\text{RelativeAbsoluteError} = \frac{\sum_{i=1}^{n} \left|PV_i - AV_i\right|}{\sum_{i=1}^{n} \left|AV_i - \overline{AV}\right|}$$

Where n is the number of projects in the historical data set, $PV_i$ is the predicted effort value of a project i, $AV_i$ is the actual effort value of a project i, and $\overline{AV}$ is the mean of all actual effort values calculated as follows:

$$\overline{AV} = \frac{1}{n} * \sum_{i=1}^{n} AV_i$$

The root relative squared error is calculated as follows [Gep11b]:

$$\text{RootRelativeSquaredError} = \sqrt{\frac{\sum_{i=1}^{n} \left(PV_i - AV_i\right)^2}{\sum_{i=1}^{n} \left(AV_i - \overline{AV}\right)^2}}$$

The error rates used for the comparison of the effort for the new project are Mean of the Magnitude of Relative Error (MMRE) and Median of the Magnitude of Relative Error (MdMRE).

The MRE is calculated as follows:

$$\text{MRE} = \left|\frac{\text{EstimatedEffort} - \text{ActualEffort}}{\text{ActualEffort}}\right| * 100\%$$

Where EstimatedEffort is the effort estimated by an effort estimation technique, such as human expert based or our effort estimation methodology, and ActualEffort is the real effort value of the project for which the effort is being estimated. Next, MMRE is calculated by using the following formula:

$$\mathrm{MMRE} = \frac{\sum_{i=1}^{n} \mathrm{MRE}_i}{n}$$

Where $\mathrm{MRE}_i$ is the MRE of a single project's effort estimation and n is the number of new projects being estimated.

MdMRE is calculated by finding the median MRE of an effort estimation value among the projects being estimated.

## 6.2. Discussion

Table 6-5 presents the results of the generation of the effort estimation model using the Method EEM1 of the methodology using the best performing regression algorithms for case study 1. The algorithms are available in WEKA tool. The MultilayerPerceptron algorithm is artificial neural network based algorithm. It has been found that LinearRegression algorithm has performed the best in terms of the correlation coefficient, relative absolute error, and root relative squared error in Trial 8. The result of trial 2 where we have performed regular linear regression was not good. The correlation coefficient was negative and error rates were high.

In trial 3, the results were improved after running feature subset selection on all features and performing the regression analysis on the reduced feature subset to derive the effort estimation model from the historical dataset. The Linear Regression has achieved the correlation coefficient of 0.9439 and error rates have dropped. It has performed better than the ANN based algorithm, MultilayerPerceptron. On the other hand, the Least Median Square algorithm was worse than MultilayerPerceptron algorithm in terms of the correlation coefficient and error rates.

In the trial 4, we can observe that results have improved after converting the nominal feature values into numerical ones. However, the Least Median Square performed better than other algorithms in this trial by achieving the correlation coefficient of 0.9697, relative absolute error of 38.8387% and root relative squared error of 38.784%. The Linear Regression algorithm in this trial has achieved a correlation coefficient of 0.9571, but its relative absolute error was slightly better than that of the Least Median Square.

In trial 5, the results have degraded and the correlation coefficient and error rates have dropped. For example, the MultilayerPerceptron algorithm has performed worse than MultilayerPerceptron algorithm in the trials 3 and 4. A similar situation arises with the Least Median Square algorithm.

Table 6-5: The best performing algorithms in Effort Estimation Model (Method EEM1) for case study 1

| **Trial #** | **Algorithm** | **Correlation Coefficient** | **Relative Absolute Error (%)** | **Root relative squared error (%)** |
|---|---|---|---|---|
| 8 | LinearRegression | 0.9775 | 31.2926 | 35.4589 |
| 4 | LeastMedSq | 0.9697 | 38.8387 | 38.784 |
| 4 | LinearRegression | 0.9571 | 37.9866 | 44.0302 |
| 3 | LinearRegression | 0.9439 | 40.2121 | 54.4069 |
| 8 | MultilayerPerceptron | 0.9285 | 36.7743 | 42.3345 |
| 8 | LeastMedSq | 0.8248 | 49.6174 | 54.6742 |
| 3 | MultilayerPerceptron | 0.8183 | 53.4167 | 55.9571 |
| 4 | MultilayerPerceptron | 0.7624 | 51.6075 | 61.602 |
| 7 | LeastMedSq | 0.7489 | 57.4825 | 69.5129 |
| 6 | LinearRegression | 0.7444 | 72.6686 | 101.1395 |
| 3 | LeastMedSq | 0.7117 | 61.8939 | 70.1565 |
| 6 | MultilayerPerceptron | 0.6878 | 67.419 | 69.7698 |
| 5 | MultilayerPerceptron | 0.6843 | 62.58 | 69.6723 |
| 7 | MultilayerPerceptron | 0.6484 | 68.2768 | 72.2187 |
| 5 | LeastMedSq | 0.5631 | 71.9888 | 83.4221 |
| 6 | LeastMedSq | 0.093 | 104.7056 | 121.5743 |
| 2 | LinearRegression | -0.6429 | 100% | 100% |

In trial 6, the results have improved after converting the nominal feature values into numerical ones. In addition, we performed similar activities as in the trial 5. For instance, the Linear Regression algorithm has achieved a correlation coefficient of 0.7444 and a relative absolute error of 72.6686. On the other side, its root relative squared error was still high: 101.1395%. The MultilayerPerceptron algorithm has achieved a better correlation coefficient of 0.6878 compared to the MultilayerPerceptron algorithm in trial 5, but its error rates did not improve much.

In trial 7, the results were better than in trials 5 and 6. However, the results of trials 3 and 4 for the algorithms LinearRegression and Least Median Square were better than those for the trial 7. In trial 8, we have applied our approach completely to all steps of the effort estimation. It can be observed that the Linear Regression based effort estimation model was the best in trial 8 in terms error rates (Figure 6-1). The trial 4 had the second best Linear Regression based effort estimation model.

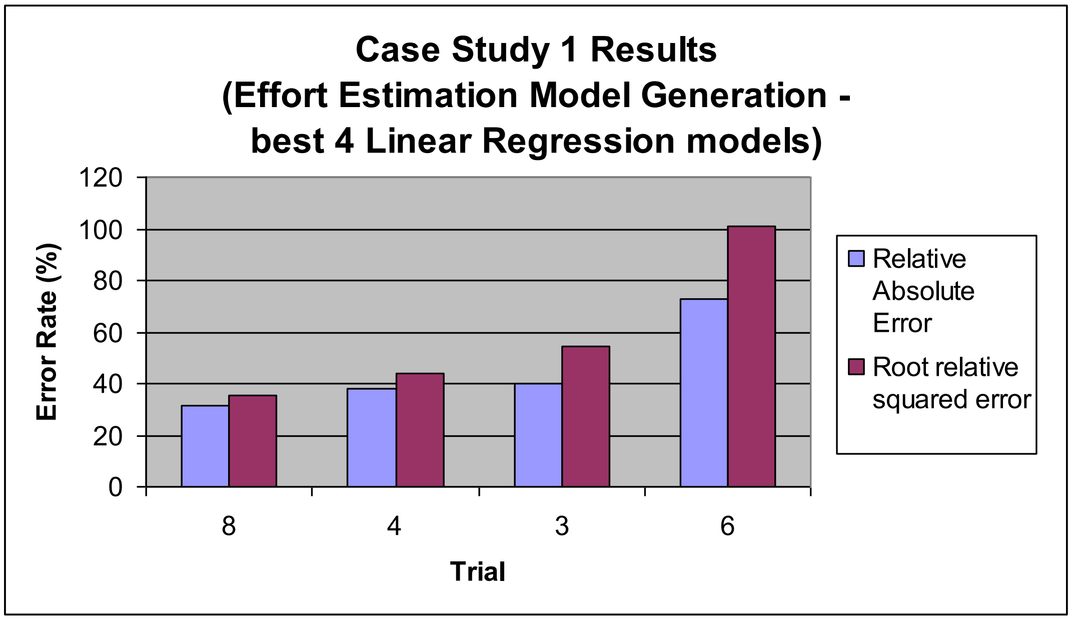


Figure 6-1: Best 4 Linear Regression models generated in Case Study 1

Trial 8 had also the best ANN based effort estimation model among all trials in terms of the error rate (Figure 6-2). However, the second best ANN based effort estimation model was in trial 3.

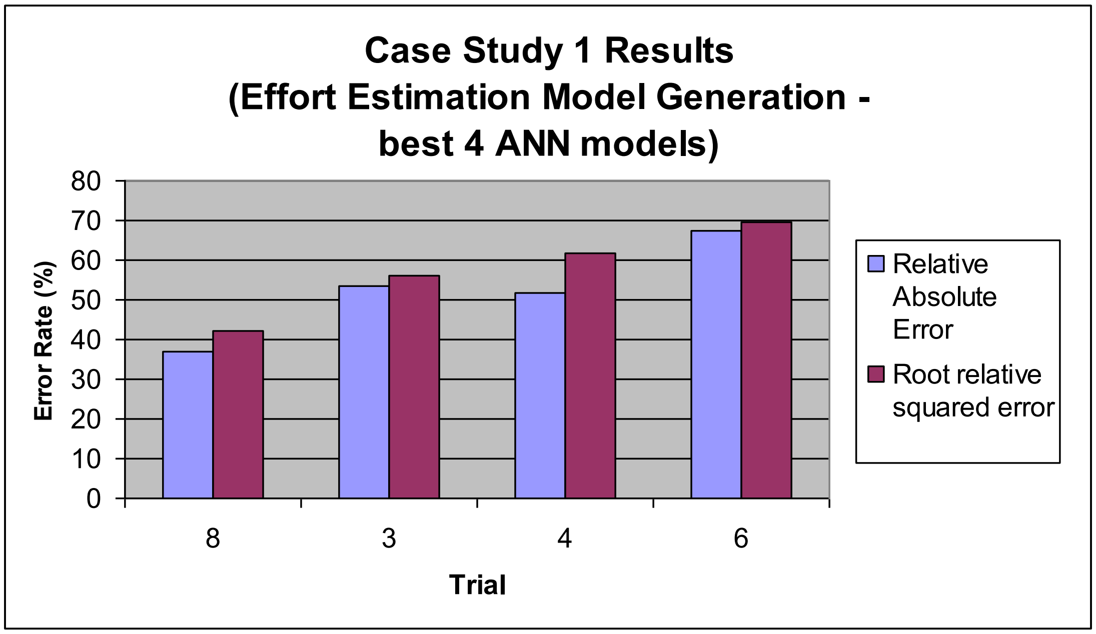


Figure 6-2: Best 4 ANN models generated in Case Study 1

We can observe the following for the effort estimation model generation phase based on the results:

- Running the regression analysis on all features in the set degraded the quality of the generated effort estimation model for trial 5 in comparison to trial 3.
- Running the regression analysis on the selected feature subset has improved the results for trials 3, 4, 7, and 8 in comparison to running the regression analysis on all features in the set for trials 5 and 6.
- Conversion of the nominal feature values to numerical values has improved results for trial 4 compared to trial 3, for trial 6 compared to trial 5, and for trial 8 compared to trial 7.
- Performing Feature Subset Selection on nominal features only, instead of performing Feature Subset Selection on all features (numerical and nominal), has improved the correlation coefficient and error rates for trial 8 when it was used in combination with other techniques of our approach.

Using the effort estimation model of trial 8, which was the best among trials [3-8], we performed effort estimation for the new projects. The subjective features of the new project were estimated using our approach. The results prove that our approach performs well during the generation of the effort estimation model. However, the number of projects used to generate the effort estimation model was not very high. The effort estimation model generated using trial 8 had the following form:

$$\begin{aligned}&\text{Effort} = 22.8054 * \text{NumberOfDevelopers} - 101.2604 * \text{ProjectComplexity} + \\ &98.1228 * \text{PerformanceImpact} - 33.6765\end{aligned}$$

We can see that the model includes the impact of performance (NFR) on effort, the project complexity, and number of developers. One can notice that Size was not part of the effort estimation model. This could be explained by the fact that Size and other features, such as impact of NFRs on effort, were found to be correlated and the feature reduction algorithm has excluded them as being redundant for our data set.

Effort estimation for new projects using our approach performed quite well (Table 6-6). The correlation coefficient for the LinearRegression model was 0.7481, while the MMRE was 21%. On the other hand, the human expert estimates were prone to have higher MRE. The similar situation was for the regular linear regression done in trial 2 (Figure 6-3).

Table 6-6: Results of both case studies

| **Trial #** | **Case Study #** | **New Project Problem Domain** | **Historical Data Set Problem Domain** | **MMRE (%)** | **MdMRE (%)** | **Correlation Coefficient** |
|---|---|---|---|---|---|---|
| 1 | 1 | PD1 | PD1 | 54 | 61 | N/A |
| 2 | 1 | PD1 | PD1 | 41 | 40 | 0 |
| 8 | 1 | PD1 | PD1 | 21 | 19 | 0.75 |
| 1 | 2 | PD2 | PD2 | 23 | 17 | 0.94 |
| 2 | 2 | PD1 | PD1+PD2+PD3+PD4 | 3776 | 3336 | 0.9121 |
| 2 | 2 | PD2 | PD1+PD2+PD3+PD4 | 35 | 31 | 0.9141 |
| 3 | 2 | PD1 | PD2 | 453 | 515 | 0.91 |
| 3 | 2 | PD1 | PD3 | 5168 | 4403 | 0 |
| 3 | 2 | PD2 | PD1 | 96 | 96 | 0 |
| 3 | 2 | PD2 | PD3 | 182 | 170 | 0 |
| 4 | 2 | PD2 | PD1+PD2+PD3+PD4 | 31 | 24 | 0.9177 |

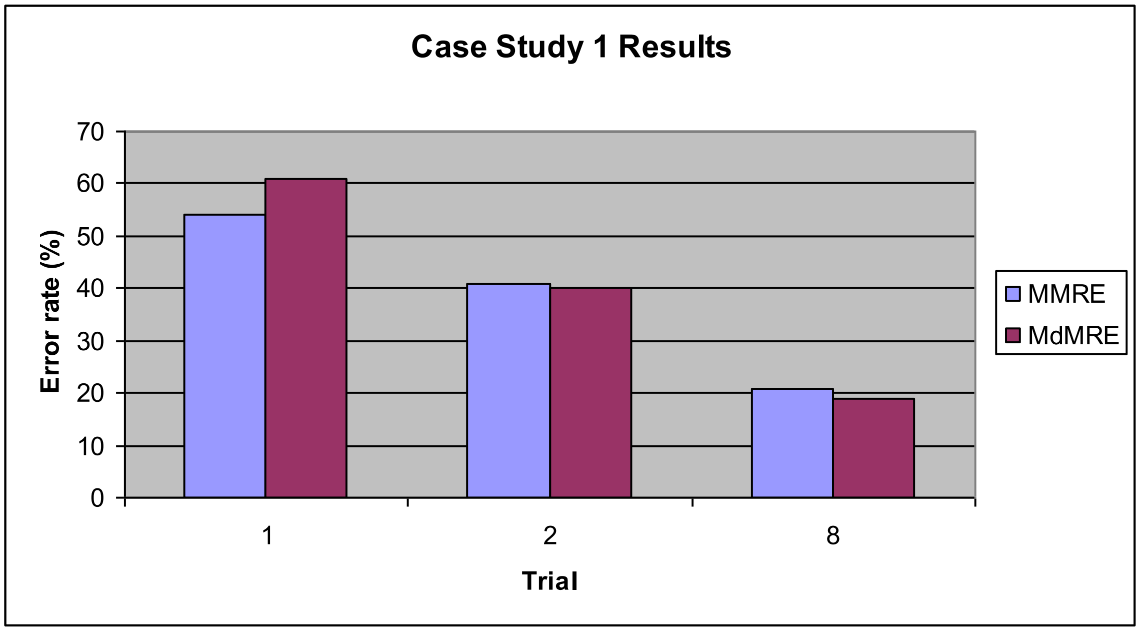


Figure 6-3: Case Study 1 Results (effort estimation for new project)

We can also observe in Case Study 2 that in trials 2 and 3 the estimation of the effort for new projects did not perform well when our approach was not used. In fact, the MMRE and MdMRE have deteriorated significantly in some cases (Figure 6-4). For instance, in trial 2, the effort estimation for new projects of problem domain PD1 using the historical dataset containing projects of four problem domains deteriorated results with MMRE reaching almost 3776%. The effort estimation for new projects of problem domain PD2 using the historical dataset containing projects of four problem domains deteriorated results less than for the similar experiment in trial 2 for the new projects of problem domain PD1. However, the effort estimation MMRE of new projects of problem domain PD2 was 35% which is still higher than that for trial 1 of case study 2.

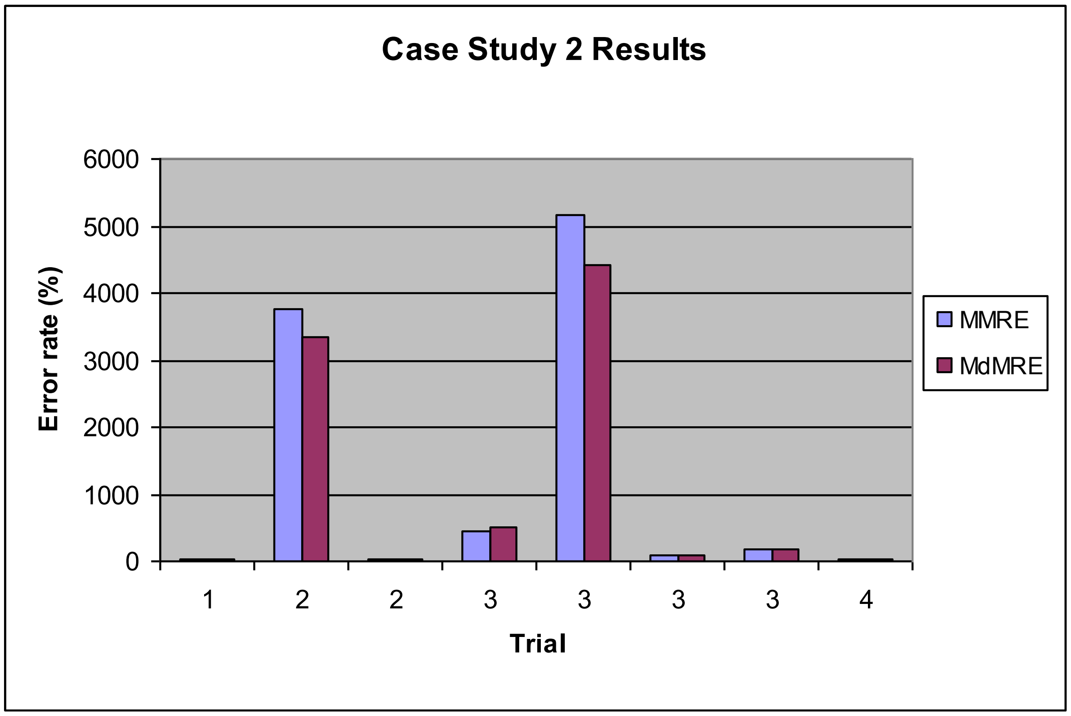


Figure 6-4: Case Study 2 Results (effort estimation for new project)

In trial 3, effort estimation of the new project of problem domain PD1 using the historical dataset containing projects of another problem domain, such as PD2 or PD3 deteriorated results, with MMRE increasing significantly to 453% or 5168% in comparison to the results of effort estimation performed in trial 8 of case study 1. Also, the effort estimation of the new project of problem domain PD2 using a historical dataset containing projects of another problem domain, such as PD1 or PD3, deteriorated results, with MMRE increasing significantly to 96% or 182% in comparison to the results of effort estimation performed in trial 1 of case study 2. When our approach was used in trial 1 of case study 2, the estimation of the effort for the new projects of problem domain PD2 was done well with MMRE of 23% and MdMRE of 17%. On the other hand, when our approach was used in trial 4 of case study 2 without taking into account the impact of problem domain, the MMRE and MdMRE improved compared to trial 2 results. However, trial 4 result was still slightly worse than trial 1 result where our methodology was used and the problem domain was taken into account. We can observe based on the results of case study 2 that our approach performed better than the regular approach, which used all projects from all problem domains. In comparison, the study done by [LPH02] on a small set of 19 project has shown that COCOMO II achieved an MMRE of 31% (section 3.1.1.2.4) and SEER-SEM achieved an MMRE of 35%. Also, we have noticed that not using our methodology and using projects with problem domains different from the problem domain of new projects have significantly degraded the accuracy of effort estimates in both case studies.

The generated effort estimation model used to estimate the effort of new projects of problem domain PD2 in trial 1 has the following form:

$$\begin{aligned}&\text{Effort} = 14.9767 * \text{Size} + 3099.4235 * \text{Primary ProgramLanguage} = \text{Net}, \text{C}++, \text{C\#},\\ &\text{Java}, \text{ASPNet}, \text{COBOL}, \text{FOCUS}, \text{VisualBasic}, \text{PLI} +\\ &9882.7283 * \text{Primary ProgramLanguage} = \text{PLI} - 1001.2267\end{aligned}$$

We can see that the model includes Size and the primary programming language feature. Other features such as project complexity, problem duration, and development platform are not included, because the feature reduction algorithm has identified them to be correlated among each other and excluded them as being redundant for this data set.

The next chapter presents the conclusions and the future work.

# Chapter 7: Conclusions and Future Work

In this thesis, we have presented a novel effort estimation methodology that can be applied in the early stages of software development projects to estimate the effort of new projects. The effort estimated by our methodology includes overall effort needed to design, implement, and test the software product and to manage corresponding development project in which it is being developed. The results show that our approach performs well by increasing the correlation coefficient and decreasing the error rate of the generated effort estimation models and achieving more accurate effort estimates for new projects. We have developed an effort estimation model based on the historical data of previous projects to estimate software development effort during the requirements specification phase. Also, we have objectively assessed the impacts of different NFRs and different problem domains on the estimation of software development effort. Moreover, we made the effort estimation model robust by dynamically reducing the feature space using both statistical and semantic techniques. Furthermore, the questionnaire has been designed to collect important information about new and past historical projects. Finally, we have implemented the effort estimation tool based on the methodology described in the thesis to prove our concept.

In Table 7-1, we present a summary of effort estimation methodologies using parametric models to compare them with our methodology based on six different factors, such as functional size method and consideration of Impact of NFR on effort. We can observe that only half of the surveyed methods support functional size measurement and early effort estimation. There are only few methods that support COSMIC method to measure functional size. Also, very few methods take into account impacts of NFRs, problem domains and support feature reduction.

Table 7-1: Summary of effort estimation methodologies using parametric models

| **Criterion** / **Methodology** | **Functional Size** | **Early effort estimation** | **Problem domain specific** | **Accuracy** | **Impact of NFR on effort** | **Feature reduction** | **Proprietary** |
|---|---|---|---|---|---|---|---|
| Our methodology | Yes (CFP) | Yes | Yes | Yes | Yes | Yes | No |
| COCOMO II [BAB+00] | Yes (FP, SLOC, AP) | Partially | No | No | No | No | No |
| Method [Kas09b] | Yes (CFP) | Yes | No | Yes | Yes | No | No |
| SEER-SEM [Gal08] | Yes (FP) | Yes | Unknown | Unknown | No | Unknown | Yes |
| Kultur Method [KKB09] | Yes (CFP, FP) | Yes | No | Yes | No | No | No |
| Mendes Method [MMFG07] | No | No | No | Yes | No | No | No |
| Martin Method [MPYT05] | No (SLOC) | No | No | Yes | No | No | No |
| Select Estimator [BC00] | Yes (by counting elements) | Yes | No | Unknown | No | No | Yes |
| Putnam's SLIM [BC00] | No (SLOC) | No | No | No | No | No | No |
| Albrecht and Gaffney Method [AG83] | No (FP to SLOC conversion) | Yes | No | No | No | No | No |
| Checkpoint [BC00] | Yes (FP) | Yes | No | Unknown | No | Unknown | Yes |
| ESTIMACS [BC00] | Yes (FP) | No | No | Unknown | No | No | Yes |
| PRICE-S [Pri11] | Yes (FP, SLOC) | Yes | Yes | Unknown | No | Unknown | Yes |
| Parthasarathy Method [Par07] | Yes (FP, etc.) | Yes | No | Unknown | No | No | No |

The list of open problems is restated in Table 7-2 with the corresponding sections that address them in this thesis.

Table 7-2: Linking Open Problems to their corresponding solutions

| **Open Problem** | **Link to the Answer** |
|---|---|
| [OP1] The estimation of effort has the tendency to include human estimator's subjectivity leading often biased results. | Sections 4.1.2 and 4.2.3 |
| [OP2] The impact of NFR on effort is often not considered in the effort estimation. | Sections 4.1.2 and 4.2.3 |
| [OP3] The quantification of the impact of NFR on effort is not performed objectively. | Section 4.2.3 |
| [OP4] The effort estimation techniques using historical database do not clearly distinguish clustering of projects by problem domain to achieve a better precision. | Sections 4.1.1 and 4.2.1 |
| [OP5] The effort estimation methodologies do not always allow the user to perform feature reduction techniques. | Section 4.1.4 |
| [OP6] Data collection from historical projects is not clearly defined. | Section 4.3 |

Our proposed methodology is practical and easy to learn as described in chapter 4 and section 4.1.5. It can be applied in the context of industry based organizations that require fast and early effort estimation for new projects. Organizations that already collect NFRs and results of past projects will be able to easily start using this methodology. On the other hand, organizations without current practice of NFR and past projects data collection will be able to start building their historical database and move towards more scientific, systematic and reproducible effort estimation practices. Our methodology will help companies to improve the accuracy of project planning and effort estimation, which in turn can aid them in successful execution of their projects.

The previous work presented in [HKO08] showed that the FRs and NFRs can automatically and effectively be extracted from software requirements document using natural language processing techniques, and the recent work [HKO10, HOK09] has shown that the functional size of the software can be computed objectively from any form of unrestricted textual representation of FRs. In the future, we plan to fully automate our methodology and to integrate it with the work presented in [HKO10] in order to obtain the Size measurements automatically in the early stages of software development projects.

Also, we would like to research the interrelation between the NFRs in order to identify how the interaction of a group of NFRs contributes towards the overall effort of software development project. For example, one area of study could be to identify how conflicting NFRs such as Performance and Security interact with each other and impact overall effort to develop software product.

In addition, we are looking forward to study the impact of the choice of architectural decisions on the effort estimation value. In particular, we are working on an approach that relies on a quantitative assessment of the impact of architectural tactics on quality requirements on the one hand, and the impact of incorporating these tactics in architectural patterns on the other hand. We will then incorporate this approach into the study discussed within this thesis to generate a range of effort estimation values against a set of architectural patterns. Also, we plan to research how to take into account the impact of the development methodologies such as Test Driven Development and Agile Software Development (ASD) on the effort estimation. Moreover, we would like to study what optimal level of detail SRS needs in order for a project to have more precise effort and size estimations. This will allow projects and processes based on ASD to improve further the way requirements are specified and detailed and to optimize software estimation results. Finally, we plan to collaborate with industry leaders such as SAP in order to use more actively our methodology in the future as part of industry based projects.

# Bibliography

[ADOSS03] Abran, A., Desharnais, J. M., Oligny, S., St-Pierre, D., & Symons, C. (2003). COSMIC FFP – Measurement Manual (COSMIC implementation guide to ISO/IEC 19761:2003), École de technologie supérieure – Université du Québec, Montréal, Canada.

[AG83] Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering, 9*, 639-648.

[AHKO11] Abdukalykov, R., Hussain, I., Kassab, M., & Ormandjieva, O. (2011). Quantifying the Impact of Different Non-Functional Requirements and Problem Domains on Software Effort Estimation. Accepted at the 9th International Conference on Software Engineering Research, Management and Applications (SERA'11) and will be published in the proceedings of the conference, August 10-12, 2011, Baltimore, MD, USA.

[ANC08] Azzeh, M., Neagu, D., & Cowling, P. (2008). Improving Analogy Software Effort Estimation using Fuzzy Feature Subset Selection Algorithm. Proceedings of the 4th International Workshop on Predictor Models in Software Engineering (pp. 71-78). Leipzig, Germany: ACM.

[AS00] Angelis, L., & Stamelos, I. (2000). A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Engineering, 5, 35-68.

[ASM01] Angelis, L., Stamelos, I., & Morisio, M. (2001). Building a software cost estimation model based on categorical data. Software Metrics Symposium. METRICS 2001. Proceedings. Seventh International, vol., no., pp.4-15.

[BAB+00] Boehm, B., Abts, C., Brown, A. W., Chulami, S., Clark, B. K., & Horowitz, E. (2000). *Software Cost Estimation with COCOMO II* (with CD-ROM). Englewood Cliffs, NJ: Prentice-Hall.

[BC00] Boehm, B., & Chulani, S. (2000). Software development cost estimation approaches – a survey. Technical Report. University of Southern California and IBM Research, Los Angeles, USA.

[BHS03] Baader, F., Horrocks, I., & Sattler, U. (2003). Description logics as ontology languages for the semantic web, in Lecture Notes in Artificial Intelligence, Springer. Retrieved from http://www.cs.man.ac.uk/~horrocks/Publications/download/2003/BaHS03.pdf/

[Boe00] Boehm, B. (2000) Safe and Simple Software Cost Analysis. IEEE Software, 17 (5), 14-17.

[Boe81] Boehm, B. (1981). *Software engineering economics.* Englewood Cliffs, NJ: Prentice-Hall.

[BL01] Burgess, C.J., & Lefley, M. (2001). Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology, 43*, 863-873.

[BOM08] Braga, P. L., Oliveira, A. L., & Meira, S.R. (2008). A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation. *Proceedings of the 2008 ACM symposium on Applied computing*. Fortaleza, Ceara, Brazil.

[CBS99] Chulani, S., Boehm, B., & Steece, B. (1999). "Calibrating Software Cost Models Using Bayesian Analysis," *Proceedings 1999 ISPA/SCEA Conference*.

[CCT01] Chang, C. K., Christensen, M. J., & Tao, Z. (2001). Genetic algorithms for project management. Analysis of Software Engineering, 11, 107-139.

[COS09] COSMICON. (2009). The COSMIC Functional Size Measurement Method. Retrieved May 14, 2011 from http://www.cosmicon.com/methodV3.asp

[Fir03] Firesmith, D. G. (2003). Common concepts underlying safety, security, and survivability engineering, Technical Note CMU/SEI-2003-TN-033, Carnegie Mellon Software Engineering Institute.

[FMG05] Fischman, L., McRitchie, K., & Galorath, D. D. (2005). Inside SEER-SEM. *CrossTalk, 18.* Retrieved May 13, 2010 from http://www.crosstalkonline.org/storage/issue-archives/2005/200504/200504-Fischman.pdf

[FP97] Fenton, N. E., & Pfleeger, S.L. (1997). Software Metrics: A rigorous and Practical Approach, International Thomson Computer Press.

[FTAS08] Feldt, R., Torkar, R., Angelis, L., & Samuelsson, M. (2008). Towards individualized software engineering: empirical studies should collect psychometrics, In Proceedings of the 2008 international Workshop on Cooperative and Human Aspects of Software Engineering (Leipzig, Germany, May 13 - 13, 2008), CHASE '08, ACM, New York, NY, (pp. 49-52).

[Gal08] Galorath. (2008). SEER for software development: estimating software projects. Retrieved May 13, 2010 from http://galorath.com/index.php/products/software/C5

[Gep11a] Gepsoft Inc. (2011). Relative Absolute Error. Retrieved June 10, 2011 from http://www.gepsoft.com/gxpt4kb/Chapter10/Section1/SS08.htm

[Gep11b] Gepsoft Inc. (2011). Root Relative Squared Error. Retrieved June 10, 2011 from http://www.gepsoft.com/gxpt4kb/Chapter10/Section1/SS07.htm

[GHL09] Gencel, C., Heldal, R., & Lind, K. (2009). On the Relationship between Different Size Measures in the Software Life Cycle. *Software Engineering Conference. APSEC '09. Asia-Pacific* , vol., no., pp.19-26, 1-3

[GJ07] Grimstad, S., & Jorgensen, M., (2007). The impact of irrelevant information on estimates of software development effort.

[Hal00] Hall, M. A. (2000). *Correlation based Feature Selection for Machine Learning*. Doctoral dissertation. University of Waikato, Waikato, New-Zeland.

[HFH09] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I.H. (2009). The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1

[HHRC04] Huang, X., Ho, D., Ren, J., & Capretz, L. (2004). A neuro-fuzzy tool for software estimation. Proceedings. 20th IEEE International Conference on Software Maintenance, (p. 520).

[HKO08] Hussain, I., Kosseim, L., & Ormandjieva, O. (2008). Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents. In *LNCS: Natural Language and Information Systems* (Vol. 5039/2008, pp. 287-298). Germany: Springer-Verlag.

[HKO10] Hussain, I., Kosseim, L., & Ormandjieva, O. (2010). Towards Approximating COSMIC Functional Size from User Requirements in Agile Development Processes Using Text Mining. In *LNCS: Natural Language Processing and Information Systems* (Vol. 6177/2010, pp. 80-91). Germany: Springer-Verlag.

[HOK09] Hussain, I., Ormandjieva, O., & Kosseim, L. (2009). Mining and Clustering Textual Requirements to Measure Functional Size of Software with COSMIC. *Proceedings of the International Conference on Software Engineering Research and Practice (SERP 2009).*

[Hus11] Hussain, I. (2011). L.A.S.R. (Live Annotation of Software Requirements). Retrieved July 14, 2011 from http://users.encs.concordia.ca/~nlp-se/lasr/

[IAK02] Idri, A., Abran, A., & Khoshgoftaar, T. M. (2002). Estimating software project effort by analogy based on linguistic values. Proceedings of International Software Metrics Symposium (pp. 21-30). Ottawa. Canada: IEEE Press.

[IEEE83098] IEEE Std. 830-1998. (1998). IEEE recommended practice for software requirements specifications, IEEE Transactions on Software Engineering.

[IKA02] Idri, A., Khoshgoftaar, T. M., & Abran, A. (2002). Can neural networks be easily interpreted in software cost estimation? *Proceedings of IEEE International Conference on Fuzzy Systems,* (pp. 1162 – 1167).

[ISB11] International Software Benchmarking Standards Group (ISBSG). (2010). About ISBSG. Retrieved June 2, 2011 from http://www.isbsg.org/isbsgnew.nsf/webpages/~GBL~About%20Us

[ISO1414307] International Organization for Standardization. (2007). ISO/IEC IS 14143-1:2007: Information technology - Software measurement - Functional size measurement - Part 1: Definition of concepts.

[ISO912601] International Standard ISO/IEC 9126-1. (2001). Software engineering – Product quality – Part 1: Quality model. ISO/IEC 9126-1:2001, 200.

[Jor04] Jorgensen, M. (2004). Regression Models of Software Development Effort Estimation Accuracy and Bias. *Empirical Software Engineering, 9,* 297-314.

[Kas09a] Kassab, M. (2009). *Non-Functional Requirements: Modeling and Assessment.* VDM Verlag.

[Kas09b] Kassab, M. (2009). *Formal and Quantitative Approach to Non-Functional Requirements Modeling and Assessment in Software Engineering.* Doctoral dissertation. Concordia University, Montreal, Canada.

[KR92] Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. *In Machine Learning: Proceedings of the Ninth International Conference*.

[KKB09] Kultur, Y., Kocaguneli, E., & Bener, A. B. (2009). Domain specific phase by phase effort estimation in software projects. *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, vol., no., pp.498-503.

[LPH02] Lum, K., Powell, J., & Hihn, J. (2002). Validation of Spacecraft Software Cost Estimation Models for Flight and Ground Systems. Price Systems Inc. Retrieved August 2, 2011 from http://www.pricesystems.com/white_papers/SW%20Validation%20Paper%20ISPA%2002.pdf

[LS05] Liebchen, G. A., & Shepperd, M. (2005). Software Productivity Analysis of a Large Data Set and Issues of Confidentiality and Data Quality, *In*

*Proceedings of the 11th IEEE international Software Metrics Symposium* (September 19 - 22, 2005), METRICS, IEEE Computer Society, Washington, DC, 46.

[LWHS01] Lokan, C., Wright, T., Hill, P. R., & Stringer, M. (2001). Organizational Benchmarking Using the ISBSG Data Repository, IEEE Software, 18(5), (pp. 26-32).

[LW03] Leffingwell D., & Widrig, D. (2003). Managing Software Requirements: A Unified Approach, The Addison-Wesley Object Technology Series.

[Mat11] MathWorks Inc. (2011). The Language of Technical Computing. Retrieved June 10, 2011 from http://www.mathworks.com/products/matlab/

[Mcc06] McConnell, S., (2006). Software Estimation – Demystifying the Black Art. Microsoft Press.

[Men10] Menzies, T. (2010). Selecting Quality Data. Software Engineering Institute (SEI).
Retrieved May 31, 2010 from
http://www.sei.cmu.edu/measurement/research/upload/Menzies.pdf

[Mic11] Microsoft Corp. (2011). Windows Intune: Software Categories.
Retrieved May 11, 2010 from
http://onlinehelp.microsoft.com/en-us/windowsintune/ff399004.aspx

[MF00] Maxwell, K. D., & Forselius, P. (2000). Benchmarking Software-Development Productivity, IEEE Software, 17(1), (pp. 80-88).

[MM08] Mendes, E., & Mosley, N. (2008). Bayesian Network Models for Web Effort Prediction: A Comparative Study. *IEEE Transactions on Software Engineering, 34,* 723-737.

[MMFG07] Mendes, E., Martino, S. D., Ferrucci, F., & Gravino, C. (2007). Effort Estimation: How Valuable is it for a Web Company to Use a Cross-company Data Set, Compared to Using Its Own Single-company Data Set? *Proceedings of the 16th international conference on World Wide Web* (pp. 963-972). ACM.

[MJ03] Molkken, K., & Jorgensen, M., (2003). A Review of Surveys on Software Effort Estimation. (p. 223). IEEE Computer Society.

[MP08] Magazinovic, A., & Pernstål, J. (2008). Any other cost estimation inhibitors? *In Proceedings of the Second ACM-IEEE international Symposium on Empirical Software Engineering and Measurement,* Kaiserslautern, Germany, ESEM '08. ACM, New York, NY, (pp. 233-242).

[MPYT05] Martin, C. L., Pasquier, J. L., Yanez, C. M., & Tornes, A. G. (2005). Software development effort estimation using fuzzy logic: a case study. *Computer Science, 2005. ENC 2005. Sixth Mexican International Conference on*, vol., no., pp. 113- 120

[Par07] Parthasarathy, M. A. (2007). Practical Software Estimation. Infosys. Addison-Wesley.

[PB05] Park, H., & Baek, S. (2005). An empirical validation of a neural network model for software effort estimation. Expert Systems with Applications, 35, 929-937.

[Pri11] PRICE Systems. (2011). How to Estimate Software Projects. Retrieved May 31, 2010 from http://www.pricesystems.com/products/popup_software.htm

[PSR05] Pendharkar, P.C., Subramanian, G. H., & Rodger, J. A. (2005). A Probabilistic Model for Predicting Software Development Effort. *IEEE Transactions on Software Engineering, 31*, 615 – 624.

[PWL05] Pfleeger, S. L., Wu, F., & Lewis, R. (2005). Software Cost Estimation and Sizing Methods, Issues and Guidelines. RAND Corporation.

[Put81] Putnam, L. H. (1981). SLIM: a quantitative tool for software cost and schedule estimation. *Proceedings of NBS/IEEE/ACM Software Tool Fair* (pp. 49-57). McLean, VA: Quantitative Software Management Inc.

[Qui86] Quinlan, R. R. (1986). Induction of decision trees. Machine Learning, 1:81–106.

[SADS03] Stamelos, I., Angelis, L., Dimou, P., & Sakellaris, E. (2003). On the use of Bayesian belief networks for the prediction of software productivity. Information and Software Technology, 45, 51-60.

[Sas11] SAS Inc. (2011). Analytics. Retrieved June 10, 2011 from http://www.sas.com/technologies/analytics/index.html

[Sch10] Schwarz, C. (2010). *Scales of Measurement.* Retrieved June 2, 2011 from Simon Fraser University: http://www.stat.sfu.ca/~cschwarz/Stat-301/Handouts/node5.html

[Shu00] Shukla, K. K., (2000). Neuro-genetic prediction of software development effort. *Information and Software Technology, 42*, 701-713.

[SMLE02] Shan, Y., McKay, R. I., Lokan, C. J., & Essam, D. L. (2002). Software Project Effort Estimation Using Genetic Programming. *Proceedings of International Conference on Communications* (pp. 1108-1112). Canberra, ACT, Australia: IEEE Computer Society.

[SSK96] Shepperd, M., Schofield, C., & Kitchenham, B. (1996). Effort estimation using analogy. *Software Engineering, Proceedings of the 18th International Conference on*, vol., no., pp.170-178.

[Sta01] StatSoft Inc. (2011). Elementary Concepts in Statistic. Retrieved May 14, 2011 from http://www.statsoft.com/textbook/elementary-concepts-in-statistics/

[Ste46] Stevens, S. S. (1946). "On the Theory of scales and Measurement". Science 103, 1946, pp. 677-680.

[SX07] Sadana, V., & Xiaoqing, F., L. (2007). Analysis of Conflicts among Non-Functional Requirements Using Integrated Analysis of Functional and Non-Functional Requirements. *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, vol.1, no., pp.215-218, 24-27.

[TEMPLATE09] Scenario Plus, Qualities and Constraints, or Non Functional Requirements Template. (2009). Retrieved from

http://www.scenarioplus.org.uk/download_nfrs.html

[YHLWB08] Yang, Y., He, M., Li, M., Wang, Q., & Boehm, B. (2008). Phase distribution of software development effort, *In Proceedings of the Second ACM-IEEE international Symposium on Empirical Software Engineering and Measurement* (Kaiserslautern, Germany, October 09 - 10, 2008), ESEM '08, ACM, New York, NY, (pp. 61-69).

# Appendix A

## Questionnaire to collect project information

**Background**

The present questionnaire is used as part of the effort estimation to identify how various factors and requirements affect the overall software effort for specific development projects that we are studying. The results of the questionnaire are strictly confidential and will be used within this effort estimation task. We are interested only in the aggregated results of the questionnaire and your personal responses will not be kept.

The relevant terms and concepts mentioned in the questionnaire are explained in details in Appendix section. Rating based answers should be marked by a symbol of "X".

Your participation and feedback in this survey are highly appreciated and will be helpful in correctly identifying the important factors affecting software effort.

**Intended Audience**

Software project manager, scrum master or member of project who has good knowledge and overview of the project

## Questions

**Date of Questionnaire:** [Date when the questionnaire is filled in]

**General Questions**

1. How many years of experience the development team have with the development environment and languages (e.g. Java, Eclipse SDK, or any other relevant technology for the problem domain or organization) at the time of execution of the project?

- Number of years of experience using the development environment [NAME]:
- Number of years of experience using the development language [NAME]:

**2.** How many development locations (geographical) were involved in the development of the project? (e.g. 1, 2, 3, etc.)

Number of development locations:

| Rate the impact on the project effort of the following factors: | Very Low | Low | Nominal | High | Very High |
|---|---|---|---|---|---|
| **3.** complexity of the product (related to Functional Requirements) | | | | | |
| **4.** the delivery deadline (schedule) (e.g., delivery of the project in accelerated format [more effort spent in the beginning of the project] or in stretched-out format [effort is stretched over the long period of time] ) | | | | | |
| **5.** the risk management done as part of your project | | | | | |
| **6.** the team cohesion for the project you were part of (e. g. difficulty in synchronizing project stakeholders (users, customers, developers, others) due to differences of their objectives). | | | | | |

**7.** If you have any additional comments regarding your answers to general questions mentioned above, please provide it here:

**Non-Functional Requirements (see definitions in Appendix)**

| Rate the impact on the project effort of the following non-functional requirements: | ++ (increases effort significantly) | + (increases effort slightly) | 0 (no impact on effort) | - (decreases effort slightly) | -- (decreases effort significantly) | N/A |
|---|---|---|---|---|---|---|
| **8.** Reliability | | | | | | |
| **9.** Efficiency | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **10.** Quality-In-Use | | | | | | |
| **11.** Portability | | | | | | |
| **12.** Configurability | | | | | | |
| **13.** Maintainability | | | | | | |
| **14.** Dependability | | | | | | |
| **15.** Security | | | | | | |
| **16.** Accessibility | | | | | | |
| **17.** Constraint | | | | | | |
| **18.** Accuracy | | | | | | |
| **19.** Usability | | | | | | |

**20.** If you have any additional comments regarding your answers to questions related to non-functional requirements mentioned above, please provide it here:

# Appendix B

## Definition of Terms used in the Questionnaire

The following table contains definitions of terms used in the questionnaire including NFR definitions developed by Mohamad Kassab in [Kas09b].

Table A-1: Definition of important terms including NFRs used in the questionnaire. [Kas09b]

| **Term** | **Description** | **Example** |
|---|---|---|
| Accessibility | "The degree to which a product is accessible by as many people as possible." [Kas09b] | System is capable of voice input for those who can't use regular IO devices. |
| Accuracy | "The capability of the software product to provide the right or agreed results or effects with the needed degree of precision." [Kas09b] | The model should accurately represent the semantics of the domain as perceived by the stakeholder(s). |
| Configurability | "In Communications or computer systems, a configuration is an arrangement of functional units according to their nature and number." [Kas09b] | Application provides feature customization. |
| Dependability | "The ability to deliver service that can justifiably be trusted by users." [Kas09b] | Application has different modules, in case one module is not functional, this does not impact on other modules. |
| Constraint | "Constraints are defined in [LW03] as restrictions on the design of the system, or the process by which a system is developed, that do not affect the external behaviour of the system but that must be fulfilled to meet technical, business, or contractual obligations." [Kas09b] | Economic Constraint, Operating Constraint, Political / Cultural Constraint, Business Rule, etc. |
| Efficiency | "The amount of computing resources and code required by a program to perform its function." [Kas09b] | Device Efficiency, Resource Behaviour, Performance, Time Behaviour, etc. |
| Maintainability | "The ability to change the system to | Changeability, Extensibility, |

| | | |
|---|---|---|
| | deal with new technology or to fix defects.” [Kas09b] | Correct-ability, etc. |
| N/A | Not Applicable | |
| NFR | Non-functional requirement | |
| Portability | “The ability of the system to run under different computing environments.” [Kas09b] | System supports more than one OS and entire range of 32bit hardware. |
| Quality-In-Use | “The capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use. / (Quality in use is the user's view of the quality of an environment containing software, and is measured from the results of using the software in the environment, rather than properties of the software itself.)” [Kas09b] | The increase in customer approval ratings from surveys, the increase in revenue from returning customers needs to be tracked as a customer feedback and considered as customer satisfaction. |
| Reliability | “The ability of a system or component to perform its required functions under stated conditions for a specified period of time.” [Kas09b] | In case of a failure of either critical or less critical operations, the system has to recover as fast as possible, not more than 1-2 days. |
| Security | “A measure of the system's ability to resist unauthorized attempts at usage and denial of service while still providing its services to legitimate users.” [Kas09b] | The system should provide its functionalities with high confidentiality when it is required. |
| Usability | “The ease with which a user can learn to operate, to prepare inputs for, and to interpret outputs of a system or component.” [Kas09b] | Software should be able to make smart gauss, and offer related tool tips. |