ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
PH.D.

BY
Francisco VALDÉS

DESIGN OF A FUZZY LOGIC SOFTWARE ESTIMATION PROCESS

MONTREAL, DECEMBER 14TH, 2011

© Copyright 2011 reserved by Francisco Valdés

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED
BY THE FOLLOWING BOARD OF EXAMINERS

M. Alain Abran, Thesis Supervisor
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

Mme Sylvie Nadeau, President of the Board of Examiners
Département de génie mécanique à l'École de technologie supérieure

Mme Sylvie Ratté, Examiner
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Hakim Lounis, External Examiner
Département d'informatique à l'Université du Québec à Montréal

THIS THESIS WAS PRESENTED AND DEFENDED

BEFORE A BOARD OF EXAMINERS AND PUBLIC

ON DECEMBER 1ST, 2011

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AKNOWLEDGEMENTS

I have been indebted in the preparation of this thesis to my supervisor, M. Alain Abran of École de technologie supérieure (ÉTS) de l'université du Québec, whose patience, kindness, procedural guidance and encouragement, as well as his exceptional academic experience, have been invaluable to me.

I am extremely grateful to the Conacyt, who gave me a unique opportunity and support to improve my professional skills and to live an invaluable experience.

To Renata, Ivanna and Miranda, my daughters, my joy, my reason to keep moving.

To Nancy Pino who let me know that when you really want something, you gonna get it no matter what.

My parents, Francisco Valdés and Martha Souto, always have been a constant source of full support – emotional, moral and of course financial – during my studies.

My brothers Hugo, Alejandro y Martha, an excellent emotional support too.

I am grateful to the excellent staff of the Gélog (Software Engineering Research Laboratory at ETS) and to Estefanía Fuentes, the Responsable du développement Latino - américain/ étudiants Conacyt Office at ETS.

DESIGN OF A FUZZY LOGIC SOFTWARE ESTIMATION PROCESS

Francisco VALDÉS

RÉSUMÉ

Cette recherche décrit la conception d'un processus avec logique floue pour l'estimation des projets de logiciels.

Il y a des études qui montrent que la plupart des projets de logiciels excèdent leur budget ou dépassent leur calendrier prévu, et ce même si depuis des années les organisations font des efforts pour augmenter le taux de réussite des projets de logiciels en rendant le processus plus facile à gérer et, par conséquent, plus prévisible.

L'estimation du projet est un enjeu important, car c'est la base pour quantifier, allouer et gérer les ressources nécessaires à un projet. Lorsque les estimations de projets logiciels ne sont pas effectuées correctement, les organisations font face un risque élevé dans leurs projets et cela peut mener à des pertes pour l'organisation au lieu des profits prévus et justifiant le démarrage des projets.

Les estimations les plus importants doivent être effectuées au début du cycle de développement (i.e. à la phase de conceptualisation des projets): à ce moment là, l'information est disponible seulement à un niveau très élevé d'abstraction, et souvent elle est fondée sur un certain nombre d'hypothèses non vérifiables.

L'approche généralement utilisée pour estimer les projets dans l'industrie du logiciel est celle basée sur l'expérience des employés dans l'organisation, aussi nommée l'appoche par 'jugement d'experts'. Bien sûr, il y a un certain nombre de problématiques reliées à l'utilisation de ces jugements d'experts en estimation: par exemple, les hypothèses sont implicites et l'expérience est fortement liée aux experts et non pas à l'organisation.

Le but de recherche de cette thèse était de concevoir un processus d'estimation de projets de logiciels capable de tenir compte du manque d'informations détaillées et quantitatives dans les premières phases du cycle de vie du développement logiciel.

La stratégie choisie pour cette recherche tire partie des avantages de l'approche fondée sur l'expérience qui peut être utilisée dans les phases précoces de l'estimation de projets de logiciels, tout en tenant compte de certains des problèmes majeurs générés par cette méthode d'estimation par jugements d'experts. La logique floue a été proposée comme approche de recherche parce que c'est une façon formelle pour gérer l'incertitude et les variables linguistiques disponibles dans les premières phases d'un projet de développement d'un logiciel: un système à base de logique floue permet d'acquérir l'expérience de l'organisation par l'intermédiaire des experts et de leurs définitions de règles d'inférence.

Les objectifs de recherche spécifiques à atteindre par ce processus d'estimation améliorée sont:

A. Le processus d'estimation proposé doit utiliser des techniques pertinentes pour gérer l'incertitude et l'ambiguïté, comme le font les practiciens lorqu'ils utilisent leur 'jugement d'experts' en estimation de projets logiciel: le processus d'estimation proposé doit utiliser les variables utilisées par les praticiens.
B. Le processus d'estimation proposé doit être utile à un stade précoce du processus de développement logiciel.
C. Le processus d'estimation proposée doit préserver l'expérience (ou la base de connaissances) pour l'organisation et inclure un mécanisme facile pour définir l'expérience des experts.
D. Le modèle proposé doit être utilisable par des personnes avec des compétences distinctes de celles des 'experts' qui définissent le contexte d'origine du modèle d'estimation proposé.
E. Pour l'estimation dans le contexte des premières phases, un processus d'estimation fondé sur la logique floue a été proposée, soit : 'Estimation of Projects in a Context of Uncertainty - EPCU''.

Une caractéristique importante de cette thèse est l'utilisation, pour fin d'expérimentation et de vérification, d'informations provenant de projets provenant de l'industrie au Mexique.

La phase d'expérimentation comprend trois scénarios:

Scénario A. Le processus d'estimation proposé doit utiliser les techniques pertinentes pour une gestion de l'incertitude et de l'ambiguité afin de faciliter la tache aux intéressés de réaliser ses estimations. Ce processus doit prende en compte les variables que les intéressés utilisent.

Scénario B. Ce scénario est similaire au scénario A, sauf qu'il s'agit de projets en démarrage, et pour lesquels les valeurs finales de durée et de coûts ne sont pas disponibles pour fin de comparaison.

Scénario C. Afin de remédier au manque d'informations par rapport au scénario B, le scénario C consiste en une expérience de simulation.

Ces expérimentations ont permis de conclure que compte tenu des projets examinés dans les 3 scénarios, l'utilisation du processus d'estimation défini – EPCU - permet d'obtenir de meilleurs résultats que l'approche par opinions d'experts et peut être utilisée pour l'estimation précoce des projets de logiciels avec de bons résultats.

Afin de gérer la quantité de calculs requis par le modèle d'estimation EPCU et pour l'enregistrement et la gestion des informations générées par ce modèle EPCU, un outil logiciel a été conçu et développé comme prototype de recherche pour effectuer les calculs nécessaires.

DESIGN OF A FUZZY LOGIC SOFTWARE ESTIMATION PROCESS

Francisco VALDÉS

ABSTRACT

This thesis describes the design of a fuzzy logic software estimation process.

Studies show that most of the projects finish overbudget or later than the planned end date (Standish Group, 2009) even though the software organizations have attempted to increase the success rate of software projects by making the process more manageable and, consequently, more predictable.

Project estimation is an important issue because it is the basis for the allocation and management of the resources associated to a project. When the estimation process is not performed properly, this leads to higher risks in their software projects, and the organizations may end up with losses instead of the expected profits from their funded projects.

The most important estimates need to be made right in the very early phases of a project when the information is only available at a very high level of abstraction and, often, is based on a number of assumptions.

The approach for estimating software projects in the software industry is the one typically based on the experience of the employees in the organization. There are a number of problems with using experience for estimation purposes: for instance, the way to obtain the estimate is only implicit, i.e. there is no consistent way to derive the estimated value, and the experience is strongly related to the experts, not to the organization.

The research goal of this thesis is to design a software estimation process able to manage the lack of detailed and quantitative information embedded in the early phases of the software development life cycle.

The research approach aims to leverage the advantages of the experience-based approach that can be used in early phases of software estimation while addressing some of the major problems generated by this estimation approach.

The specific research objectives to be met by this improved software estimation process are:

A. The proposed estimation process must use relevant techniques to handle uncertainty and ambiguity in order to consider the way practitioners make their estimates: the proposed estimation process must use the variables that the practitioners use.
B. The proposed estimation process must be useful in early stages of the software development process.
C. The proposed estimation process needs to preserve the experience or knowledge base for the organization: this implies an easy way to define and capture the experience of the experts.

D. The proposed model must be usable by people with skills distinct from those of the people who configure the original context of the proposed model.

In this thesis, an estimation process based on fuzzy logic is proposed, and is referred as the 'Estimation of Projects in a Context of Uncertainty - EPCU'.

The fuzzy logic approach was adopted for the proposed estimation process because it is a formal way to manage the uncertainty and the linguistic variables observed in the early phases of a project when the estimates need to be obtained: using a fuzzy system allows to capture the experience from the organization's experts via inference rules and to keep this experience within the organization.

The experimentation phase typically presents a big challenge, in software engineering in particular, and more so since the software projects estimates must be done "*a priori*": indeed for verification purposes, there is a typically large elapsed time between the initial estimate and the completion of the projects upon which the 'true' values of effort, duration and costs can be known with certainty in order to verify whether or not the estimates were the right ones.

This thesis includes a number of experiments with data from the software industry in Mexico. These experiments are organized into several scenarios, including one with re-estimation of real projects completed in industry, but using – for estimation purposes - only the information that was available at the beginning of these projects.

From the experiments results reported in this thesis it can be observed that with the use of the proposed fuzzy-logic based estimation process, estimates for these projects are better than the estimates based on the expert opinion approach.

Finally, to handle the large amount of calculations required by the EPCU estimation model, as well as for the recording and the management of the information generated by the EPCU model, a research prototype tool was designed and developed to perform the necessary calculations.

INDEX

Page

TABLES LIST

Page

FIGURES LIST

Page

ABBREVIATIONS AND ACRONYMS LIST

B2B	Business to Business
BRE	Balanced Relative Error
CBR	Case Based Reasoning
CFP	COSMIC Function Points
CIO	Chief Information Officer
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
COBIT	Control Objectives for Information and Related Technology
COCOMO	COnstructive COst MOdel
COSMIC	Common Software Measurement International Consortium
CW	Computing with Words
DBMS	Database Management System
EMRE	Magnitude of Relative Error relative to the estimate
EPCU	Estimation of Projects in a Context of Uncertainty
EPEI	Estimación de Proyectos en Entornos de Incertidumbre (Spanish translation of EPCU)
FFP	Full Function Points
FISMA	Finnish Software Measurement Association
FL	Fuzzy Logic
FP	Function Points
FPA	Function Points Analisys
FSM	Functional size measurement
IBRE	Inverted Balanced Relative Error
IEC	International Electrotechnical Commission
IFPUG	International Function Points User Group
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
KA	Knowledge Area
KDSI	Thousands of lines of delivered source instructions

MMRE	Mean Magnitude of Relative Error
MOPROSOFT	Modelo de Procesos para la Industria del Software (Software Industry Process Model)
MRE	Magnitude of Relative Error
NESMA	Netherlands Software Metrics Association
NL	Natural Languaje
OO	Object Oriented
OTAN	North Atlantic Treaty Organization (NATO)
PMI	Project Management Institute
PNL	Precisiated Natural Language
PRED	Criterion represents a proportion of a given level of accuracy
PSP	Personal Software Process
RMS	Root of the Mean Square
SDMRE	Standard Deviation of MRE
SEI	Software Engineering Institute
SPSS	Statistical Package for the Social Sciences
SRS	Software Requirements Specification
SWEBOK	Software Engineering Body of Knowledge
TSP	Team Software Process
UCP	Use Case Points
UML	Unified Modeling Language
USA	United States of America
WG	Work Group

SYMBOLS AND UNITS LIST

gu	Generic unit asociated to the x axis in the membership function.
mv	Membership value asociated to the y axis in the memebrship functions.
ph	Person hour hour of work.
mm	Man month of work.

INTRODUCTION

Software Project Estimation in the Early Project Phases

Information is acquired in a gradual way throughout the software development life cycle (Figure 1.1): for instance, at the conceptualization phase most of the information available is at a very high level of abstraction and it is often based on a number of assumptions (documented or implicit) which can be neither verified nor precisely described at that point in time. This leads to the challenge of having to make decisions on project budgets on the basis of incomplete and, at times, unreliable information.

Consequently, software project estimates of effort and duration based on such incomplete and not fully reliable information should not be expected to be accurate: such estimates are to be associated with potentially significant ranges of variance. Still, even at this early phase of a software development process, management must rely on such incomplete information for decision making purposes.

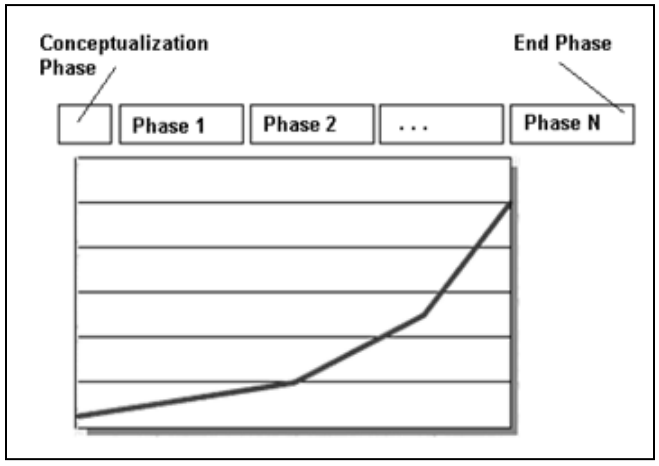


Figure 0.1 Information Acquisition Process through the Software Development Phases

In Figure 0.1, the *x* axis is the time and the *y* axis represents the relative quantity and detail of information acquired trough time (from none before a project begins, to all at the end of a project).

There is an imperative need for the organizations to estimate in the early phases of the software projects in order to plan and manage business and resources:

- on the one hand, the decision to launch a project is often determined by considering, in particular, the 'perceived' (or "subjective") importance of the project outcome for the organization that is, delivering the product (a system);
- on the other hand, and concurrently, the organization has to minimize the risks that it may not be possible to complete the project within the time-to-market required.

Improvements to the software estimation techniques in this context are therefore welcome in order to improve the decision making process and to decrease related risks.

It is during the initial project phases when dealing with rough information that the most important estimates often need to be made: that is, when the software is conceptualized (i.e. in the feasibility phase when the information is often vague and imprecise).

"An estimation is a prediction that is equally likely to be above or below the actual result" (DeMarco, 1982).

Morgenshtern (2007) mentions the following usages of projects estimation: project selection, staffing, scheduling, monitoring and control, team performance assessment, and marketing.
In the past 40 years, many estimation models and tools have been developed: most of these models focus on estimating effort, and the unit most often used is the man/month (MM).

Software is different from other systems: in physical systems, the attributes are usually fully identified and described in terms of measures and quantities, but this is not yet so with software. In the early stages of software development, when a software system is

conceptualized, the information available is initially only at a very high level of abstraction, and it is often based on a number of assumptions which can be neither verified nor precisely described at that time. This has been illustrated with the cone of uncertainty (Boehm, 1981) - see Figure 0.2.

The original conceptual basis of the cone of uncertainty was developed by Barry Boehm who referred to the concept as the "Funnel Curve" (Boehm, 1981); later, McConnell (2006) has used the expression "Cone of Uncertainty" to describe this concept.

This Figure 0.2 represents that in the early phases the variability in the estimates is higher than in the later phases: the variation proposed by Boehm (1981) in early phases is [-25%, 400%] in this cone of uncertainty.

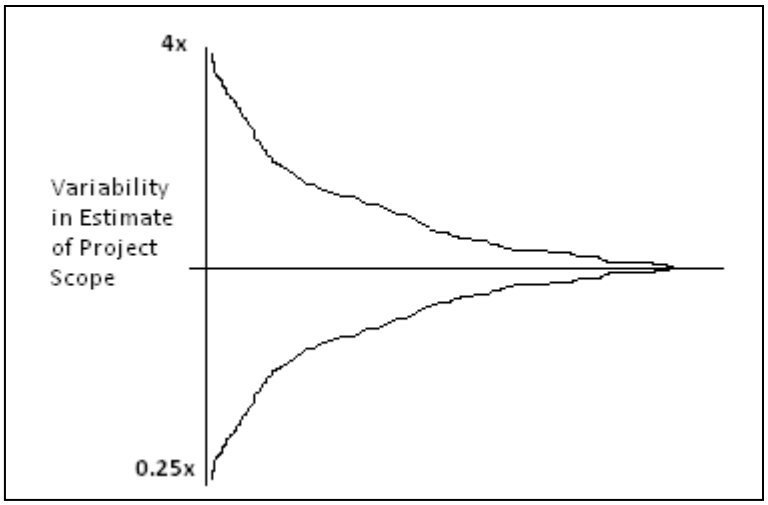


Figure 0.2 Cone of Uncertainty -Adapted from Boehm (1981)

Considering this high uncertainty context in the early phases, Morgenshtern *et al.* (2007) suggest that “estimates are, in reality, guesses regarding future performance based on available knowledge. As such, their accuracy is affected by the extent of uncertainty regarding the task to be estimated. Uncertainty is associated, inter alia, with definitions of requirements, choice of technological solutions, innovativeness of needs, and customer characteristics” (Morgenshtern, 2007).

An estimation technique typically used in industry is the one based on the ‘opinions’ of the organization’s employees, that is, an estimation technique based on their work experience. This experience-based estimation approach (also known as: ‘expert judgment’ or ‘intuitive approach’) considers informally an unspecified number of quantifiable and non-quantifiable variables that other estimation models based on statistical techniques cannot take into account.

Of course, there are a number of problems with using experience to make estimates, notably the following ones:

- experience is specific to the people and not to the organization;
- estimation expertise is neither well described nor well understood;
- this estimation expertise is hard to assess;
- a human is implicit in the social context and the estimation is affected by this social factor, and
- this estimation expertise cannot be replicated systematically.

In summary, with experience-based estimation, the people expertise cannot be used without the people who possess it.

In spite of these problems, the experience-based estimation approach is still valuable to an organization, and presents some advantages since it can:

- manage qualitative and linguistic variables;
- manage or work with uncertainty;
- create commitment for the people or team to reach the estimated value.

A challenge with this experience-based estimation approach is to figure out how to benefit from it and use it in combination with other estimation techniques, including algorithmic or non-algorithmic-based estimation techniques.

Measurement of the Inputs to the Estimation Models

Any estimation model has a strong relation with the measurement process of the input variables used to generate the estimate. This means that the measurement process is the basis of the estimation model: when the measurement of the input variables for an estimation model is reliable, then there can be more confidence in the use of the estimation model which quality has been documented on the basis of past completed projects.

Abran (2008) refers to an audit report on the inputs to the estimation process: this report should include audit results on the accuracy of and completeness of estimation inputs such as: the functional size of the software product, the resources needed for the development process and the process components themselves. The credibility of the input variables used in the estimation process impact the quality of outcomes of the estimation model.

Measurement in software engineering is challenging, including for the following reasons:
• Software engineering is 'young' (at most 40 years). The term was used for the first time by Fritz Bauer in the first software development conference organized by the Science Committee of OTAN in Garmisch (Germany), 1968.
• Early publications about software measurement date back to the early 70's (Santillo, 2006).
• Most of the software attributes are currently mostly described in a qualitative manner rather than quantitatively (Idri, 2004) and depend on human views.

In estimation models, a number of project variables such as complexity, maintainability, team integration and so on are of a categorical nature: for instance, complexity of software is often classified using ordinal categories (simple, medium complexity or very complex). One of the problems related to the categorical features of a software project, is that the experience of the humans is directly involved in their categorization process (that is, making the judgment call to classify it within one of the previously agreed categories). Thus the humans often use linguistic and categorical values (i.e. very small, small, large and very large) to

describe and evaluate such variables rather than using numerical values of a ratio scale type in order to quantify such software and software projects attributes. This use of linguistic and categorical values leads to some imprecision in both the evaluation of such variables and how such variables are taken into account in estimation models.

Software Measurement and Fuzzy Logic

Morgenshtern *et al.* (2007) identify four (4) dimensions that impact estimation accuracy:

1. Project uncertainty: the amount of uncertainty perceived by those who had to estimate the duration of the project tasks and the effort required to carry them out.
2. Estimation processes: the various processes that contribute, either directly or indirectly, to the generation of the estimated project duration and effort.
3. Development management processes: controlling actual performance against estimates and updating as appropriate, carrying out systematic risk assessments to validate the estimates, and implementing managerial policies that promote the commitment of team members to the estimates that constitute their project plan.
4. Estimator's experience: years of experience and number of projects with similar technologies and systems the estimator was involved with.

The project uncertainty can be described and contextualized by linguistic values: "it is not possible to measure it, however it is possible to contextualize it" (Valdés, 2007). A challenge is to convert the linguistic values to valid numerical values, preferably on a ratio scale. The software community often uses categorical data or intervals to represent these linguistic values (See Figure 0.3).

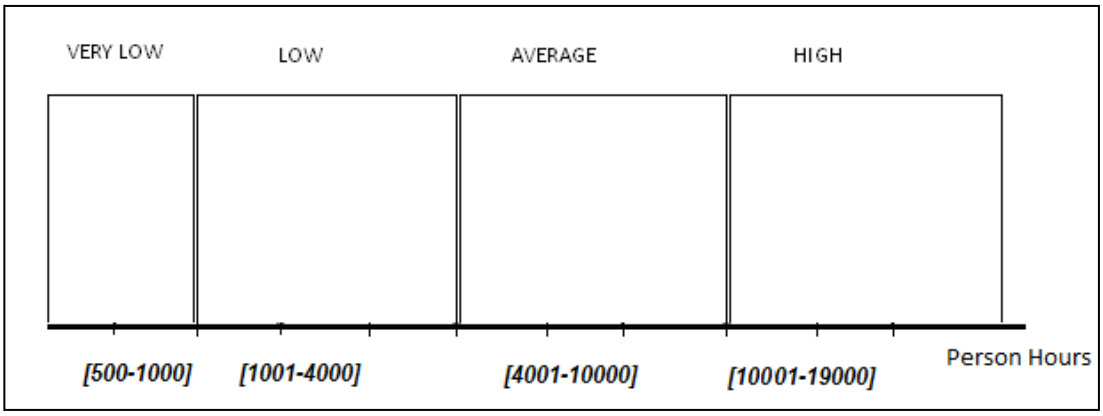


Figure 0.3 Effort represented by person hours [ph] intervals identified as categorical data

In Figure 0.3, four intervals are represented: from 500 person-hours (ph) to 19,000 ph. From the left to right the intervals are: very low [500-1000] ph, low [1001-4000] ph, average [4001-10000] ph and high [10001-19000] ph.

The use of intervals is based on the need to explain the reality with the available scales. "One way of distinguishing between real-world objects or entities is to describe their characteristics. Measurement is one such description. A measure is simply a mapping from the real, empirical world to a mathematical world, where we can more easily understand an entity's attributes and relationship to other entities. The difficulty is in how the mathematical behavior is interpreted and what it means in the real world" (COSMIC Measurement Practice Commitee, 2007).

In this context, classical tools for measurement (i.e. Aristotle logic, statistics) do not mimic the way in which the humans interpret the linguistic values: these classical tools cannot interpret the linguistic values as humans do since these tools were not created to handle the imprecision and uncertainty as the humans do.

There exist a number of techniques, such as fuzzy logic, to handle quantitatively imprecision and uncertainty: fuzzy logic (FL) is a formal quantitative framework that captures the vagueness of humans' knowledge expressed via natural language: "Basically, fuzzy logic is a precise logic of imprecision and approximate reasoning. More specifically, fuzzy logic may

be viewed as an attempt at formalization/mechanization of two remarkable human capabilities:

1. The capability to converse, reason and make rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility – in short, in an environment of imperfect information.
2. The capability to perform a wide variety of physical and mental tasks without any measurements and any computations" (Zadeh, 2008).

Using such techniques that can manage uncertainty may help design better software estimation models.

Thesis organization

This thesis is organized in eight chapters. Chapter 1 presents the state of the art on software estimation models. Chapter 2 presents the research goal and the specific research objectives. Chapter 3 presents the methodology designed for this research project. Chapter 4 presents the initial design of our proposed EPCU estimation model. Chapter 5 presents the design of the prototype tool built to facilitate the use of the estimation model by automating the amount of calculations required by the proposed EPCU model. Chapter 6 presents the experimentats set up to analyze the performance of the proposed estimation model and experiments results are also presented and discussed. Chapter 7 presents the the conclusions and future work.

CHAPITRE 1

STATE OF THE ART

1.1 Introduction

The software engineering discipline is not yet as mature as other scientific disciplines: most of the measures designed for software products are still based on researcher's intuition rather than rigorous designs and strong experimentations.

In this chapter an overview of the software engineering discipline maturity is described, followed by a focus on the estimation techniques, including a classification. This chapter includes next a discussion on the estimation of software project duration.

This chapter also presents the more frequently used quality criteria for the estimation techniques found in the literature; it also presents an overview of the evolution of the estimation models, followed by the evolution of the functional size measurement methods.

Finally this chapter describes the use of fuzzy logic for estimation purposes.

1.2 Software engineering

The Software Engineering discipline is not yet as mature as other scientific disciplines: most of the measures designed for software products are still based on researchers' intuition rather than on rigorous designs and strong experimentations: Abran (2010, 2008) presents a number of analyses of the designs of COCOMO, COCOMO II, Function Points, Use Case Points, Halstead's metrics (commonly referred to as 'software science' - (Halstead, 1977)) and the cyclomatic complexity number (Mccabe, 1995, 1996). These analyses illustrate a number of the weaknesses of these software measures.

The IEEE and ISO 19759 Guide to the Software Engineering Body of Knowledge - the SWEBOK Guide (Abran, 2004) – presents a taxonomy of all the knowledge areas (KA) recognized as part of the software engineering discipline. While measurement is an important aspect of all SWEBOK KA, it is in the Software Project Planning topic where the estimation topic is presented specifically, and in the Software Engineering Measurement topic where measurement programs are presented - see Figure 1.1.

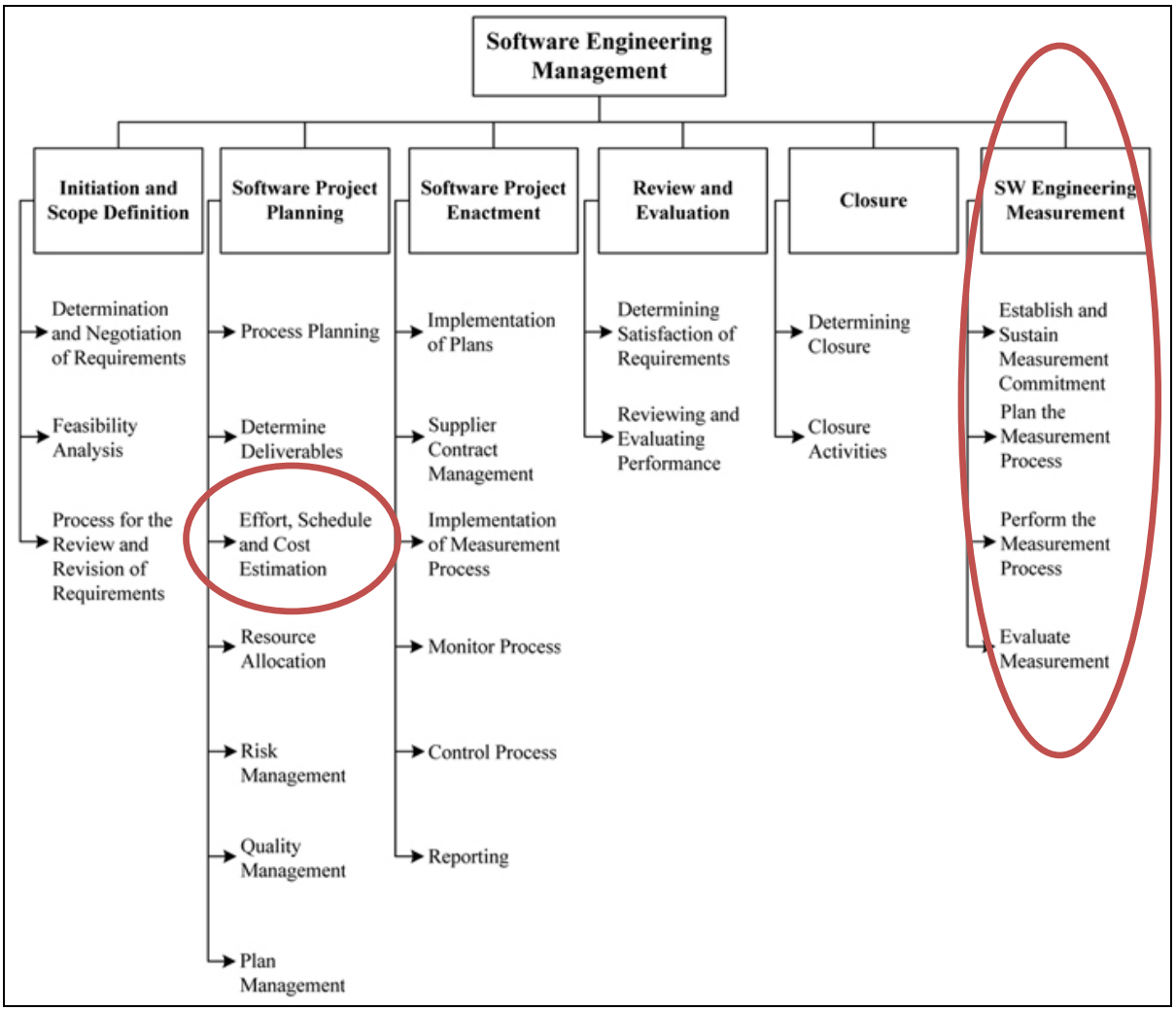


Figure 1.1 Breakdown of topics for the Software Engineering Management KA- Adapted from Abran (2004)

The immaturity is a peculiarity of software engineering relative to the other classical engineering and scientific disciplines (Habra, 2008; Abran, 1998). A symptom of the lack of maturity is the limited number of internationally accepted software measurement methods.

In mature disciplines it is possible to observe international consensus about measurement, as evidenced through established measurement methods and their respective etalons. In the software domain there exist international standards only for the functional size measurement, including the ISO 14143 series prescribing key concepts of the entity and the attribute to be measured. To date, ISO has recognized five (5) functional size measurement methods for software as compliant to ISO 14143:

A. One is referred to as a 2nd generation of functional size measurement methods: COSMIC – ISO 19761
B. Four (4) are considered as 1st generation of functional size measurement methods: MKII: ISO 20698, IFPUG: ISO 20926, NESMA: ISO 24570 and FISMA: ISO 29881.

This means that even for the measurement of the functional size of software there is not yet a single universally accepted way of measuring it.

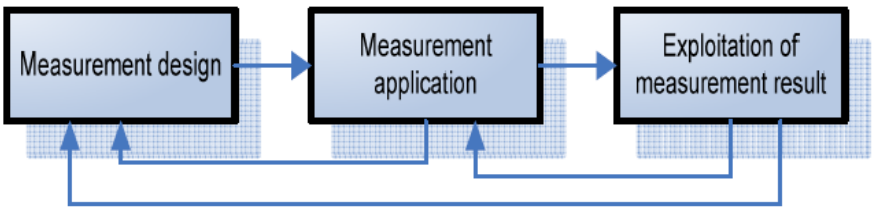
Hundreds of software measures (akin to software 'metrics') have been proposed in the software engineering field, but there is not yet a widely accepted framework or consensus on how to conduct an analysis of the measures proposed, including to make comparative studies of the various 'metrics' proposed to measure the same attribute (Habra, 2008).

Indeed, the measurement in software engineering is not as mature as in other disciplines and some researchers are looking into metrology to improve the measurement foundations of software engineering (Condori-Fernandez, 2008; Habra, 2008, Abran 2010).

Considering this, it would be challenging to consider software estimation more mature that measurement in software engineering.

In the literature, Habra (2008) refers to the decomposition made by Jacquet *et al.* (1997) who divide the measurement life cycle into three consecutive phases. Even while these phases are presented as consecutive, in practice they can be viewed as iterative – see Figure 1.2. This decomposition is referenced by Abran (2010) as the Measurement Context Model.

In this Measurement Context Model the first phase is considered the most important: it consists in the definition of what will be measured and what is the objective behind it.



1- Design	2 - Application	3 - Exploitation
Verification of the design of a measurement method (with respect to the objective and measurement principle)	Verification of the actual of the measurement procedures, including the measurement instruments or steps, and of results obtained	Verification of the quality of the models of relationships across multiple measurement results

Figure 1.2 Measurement Context Model (Abran, 2010) With the Author's authorization.

Two definitions need to be presented here: these definitions are related to the Measurement Context Model and are referenced in the International Vocabulary of Basic and General Terms in Metrology (ISO, 1993).

"Measurement method: A measurement method is a logical sequence of operations, described generically, used in the performance of measurements.

Measurement procedure: A measurement procedure is a set of operations, described specifically, used in the performance of particular measurements according to a given method".

These definitions are related to the two first phases in the Measurement Context Model. Abran (2010) clarifies the substeps for each Measurement Context Model phase in a diagram – see Figure 1.3.

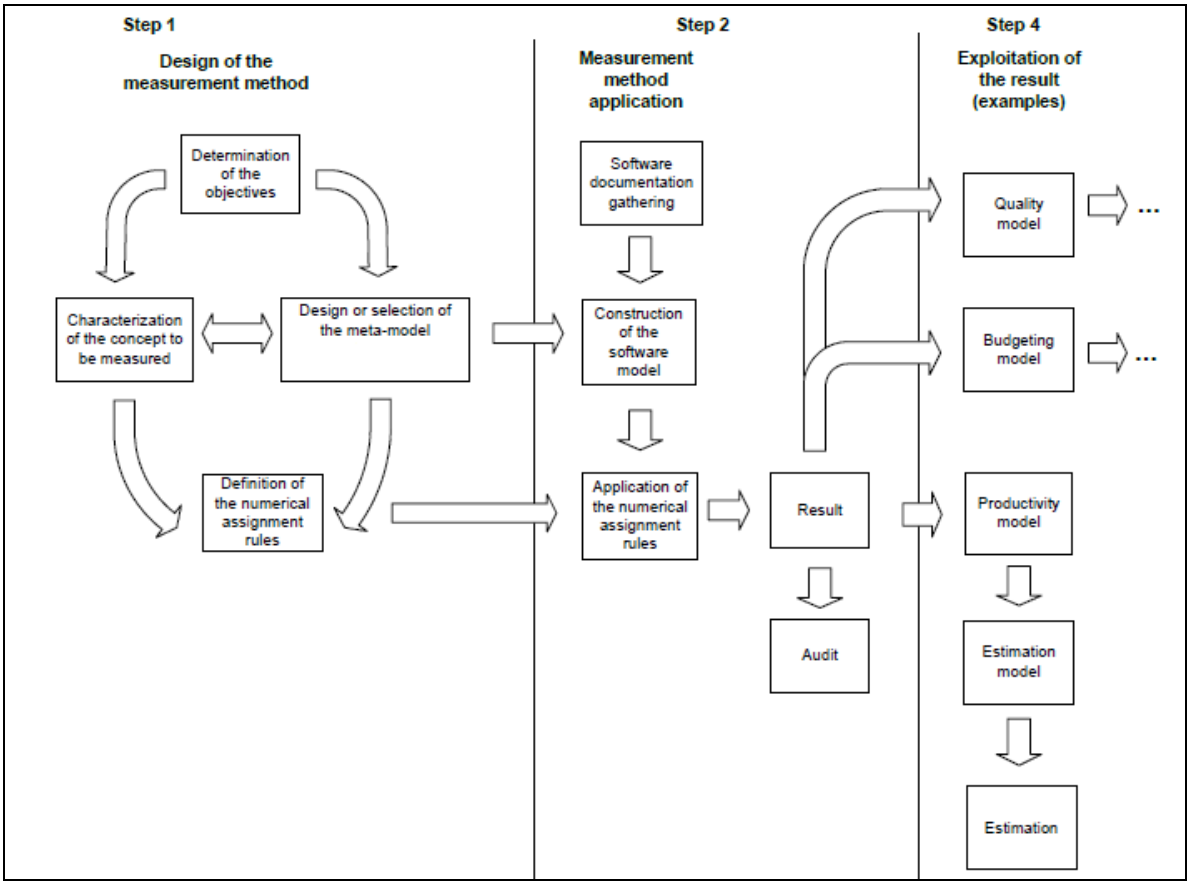


Figure 1.3 Measurement Context Model - Detailed Levels (Abran, 2010) with the author's authorization.

Naturally, the managers are more interested in Phase 3: Pfleeger *et al.* (1997) mention that "customers encourage product assessment because they are interested in the final product's characteristics, regardless of the process that produced it"; however, if the previous phases are not based on sound foundations from a measurement perspective, the third one might not produce good results. If the measurement methods have been previously well defined and are available, the first phase is not necessary.

1.3 Classification of Software Estimation Techniques

In the software engineering field, a number of estimation models and tools have been developed over the past 40 years in order to help predict important attributes about the software projects to be developed, such as the duration, effort and cost.
A general classification of estimation models is presented in Abran (2008):

- *A priori*
- *A posteriori*

The "*a posterior"* estimation model approach is built considering completed projects, when all the variables used as inputs to the estimation models are known, as well as the output variable which can be used to evaluate the models built.

The "*a priori"* estimation model approach is used at the beginning of the projects when the variables used as input are often imprecise and uncertain, typically using a technique based on informal personal or organizational experience, and when there is, of course, no data available on the projects completed.

In the literature there are several approaches of estimation techniques classification (Idri, 2001; Shepperd, 1996; Idri, 2002) based on the model and the information considered to make the estimations. One such classification approach (Shepperd, 1996) classifies the techniques into three categories:

A. Expert judgment
B. Algorithmic models
C. Analogy

A. The expert judgment (referred to in this thesis as an 'experience-based' approach) can be hardly considered as a technique because the means of deriving an estimate are not explicit. However the estimation approach typically used in industry is this one based on the experience of the employees in the organization: i.e. the 'expert judgment' based on people's

experience. Of course, there are a number of problems with using experience to provide estimates.

Hill *et al.* (2000) mention - "Perhaps the most common approach to estimating effort is to consider the opinions of experts. This does not require the existence of historic data and is particularly useful at the start of system development when requirements are vague and changing, and it is ballpark figures that are required".

Two approaches are described by Shore (2008): the first approach emphasizes what "should" be done, and assume that a rational and consistent approach is followed. The second approach is focused on how the organization's individuals actually behave and make decisions: this corresponds to the "behavioral" view.

B. The algorithmic models are the most documented in the literature. Examples of this category of models include COCOMO-based models (Boehm, 1981, 2001), Function Points based models (IFPUG, 2005; Kitchenham, 1997) and Use Case Points based models (Ribu, 2001). Some of these models, such as COCOMO are based on inputs within pre-defined intervals, while other models, such as the function points based models, are derived from statistical or numerical analysis of some historical data set about projects completed. The statistical techniques most frequently documented in these algorithmic models are the simple/multiple/stepwise regression. Other statistical techniques used in such estimation models are the Bayesian approach, principal components analysis and polynomial interpolation.

Some disadvantages for this category of algorithmic models are documented in Idri (2001, 2002):

- The prediction function form is pre-determined. For example: in the exponential model, Effort = α x size$^{\beta}$, where α represents the productivity coefficient and β represents the coefficient of economies/diseconomies of scale.

- This category of models needs to be adjusted to local contexts: the models are often obtained in some source contexts that will be different from the target contexts.
- These algorithmic models need historic data, and many organizations do not have this information. Additionally, collecting such effort and cost data may be both expensive and time consuming (Morgenshtern, 2007).

C. The analogy technique is considered as a systematic form of expert judgment. An example of using analogy estimation is the complex human intelligence: the analogy approach uses information that is more imprecise and vague than precise and certain. Some researchers (Myrtveit, 1999; Shepperd, 1996; Idri, 2004) are paying attention to the analogy approach because of its similarity with the expert judgment. The analogy approach is based on a Case Based Reasoning (CBR) approach (Kolodner, 1993) that includes four steps:

- Characterization of cases.
- Storage of past cases.
- Retrieval of similar cases to use analogies.
- Use the retrieved cases to solve the target case (case adaptation).

The analogy technique presents some disadvantages, in particular with respect to the knowledge required to identify analogy cases and in the computational effort.

Park (1994) has provided some insights on the software estimation processes:

- "Estimates are made by people, not by models. They require reasoned judgment and commitments to organizational goals that cannot be delegated to any automated process.
- All estimates are based on comparisons. When people estimate, they evaluate how something is like, and how something is unlike, things that they or others have seen before.
- Before people can estimate, they must acquire knowledge. They must collect and quantify information from other projects, so that they can place comparative evaluations on demonstrably sound footings."

1.4 Estimation Techniques in the Literature

A significant proportion of research on software estimation has focused on linear regression analysis; however, this is not the unique technique that can be used to develop estimation models. An integrated work about these estimation techniques has been published by Gray (1997) who presented a detailed review of each category of models.

- Least Squares Regression. "Linear least squares regression operates by estimating the coefficients in order to minimize the residuals between the observed data and the model's prediction for the ith observation. Thus all observations are taken into account, each exercising the same extent of influence on the regression equation, even the outliers" (Gray, 1997).

- Robust Regression. "Robust regression analysis has been used to avoid the impact of outliers in the models. The general idea behind robust regression is that by changing the error measure (from least squares) the model can be made more resilient to outlying data points. There are several robust regression models" (Gray, 1997).

- Neural networks. "The most common model-building technique used in the literature as an alternative to least mean squares regression is back-propagation trained feed-forward neural networks (back-propagation networks)" (Gray, 1997). "The neural networks take problems previously solved in order to build a decision taking system" (Ponce, 2010).

- Fuzzy Systems (Adaptive). "Fuzzy systems have been used in only a few publications for software estimation models. A fuzzy system is a mapping between linguistic terms, such as "very small", attached to variables. Thus, an input into a fuzzy system can be either numerical or linguistic, with the same applying to the output" (Idri, 2000, 2001, 2002; Gray, 1997).
- Hybrid Neuro-Fuzzy Systems. MacDonell mentions that "researchers (Horikawa, 1992; Jang, 1993) have attempted to combine the strengths of neural networks and fuzzy

systems while avoiding most of the disadvantages of each. This has resulted in a wide range of possibilities for hybridizing the two techniques. While all of these techniques are different in some way, they share the same basic principles: an adaptive system that can deal with easily comprehended linguistic rules and that permits initialization of the network based on available knowledge" (Gray, 1997).

- Rule Based Systems. "Rule-based systems have been used in very few cases for modeling software projects estimation. A rule-based system is organized around a set of rules that are activated by facts being present in the working memory, and that activate other facts" (Gray, 1997).

- Case-Based Reasoning. "Is a method of storing observations, such as data about a project's specifications and the effort required to implement it, and then when faced with a new observation retrieving those stored observations closest to the new observation and using the stored values to estimate the new value, in this case effort. Thus a case-based reasoning system has a pre-processor to prepare the input data, a similarity function to retrieve the similar cases, a predictor to estimate the output value, and a memory updater to add the new case to the case base if required" (Gray, 1997).

- Regression and Classification Trees. "Regression and classification trees, while based on the same principle, each have a different aim. Regression trees can be used when the output value to be predicted is from the interval domain, while classification trees (also known as decision trees) are used to predict the output class for an observation, that is to say, from the nominal or ordinal data scale. Both algorithms work by taking a known data set and learning the rules needed to classify it" (Gray, 1997).

1.5 Issues in the estimation of software project duration

The three major constraints on projects include typically project effort (i.e. as a substitute for costs), project schedule and the number of functions to be delivered (i.e. project scope).

Within this set of "triple Constraints", the schedule or the time-to-market, is often the hardest to control by the managers: in addition to the other two constraints (cost, scope) which influence the schedule, there are a number of other project variables that may impact project schedule.

It is generally recognized (Bourque, 2007) that requirements define the project size (scope), which impact the effort needed to develop it, which then drives the project duration – see Figure 1.4. This relation between effort and duration is not necessarily linear (Oligny, 2000).

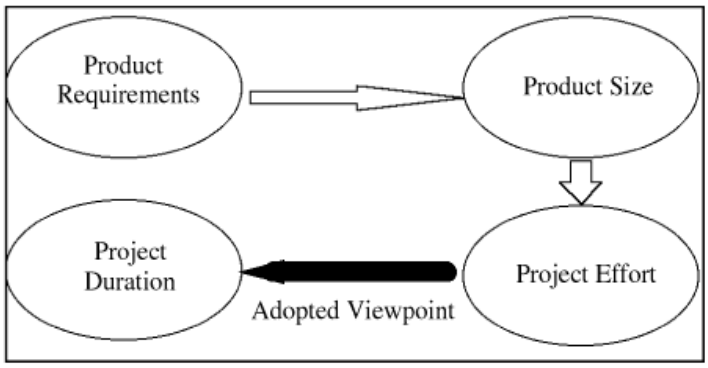


Figure 1.4 Example of a strategy to estimate project duration (Bourque, 2007), with permission.

The influences of other variables may vary: some are related to the project environment and others to the project itself. Bourque, 2007 illustrates one estimation strategy – Figure 1.4 – whereas the estimate of project duration is based on an estimate of project effort, which is itself based on estimate of product size and product requirements.

A specific project where its software size and project effort has already been pre-defined is taken as an example.: if the effort for this software size is estimated at 480 person-hours, and if all the tasks have to be strictly executed sequentially, this project will require one person working 8 hours a day: this means that the project duration will be estimated at 3 months. However, if the project manager leadership is very poor, it is very possible that the project will require a longer schedule; there can be other variables that can also impact the schedule of the project.

In the actual competitive business context, the software development organizations are often more interested in time-to-market than in the cost of the software: not because the cost is not important, but because the opportunity of delivering a project within the promised schedule is crucial to adapt to the competitive environment.

Morgenshtern *et al.* (2007) have identified some interesting estimation-related issues that need to be considered:

- project uncertainty has a stronger effect on duration estimate than on effort estimation errors.
- the effect of the estimators' experience is more significant for duration estimation than for effort estimation.
- effort estimation and duration estimation are driven by somewhat different processes, and that the respective errors are affected by different factors.

From these issues, it is possible to observe that if the uncertainty cannot be managed in the early phases, the project duration estimation will be impacted.

Estimation of project duration has often a strategic value for organizations. Most of the literature on estimation focuses on improving estimation of effort and does not often address directly the estimation of the project duration. The studies made on the duration estimation usually follow an approach in which the basis is the effort obtained by any kind of estimation model, and which forms next the basis for estimation project duration (Bourque , 2007).

A simple example: for a software project that has been estimated to require 200 person-hours, the duration can be estimated on the basis of a number of variables, such as:

- How many developers will design and construct the software?
- How much experience do they have with the development tools set used?
- The cohesion of the development team, and so on.

The full set of variables can be combined into a complex model of relationships to produce an estimate of the project duration, but the estimation result is not necessary better than its estimation based only on the effort estimated. A project may be influenced by many parameters at the same time, their impact being distinct from each other: some might have a major impact in a specific project, while others might be almost irrelevant (Kadoda, 2000).

Duration estimation is a major challenge, due for instance to the uncertainty of the information available to make the estimation and second on how to use the available information in order to obtain a result. Park (1994) referenced in Oligny *et al.* (2000) mentions that "It does imply though, that, software duration estimation is a somewhat complex problem and that applying these models correctly requires much expertise and commitment".

1.6 Estimation Models: Quality criteria

The software measurement and estimation literature (Idri, 2000, 2001, 2004; Kolodner, 1993; Shepperd, 1996; Myrtveit, 1999; Shepperd, 1996; Gray, 1997, Abran, 2010) presents the following quality criteria to evaluate estimation models:

- The Magnitude of Relative Error (MRE), defined usually by:

$$MRE = \left| \frac{Actual - Estimated}{Actual} \right| \quad (2.1)$$

$$\%\ MRE = 100\ x \left| \frac{Actual - Estimated}{Actual} \right| \quad (2.2)$$

- The prediction level Pred.

$$Pred(l) = \frac{K}{N} \quad (2.3)$$

The Pred criterion represents a proportion of a given level of accuracy, where *k* is the number of projects in a sample of size N for which the MRE <= l. Usually, a good prediction level is a Pred (25%) = 75%.

Considering the MRE as the base criterion, the accuracy of the estimation can also be measured by the Mean Magnitude of Relative Error (MMRE) and the Median Magnitude of Relative Error (MdMRE). The major advantage of the median over the mean is that the median is not sensitive to the outliers; so the median is a more appropriate as a measure of central tendency of a skewed distribution.

Other criteria may need to be evaluated: the proportion of deviation in the estimates in particular because of the risk to make very erroneous estimates. This can be measured by the Standard Deviation of MRE (SDMRE defined as the root of the mean square error (RMS)).
The SDMRE = RMS is defined usually by:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Actual_i - Estimated_i)^2} \quad (2.4)$$

These quality criteria are used throughout the documents reviewed in the literature, including for comparing results generated by the estimation models.

In the literature, Stensrud (2002) and Gray (1999) mention some problems in the use of MRE as a selector between estimation models, indicating that the MRE is not independent of the projects size: the MRE is larger for small projects than for large projects, that means that MRE is negatively correlated with project size (Stensrud, 2002).

These authors (Stensrud, 2002) have proposed some other alternative quality criteria such as the Magnitude of Relative Error relative to the Estimate (EMRE), the Balanced Relative

Error (BRE) and the Inverted Balanced Relative Error (IBRE); these criteria are not very often used in the literature.

The quality expected of the estimation results obtained by any estimation model is very important : Abran (2010) mentions that "if the estimation model is used very early on in the life cycle when only scanty information is available (such as at the pre-feasibility stage), then most of the input numbers are 'guestimates' and are not derived from the application of rigorous measurement procedures; these 'guestimates' are indeed numbers, but with very little strengths in terms of precision, repeatability and reproducibility. Of course, the estimate (eg. the output number) produced by an estimation model based on these 'guestimates' in inputs cannot produced anything but 'guestimates' as output, with a level of 'goodness' that cannot of course be greater than the quality of the inputs" (Abran, 2008).

1.7 Evolution of the Estimation Models

In the literature on software estimation, a number of estimation models were identified, most of them algorithmic models. For instance, Boehm (1981) published the COCOMO model, one of the first documenting publicly the project database used: this COCOMO model used a set of 63 projects. This COCOMO model has 17 attributes: two of these refer to the thousands of lines of delivered source instructions (KDSI) and the project type (organic, semi-detached, and embedded). The other 15 attributes are related to the software environment.

The weakness of this model is that there were a lot of assumptions about the correct use of the model; these assumptions are challenging to meet in real projects (for example: the project will enjoy good management, the users requirements will not change substantially, etc.). Another weakness in this type of estimation models is the use of lines of code as a primary input which, of course, cannot be estimated accurately early in the project life cycle.

In 1996, Shepperd *et al.* (1996) made a comparison between three estimation techniques (analogy, linear regression and stepwise regression) using several datasets and concluded that it would seem that estimation by analogy is a superior technique than regression, since it can produce better estimation with respect to the quality criteria evaluated in the study, even when a statistical relation cannot be found and is a more intuitive method.

However, Shepperd *et al.* (1996) also identified a number of problems:

- as in algorithmic models, it is not clear what is the effect of old data points: when an organization develops some projects and successively introduces new technology the older data points will be increasingly misleading.
- it is not clear why different sets of variables and methods used are more or less successful with different data sets.

Myrtveit *et al.* (1999) made a comparison between multiple regression models, analogy models (using the analogy tool Angel) and the expert judgment approach using a set of 48 projects. These authors found that the results in an experiment are sensitive to a number of factors, in particular to the data (cleaning data, number of data points, number of independent variables, interval between the smallest and the largest project, the homogeneity), the experimental set up and the analysis.

Another finding from Myrtveit *et al.* (1999) is that the statistical methods produce information that is just one or several inputs to make a decision; there are other aspects that impact the estimation process, such as the experiences and the environment. When using human subjects, their skill level impacts on the results, and when the outliers are removed, the results favor the regression models.

In 2000, Kadoda *et al.* (2000) analyzed the Case Based Reasoning (CBR) using the Desharnais database with 77 projects; they found, in a general way, that estimation by analogy generates better results than step wise regression. They also identified that the presence of extreme outliers can have a major impact upon estimation accuracy. So

increasing the dataset does not necessarily enhance the accuracy of the estimation models. Configuring a CBR prediction system is a non-trivial task: a lot of decisions need to be made in the configuration phase. They conclude that simple similarity measures while using CBR present three major inadequacies:

- are computationally intensive,
- the algorithm is intolerant to noise and of irrelevant features,
- cannot handle categorical data other than binary values.

In the same year, Idri *et al.* (2000) introduce the use of fuzzy logic to tackle the problems of linguistic variables in the COCOMO model. They use the dataset of the COCOMO model with 63 projects.

The same authors (Idri *et al.*, 2001) developed the fuzzy analogy approach that can be used when the software projects area described by categorical or numerical data. This approach improves the classical analogy procedure and represents the data using fuzzy sets, handling the imprecision and uncertainty when describing a software project.

In order to validate the fuzzy analogy approach, Idri *et al.* (2002) made a comparison about Fuzzy Analogy, Fuzzy COCOMO, Classical Analogy and Classical Intermediate COCOMO'81, and they suggested a ranking against the performance of the models analyzed:

1. Fuzzy Analogy
2. Fuzzy Intermediate COCOMO '81
3. Classical Intermediate COCOMO '81
4. Classical Analogy.

This study concludes that using fuzzy logic with the estimation by analogy tolerates imprecision and uncertainty in its inputs (cost drivers).

Idri *et al.* (2004) extended the study previously developed, analyzing the performance of Fuzzy Analogy, Fuzzy COCOMO, Classical Analogy and Classical Intermediate COCOMO'81, and including the CBR.

Until now the studies have been carried on to compare the use of algorithmic models against other models that are not based in statistical or numerical analysis; the result from this type of models - specifically those which use fuzzy logic, is that the models were tolerant to the imprecision and have the ability to work with uncertainty because of their use of fuzzy logic.

A summary of some of the estimation models developed since the 1980's is presented in Annex I and includes: the author, the estimation model or technique used, the dataset analyzed, some insights into the proposed work, as well as some strengths or advantages, and some weaknesses or disadvantages of each software estimation model.

1.8 Functional Size Measurement (FSM) method

The first Functional Size Measurement (FSM) method was proposed by Allan Albrecht of IBM in 1979 (Albrecht, 1979), that is Function Point Analysis (FPA). This was the first software measurement method without a technology bind and its design was based only on what the system users could see from the outside (Abran, 2010). Naturally the initial design of FPA is applicable only to the specific software type upon which its design was based, that is the 'management information systems' (MIS). Subsequently, a number of variants from the Albrecht/IFPUG FPA approach have been proposed to improve the measurement of software functional size, and to extend its domain of applicability – see Figure 1.5 (Abran, 2010).

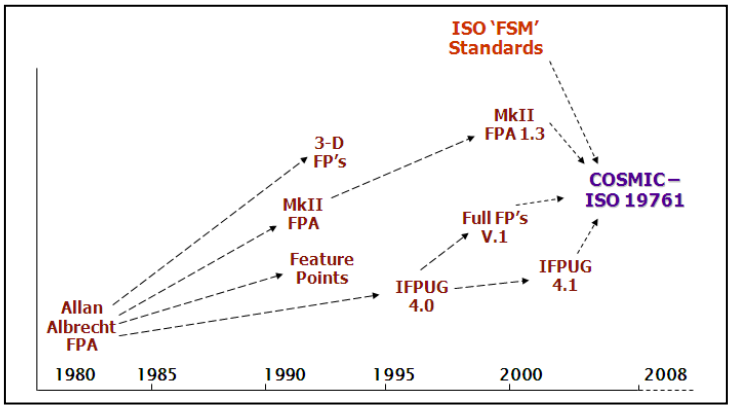


Figure 1.5 Evolution of functional size measurement methods (Abran, 2010) - With permission.

In 1994, a new Working Group 12 (WG12) of the ISO/IEC Joint Technical Committee 1, Sub-Committee 7 (Software Engineering) was established to seek to establish an international standard for functional size measurement. WG12 decided that the first step was to establish the basic principles of FSM. This initiative led to the publication of ISO 14143-1:1997 ('Information Technology – Software measurement – Functional size measurement – Definition of concepts'). Other technical reports in the 14143 series cover related topics like conformity assessment, verification of a candidate FSM method and the definition of types of software domains for FSM.

In the late 90's a set of organizations from USA, Canada and Japan funded a research project to extend the domain of application of functional size measurement (Abran, 2010) that was Management Information Systems (MIS) to real-time and embedded software domain. The research project was conducted by Dr. A. Abran and included five steps:

- Step 1. Literature review
- Step 2. Proposal for an extension to FP to real-time software
- Step 3. Field tests of the designed prototype
- Step 4. Analysis of measurement results
- Step 5. Public release

The method initially named 'Full Function Points' (FFP) was released in 1997 and introduced new transactional function types to the traditional FPA method (Abran, 2010).

In 1998, some experts of WG12 met informally in London to initiate the next effort to develop a new FSM Method based on FFP and aimed to meet the following constraints and objectives:

- Starting from established FSM principles.
- Aimed to be compliant with ISO/IEC 14143/1:1997 from the outset.
- Considering the experience of previously developed FSM methods.
- The re-designed FSM method had to be equally applicable to MIS/business software, to real-time and infrastructure software (e.g. as in operating system software) and to hybrids of these.

From the London meeting arose the 'COSMIC Group': the Common Software Measurement International Consortium. The first official version of its method, 'COSMIC-FFP v2.0' was published in October 1999, initiating the '2nd generation' of functional size measurement methods. In 2003, the version 2.2 was published as ISO 19761, and the version 3.0 in 2007. The latest version is the v3.0.1 and was published in May 2009.

The FFP method is considered as the version 1.0 of the COSMIC method. All versions of the COSMIC measurement method are available on the COSMIC Group web at www.cosmicon.com.

The COSMIC 'Advanced and Related Topics' document (COSMIC 2007) describes two approaches for approximate sizing (COSMIC 'Advanced and Related Topics', 2007):

- Early sizing: this approach is to be used early in the life of a project, before the Functional Users Requirements (FUR) are detailed and specified.
- Rapid sizing: this approach is to be used when there is not enough time to measure the required software piece using the standard method.

These two approaches can be considered in the early phases of a development project. In both of the approximate sizing approaches, a first task is to identify artifacts of the software piece at some higher level of granularity (the standard level of granularity for the COSMIC

method is the functional process), and to size them using a locally-calibrated scaling factor; these locally-calibrated measures can next be converted to the COSMIC units (i.e. CFP) using a scaling factor. This solution needs an organization history data in order to obtain an adequate scaling factor.

1.9 Why using Fuzzy Logic for Estimation?

Fuzzy logic (FL) is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth - truth values between "completely true" and "completely false". It was introduced by Dr. Lotfi Zadeh of the University of Columbia in Berkeley in the 1960's as a means to model the uncertainty of natural language (Casals, 1997; Zadeh, 1998).

Because of the lack of information in the early phases of software development, most of the data to be collected at estimation time have to be expressed in a subjective way, using qualitative or linguistic variables.

The way these variables are handled next in most of the algorithmic estimation models has been described as a weakness for such models (Idri, 2001, 2002, 2004; Shepperd, 1996).

As mentioned previously, the estimation approach most often used in the industry is the experience-based judgment approach: based on its past experience, the estimator evaluates the variables present in the early phases using linguistic variables and he is capable to analyze qualitatively the relationships between the variables and to assess subjectively the impact in a quantitative way. This process may appear to be simple but, in practice, it is a complex process of human reasoning applied to the estimation of software projects.

The benefits of using fuzzy logic have been demonstrated in a number of other knowledge areas, such as: control, signal decoding, pattern recognition (Ponce, 2010).

The use of fuzzy logic in decision support models involves some elements that can simulate the way in which the humans do their reasoning. If this fuzzy logic approach can be used for generating an estimation model, then the axioms in parametric models proposed by Park (1994) could be modified, or at least redefined.

A number of analogies related to the insights identified by Park (1994) has led to the investigation of the use of fuzzy logic for software estimation purposes:

- All estimations techniques are based on comparisons. The fuzzy logic engine is based on inference rules alike the humans when making their qualitative comparisons.
- Before people can estimate, they must acquire knowledge. An important part in a fuzzy logic model is the "rulebase": this is a set of inference rules that represent the expert knowledge. Some factors that affect duration and effort estimation are described in Morgenshtern (2007).

In the software estimation field, the results generated considering the modeling capabilities comparison of distinct techniques presented by Gray *et al.* (1997) show the appropriateness of each method based on the conceptual requirements of modeling methods as presented in Table 1.1 where each columns represents (as in (Gray, 1997)):

- Model free: Refers to the ability of the modeling technique to determine its own structure, rather than relying on the developer to provide the form of the relationship between inputs and outputs. As an example, when developing a regression model it is necessary to specify which variables should be transformed and what type of transformation should be used. With a neural network, an appropriate approximate transformation will be found by the network when training.
- Can resist outliers: Refers to the model's robustness of estimation when faced with a data set containing outliers.
- Explains output: The capability for a user to see how a model arrived at its conclusions.

- Suits small data sets: One of the major problems in the development of models is the size of the dataset: there is not always enough data. This column refers the model's robustness of estimation when faced with a small data set.
- Can be adjusted for new data: Refers to the issue of whether additional data can be added or whether the entire model must be regenerated on the combined data set must be considered.
- Reasoning process is visible: This can be important for the purpose of verification as well as theory building and gaining and understanding of the process being modeled.
- Suit complex models: Is related to the issue of model-free estimation and the ability to add expert knowledge.
- Include known facts: Refers to the capability to include known information into a model: that is, to initialize a model with known facts (expert knowledge) and then use data to improve and refine it.

With fuzzy logic, only two criteria (criteria: "can resist outliers" and the "can be adjusted for new data") are partially met, while all the other criteria are fully satisfied.

This is an important insight. There are also some other studies in which the use of fuzzy logic offers better reality representation than the traditional techniques and, consequently, more confident estimates, basically because this approach can manage adequately the linguistic variables (Idri, 2002).

Table 1.1 Comparison of estimation techniques in terms of modeling capabilities, Adapted from (Gray, 1997)

Technique	**Model Free**[1]	**Can resist outliers**	**Explains output**	**Suits small data sets**	**Can be adjusted for new data**	**Reasoning process is visible**	**Suit complex models**	**Include known facts**
Least Squares Regression	**N**	**N**	P	**N**	**N**	Y	**N**	P

[1] (Yes = "Y", No= "N", Partially = "P")

Technique	Model Free	Can resist outliers	Explains output	Suits small data sets	Can be adjusted for new data	Reasoning process is visible	Suit complex models	Include known facts
Robust Regression	**N**	Y	P	P	**N**	Y	**N**	P
Neural networks	Y	**N**	**N**	**N**	P	**N**	Y	P
Fuzzy Systems (Adaptive)	Y	P	Y	Y	P	Y	Y	Y
Hybrid Neuro-Fuzzy Systems	Y	P	Y	P	P	P	Y	Y
Rule Based Systems	**N**	N/A	Y	N/A	N/A	Y	Y	Y
Case-Based Reasoning	Y	P	Y	P	Y	P	Y	**N**
Regression Trees	Y	Y	Y	P	Y	P	Y	P
Classification or Decision Tress	Y	Y	Y	P	Y	P	Y	P

Gray *et al.* (1997) describes a basic fuzzy system in the following way: "A fuzzy system as considered here, although as noted above there are different types, is made up of three main components, as illustrated in Figure 1.6:

1. The membership functions represent how much a given numerical value for a particular variable fits the term being considered. In order to do this a fuzzification process is needed, that means a process to convert a crisp value into a membership function value.

2. The rulebase which can be obtained from people with experience in specific problem understandings of the relationships being modeled and refined (or even obtained in the first case) using various data-driven adaptation techniques. The rulebase performs the mapping between the input membership functions and the output membership functions. The greater the input membership degree, the stronger the rule fired by the inference engine, and thus the stronger the pull towards the output membership function.

3. Since several different output memberships could be contained in the consequences of the "if-then" rules fired, a defuzzification process, the third component, is carried out to combine the outputs into a single label or numerical value as required".

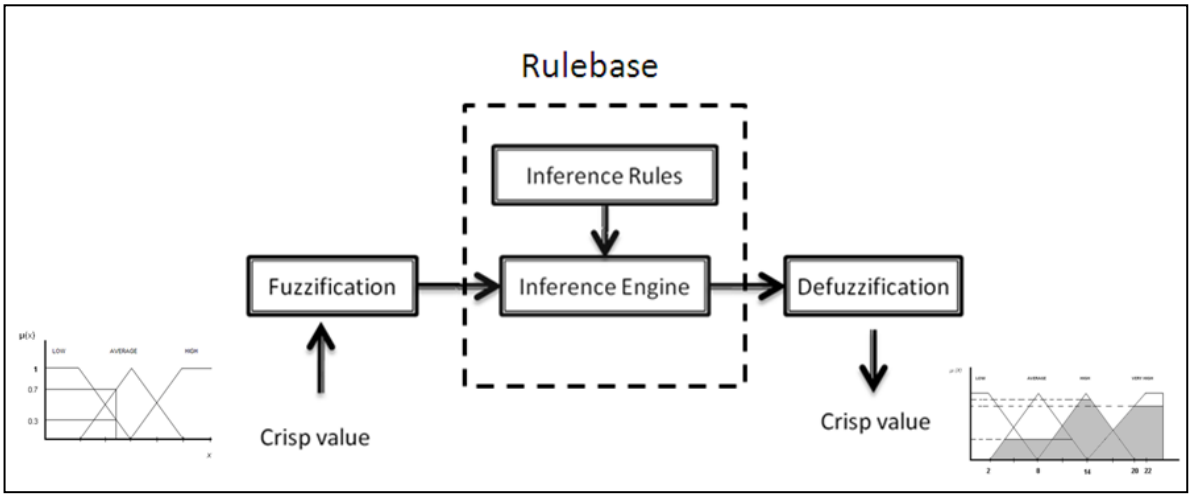


Figure 1.6 Basic fuzzy logic system

"Basically, fuzzy logic is a precise logic of imprecision and approximate reasoning. More specifically, fuzzy logic may be viewed as an attempt at formalization/mechanization of two remarkable human capabilities:

1. The capability to converse, reason and make rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility – in short, in an environment of imperfect information.
2. The capability to perform a wide variety of physical and mental tasks without any measurements and any computations" (Zadeh, 2008).

There are some elements proposed by the fuzzy logic that make it useful in the management of uncertainty and imprecision, such as fuzzy sets theory which is basically a theory of classes with unsharp boundaries. Some features expressed by Zadeh (2008) are considered an extension of the classical set theory. Other elements are the linguistic variables and the 'if –

then' rules: the key idea with these elements is centered in the use of information compression.

The representation of a linguistic variable in a fuzzy logic function is gradual (unsharp) between the boundaries, rather than abrupt and stepwise as in algorithmic models.

The fuzzy set approach deals with linguistic variables or qualitative variables. The qualitative aspect is related to the scale in which the attributes are measured: a classification review in the literature on scale type indicates that there are five types: nominal, ordinal, interval, ratio and absolute (Abran, 2010).

The categorical attributes have a nominal or ordinal scale type:

- The nominal scale type is the lowest scale type level and only allows the classification into categories.
- The ordinal scale type provides additional information to order the categories.

It seems then to be logical that imprecision and uncertainty cannot be avoided early on in an estimation process, so the need of a framework to manage this uncertainty is fundamental to develop an estimation model for its use early in the development phases: that is why the fuzzy logic is selected in this research project in order to measure the independent variables that affect the project result.

In the fuzzy logic (FL) theory some main features are described by Zadeh (2008):

- Linguistic variables and fuzzy if-then rules
- Fuzzy Logic-generalization
- The concept of precisiation and cointension
- Natural Language (NL)-computation, computing with words (CW) and precisiated natural language (PNL)
- Computational theory of perceptions
- Possibility theory

- Computation with imprecise probabilities
- Fuzzy logic as a modeling language.

Zadeh (2008) defines the precisiation as "an operation which transforms an object, p, into another object, p*, which is more precisely defined, in some specified sense, than p". The reverse applies to imprecisiation.

"In the realm of this discourse p is usually a proposition, predicate, question, command or, more generally, a linguistic expression which has a semantic identity, and the need to differentiate between the value precisiation (value precision) and the meaning precisiation (meaning precision). For example:

X=5 value precisiation X= small meaning imprecisiation
X= small meaning imprecisiation X= small (defined by a fuzzy set) meaning precisiation

The fuzzy logic features, and specifically the precisiation, allow managing and supporting the uncertainty associated to the qualitative variables available at the early stages in the software development process: this gives a more realistic model because the constraints have some elasticity and are not precisely defined as usually happened in the quantitative models that are inelastic.

Even though the rationale is that in many cases precision carries a cost, in such cases, deliberate value imprecisiation serves a useful purpose because it provides a way of reducing the mentioned cost.

The precisiation concept offered by fuzzy logic is also important because it is related to the cointension (Zadeh, 2008): that is, a measure of the degree of how the number associated fits to the perception of the concept. Then the cointension of p* in relation to p, C(p*,p) is a qualitative measure of the degree of proximity of the i-meanings of p* and p. p* is cointensive if the degree of proximity is high" (Zadeh, 2008).

1.10 A number of issues in the software estimation process

Why, even after 40 years of research on software estimation, is the estimation approach most often used in industry still based on the experience of the estimators?
If the early phase context of software estimation is analyzed, three basic elements are found:

1. Imprecise or vague information.
2. High uncertainty (the origin is the lack of information).
3. Most of the variables to consider when estimating are linguistic variables.

To tackle these issues, the estimation method up to now capable to manage the uncertainty and linguistic variables has been the 'expert' judgment in experience-based estimation.

A common opinion about software engineering is that it is different from engineering and other sciences because the software products are intellectual products rather than physical objects. However this approach is not well supported. Even in other sciences the measurements are made using models, e.g. a representation of reality, not the physical objects. The problem here is to determine the right model that enables the attribute to be measured. An example of the use of models to measure physical phenomena is the ondulatory model of light: this model is determinant for the measurement of the speed of light (Habra, 2008).

In such a context, any improvement to an estimation technique or a new one that helps to model more adequately the context in which the early estimation in software developments is to be made is therefore welcome in order to improve the decision making process. If this new or improved technique solves some of the problems attached to the use of the experienced judgment, is an improvement.

CHAPITRE 2

RESEARCH OBJECTIVE

2.1 Motivation

There are a number of studies that show that a significant portion of the software projects finish over budget or late over the planned schedule (Standish Group, 2004, 2009):

" …the results show a marked decrease in project success rates, with 32% of all projects succeeding which are delivered on time, on budget, with required features and functions, 44% were challenged which are late, over budget, or with less than the required features and functions and 24% failed which are cancelled prior to completion or delivered and never used" (Standish Group, 2009).

In Table 2.1, it is shown how the benchmarks gathered by the same study have evolved through the years.

Table 2.1 Standish Group benchmarks over the years- Adapted from Laurenz (2010)

Year	Successful Project (%)	Overrunning Projects (%)	Failed (Cancelled) Projects (%)
1994	16	53	31
1996	27	33	40
1998	26	46	28
2000	28	49	23
2004	29	53	18
2006	35	46	19
2009	32	44	24

To address these challenges, an approach often used by organizations in order to improve the outcomes described above, is to adopt an “operational improvement approach”: this means to standardize the process to develop software and to manage IT in general. Some process

improvement models focus on defined standardized processes such as: CMM (Chrissis, 2007), CMMI (Chrissis, 2007), ITIL (Information Technology Infrastructure Library), COBIT (Control Objectives for Information and Related Technology) and MoProSoft (Modelo de Procesos para la Industria del Software). This “operational improvement approach" aims to increase the project’s success percentage by making the process more manageable and, consequently, more predictable.

Morgenshtern *et al.* (2007) suggest that additional project management practices affect the project duration, and are directly related to the project duration estimation errors. The project management practices most often involved are: progress control, updating of work plans and assessing the risks in the projects.

The Standish Group defines a successful project solely by adherence to an initial forecast of cost, time, and functionality (Laurenz, 2010). But what if the estimates are not good enough? Many projects may be considered as failures from a project perspective because their estimation at the early phases in a project is made with a high uncertainty environment, and without a systematic process.

The research motivation of this project is to improve the software estimation process: this is a major challenge for any organization that develops software.

2.2 The research goal and research objectives

The research goal of this thesis is to design of a software estimation process able to manage the lack of detailed and quantitative information embedded in the software development process, and particularly in the early stages of the software development life cycle.

The research strategy selected in this thesis aims to benefit from the advantages of the experience-based approach that can be used in early phases of software estimation while addressing some of the major problems of this estimation approach by experienced judgment.

The research objectives to be met by this improved software estimation process are:

A. The proposed estimation process must use relevant techniques to handle uncertainty and ambiguity in order to consider the way practitioners make their estimates: the proposed estimation process must use the variables that the practitioners use (qualitative) in estimating.
B. The proposed estimation process must be useful in early stages of the software development process.
C. The proposed estimation process needs to preserve the experience or knowledge base for the organization: this implies an easy way to define the experience of the experts.
D. The proposed model must be usable by people with skills distinct than the people who configure the original context of the proposed model.

The objective "A" is related to the management of the qualitative or linguistic variables and the work with uncertainty that can be handled by the experience-based approach.

The objective "B" specifies the moment in which the model must be useful, that is in the early phases.

The objective "C" aims to solve problems related to the estimation expertise which belongs to the expert, and it is hard to assess.

The objective "D" is fundamental because it is the enabler to the systematic replication of the expertise, and without this objective the model would be like the experience-based approach, bounded to the experts' experience.

The constraints that will be addressed in this research are:

1. The information acquired in the early stages when developing a software project is rough, has a lot of imprecision and high uncertainty. This will not change since at these stages the concept of the software to be developed is defined at a very abstract level by the user.

2. The estimation process most frequently used in practice is based on the staff experience; unfortunately, there is not actually a way to replicate systematically this knowledge in order to estimate projects in an organization.
3. Actually there is no estimation process that can tackle the information vagueness of the very early stages such as the feasibility stage.

Even though the proposed model could be taylored for estimating distinct dependent variables (such as duration, effort or cost), the experimental part of this research will focus on estimating project duration for the following reasons:

- The project duration has a strategic value for the organizations.
- It is assumed that in the current high competitive industry the time-to-market is a very important element that drives the software development.
- The algorithmic models actually generated use as a basis the effort in order to estimate duration; however the relation between effort and duration is not linear (Bourque, 2007). This leads to the need to use models based on effort in conjunction with other duration models (Oligny *et al.*, 2000).

2.3 Research approach

To tackle the research goal, objectives and constraints defined for this research, the strategy selected is to design a new software estimation process using fuzzy logic as its basis.

This research work will draw knowledge from four disciplines: statistics, metrology, software engineering and fuzzy logic - see Figure 2.1.

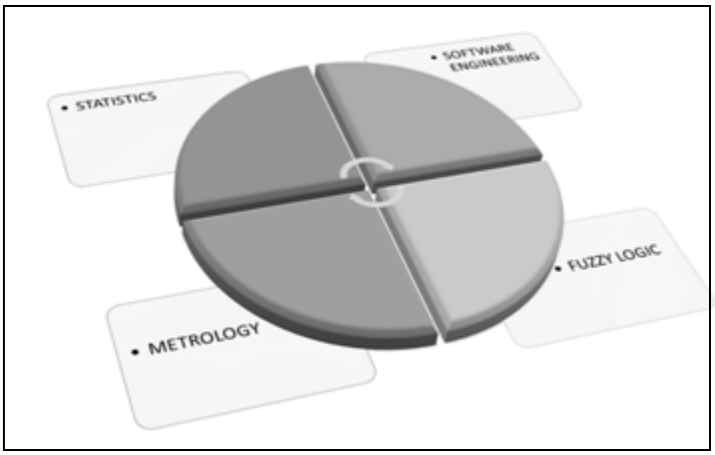


Figure 2.1 Disciplines that support the research

2.4 Statistics

The statistics discipline will be used to demonstrate the quality of the estimation process proposed. Most of the literature related to software estimation uses some quality criteria based on statistics (see Chapter 1). Understanding of statistics is important in order to use appropriately the statistical techniques and tools.

2.5 Metrology

The metrology will contribute to tackle the non uniformity in the distinct units of measurement; the metrology includes rigorous definitions of measurements standards and their management. The metrology has evolved and has been extended over the past century to new technological areas like electricity, photometric and time measurement (Condori-Fernandez, 2008); in these areas, the metrology is recognized as mature and the measurement standards are fully known, thereby enabling their wide use.

This discipline of metrology will be used to take into consideration the generally accepted knowledge in measurement to ensure a sound foundation to the estimation process to be proposed. The basic reference in this discipline is the ISO “International Vocabulary of Basic and General Terms in Metrology” (ISO, 2007).

2.6 Fuzzy logic

The fuzzy logic discipline (Zadeh, 2008; Zadeh et al., 2008) will enable the model and the estimation process to handle the vagueness of the information acquired in early phases in a software development project. It will help to support the uncertainty about the meaning of linguistic values used by the "estimators" when making the estimations.

The fuzzy logic first follows a path in order to convert a crisp value (a number referenced to a context) into fuzzy values (membership values); next these fuzzy values are evaluated with the inference rules defined (i.e. knowledge) using theory rules (for instance: t-norm, t-conorm). The fuzzy values obtained need to be converted back into a crisp value that makes sense and can be used as a basis to take decisions. The fuzzy logic scheme to produce a crisp value was shown in Figure 1.6.

Zadeh (2008) describes some fuzzy logic features, one of which is the fuzzy logic generalization (FL-generalization): this feature is related to "any bivalent-logic-based theory, T, may be FL-generalized, and hence upgraded, through addition to T of concepts and techniques drawn from fuzzy logic.

For the set theory there are three basic operations: union (any valid union operator is known as t-conorm too), intersection (any valid intersection operator is known as t-norm too) and the complement.

If A and B are crisp sets:

$$\begin{aligned} B\text{-}A &= \{x \mid x \in B \text{ and } x \notin A\} \\ \bar{A} &= \{x \mid x \notin A\} \\ A \cup B &= \{x \mid x \in A \text{ or } x \in B\} \\ A \cap B &= \{x \mid x \in A \text{ and } x \in B\} \end{aligned} \qquad (3.1)$$

As for the crisp sets, for the fuzzy sets these are the distinct operators that enable the basic operations described above; the big difference is that for the crisp sets all the possible operators for the same operation lead to the same results. For the fuzzy sets, this does not happen because distinct operators lead to distinct values when their arguments are values between 0 and 1 (Casals, 1997).

This behavior has generated a number of studies to explore which operator to use in the fuzzy sets (Milos, 1999); however, the standard operators proposed by Zadeh (1965, 1988) are used most often. The standard operators are:

$$\begin{aligned} \mu\bar{A}(x) &= 1 - \mu A(x) \\ \mu A \cup B\,(x) &= \mathbf{max}\ \{\, \mu A(x), \mu B\,(x) \,\} \\ \mu A \cap B\,(x) &= \mathbf{min}\ \{\, \mu A(x), \mu B\,(x) \,\} \end{aligned} \qquad (3.2)$$

The use of the maximum as union operator avoids the case in which an element that belongs to AUB has a low membership value of one of the two sets. On the opposite way, the minimum as intersection operator avoids that an element can belong to A∩B with a membership value higher than any of the sets.

The inference rules are defined in the "if -then" form:

$$\begin{aligned} &\text{If A and B, then Z } (A \cap B) \\ &\text{If A or B, then Z } (A \cup B) \end{aligned} \qquad (3.3)$$

Where:

- A is a fuzzy set for one input variable,
- B is a fuzzy set for another input variable, and
- Z is the fuzzy set for the output variable.

2.7 The proposed estimation process based on fuzzy logic

The proposed fuzzy logic estimation process includes six steps - See Figure 2.2:

1. Identification of the input variables,
2. Specification of the output variable,
3. Generation of inference rules,
4. Fuzzification,
5. Inference rules execution, and
6. Defuzzification.

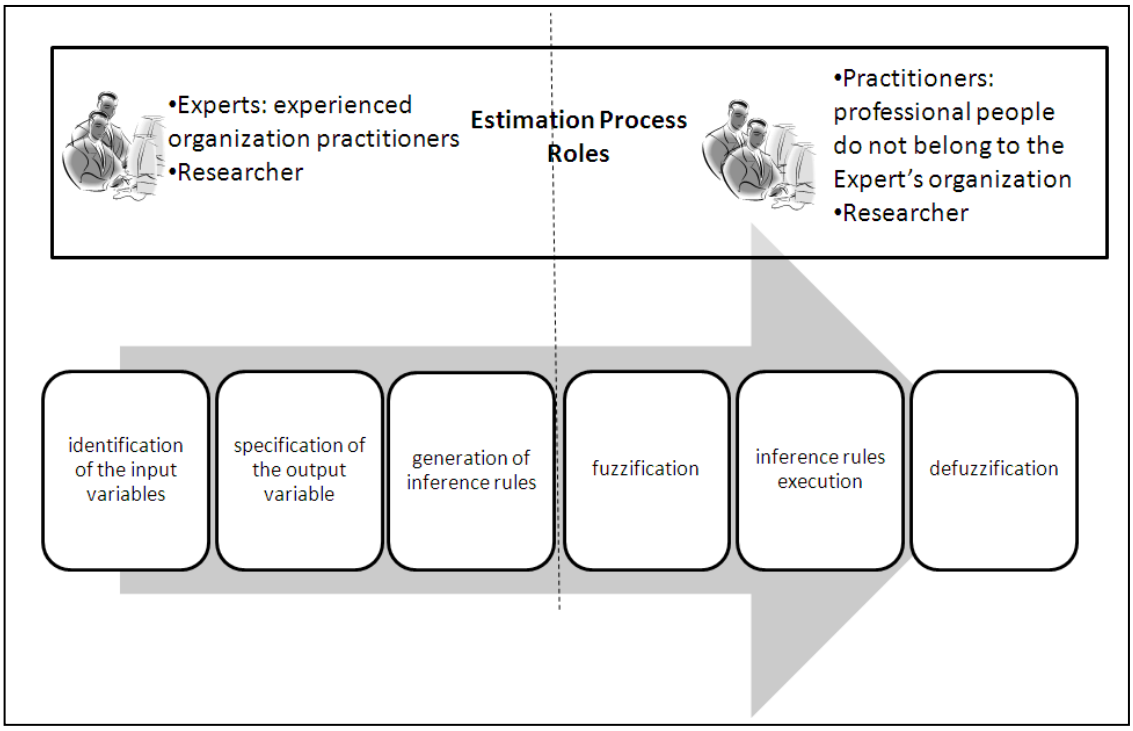


Figure 2.2 The steps in a fuzzy logic estimation process

The first three steps are related to the configuration of the estimation process: this configuration process generates an estimation model or EPCU "context" for estimating a specific project. The last three steps are related to the use of the model generated in order to obtain estimates for a specific project.

2.8 Step 1: Identification of the Input Variables

The goal of step 1 is to get the experienced "experts" (practitioners in a software development for a specific organization) to identify and assess the most significant input variables for a project or kind of projects, such as: software size, software complexity, team skills, knowledge of the software development process or its implementation phase, the leader's skills, the customer or provider organization's environment, knowledge of the tools to be developed in the project, customer commitment, the stakeholders involved, and so on.

In this step, the experts must define next the fuzzy sets for each of the input variables they selected. This means that they must agree on a classification scheme for each variable, typically in terms of linguistic values (and only the linguistic categories that make sense to them in practice). For example, for the input variable complexity, its fuzzy set could be defined as a classification such as: low, average or high (i.e. linguistic values). It is possible to define more linguistic values but it is not necessarily useful in some contexts because the differences between each one could be too fine grained for a context (for example: very low, quite low and low).

Also required is the definition of membership function domain(s) to represent the opinions of the experienced practitioners about the input variables for a specific project to be estimated (meaning precisiation).

2.9 Step 2: Specification of the Output Variable

The previous step 1 is repeated for the selected output variable, for example project duration. A classification for the output has also to be defined in a fuzzy set that represents it.

2.10 Step 3: Generation of the Inference Rules

In step 3, all the fuzzy sets belonging to each input variable must be combined in 'if…, then…' form:

$$\text{If } x \text{ and } y, \text{ then } z$$
$$\text{If } x \text{ or } y, \text{ then } z; \qquad (3.4)$$

where x is a fuzzy set for one input variable, y is a fuzzy set for another input variable and z is the fuzzy set for the output variable. All the fuzzy sets for each input variable must be combined to generate the rulebase.

2.11 Step 4: Fuzzification

The goal of step 4 is to obtain fuzzified values as a consequence of opinions about those values put forward by an experienced practitioner. With the membership function defined for all the input variables, a value assignment that represents an opinion from the people needs to be requested for each variable. This will create fuzzy values to be used in the next step to execute the rulebase.

2.12 Step 5: Inference Rule Execution

The fifth step consists of executing the rulebase by substituting the fuzzy values obtained in the previous step. The Inference Rule execution must follow the rules of fuzzy logic (Zadeh operator), such as:

$$\text{Value (P or Q)} = \max \{\text{value (P), value(Q)}\}$$
$$\text{Value (P and Q)} = \min \{\text{value (P), value(Q) }\} \qquad (3.5)$$

2.13 Step 5: Defuzzification

The defuzzification in step 5 is developed in order to obtain a crisp value for the final estimate. Examples of such defuzzification methods are: Max-Min, Max-Dot, Max-Product, Centroid Average, and Root Sum Square (RSS).

There are five (5) defuzzification methods referenced by Wong (1995), however the centroid average or center of gravity provides a better solution than other methods (Zadeh, 2008):

1. Centroid average or Center of gravity
2. Maximum center average
3. Mean of maximun
4. Smallest of maximum
5. Largest of maximum

CHAPITRE 3

RESEARCH METHODOLOGY

The research methodology proposed to reach the research objectives includes five phases – see Figure 3.1. The first phase "Literature Review" (see Chapter 1) has collected and analyzed the necessary information about software measurement and software estimation.

The literature review of the past 40 years has focused to acquire the necessary information on how to evaluate and develop software estimation models, the quality criteria to evaluate the estimation models and some other important issues related to this research.

The other purpose for this literature review phase was to acquire information about the disciplines needed for this research, such as statistics, metrology, fuzzy logic and software engineering.

The second phase of the research methodology is the "Building of the Fuzzy Logic Model". The estimation model to be built will be referred to as the 'Estimation of Projects in Context of Uncertainty' - the EPCU model - and will focus on defining the Software Estimation Process that can provide information to the decision makers, including information about the quality and the confidence of the model.

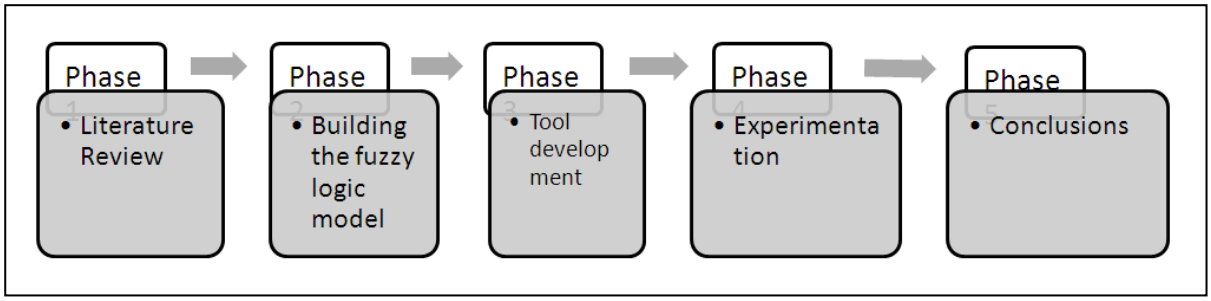


Figure 3.1. Methodology Research Phases

It will be fundamental that the proposed model handle the qualitative or linguistic variables in a formal way. That is why the mathematical framework selected to develop the model is the fuzzy logic.

The third phase of this research, "Tool development", will focus on the design and development of a software prototype tool to help and simplify the storage and experimentation process. This decision was made because processing manually the information needed for the experimentation would have been a very time consuming task.

The prototype tool will initially aim at statistical analyses; these kinds of studies will be made with the statistical tool SPSS v17 or Excel in some cases. The definition and the detailed design of the prototype tool will be described in chapter 5.

The fourth phase of this research project, "Experimentation", will focus on determining if the model developed has reached the objectives and contributes positively to the goal of the research.

It will consist in a number of experiments to test the estimation model developed.

Each of the experiments will need as inputs:

1. The estimation model proposed,
2. The tool in order to facilitate the experimentation,
3. Information about projects, including information about each project at its inception, as well as information about these projects once completed (if it is available).

To evaluate the performance of an estimation model, the following set of information is needed:

- The set of information available at the very early stages of the development process.
- The set of information available for the same projects once completed.

Usually the performance of the estimation models are evaluated with finished projects, so there is a need of finished projects to evaluate the performance model.

Considering these issues the experimentation will be designed to test the model in a context similar to the context of the early phases. The experimentation proposed for this research will:

1. Use the model with a set of industry projects that were already completed and for which the necessary information was available both at their inception as well as once completed. A part of this experiment will consist in collecting a set of completed projects that had been estimated using an experience-based approach in the Mexican software industry and in conjunction with the people who had participated in the original estimation. A fuzzy logic-based estimation model will be generated and used to estimate the finished projects.
2. Use this set of completed projects to simulate an early estimation. In this "*a priori"* context, the participants will be provided with the description of the software requirements for a set of projects as they were described in the early project phases. For this experiment, it requires that a re-documentation be done by the researcher of the very early drafts of the preliminary statement of the scope of all the software to be developed. This re-documentation of the early software requirements will be based on the availability of project documentation in each participating organization and in the experts' memories. The re-documentation has to be performed at a very high level of abstraction, as is typically done by users at the conceptualization or feasibility stage in a software development process.
3. Use the fuzzy logic-based estimation model in order to estimate some projects in the real early phases. This experiment will be made in order to analyze how the model must be used in early phases situations.

The results of these experiments will be analyzed to verify if the proposed fuzzy logic-based estimation process covers the other research objectives stated, including if could be usable by

people with skills distinct than the people who configure the original context of the proposed model and the comparison of the model performance against the experience-based approach.

The last phase of the research is the preparation of conclusions and the identification of further work for future improvements to the estimation process proposed in this research.

CHAPITRE 4

THE DESIGN OF THE EPCU MODEL

4.1 Introduction

Considering the estimation in the early phases of software development and the defined environment it is possible to identify the set of concepts of the fuzzy logic-based EPCU estimation model – see Figure 4.1:

> A project is influenced by many parameters (independent variables) at the same time, their impact being distinct from each other: some have a major impact in a specific project, while others might be almost irrelevant for this same project. This may be dependent on their magnitude and the relation between each other, leading to distinct project performance (dependent variables: time, cost, effort).

If there was a way to measure exactly each input variables and to determine precisely their impact and the relation to each other variables, the estimation process would be simpler.

However, it is known that in the very early phase of a software related project not all the parameters can be determined and measured exactly: most of them are qualitative and the uncertainty is very high because the definition of the project (i.e. the information available at that time) is still at a very high abstraction level (McConnell, 2006).

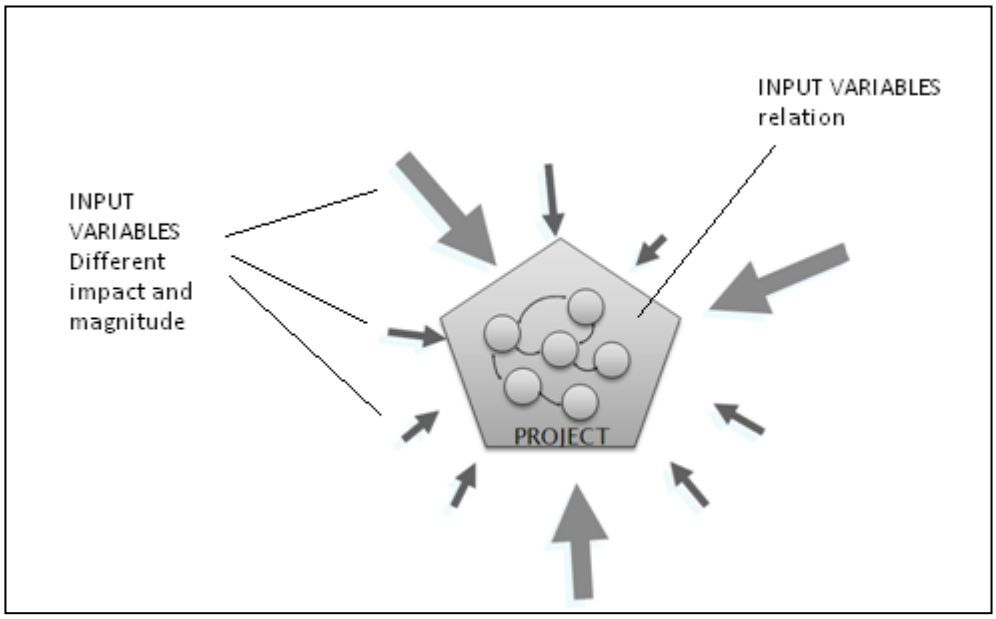


Figure 4.1 The set of concepts for the EPCU model

The measurement of the inputs is a fundamental issue in the estimation process - see Figure 4.1. No matter if the inputs are quantitative or descriptive (Abran, 2008), these inputs need to be reliable in order to lead a good estimation result.

Considering the importance of the reliability of the input variables, there is a need to manipulate the input variables (linguistic) over a formal framework: this is one of the reasons why the fuzzy logic theory has been selected.

At the beginning of a software project, there is, on the one hand, a preliminary scope statement, or early requirements, and different options to develop it - see Figure 4.2. These options represent distinct software development processes which may come from different software providers. At the end of the project no matter how the software has been developed, the functionality must be delivered: some of the differences in project effort may come from some providers being cheaper than the others, or with more or less quality or with longer project schedule. The drivers that determine this are the parameters (input variables) and its influence for a specific context.

On the other hand, for each software provider, the size of the same set of requirements could be assessed qualitatively, for estimation purposes, as large by one provider, or average by another software provider: so there it is a need to consider the output variable in reference to a specific context too.

Depending on the context in which a specific set of functions is developed, the dependent variables (such as: costs, effort, duration, quality and so on) may vary considerably - See Figure 4.2.

The EPCU “context” is therefore defined as:
"a set of variables (inputs and output) and the relations that affect a specific project or a set of similar projects".

This definition is important because in the use of the EPCU model the practitioners will provide in input their opinions about the context of the project, rather than an estimate of the output variable using an intuitive approach - see Figure 4.2.

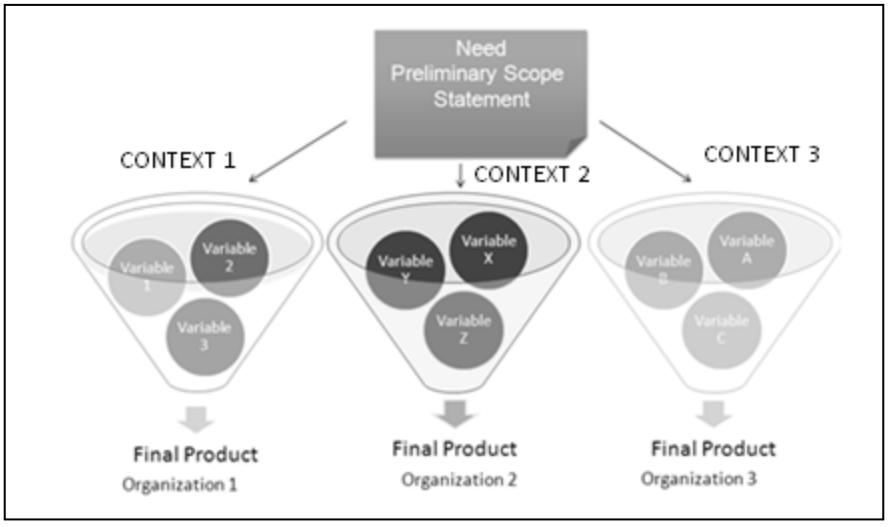


Figure 4.2 Distinct contexts for the same set of requirements

4.2 Description of the EPCU Process

The process designed in this research is referred to as an Estimation of Projects in a Context of Uncertainty (EPCU) and is designed using six process steps:

1. Identification of the input variables
2. Specification of the output variable
3. Generation of the inference rules
4. Fuzzification
5. Inference rule execution
6. Defuzzification.

Steps 1 to 3 are related to the configuration of the inputs to the EPCU model and steps 4 to 6 are related to the use of the model once it has been defined. An overview of these six steps required for the fuzzy logic estimation process are described next in more details.

4.3 Step 1: Identification of the input variables

The purpose of this step is to elicit the most significant input variables for a project (or a kind of projects) from the experienced practitioners in an organization (independent variables like: software size, software complexity, team skills, and so on) (See Chapter 6 for examples of its usage).

It is natural for the practitioners (and even experts) to differ in their opinions of some variables. To deal with this, fuzzy logic is used in a step known as fuzzification, which is described in step 4.

In this first step experienced practitioners must define the fuzzy sets for each variable selected for the model, which means that they must classify the variables in terms of linguistic values which they can evaluate on the basis of their own experience.

Step 1 also requires to define the membership function domain to represent the opinions of the experienced practitioners about these fuzzy sets for each input variables. By the end of this step, the membership functions that represent the behaviors of the variables in terms of the fuzzy sets are defined (See Figure 4.3).

4.4 Step 2: Specification of the output variable

For step 2, the objective is to define the fuzzy sets for output variable and the membership function domain to represent the opinions of the experienced practitioners about the fuzzy sets defined for the output variable.

It is recommended that the membership functions for the input or output variables be normalized: this implies that the value range in the "*y*" axis should always between 0 and 1 - see Figure 4.3.

4.5 Step 3: Generation of the Inference Rules

All the fuzzy sets belonging to each input variable must be combined into 'if…, then…' form:

If *x* and *y*, then *z*
If *x* or *y*, then *z* (4.1)

Where:

- *x* is a fuzzy set for one input variable,
- *y* is a fuzzy set for another input variable, and
- *z* is the fuzzy set for the output variable, resulting from the *x* and *y* relationship.

All the fuzzy sets for each input variable must be combined to generate the rulebase.

4.6 Step 4: Fuzzification

Once the membership function is defined for all the input and output variables, and the relations between them are stated by the experienced practitioners, a practitioner opinion needs to be requested for each input variable for a specific project to be estimated. The goal of this step is to obtain fuzzified values as a consequence of opinions put forward by an experienced practitioner for each of the input variables.

This means that a membership function must be evaluated with the values provided as inputs by the people that need to estimate the project (known as "practitioner" in the experimentation - Chapter 6). If the membership function defined for each input variable is using three fuzzy sets, the fuzzification process can look like Figure 4.3.

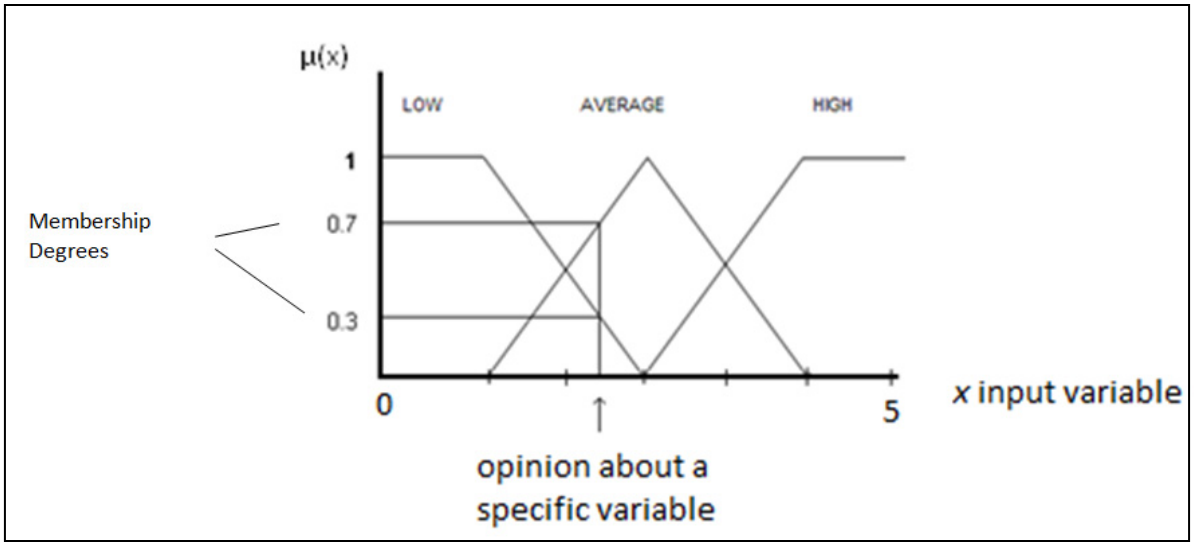


Figure 4.3 Example of a fuzzy membership function and defuzzification

In Figure 4.3, a crips value is assigned to an input variable (horizontal axis) by a practitioner in a reference (between 0 and 5) considering his own experience; this crisp value is transformed next in membership degrees using the membership functions for the fuzzy sets defined for a specific input variable.

The relation between the practitioner opinion and the fuzzified values is denoted by a function known as membership function $\mu(x)$ (Figure 4.3).

The input variables used are a linguistic value (low, average or high), the range of the possible values or the domain function (x) is [0, 5]. If the variable is quantitative the range can be defined by the historic values: for example if the numbers of programmers for a typical project in a specific environment is from 2 to 4, the domain function is [2, 4]. For most of the experiments qualitative variables will be used, and the [0, 5] reference values will be stated.

Because the membership function is normalized for all the membership functions, the range for the possible values for the $\mu(x)$ is [0, 1].

4.7 Step 5: Inference Rule Execution

This fifth step consists in executing the rulebase by substituting the fuzzy values obtained for each input variable fuzzy set. The execution of the rulebase is made following the rules defined by the fuzzy logic theory, such as:

$$\text{Value (P or Q)} = \max \{\text{value (P), value(Q)}\}$$
$$\text{Value (P and Q)} = \min \{\text{value (P), value(Q) }\} \qquad (4.2)$$

4.8 Step 6: Defuzzification

The defuzzification step aims to convert the fuzzy values related to the distinct fuzzy sets that describe the behavior of the output variable into a crisp value that represents a valid value for the output variable.

Intuitively this is the inverse process to the fuzzification operation: the scale types used are the same. However, the output variable could have distinct units because it is the dependent variable (for instance, the units can be defined as calendar months for the duration as the output variable).

The output variable is defined as a membership function with several fuzzy sets, all of them in the function domain defined for each project considering the organization history and the expert knowledge. An example of the output variable membership function is shown in Figure 4.4.

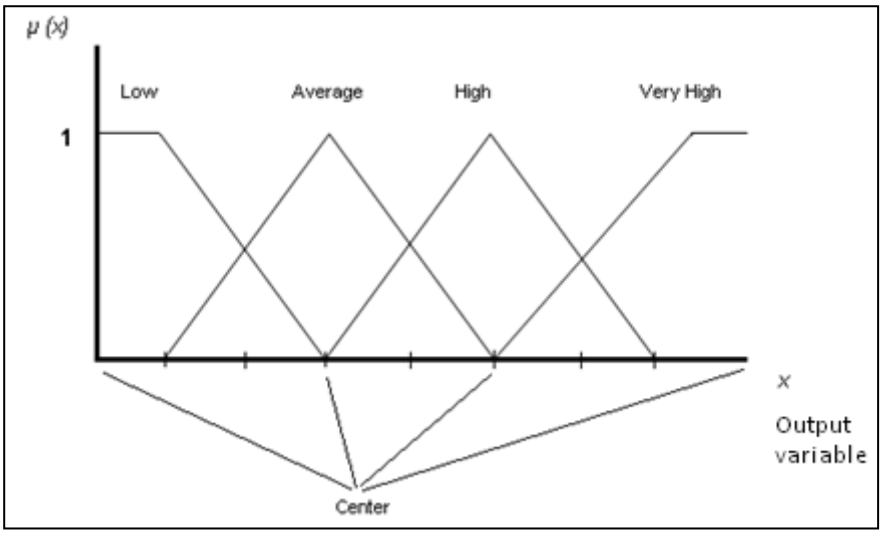


Figure 4.4 Output variable membership function

In Figure 4.5, the shaded area means all the possible values that can take the output variable, considering all these areas and using the algorithm defined a crisp value is calculated.

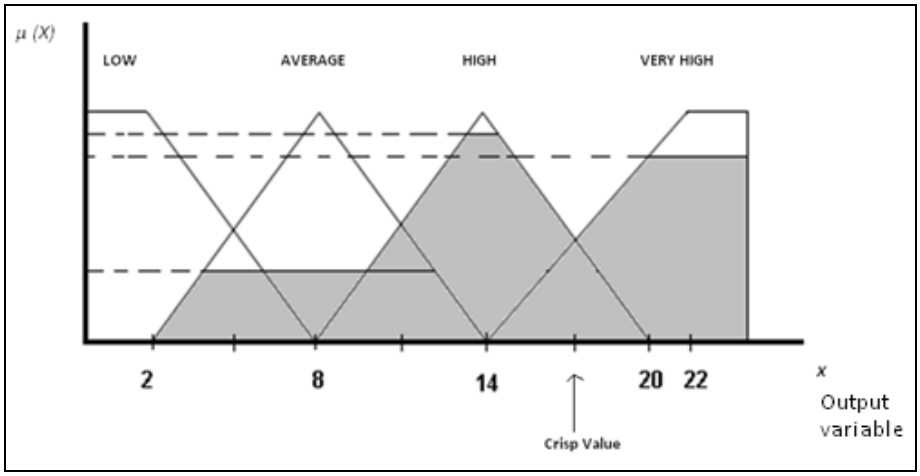


Figure 4.5 Example of a fuzzy membership function and defuzzification

The EPCU estimation generated in the experiments (presented in Chapter 6) will be obtained using RSS and then by computing the 'fuzzy centroid' of the area.

This defuzzication method was selected in this research project because when using the RSS all the "fired" rules from the rulebase are considered for each fuzzy set defined for the output variable, not the maximum or the minimum value: this selection gives the best weighted influence to all the inference rules involved.

After the use of the RSS, the computation of the fuzzy centroid of the composite area generated is calculated. Even though it is more complex mathematically than the other defuzzication methods, it provides better solutions than other methods (Zadeh *et al.,* 2008)

The two sub-steps for obtaining the crisp value are:

1. Obtain the strength for each fuzzy set belonging to the output membership function (RSS).
2. Obtain the fuzzy centroid of the area.

4.9 Sub-step 6.1. Obtain the strength for each fuzzy set belonging to the output membership function (RSS)

Considering the values obtained in the Inference Rule execution – step 5, the strength for each fuzzy set defined for the output variable is calculated with the following formula:

$$FS_k = \sqrt{\Sigma_i R_i^{\,2}} \qquad (4.3)$$

where FS_k , is the fuzzy set defined by a same linguistic value.

R_i is the rule that fired a specific fuzzy set.

4.10 Sub-step 6.2. Obtain the fuzzy centroid of the area

The weighted strengths of each output member function are multiplied by their respective output membership function center points and summed up. The area obtained is divided by

the sum of the weighted member function strengths, and the result is taken as the crisp output.

$$\text{Crisp Value } (FS_k) = \text{Centroid} = \frac{\Sigma\ (\text{“}FS_k\text{”}\ \text{center} * \text{“}FS_k\text{”}\ \_\text{strength}\ )}{\Sigma\ (\text{“}FS_k\text{”}\ \_\text{strength})} \qquad (4.4)$$

where F_{Sk} , is the fuzzy set defined by the same linguistic values.

4.11 Overview of the roles and responsibilities in the EPCU model

To clarify the two set of roles involved in A) the configuration and B) usage of the EPCU model, an overview is presented in Table 4.1. The left-most column presents the phase (A) configuration and (B), the usage; the second column to the right, the steps from 1 to 6; the third column, the Role 1 of those configuring and using the EPCU model; the right-most column, the role of the researcher in Role 2.

Table 4.1 Overview of the roles involved in the configuration of the EPCU model

Phases	Step	Role 1	Role 2
A) Configuration of the EPCU model:generation of a specific EPCU context	1. Identification of the input variables	Experienced practitioner who defines, using his own experience, the input variables that have most influence on the output variable (i.e. project duration).	Researcher (support): Participates as a guide for the definition provided by the organization's experienced practitioner
	2. Specification of the output variable	Experienced practitioner who defines, using his own experience, the range (function domain) for the output variable (i.e. the possible "time frame" – duration - for the project).	Researcher (support): Participates as a guide for the definition provided by the organization's experienced practitioner.
	3. Generation of the inference rules	Experienced practitioner who defines, using his own experience, the relations of the input variables with the output variable in rules of the "if-then" form.	Researcher (support): Participates as a guide for the definition provided by the organization's experienced practitioner.

Phases	Step	Role 1	Role 2
B) Usage of the EPCU model through the use of the EPCU context defined	4. Fuzzification	The practitioner, who must estimate a project, assigns a value for each of the input variables defined, considering the project to be estimated.	The researcher implements the fuzzification – once in the structure of the EPCU model.
	5. Inference rule execution		The researcher implements the inference rule execution – once in the structure of the EPCU model
	6.Defuzzification		The researcher implements the defuzzification – once in the structure of the EPCU model

In summary, the researcher defines and implements the 'shell' of the estimation prototype (see the right-most column in Table 4.1), while:

1. the experience practitioner(s) selects the input variables for a generic context of estimation, and their expected ranges of variation for each type of input variables;
2. the experience practitioner(s) select the output variable for a generic context of estimation, and its expected range of variation;
3. the experience practitioner(s) define the inference rules that relate the input variables to the output variable;
4. any practitioner who needs to do an estimation for a similar project within a similar context can next choose the specific values for the input variables defined; the EPCU model gives them next an estimate using the inference rules defined in the previous steps by the experience practitioner(s).

4.12 Analysis of the measurement scale types within the EPCU model

Dickes (1994) referenced in Abran (2010) mentions that for some attributes, like the distance between two points, the measurement rules are quite simple. For others, such as for abstract attributes, it is more complicated. In these cases, the definition is made by stating explicitly how the concept is decomposed into sub-concepts.

To analyze the scale type of the measurement steps in the EPCU model, the focus of this analysis must be on the following three sub-concepts (See Figure 1.6):

- Fuzzification
- Rules execution
- Defuzzification

A summary of the mathematical operations valid for the distinct scales types is presented in Table 4.2 – see also (Abran, 2010).

Table 4.2 Scale types operations, with permission (Abran, 2010)

Scale type		Admissible Transformation	Operations	Examples
Nominal	(R,=)	f unique	name, distinguish	colors, Shapes
Ordinal	(R,>=)	f strictly increasing monotonic function	rank, Order	preference, hardness
Interval	(R,>=,+)	f(x)= ax + b, a>0	Add	calendar time, temperature Celsius
Ratio	(R,>=,+)	f(x)= ax, a>0	add, multiply, Divide	mass, distance, absolute temperature (degrees Kelvin)
Absolute	(R,>=,+)	f(x)= x	add, multiply, Divide	entities count

4.13 Fuzzification (i.e. step 4)

In Figure 4.3, the possible values of x and $\mu(x)$ are on a ratio scale type: this scale type allows to build ratios among ratio scale variables. Physical measurements of height, weight, length are typically ratio variables. It is meaningful to say for example that 10 meters is twice as long as 5 meters. This is because there is a natural zero.

For the scale type analysis for step 4, it is possible to consider for the x axis a ratio scale type; in order to extrapolate the concept, the unit of x axis is defined as a "generic unit" [gu] because there is not a specific unit for the qualitative variables from which the practitioner assigns a value that represents their opinion about the specific variable. The practitioner

expresses his opinion in a specific range that has a natural zero. For the y axis the units are the "membership value" [mv] obtained by the relation denoted by $\mu(x)$.

Considering the scales types for each axis and the units stated in the previous paragraph, the fuzzication scale type analysis is shown in Table 4.3.

Table 4.3 Fuzzification scale type analysis

Object	Operation	Scale Type (From)	Scale Type (To)	Mathematical Validity	Transformation to other Scale type
Membership function	Fuzzification	Ratio	Ratio	Yes	No

In the fuzzification operation, the main object is the membership function (input/output), as can be seen in Figure 4.3; this operation requires a value in the x axis, $x \varepsilon R$. This value is denoted as a ratio scale type with a unit [gu].

With this value a function execution is made. $\mu(x)$: this takes the x input value (ratio scale) and converts it to the membership function domain (a ratio scale in an specific range [0,1]). Then, $y \varepsilon R$, and the unit is [mv]. Considering the scale types in this function evaluation, an original ratio scale value is converted to another ratio scale value, so this is mathematically valid: $x, y \varepsilon R$.

4.14 Inference Rule execution (i.e. step 5)

As mentioned in the fuzzification analysis, for the y axis the units are the "membership value" [mv] obtained by the relation denoted by $\mu(x)$. In this sub-concept analyzed, the rulebase execution is the application to the basic operators to each defined rule; the result will be a membership value assigned to a fuzzy set of the output variable - this result will have [mv] units too.

The scale types used are ratio scale types for the A fuzzy set, B fuzzy set and for the Z fuzzy set; this represents a membership value in each of the fuzzy sets.

Considering the scales types and the units stated, the analysis is shown in Table 4.4.

Table 4.4 Rulebase execution scale type analysis

Object	Operation	Scale Type (From)	Scale Type (To)	Mathematical Validity	Transformation
Basic operator	Rulebase evaluation	Ratio	Ratio	Yes	No

In the rules execution operation, there are membership values from the input variables and a basic operator defined by the fuzzy logic. As it was mentioned, the membership value has a ratio scale type with a unit [mv].

When applying the rules defined by the fuzzy logic, the value obtained is dependent on the operator, the maximum (or) or the minimum (and) membership value; so the original scale type is a ratio scale type with a [mv] and the final scale type is a ratio scale type with a [mv] too.

This operation, considering the scale types, is mathematically valid.

4.15 Defuzzification (step 6)

In the defuzzification step the goal is to move from a membership value to a crisp value: it is the opposite process to the fuzzification. As seen previously, the *x* axis and the *y* axis are ratio scale types with units defined as [gu] and [mv] respectively. The defuzzification procedures used consist in two sub-steps that will be analyzed individually.

4.16 Sub-step 6.1. Obtain the strength for each fuzzy set belonging to the output membership function (RSS).

Considering the values obtained in the inference rule execution step, the strength for each fuzzy set defined for the output variable is obtained with the following formula:

$$FS_k = \sqrt{\Sigma R_i^{\,2}} \qquad (4.5)$$

Where:

- FS_k is the fuzzy set defined by a same linguistic value.
- R_i is the rule that fired a specific fuzzy set.

For this sub-step the scale type for each R_i is a ratio scale type with unit [mv]; this implies that the FS_k will have a ratio scale type with a unit [mv], because the units after adding $[mv]^2 + [mv]^2 = [mv]^2$ and $([mv]^2)^{½} \Rightarrow FS_k = [mv]$.

Considering the scales types and the units stated, the analysis results are shown in Table 4.5.

Table 4.5 Strength for each fuzzy set belonging to the output membership function (RSS) scale type analysis

Object	Operation	Scale Type (From)	Scale Type (To)	Mathematical Validity	Transformation
Strength for each fuzzy	RSS	Ratio	Ratio	Yes	No

4.17 Sub-step 6.2. Obtain the fuzzy centroid of the area.

The weighted strengths of each output member function are multiplied by their respective output membership function center points and summed up. The area obtained is divided by the sum of the weighted member function strengths, and the result is taken as the crisp output.

It was stated that the FS_k or "FS_k" _strength is a ratio scale type with a membership value [mv] unit. The center is the x value for which the $\mu(x) = 1$ in a triangle fuzzy set and the minimum value for the lowest fuzzy set and the maximum value for the highest fuzzy set.

$$\text{Crisp Value } (FS_k) = \text{Centroid} = \frac{\Sigma \text{ ("}FS_k\text{" center} * \text{"}FS_k\text{" \_strength)}}{\Sigma \text{ ("}FS_k\text{" \_strength)}} \qquad (4.6)$$

Where FS_k is the fuzzy set defined by the same linguistic value.

Considering the description above, the "FS_k" _strength is a ratio scale type and the "FS_k" center is a ratio scale type; this allows the operations between the scale types with [mv] unit and [gu] unit respectively, in which the [gu] usually can be defined in terms of months or weeks if the output variable selected in, for example, project duration.

Considering the formula to obtain the crisp value in (4.7) and using the units for the fuzzy sets defined, the scale types and the units analysis is made.

crisp value (FS_k) = ("$FS_{k\ 1}$" center $_{k1}$ [gu] * "$FS_{k\ 1}$" _strength $_{k1}$ [mv] + "$FS_{k\ 2}$" center $_{k2}$ [gu] * "$FS_{k\ 2}$" _strength $_{k2}$ [mv] +" $FS_{k\ 3}$" center $_{k3}$ [gu] * "$FS_{k\ 3}$" _strength $_{k3}$ [mv] +" $FS_{k\ 4}$" center $_{k4}$ [gu] * "$FS_{k\ 4}$" _strength $_{k4}$ [mv]) / ("$FS_{k\ 1}$" _strength $_{k1}$ [mv] +" $FS_{k\ 2}$" _strength $_{k2}$ [mv] +" $FS_{k\ 3}$" _strength $_{k3}$ [mv] + "$FS_{k\ 4}$" _strength $_{k4}$ [mv])

(4.7)

crisp value (FS_k) = ([gu][mv] + [gu][mv] + [gu][mv] + [gu][mv]) / ([mv]+ [mv]+ [mv]+ [mv]) (4.8)

crisp value (FS_k) = ([gu]~~[mv]~~) / ([~~mv~~]) = [gu] (4.9)

That can be months or weeks as mentioned before.

In Table 4.6 the results of the scale types analysis are presented.

Table 4.6 Centroid scale type analysis

Object	Operation	Scale Type (From)	Scale Type (To)	Mathematical Validity	Transformation
Crisp Value	Centroid	Ratio	Ratio	Yes	No

As shown in this chapter, all the operations performed over the objects in the model are mathematically valid and the management of the units related to each scale type in the model is consistent.

4.18 Summary

In the literature reviewed (Charette, 2005; Lavagnon, 2009; Shore, 2008; Pinto,1988; Fincham, 2002; Boehm, 1981; Weinberg, 1985; Jensen, 1979; Stamey, 2006; Timothy, 2006; Jones, 2004, 2005, 2006; Dekkers, 2005), when attempting to synthesize some insights into the variables most often identified as successful or failure drivers in the software projects, a lack of consensus on the root causes for the project successes or failures was observed: it is recognized that there are several project variables, some of which can be more significant than others.

The concept of the model (Figure 4.1) mentions that in the early phases of the software project development - and in a specific environment - each project is influenced by many parameters (independent variables) at the same time, their impact being distinct from each other: some have a major impact in a specific project, while others might be almost irrelevant. This is a consequence of their magnitude and the relation between each other in generating a project result (dependent variable time: duration, cost, effort).

The proposed fuzzy logic-based EPCU estimation process is a generic model: considering this and the lack of consensus about the main variables that impact specific projects, it will depend on the users of the EPCU model to introduce into the EPCU shell the variables that better reflect the context for a specific project or a set of projects in order to define an EPCU "context" to get an estimate.

With the scale analysis, the conformity to the mathematic principles was validated, in order to state a solid base for the model.

CHAPITRE 5

DESIGN AND DEVELOPMENT OF A SOFTWARE PROTOTYPE FOR THE EPCU ESTIMATION PROCESS

5.1 Introduction

To handle the amount of calculations required by the model EPCU for all projects, as well as for the recording and the management of the information generated by the EPCU model, including estimated and actual values of the projects, a software prototype tool was designed and developed to perform the necessary calculations quickly and to support the EPCU estimation process.

The prototype tool was designed initially as a set of three modules:

1. Catalogs
2. Project Information
3. Model EPCU

Two additional modules were added later to the prototype for:

1. Project Portfolio Management
2. Reports

Figure 5.1 shows with a Unified Modeling Language (UML) model the packages representing the software modules of the prototype designed and developed.

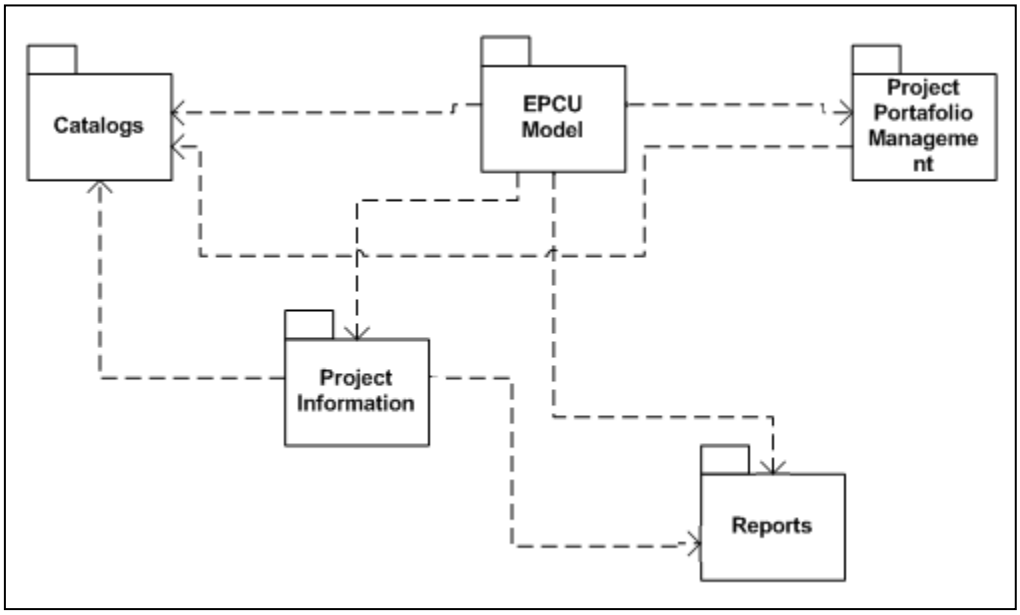


Figure 5.1 Prototype modules for the use of the EPCU estimation process

The module "EPCU Model" depends for its operation on the module "Catalogs"; similarly for the modules "Project Information" and "Project Portfolio Management". The latter bases its operation on the module "EPCU Model". The project portfolio management module uses the EPCU model to implement a portfolio approach to software projects estimation (See Chapter 7).

Finally, the module "Reports" uses the information generated by the modules "Project Information" and "EPCU Model" for its operation.

5.2 The functionality for each module

Each module has a specific objective to facilitate the configuration and usage of the EPCU model to make estimates using the EPCU model proposed in this research. The generic functionality of each module is described next, together with its use case diagram.

5.3 Module: Catalogs

The Catalog module maintains the various general information catalogs in the prototype, such as those relating to the classification of the operation units where the projects are undertaken (Area, Sub Area, Office), generated contexts, and so on.

It also manages the catalog of scenarios: in this catalog, the information for different values obtained in the estimation using the EPCU model can be displayed and can be manipulated. The basic functionality of the module is (Figure 5.2): adding new records to a specific catalog, deleting them and updating the records for each of the catalogs that it handles.

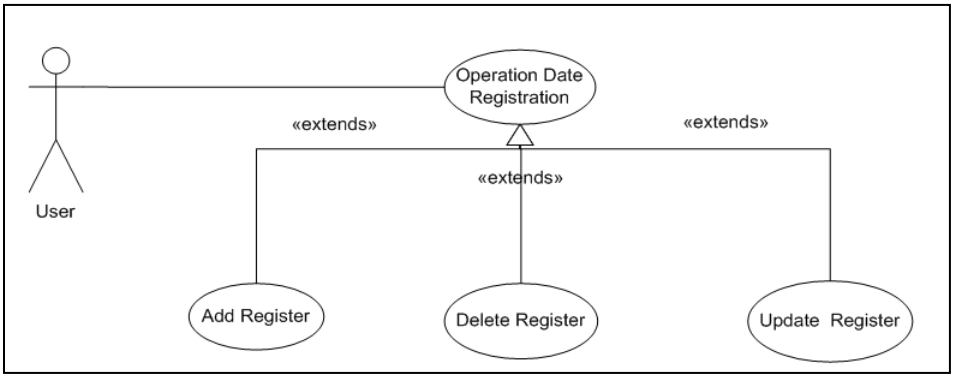


Figure 5.2 Use case diagram of the Catalogs module

5.4 Module: Project Information

The Project Information module records the project information such as: name or location within the company, duration and actual cost. Basically it is the basis to generate reports filtering out the information about the projects registered in the prototype tool.

The overall functionality of this module is shown in the use case diagram in Figure 5.3.

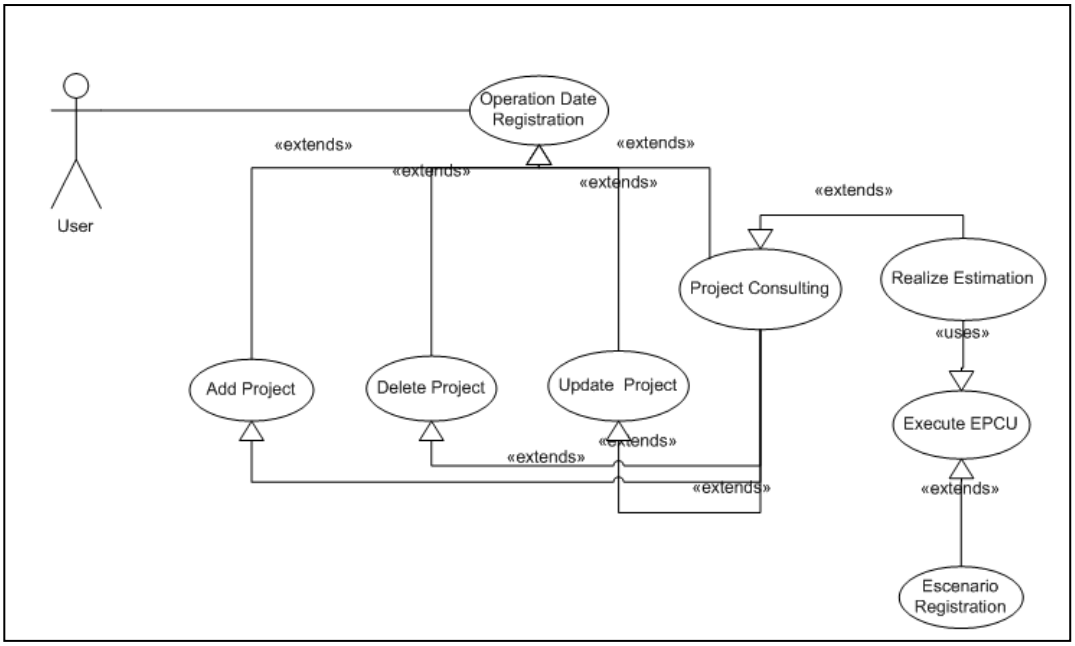


Figure 5.3 Use case diagram of the Project Information module

This module allows registering new projects for a specific operating unit; it can also provide updates to the information registered.

An estimator may require to delete any previous project and this functionality must be available.

For a project already registered, it is possible to estimate it using the EPCU tool prototype. This estimate is considered as a scenario estimate, which may be registered with the data to keep track of the value obtained and the assumptions considered.

5.5 Module: EPCU Model

The EPCU Model module implements the algorithm of the EPCU model: it can use the data recorded in the catalogs and project information modules.

The algorithm of this EPCU Model module allows the estimator to perform the basic actions to use the model, such as:

- Register the name, description and identifier of project contexts.

- Register the input variables, with all their features, such as: the number of fuzzy sets and their respective membership function, the domain of functions, etc. The fuzzy sets and their membership functions can also be defined automatically or personalized:
  - Automatically implies the division into segments of equitable ranges or domains of the function.
  - Personalized implies that the user specifically defines the membership function for each fuzzy set.
- Register an output variable. This is developed with the same characteristics as in the input variables and adding the unit of measurement of the output variable.
- Definition and recording of the inference rules representing the knowledge of the experts.
- Add an input variable; if it is considered necessary to add context variables, this functionality involves redefining the rules of inference.

Figure 5.4 shows the use case diagram of the functionality of the module EPCU Model.

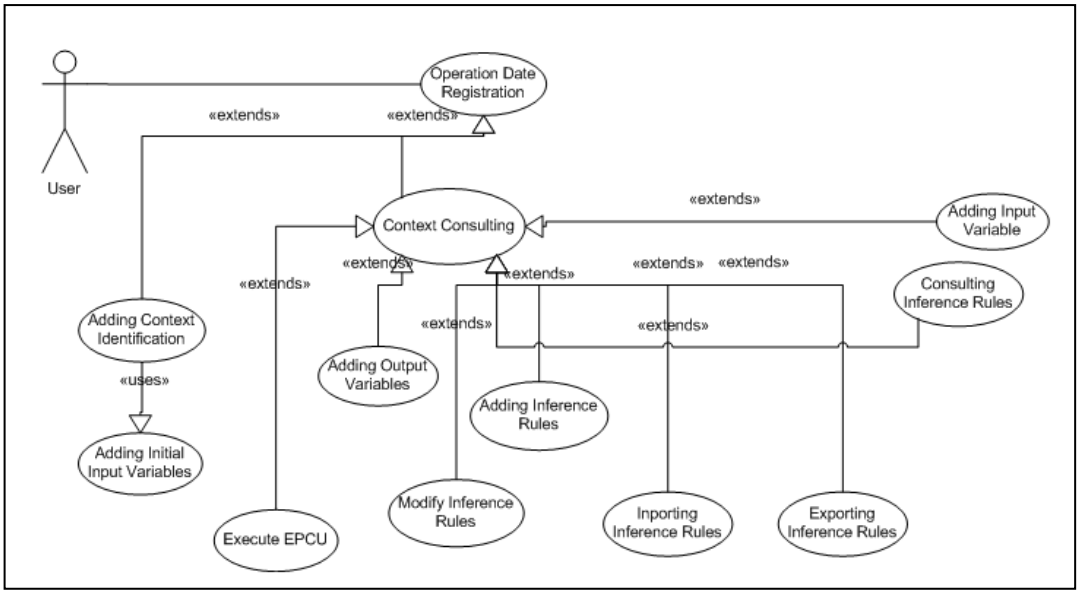


Figure 5.4 Use case diagram of the EPCU Model module

5.6 Module: Portfolio Management

This module for the Management of Projects Portfolio is defined to automatically have a representation of a quantitative value of two variables in a Cartesian Plane; the variables values are gathered using the EPCU model by defining a specific EPCU context. (See Chapter 7, the description of the EPCU model to represent a portfolio approach).

To do this, the module uses the definitions of the portfolio held in the form of catalogs. With these definitions, several projects based on the contexts defined for the selected portfolio can be analyzed.

The result is plotted on a Cartesian Plane which shows the projects and their positions in the reference plane.

Figure 5.5 shows the use case diagram of the functionality for the Portfolio Management module.

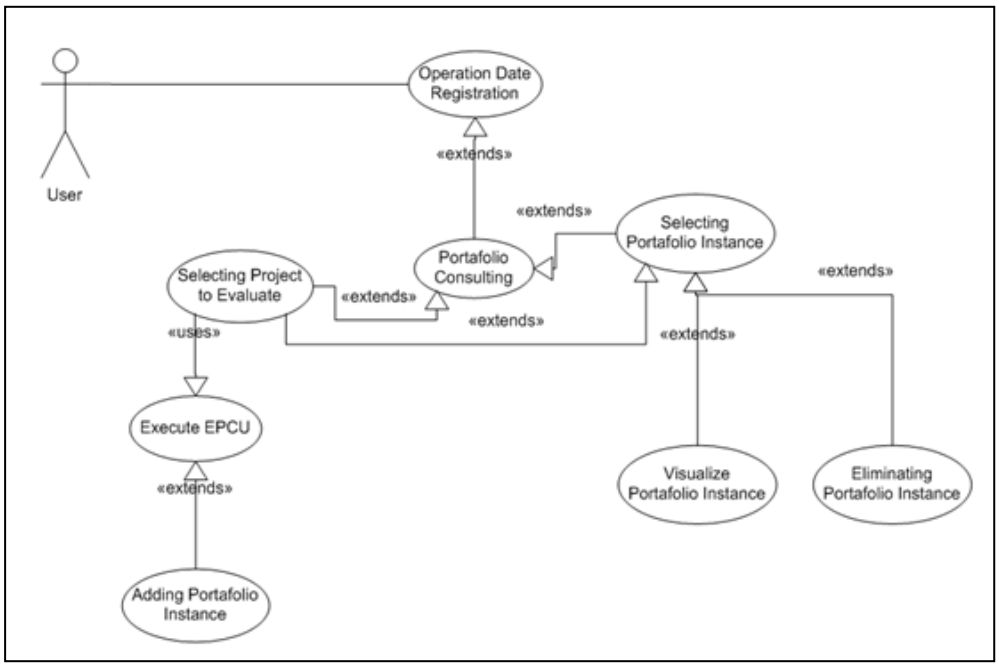


Figure 5.5 Use case diagram of the Portfolio Management module

5.7 Module: Reports

The module Reports is defined to access the information recorded in the system; the information can be filtered based on different criteria and can be presented in tabular or graphical reports in various formats.

Figure 5.6 shows the use case diagram of the functionality for the Reports module.

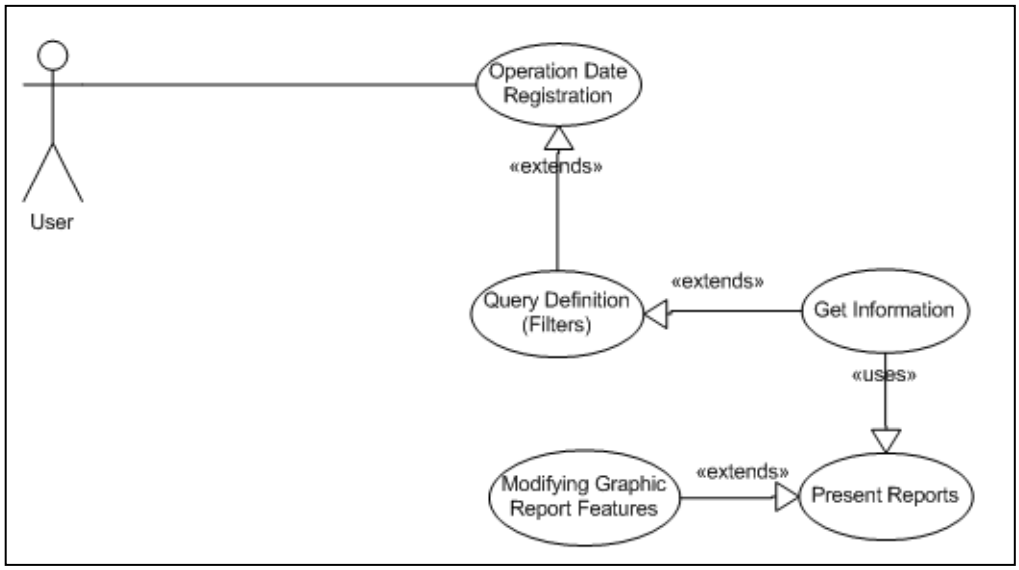


Figure 5.6 Use case diagram of the Reports module

5.8 Database

For registering the information generated, a relational database was designed to provide an adequate infrastructure. The relational diagram of the database is shown in Figure 5.7.

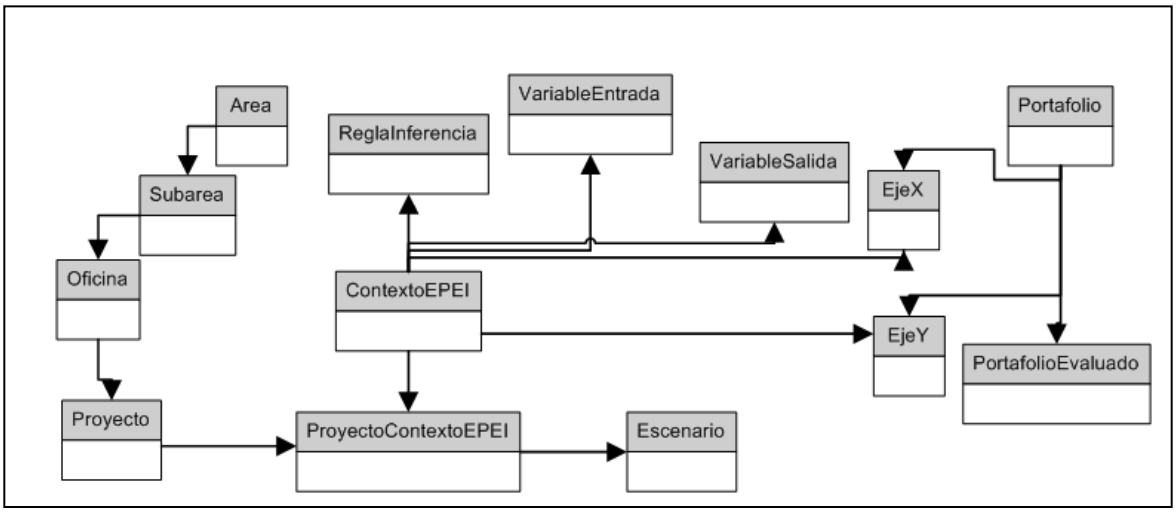


Figure 5.7 Relational database diagram for the EPCU prototype

5.9 Platform and Architecture

The EPCU prototype tool is designed for its use on a single computer (stand-alone) in a two-layer scheme, managing a database SQLAnywhere 9 and a client version for Windows developed in PowerBuilder 10.5. The architecture of the EPCU prototype tool is shown in Figure 5.8.

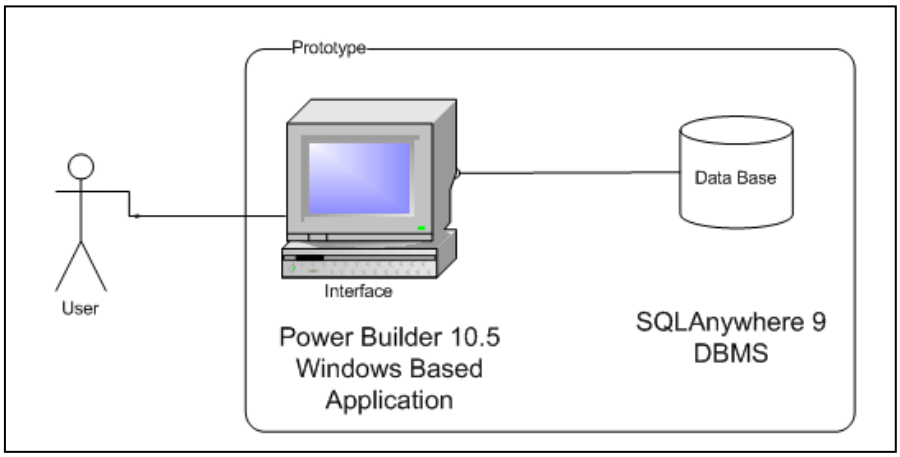


Figure 5.8 Architecture of the EPCU prototype tool

The architecture was defined considering that the use of a Windows-based tool would be more intuitive and simpler to operate; it sought to reduce the effort in defining each of the settings manually for use.

With the architecture defined, it could be easily scalable later to a client-server architecture considering an organizational database management system (DBMS) and more robustness requirements.

5.10 Programming Approach

The prototype tool was developed using an object-oriented approach. This approach seeks to reflect behavior in isolated entities promoting reuse using some key concepts such as:

- Information Hiding
- Polymorphism
- Inheritance, etc.

This will allow for easier maintenance of applications and an easier mechanism for scalability. The OO approach is broad and its benefits have been studied, documenting its benefits and disadvantages (Budd, 1991; Jacobson, 1998; Booch, 1996; Peñaloza, 1996; Sybase, 1996).

5.11 EPCU Context Definition

For the configuration of an EPCU estimation context in the prototype tool three steps are required:

1. Label the EPCU Context and definition of the input variables
2. Output variables definition
3. Inference rules definition

1- For the first step (Label the EPCU Context and definition of the input variables) the window in the prototype tool is shown in Figure 5.9.

Figure 5.9 Window for labeling the EPCU context and for defining the input variables

Once the EPCU context is labeled and the accept button is clicked, the input variables definition section is enabled. The number of variables defined in the labeling of a context, is the number of times the variables section will allow to capture the detail for a specific input variable. The data necessary to define the input variables are: the number of linguistic values (“# Rangos”), the membership function domain, begins in (“inicia en”) and ends to (“termnina en”). The labels of the linguistic values are required too ("Nombre Rangos”).

By default the membership function domain is divided into equal segments for each linguistic value.

2- The process to define the output variable (step 2) is similar to the definitions of the input variables; however, in this step the measurement unit (“unidad”) of the output variable is required. The window for step 2 is shown in Figure 5.10.

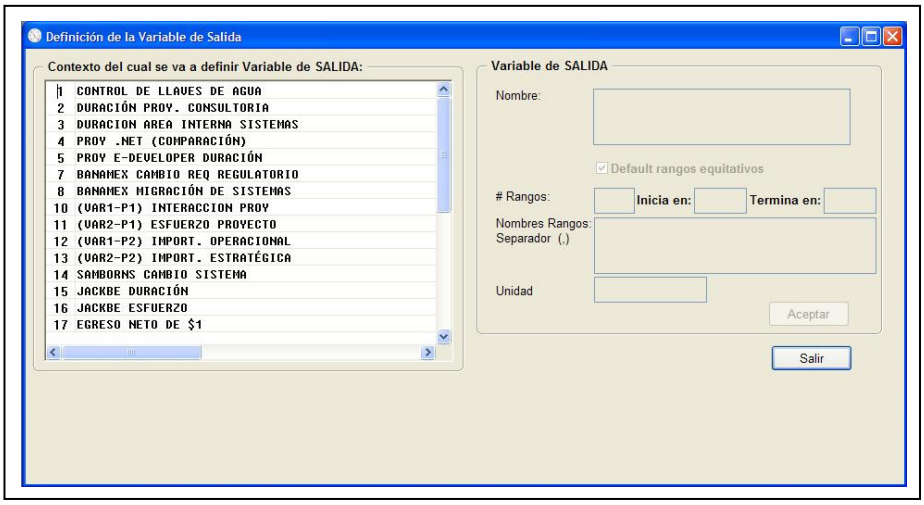


Figure 5.10 Tool Prototype: Window for the definition of the output variable

In order to capture the output variable data, the EPCU context must be selected first. The prototype tool may have several EPCU contexts.

3- Once the steps 1 and 2 are defined, the definition of the inference rules is needed. To do step 3, the possible combinations of the linguistic values for each input variable are determined automatically.

The inference rules are shown in a window one by one, in order to define the relation to the output variable - see Figure 5.11.

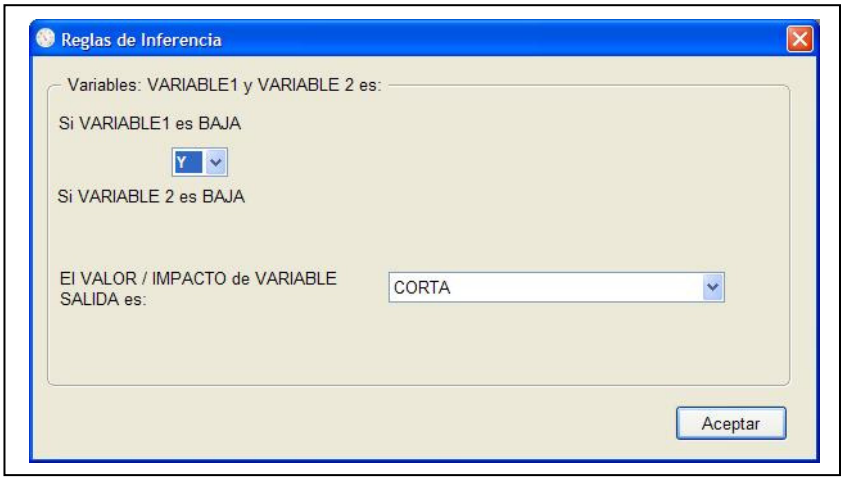


Figure 5.11 Tool Prototype: Window to define the inference rules

5.12 EPCU Context Use for Estimation

After the configuration of the EPCU context in the prototype tool, the EPCU context can be used at any time. In order to use any EPCU context defined, the estimator has to go through the following steps:

1. Select the EPCU context to use
2. Provide a valid value for each input variable
3. Execute the process

1. In step 1, a window with the list of the EPCU contexts available is presented, where the estimator selects the EPCU context to use: the input variables associated are then shown (Figure 5.12).

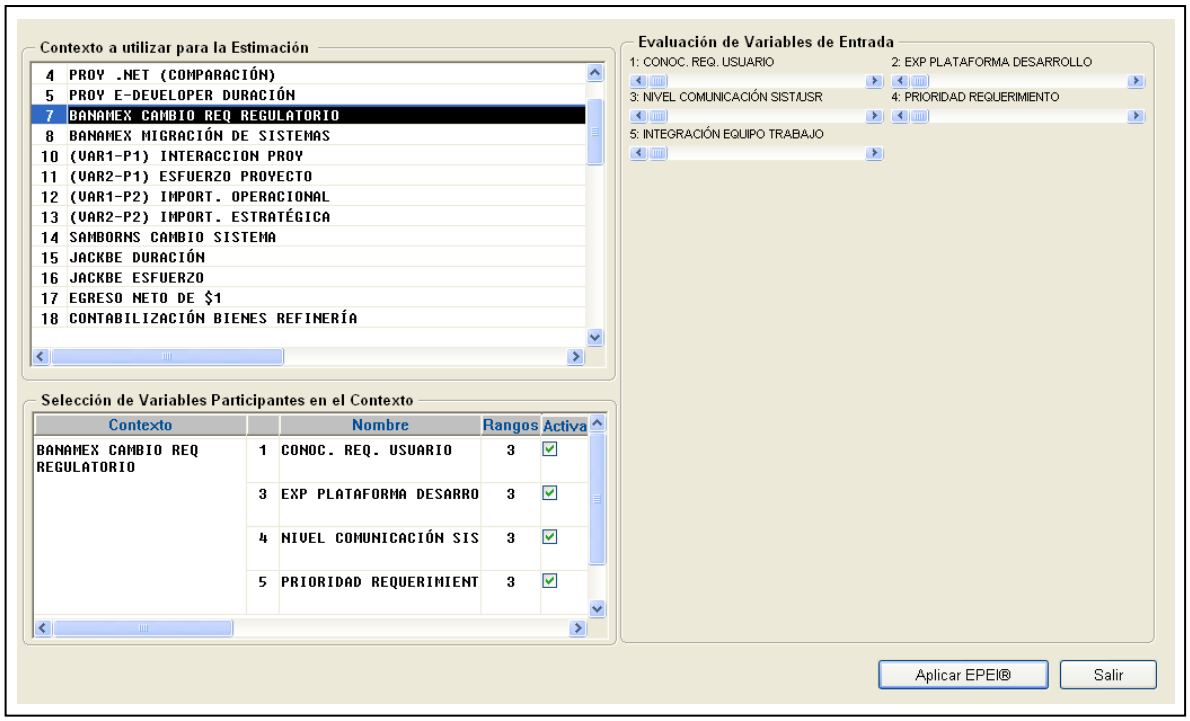


Figure 5.12 Tool Prototype: Use of specific EPCU context

2. After the input variables are shown, the value for each one is provided by the estimator using the scrollbar. The minimum value is 0, the maximum is 5, and the step increment is 0.5. (i.e. [0,5] used in this research)

3- After the values for the input variables are provided, the execution of the estimation process is executed ("Aplicar EPEI[2]"). The process makes the fuzzification, inference rules execution and defuzzification. The estimated value is presented next by the tool.

5.13 Aditionnal Functionality

The example of the windows used in other functionality like estimation scenario registration, reporting and portfolio approach, are shown in figures 5.13, 5.14 and 5.15 respectively.

Figure 5.13 Tool Prototype: Estimation scenario registration

[2] EPEI is the translation of the EPCU to Spanish language

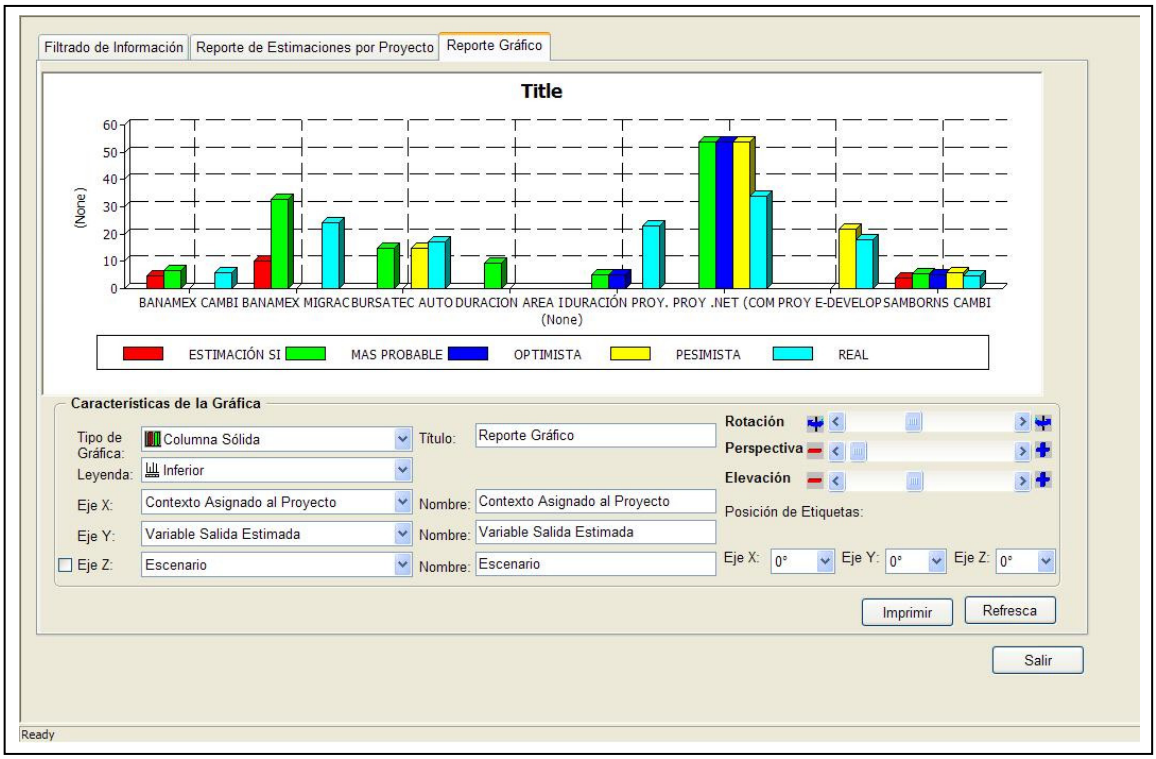


Figure 5.14 Tool Prototype: Graphic report window

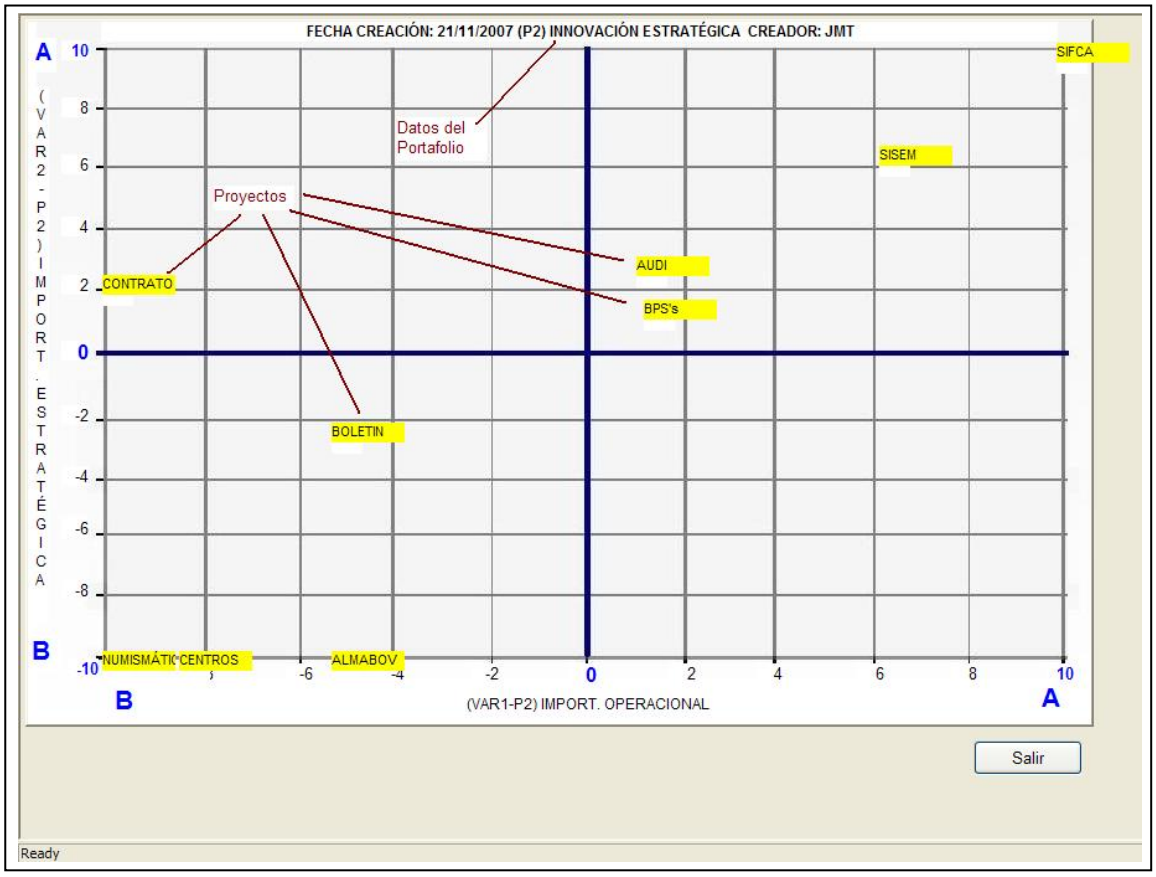


Figure 5.15 Tool Prototype: Portfolio approach Window

The use of this EPCU prototype tool will permit in the next research phase to focus the research efforts on the analysis of the results to analyze the performance, benefits and constraints of the EPCU model, thereby minimizing the effort necessary for the information recording and manipulation of data during and after the experiments.

CHAPITRE 6

EXPERIMENTATION

6.1 Introduction

The major challenge for the experimentation phase of this research is that while the software projects estimates must be done "*a priori*" in the very early phases of the projects, there is a typically large elapsed time before the projects themselves are completed and the 'true' values of effort, duration and costs can be known with certainty in order to verify whether or not the estimates were the right ones.

Considering this challenge, the experimentation was designed to test the process in a context similar as in the early phases. Therefore, the experimentation methodology proposed considers three scenarios:

- Scenario A. A specific EPCU context will be used to estimate real projects completed in industry, but using – for estimation purposes - only the information that was available at the beginning of these projects.
    - The projects selected for the experimentation are real industry projects with distinct features, and including the information on the early contexts of each project. Such early information was provided by the project leader/project manager or a set of persons who were involved in these projects.
    - In this scenario A, the real values of the projects at completion time are known: the verification of the estimates generated using experience-based approach and using the EPCU model will be done using the real data as is an "*a posteriori"* approach.

- Scenario B. In this scenario B the estimates are made before the projects are developed, but the real values for these projects are therefore not available for comparison purposes. This is an "*a priori"* approach to estimation.

- In this scenario B, several rounds of estimations will be generated in order to get more opinions about the possible final values (sensibility analyis); these extra rounds of estimation are sometime called “risk analysis”.

- Scenario C. In order to address the lack of information in scenario B, a simulation experiment will be made:
  - A subset of finished projects from the scenario A will be estimated by people who have not been involved in these projects (independent estimators): this means that the people that estimate a specific project just know the preliminary requirements as if the project were to be developed later (i.e. using only the information available at the beginning of these projects).
  - Two estimates are produced for each project and estimator: one estimate is made by the estimator himself using his experience-based approach and estimation is obtained using the EPCU model. The contexts used in the EPCU model were defined and tested in the Scenario A.
  - For both estimations, the people consider the description of the software requirements as they were described in the early phases for the real projects.

These three sets of experiments provide the opportunity to analyze how the experience of the experts stored in the rulebase can be used by other people with distinct experience and skills (this is referred to in this thesis as: experience systematic replication).

On the basis of the results of these experiments, the comparison of the estimation performance of the EPCU model with the expert judgment estimation approach will be analyzed too.

6.2 Experiments Design

6.2.1 Roles of the participants in the experiments

Conducting the experimentation required two different roles:

1. The expert who provides the information for the configuration of the EPCU model for the context of any project to be estimated (i.e. the EPCU "context"). This is the individual from the organization providing the information about the projects who selects and defines the input variables (i.e. the independent variables, the output variable (i.e. the dependent variable) to be estimated (here: project duration/effort), and the inference rules to be applied for estimating any project with the EPCU model. These participants are named Experts.
2. The independent estimators: the participants who did not participate in the project and who had access only to the preliminary requirements of the project. Their assignment of values for the input variables for a specific project is requested in order to estimate projects using the EPCU model. These participants are named Practitioners.

In some scenarios, the Practitioners and the Experts roles will be played at least by the same person:

- In scenario A, the Expert(s) define the EPCU context for estimating a known project. The same person(s) will also act in the practitioner role for assigning the inputs values using the EPCU context generated, sometimes additional practitioners were included, however they do not play as experts. The estimate will be produced by the EPCU model, not by the 'expert'.
- In scenario B the person who plays the expert role, will play the practitioner role too; sometimes there are other practitioners who do not play as experts, but for the projects in this scenario, the real value of the output variable (i.e. the duration of the project completed) is not available for comparisons.

The practitioners in the scenario C will not be familiar with the details of the software projects or with the organization's development contexts: the only basis that they will have

for estimation purposes will be their own experience. In this scenario C, two sets of estimates will be produced for the duration for a specific project:

- Using their 'own' experience-based approach, and
- Using the EPCU context generated for the specific project.

6.2.2 Experimentation phases

The experimental design consists of 3 phases:

Phase 1:

Involvement of project 'experts' for the data collection and preparation of the base material for the experiments, such as:

a) the description of the software requirements as they were described in the early project phases.
b) the descriptions of the input variables for each project (i.e. the software requirements and the context of the projects),
c) the recording of the 'experts' estimates that were on records at the beginning of the projects and the corresponding real values upon completion of the projects, such as project duration. This project information is provided by the experts: most of the times, these experts had the responsibility of the estimation of the projects. This estimation was made in two ways: with a group of people related to the project and sometimes only by the expert (this is influenced by the size of the organization).

Phase 2:

Involvement of the practitioners for assigning the values to the input variables for each of the projects to be estimated using the corresponding EPCU context. The value assignment is made using experienced judgment approach (for the scenarios A and B, the same person(s) plays the practitioner(s) and expert(s) role(s)).

Phase 3:

Data Analysis

6.3 Phase 1 - Involvement of the project experts for the data collection and preparation of the base material for the experiments

1. Selection of a set of 19 completed projects.

This step consisted of identifying a set of completed projects with the information necessary for the experimentation with the 3 scenarios: 19 projects were obtained from distinct organizations. All of the projects were real projects that had been completed or in feasibility phase in these organizations. The information for configuring the EPCU model (i.e. providing the context information for each project) was provided by the organization's experts (for scenarios A and B, the organizations' experts played the roles of both the experts and the practitioners) - see Table 6.1:

- for the scenario A, 16 projects were used,
- for the scenario B, 3 different projects were used, and
- for the scenario C a subset of 5 projects from the 16 projects in scenario A.

Table 6.1 The 19 projects used in the 3 scenarios

SCENARIO	Number of PROJECTS	Number of PRACTITIONERS	Number of EXPERTS
Scenario A	16 projects	41	31
Scenario B	3 projects	13	7
Scenario C (Projects 1, 3, 4, 5, 6)	5 projects (a subset of the 16 projects of scenario A)	84	8
TOTAL	**19 projects**	**138**	**38**

2. Description of the software requirements and the development context.

In this step, the expert provided a description of the software requirements and also described the context in which each project was developed.

An EPCU "context" is defined as a set of variables (i.e. the inputs/output of the estimation process) and the relations that affect a specific project or a set of similar projects (i.e. the rulebase).

For each distinct project, the corresponding information that was available in each organization at the time of the project's inception (that is, the independent variables known at project's inception when an "*a priori"* estimation is typically performed) were re-documented, even though the projects were actually completed and the experts had also all the information on the completed project.

This re-documentation of the early software requirements was based on the availability of project documentation in each participating organization and in the experts' recollections, as illustrated with the following example:

"The project is a .NET project to develop a B2B system for controlling the operations of shipping, transportation and delivery of packages for specialized organizations such as DHL or UPS. In addition, the B2B system must provide for contract and shipping management, package tracking, and so on".

The description of the context refers to the relevant factors that were present during project development. Most often, these factors are related to the process, the people, the organizational environment, and so on. For example:

"The responsibility of the entire project was assigned to a person who had a very good understanding about the problem domain and of the tool in which the project was developed".

3. Configuration in the EPCU prototype tool: definition of the context

Using the information described in the previous step, the prototype tool that implements the EPCU model must be configured. This means that for a specific EPCU "context" a setup must be made in the EPCU prototype tool in order to have the possibility of generating estimates using the context defined in the previous step (Step 2).

The configuration or tool setup was made by the researcher. The steps of setup the EPCU context in the prototype tool (See Chapter 5) are:

- Assign a name for the specific context.
- Configure the input variables involved, with the membership function provided by the expert.
- Configure the output variable, whit the membership function provided by the expert.
- Configure the rulebase to be executed during estimation.

Once the context is setup in the EPCU prototype tool, using it for estimation purposes is relatively easy: what is needed is only the assignment of the input values by the practitioner role for the input variables defined for the EPCU context.

6.4 Phase 2 – Involvement of the practitioners in selecting "*a priori*" input values for each of the projects to be estimated.

1. Collection of the values assigned by the practitioners for the input variables for each EPCU context used for estimating a specific project.

For this step, the practitioners were provided with the following information:

- a description of the software requirements (which were documented earlier in Phase 1 by the experts),
- a description of the development context in which each software was developed or will be developed (as provided in Phase 1 by the experts), and
- a questionnaire form (Annex XII) designed to gather from the practitioners their assignments of values for the input variables for each EPCU context defined to

estimate each project. The value assignment is materialized within a 0 to 5 range ε R (see Chapter 4). In this questionnaire, an estimate of the output variable, using their own experience as practitioners, is requested too.

Using the descriptions of the software requirements and of the development context, the practitioners provide an opinion on each input variable (i.e. value assignment) defined in a specific EPCU context generated, based on their own experience.

The values assigned for the input variables were collected from 138 practitioners (See Table 6.1) from 2005 to 2009, and estimates of the dependent output variable were generated using the EPCU model.

The different ways in which the information from the practitioners was acquired are:

1. Personal interviews.
2. Electronic data collection: an email sent to people known by the researcher.
3. Data collection within the context of courses within a continuous education program at an Institute in Mexico (Instituto Tecnológico Autónomo de México - ITAM). The questionnaires were distributed personally by the researcher.
4. Data collection during conferences. Questionnaires were also distributed by the researcher at two software measurement conferences:
   - Squeeze your Metrics (Sácale jugo a tus métricas), November 5th 2008, Circuito Tecnológico SE-AVANTARE, México D.F.
   - Expo TI 2008, Taller de Técnicas de Estimación, November 19th y 20th 2008, CANACINTRA, México, Puebla.

For each input variable, the participants had to select values within a range from 0 to 5 Ɛ R (0 being the lowest and 5 the highest).

3. **Collection of the practitioners own 'expert judgment' estimate for the duration of each project**: in this step, the practitioners provided a duration estimate for each project

(in months, weeks, or days) using the descriptions of the software requirements and of the development contexts, as well as their own experience.

6.5 Phase 3 - Scenario A. Data analysis of 16 completed projects

For this scenario A, 16 projects were estimated.

Using the model configured in the prototype tool for the first 16 distinct EPCU contexts, the 41 persons (See Table 6.1) who participated in the phase 2 as practitioners provided a value assignment on each input variable configured.

An analysis of the performance of the EPCU model is presented next (see ANNEX II for more details).

Table 6.2 Number of people who participated as practitioners, by project

Project	Practitioner by project	Practitioners Subtotal	
Project 1	5		
Project 2	1		
Project 3	3		
Project 4	1		
Project 5	1		
Project 6	1		
Project 7	3		
Project 8	2		
Project 9	5		
Project 10	2		
Project 11	5		
Project 12	2		
Project 13	2		
Project 14	2		
Project 15	3		
Project 16	3	From project 1 to project 16	41
Project 17	4		
Project 18	5		
Project 19	4	From project 17 to project 19	13
TOTAL	54		54

Figures 6.1 to 6.4 are used to analyze the performance of the EPCU model. Each figure presents the real value of the duration of the project, as well as the value estimated by the 41 practitioners for the first 16 of the 19 projects. The figures 6.1 to 6.4 show in the "*y*" axis the project duration in calendar months and, in the "*x*" axis, identification number of each of the the 41practitioners.

Figure 6.1 presents the real value (in blue) and the duration estimates in months (in red) provided by the 41 practitioners using the experience-based approach. Table 6.2 shows how many people participated as both expert & practitioner in scenario A.

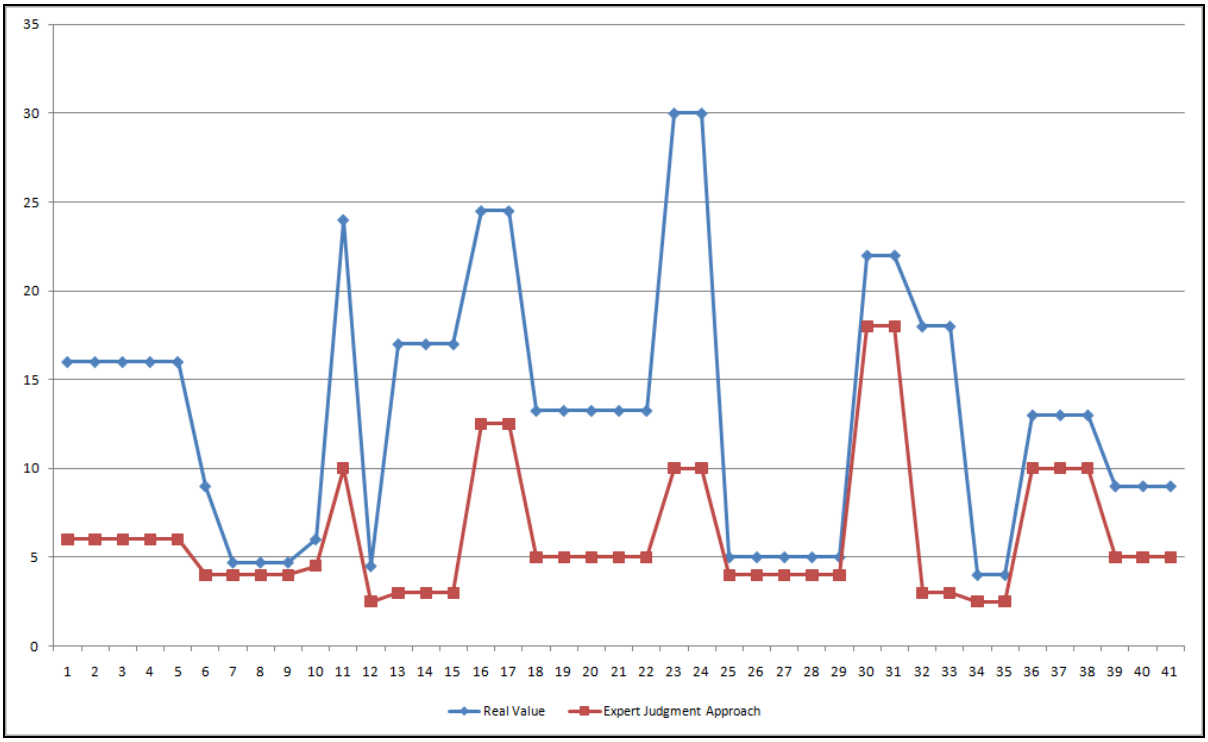


Figure 6.1 Scenario A – Duration: real values and experience-based judgment estimates (41 estimates without EPCU)

In Figure 6.1, it can be observed graphically that all (100%) of the duration estimates using the experience-based judgment approach were underestimated against the real value: the

lowest underestimate is at 15% (practitioners 7, 8, 9) and the highest underestimate is at 83% (practitioners 32, 33).

Figure 6.2 presents the duration estimates (in green) provided by the EPCU model for the values of the inputs provided by the 41 practitioners for these 16 projects (real duration values in blue). In this Figure 6.2, it can be observed graphically that the EPCU model:

- underestimates the project duration for the inputs provided by 25 practitioners (i.e. 61%), and
- overestimates for the inputs provided by the other 16 practitioners (i.e. 39%).

The smallest underestimate is 4% (practitioner 40) and the highest underestimate is 39% (practitioner 32) while the lowest overestimate is 3% (practitioner 4) and highest overestimate is 50% (practitioner 35).

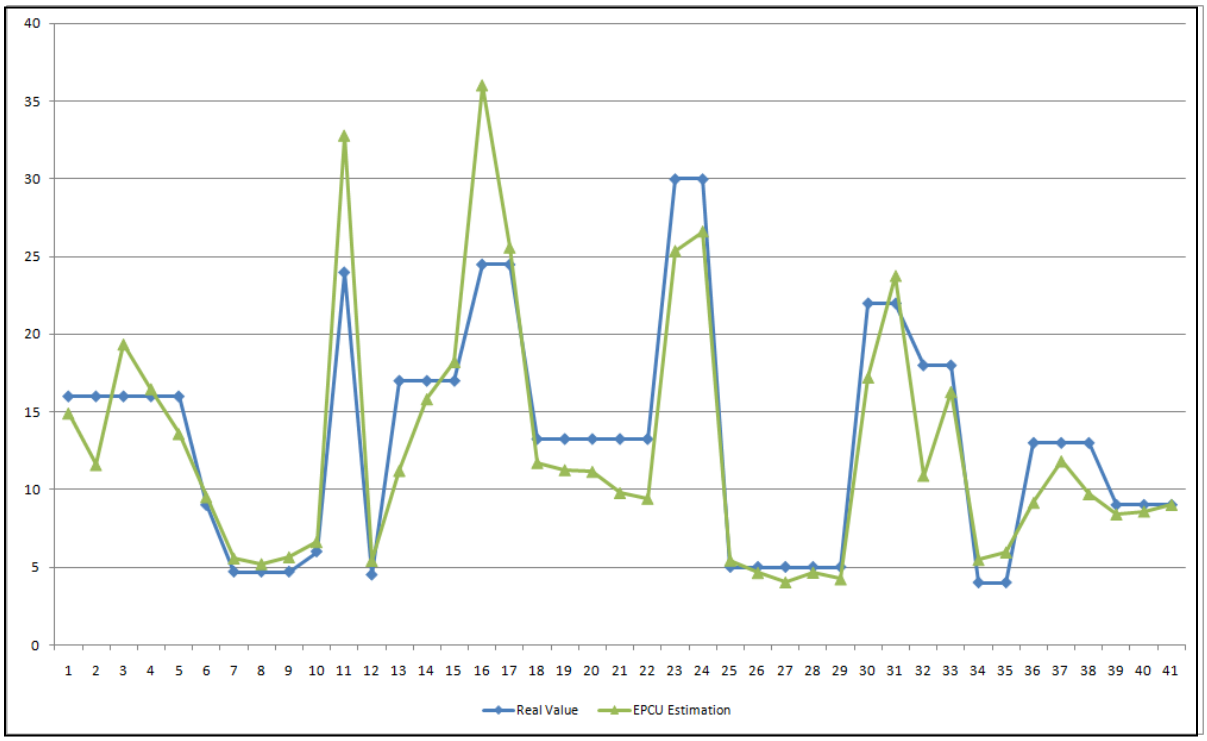


Figure 6.2 Scenario A – Duration: Real value and EPCU model estimaties (41 estimates)

Figure 6.3 presents the duration estimates provided by the 41 practitioners using the experience-based approach (in red), and the duration estimates provided by the EPCU model (in green) for the same 41 practitioners for the 16 projects.

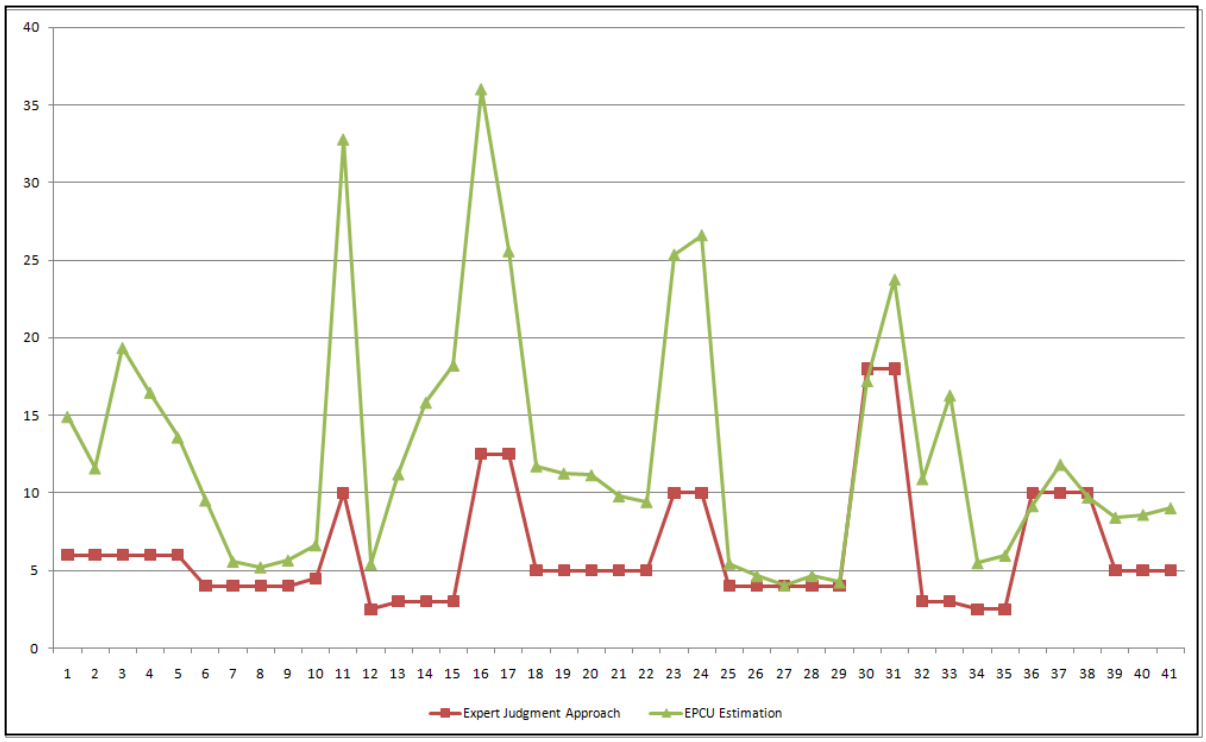


Figure 6.3 Scenario A – Duration estimates: EPCU model and experience-based judgments (41 estimates)

Figure 6.4 presents next the comparison of the results from the experience-based approach (in red), the EPCU model approach (in green) and the real value (in blue) for the duration of these 16 projects.

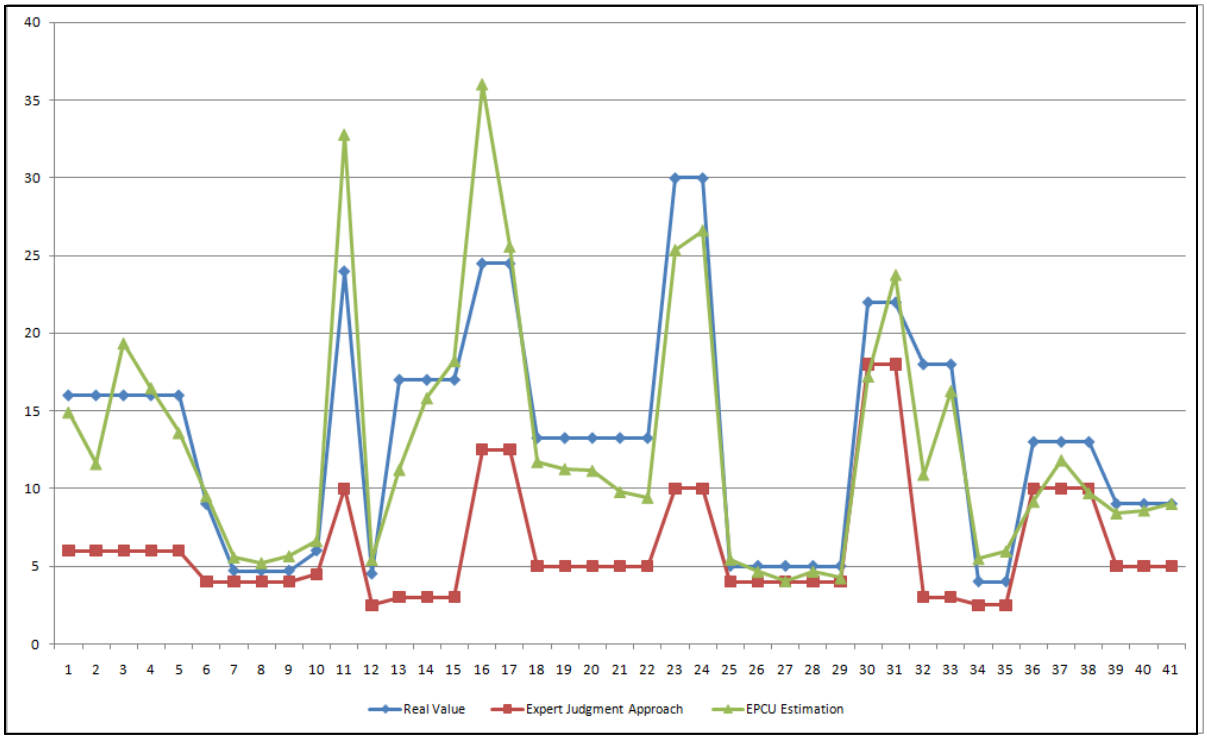


Figure 6.4 Scenario A – Duration: Real values, EPCU and Expert Judgment estimates (41 estimates)

From Figures 6.1, 6.2 and 6.3, it can be observed that there are a number of differences between the EPCU estimation results and the experience-based estimation results - See Figure 6.4:

- 100% of the projects were underestimated by the experience-based approach, and
- the estimates provided by the EPCU model for the same practitioners were either underestimated (61%) or overestimated (39%).

The descriptive statistics about the MRE for both the EPCU model estimates and the experience-based approach are shown in Table 6.3, using the statistical tool SPSS v 17: it can be observed that the mean of MRE (MMRE) of 17.5% for the duration estimates for the EPCU model (rightmost column: MRE_EPCU) is much lower than the MRE of 46.6% for the duration estimates from the experience-based approach (left column: MRE_EXP_JUDG:). The standard deviation (SDMRE) has a similar effect: 12.4% using the EPCU model and 22.7% with the experience-based approach.

Table 6.3 Scenario A: Descriptive statistics of the MRE (experience-based and EPCU approach – 41 estimates)

		MRE_EXP_JUDG	MRE_EPCU
N	Valid	41	41
	Missing	0	0
	Mean	.4661	.1751
	Median	.4900	.1490
	Mode	.20[a]	.00
	Std. Deviation	.22664	.12393
	Variance	.051	.015
	Skewness	.018	.946
	Std. Error of Skewness	.369	.369
	Kurtosis	-1.311	.199
	Std. Error of Kurtosis	.724	.724
	Minimum	.15	.00
	Maximum	.83	.50
	Sum	19.11	7.18

From Table 6.3, it can be observed that the estimates using expert judgment have a greater dispersion than the estimates generated using the EPCU model: when the kurtosis coefficient is greater than 0, this means that the data are more concentrated to the mean. This is the case for the MRE when EPCU is used. In the Expert Judgment approach the kurtosis coefficient is less than 0: this mean that the data are not concentrated to the mean, confirming a high dispersion.

The histograms of these analyses are shown in Figures 6.5 and 6.6. In these figures the *x* axis represents a MRE range determined by the tool (SPSS v 17) and the *y* axis represent the frequency, or the number of estimates in a specific range: i.e. in Figure 6.5 for the range 60% < MRE ≤ 70% the frequency is 12 cases with a MRE value within this high MRE range.

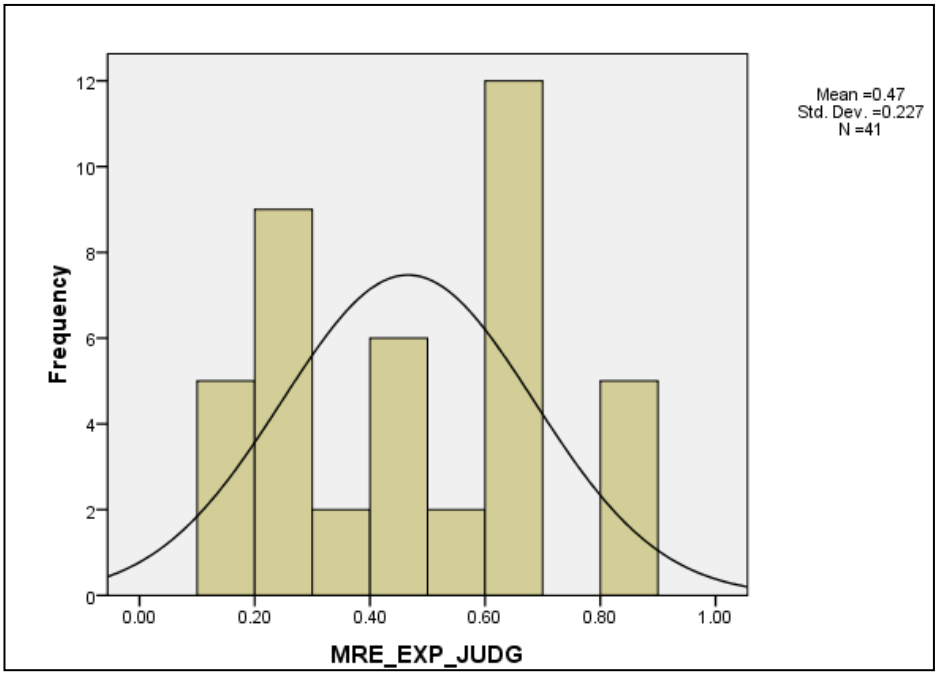


Figure 6.5 Scenario A: MRE experience-based approach distribution (41 estimates)

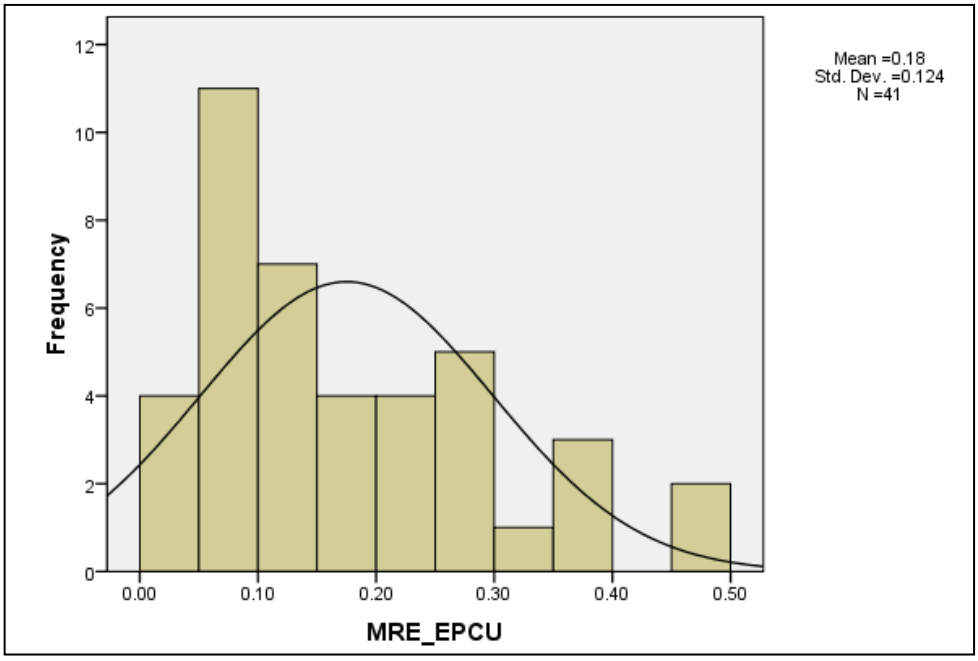


Figure 6.6 Scenario A: MRE EPCU approach distribution (41 estimates)

The skewness reflects the fact that there are more estimated points with smaller MRE than with larger MRE. The "cut-off" of the left side of the distributions in particular with the EPCU estimates means that there is no perfect estimate with error = 0.

In summary, in this scenario A of experiments, the EPCU estimation approach performed considerably better than the experience-based approach.

6.6 Phase 3 - Scenario B. *A priori* estimation data analysis

The purpose of this scenario B is to simulate the "*a priori"* approach, making several rounds of estimations in order to get more opinions about the value of the dependent variable (i.e. the output variable: project duration (for projects 17 and 19), effort (for project 18)). Scenario B included 3 additional projects in which information was available only for their very early phase: that is, for this scenario B, the final values of the output variable for the development of these 3 projects are not known: two projects were never developed, and the third one is still in progress.

Tables 6.1 and 6.2 show that 13 people were involved as practitioners for the three projects 17 to 19. For this scenario B, 7 people play both the role of experts to define the EPCU context and the role of practitioners to provide the input values to the EPCU model. 6 people play just the practitioner role.

As experts they defined the input variables to be used, the output variable and the inference rules: this means that they defined the EPCU context. As practitioners, they used next the EPCU context to provide the input variables for a specific project.

The 3 projects used in this scenario were from distinct organizations; for this reason, the experts/practitioners who participated were distinct for each project. Each project used distinct EPCU context because there were distinct.

The experiments procedures for each of these 3 projects are presented next.

Project 17 experiment

The first project in this scenario B (project 17) consists in an Oracle data mart development. The estimation using the proposed EPCU model was made by 4 practitioners (a single one of them played the role of expert to define the EPCU context). The 4 estimates were generated using the EPCU model. Each estimate reflects the point of view of each person who assigned the values to the input variables.

In order to gather an additional estimation that considers all the opinions previously obtained by the experts for each of the inputs values, an average is calculated for each input variable using the values assigned by the 4 practitioners (Estimation using the average of the input variables estimat, rightmost column in Table 6.4).

Table 6.4 Scenario B - project 17: Duration estimates for each participant

	Project Manager	Analyst 1	Analyst 2	Analyst 3	Average of inputs variables
EPCU model	5.3	3.8	5.9	5.3	4.5

Project 18 experiment

The second project in this scenario (Project 18) is from the insurance industry. The EPCU context was configured and evaluated by 5 people, that mean, all 5 play the expert role participating in the configuration of the EPCU model as a group, next the 5 people individually act as practitioners to estimate using the EPCU context configured for the Project 18; the ouput selected as the dependent variable was the project effort, in person-hour units. The estimation for this project was therefore related to the effort. The 5 practitioners who assigned the values for the input variables defined in the EPCU context also provided each an effort estimate based on their own experience, the effort gather by the 5 people in a estimation-work meeting using the experience-based approach. A sixth effort estimate (rightmost column in Table 6.5) was generated using the average of the values assigned by the 5 practitioners for each input variable.

Risk Analysis

A risk analysis was developed in order to visualize what could happen in terms of the output variable if one of the input variables reaches a risky boundary.

In order to simulate the presence of the risk, the researcher selected one of the boundary values for one variable at a time (lower [0] or higher [5]) keeping the average of the opinions for the other input variables.

A. Individually risk analysis

Taking as the basis the average value assigned for each input variable, one input variable at a time was evaluated in the risky boundary. This generated five EPCU estimates (in Table 6.6, these are labeled from scenarios 7 to 11).

B.- Combined risk analysis

The risk of the input variables does not necessarily need to be isolated: this means that, if one variable is risky, not all the other input variables are not risky, indeed, there is a high possibility that several variables may be also risky at the same time.

In order to analyze this situation, the researcher combined the individual risk analysis to several variables at the same time. This generates 6 distinct EPCU estimates (in Table 6.6 these are labeled as scenarios 12 to 17).

Table 6.5 shows the first 6 opinions generated by the experts identified by the role played in the organization who estimate the project. In Project 18, all the estimates are in person-hours (For more details, see ANNEX IV). The experience-based estimation for this project in which all the roles described in Table 6.5 participated was 2,209 person-hours.

Table 6.5 Scenario B- Project 18. Effort estimates (in person-hours) for each practitioner

Variables	Operation Manager	Technology Manager	Sales Manager	Chief Operations Manager	Project Leader	Average of input values
EPCU model	2201.8	1561.4	1528.7	1561.3	1652.7	1687.2
EXPERIENCE BASED ESTIMATION	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4

Table 6.6 shows the risk analysis estimates developed by the researcher taking as a basis the average of the opinions.

Table 6.6 Scenario B - Project 18 Risk analysis: Effort estimates generated by the EPCU model

	Individually risk analysis					Combined risk analysis					
Variables	Scenario 7	Scenario 8	Scenario 9	Scenario 10	Scenario 11	Scenario 12	Scenario 13	Scenario 14	Scenario 15	Scenario 16	Scenario 17
EPCU model	1716.8	1995.3	2170.3	2050.6	2288.6	2330.2	2571.6	2685.4	2841.4	2966.1	3750.0
EXPERIENCE BASED ESTIMATION	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4	2209.4

Project 19

The third project (project 19) is from the financial industry. This project has two main constraints: the window timeframe is about 30 months (the stakeholder (CIO) judgment approach) and the development team has to introduce the PSP and TSP practices (the staff was not familiar with these new software development practices).

- The Personal Software Process (PSP) and Team Software Process (TSP) are software process improvement technologies developed by the Software Engineering Institute (SEI) that aim to improve the performance of software teams. The PSP and TSP merge concepts from the Software Capability Maturity Model (SW-CMM), statistical methods, and team behavioral theory into an integrated process that directly benefits the individuals and teams who are responsible for producing and maintaining software.

For this third project, 5 duration estimates were generated using the EPCU model: 4 directly by 4 practitioners (one of them play the expert role too) and 1 estimate using the average of the values assigned by the 4 practitioners for each input variable. The detailed data are presented in ANNEX V.

Table 6.7 presents the duration estimates in comparison to the pre-set deadline of 30 calendar months.

There is an important observation relating to Table 6.7: the 'expert judgment' value of a 30 months deadline (i.e. duration was estimated by the CIO of the organization using his own experience) which led to this deadline being transformed into a constraint for this project.

Table 6.7 Scenario B – Project 19: Duration Estimates generated by the EPCU model

	Practitioner 1	**Practitioner 2**	**Practitioner 3**	**Practitioner 4**	**Average**
EPCU	40.0	38.1	54.0	32.0	39.1
CIO Constraint	30	30	30	30	30

As can be seen in Table 6.7, all duration estimates generated by the EPCU model using the input variables assigned by the practitioners were over the 30 months deadline: the input values provided by practitioner 4 led to the lowest difference (2 months after the deadline), while the highest difference (14 months after the deadline) came from the input values provided by practitioner 3.

6.7 Phase 3 - Scenario C. *A priori* estimation - Projects simulation data analysis

6.7.1 Experiment context and initial data analysis

On the one hand, the purpose of scenario C is to simulate an "*a priori*" estimation process. On the other hand, scenario C is used to compare the performance of the experience-based estimation approach against the EPCU estimation approach. This scenario can represent a systematic replication of the estimation expertise (experience systematic replication) that the

experts have embedded in the EPCU model through their selection of the input variables and their selection of ranges of values for these variables, as well as for the output variable.

A total of additional 84 practitioners participated in the experiment for this scenario C. These 84 practitioners had not been involved with the projects to be estimated, and were not the same people who participated in scenarios A or B. These practitioners had a large variation of experience, skills and background; they were therefore classified into three categories according to:

1. their professional experience,
2. their software development experience, and
3. whether or not their profession is IT-related.

- Each of the 84 practitioners participating in the experiment filled the research questionnaire (ANNEX XII) and had to assign values for the input variables listed in it for 5 projects: these 5 projects (Projects 1, 3, 4, 5, 6) are a subset of the 16 projects from the scenario A.
- Each of the practitioners also had to provide an experience-based approach estimate of the duration for each of these 5 projects.

The full data sample is presented in ANNEX VI, ANNEX VII. ANNEX VIII. ANNEX IX, ANNEX X.

This scenario has:

- The practitioners own experience-based estimates (i.e. without the use of the EPCU model).
- The estimates obtained for the 5 projects using the EPCU model with the value assignment for the input variables from the 84 practitioners.

The data were analyzed and the descriptive statistics were provided by the SPSS 17 statistical software: the results are shown in Table 6.8 where the mean of the MRE (MMRE) for each project are:

A) MRE of experienced-based estimates

- MRE_EJ_P1 = 47.8%,
- MRE_EJ_P2 = 57.4%,
- MRE_EJ_P3 = 111.9%,
- MRE_EJ_P4 = 54.6%,
- MRE_EJ_P5 = 53.6%

B) MRE of EPCU model estimates

- MRE_PCU_P1 = 54.6%
- MRE_PCU_P2 = 16%,
- MRE_PCU_P3 = 41.3%,
- MRE_PCU_P4 = 34.8%,
- MRE_PCU_P5 = 21.6%

Table 6.8 Scenario C: Descriptive MRE statistics for the 5 projects (Pi) using the experience-based approach and the EPCU model – 84 practitioners (Valdès, 2010)

		MRE_EJ_P1	MRE_EPCU_P1	MRE_EJ_P2	MRE_EPCU_P2	MRE_EJ_P3	MRE_EPCU_P3	MRE_EJ_P4	MRE_EPCU_P4	MRE_EJ_P5	MRE_EPCU_P5
N	Valid	52	84	52	84	52	84	53	83	51	83
	Missing	32	0	32	0	32	0	31	1	33	1
Mean		.4779	.5462	.5740	.1596	1.1185	.4132	.5464	.3378	.5362	.2157
Median		.5000	.6500	.6700	.1400	.7800	.3700	.4900	.1700	.5000	.2400
Std. Deviation		.41544	.30635	.29000	.11289	1.11629	.30296	.44459	.30996	.24901	.13404
Variance		.173	.094	.084	.013	1.246	.092	.198	.096	.062	.018
Skewness		1.050	.616	-.560	1.291	1.673	.365	2.860	1.249	-.207	1.334
Kurtosis		2.087	1.216	-.700	1.692	2.236	-1.229	12.760	.725	-1.053	2.839
Minimum		.00	.01	.00	.01	.00	.01	.06	.05	.00	.03
Maximum		2.00	1.55	1.00	.50	4.33	.94	2.83	1.20	.97	.75

The whole output for the statistic analysis with SPSS 17 is presented in ANNEX XI.

6.7.1.1 Performance of the EPCU model

The performance of the use of the EPCU model is evaluated in Table 6.9 (with 4 of the evaluation criteria described in Chapter 1) for the 5 projects with the information provided in input by the 84 practitioners.

It can be observed from Table 6.9 that:

- the best MMRE (i.e. the lowest) is obtained using the EPCU context when estimating project 2: 16%.
- the worst MMRE (i.e. the highest) is obtained using the EPCU context when estimating project for project 1: 55%.

If the MMRE is considered as the key criterion, the best rulebase performance is for project 2 (MMRE = 16%) and project 5 (MMRE = 22%), while projects 4, 3 and 1 have an MMRE of 24%, 41% and 55% respectively.

Considering next the coefficients of prediction better than 25%, the Pred(25%) criterion (bottom line of Table 6.9) are as follows:

- The most accurate estimation is obtained with the rulebase for project 2, with a prediction level of 85%, and project 5, with a prediction level of 80%.
- The least accurate estimation is obtained for project 1, with a prediction level of only 21%.

Table 6.9 Scenario C: Performance Estimation results using the EPCU model for 5 projects (Valdés, 2010)

Criteria & Projects	**P1**	**P2**	**P3**	**P4**	**P5**
MMRE	55%	16%	41%	34%	22%
MdMRE	65%	14%	37%	17%	24%
SDMRE	31%	11%	30%	31%	13%
Pred(25%)	21%	85%	42%	58%	80%

6.7.2 Experience systematic replication

The next set of analyses with Scenario C investigates whether or not the performance of the EPCU model is influenced by the expertise of the practitioners who evaluated the input variables of the projects to be estimated. For this type of analysis, the following sub-samples were identified:

1. Practitioners with more than or equal to 10 years of professional experience (or with less than or equal to10 years of professional experience).
2. Practitioners with more than 5 years in software development experience (or with less than or equal to 5 years in software development experience).
3. IT related professionals (or non-IT related professionals).

The sub-sample sizes for each project (including the full sample and the sub-sample for each classification category) are shown in Table 6.10. It is to be noted that in the questionnaires filled out, some of the "practitioners" did not include the information necessary to classify them into a specific category. Therefore:

- the sub-sample size for each category varies from 25 to 59 practitioners, as illustrated in Table 6.10.
- for project 5 in the categories "Practitioners with more than 5 years of experience in software development' and "Practitioners with less than or equal to 5 years of experience in software development", the sum of the sub-samples for each one is 75: this means that some of the people did not include the necessary information to be classified into one of either categories.
- for projects P4 and P5, one practitioner did not provide all of the information for the inputs variables: therefore, the full sample size for projects P4 and P5 is 83 instead of 84 practitioners.

Table 6.10 Scenario C – Sub-sample sizes by classification of practioners for each project

		FULL SAMPLE	MORE THAN 5 YEARS EXPERIENCE IN SOFTWARE DEVELOPMENT	LESS OR EQUAL THAN 5 YEARS EXPERIENCE IN SOFTWARE DEVELOPMENT	MORE OR EQUAL THAN 10 YEARS PROFESSIONAL EXPERIENCE	LESS THAN 10 YEARS PROFESSIONAL EXPERIENCE	NON IT PROFESSIONALS	IT PROFESSIONALS
P1	Number of Practitioners	84	31	44	45	39	25	59
	%	100%	37%	52%	54%	46%	30%	70%
P2	Number of People	84	31	44	45	39	25	59
	%	100%	37%	52%	54%	46%	30%	70%
P3	Number of People	84	31	44	45	39	25	59
	%	100%	37%	52%	54%	46%	30%	70%
P4	Number of People	83	31	43	45	38	25	58
	%	100%	37%	52%	54%	46%	30%	70%
P5	Number of People	83	31	43	45	38	25	58
	%	100%	37%	52%	54%	46%	30%	70%

The performance of the use of the EPCU model for each of the sub-samples of data related to the practitioners classification that was identified in Table 6.10 is evaluated using the same 4 criteria: MMRE, MdMRE, SDMRE and Pred(25%).

The quality criteria obtained by each classification for each of the 5 projects with the 84 practitioners for each project is shown in Table 6.11.

For all the practitioner's classification, it can be observed that the EPCU model performance is similar to the full sample. The highest MMRE is for Project 1, followed by Project 3, Project 4, Project 5 and finally Project 2.

Table 6.11 Scenario C: Performance of the EPCU model, by project, and by practitioners' categories

		FULL SAMPLE	MORE THAN 5 YEARS of EXPERIENCE IN SOFTWARE DEVELOPMENT	LESS THAN OR EQUAL TO YEARS of EXPERIENCE IN SOFTWARE DEVELOPMENT	MORE Than OR EQUAL to 10 YEARS off PROFESSIONAL EXPERIENCE	LESS THAN 10 YEARS of PROFESSIONAL EXPERIENCE	NON IT PROFESSIONALS	IT PROFESSIONALS
Number of practitioners		84	31	44	45	39	25	59
P1	MMRE	55%	54%	55%	59%	55%	52%	56%
	MdMRE	65%	54%	63%	67%	65%	51%	67%
	SD MRE	31%	30%	33%	30%	31%	31%	30%
	Pred(25%)	21%	23%	23%	16%	46%	20%	22%
Number of practitioners		84	31	44	45	39	25	59
P2	MMRE	16%	17%	14%	16%	16%	18%	15%
	MdMRE	14%	14%	14%	14%	15%	17%	14%
	SD MRE	11%	11%	11%	12%	10%	11%	11%
	Pred(25%)	85%	84%	89%	82%	87%	84%	85%
Number of practitioners		84	31	44	45	39	25	59
P3	MMRE	41%	44%	40%	40%	43%	46%	39%
	MdMRE	37%	52%	32%	35%	38%	51%	30%
	SD MRE	30%	31%	31%	29%	32%	28%	31%
	Pred(25%)	42%	39%	45%	42%	41%	32%	46%
Number of practitioners		83	31	43	45	38	25	58
P4	MMRE	34%	31%	37%	32%	36%	27%	37%
	MdMRE	17%	17%	17%	17%	17%	17%	17%
	SD MRE	31%	30%	32%	31%	31%	27%	32%
	Pred(25%)	58%	61%	56%	60%	55%	64%	57%
Number of practitioners		83	31	43	45	38	25	58
P5	MMRE	21%	24%	19%	23%	19%	22%	21%
	MdMRE	24%	24%	21%	24%	23%	24%	24%
	SD MRE	13%	16%	11%	15%	11%	12%	14%
	Pred(25%)	80%	71%	91%	76%	87%	76%	81%

From Table 6.11, it can be observed that the performance ofthe EPCU model presents a low variation, whatever the skills levels of the practitioners. In particular,

- For Project 1 :
  - the maximum MMRE is 59% and the minimum, 52% with little variation;
  - the highest SDMRE is 33%.
  - there is a much wider range of values for Pred(25%): from 16% to 46%.

- For project 2:
  - the differences across MMREs are small (highest MMRE is 18% and the lowest is 14%).
  - the highest SDMRE is 12%,
  - the Pred(25%) is one of the highest in this experiment (between 82% and 89%).

- For project 3 :
  - the difference between the maximum and the minimum MMRE is 7% with the worst MMRE is 46% and the best is 39%;
  - the highest SDMRE is 32%, the best is 28%;
  - the range for the prediction criteria Pred(25%) is from 32% to 46%.

- For projects 4 and 5:
  - the MMRE worst values are respectively 37% and 24%.
  - the highest SDMRE are respectively 32% and 16%
  - the Pred(25%) for project 4 varies from 55% to 64% (i.e. more than 50% of the sample has an error of less or equal to 25%). For project 5, the value range is better - from 71% to 91%.

In summary, all of the SDMRE values are less than 50% and most of the MMRE are lower than 50% - only project 1 has its MMRE over 50%.

An important observation is that the estimation exercise in scenario C was carried with conditions similar as when estimating in the very early stages of a development project, that is with high uncertainty and little information about the requirements, as it happens often in industry contexts.

In similar situations the variation between the original estimation and the real value reported in some literature is greater than 50% (McConnell, 2006; Laurenz, 2010; The Standish Group

International, 2004, 2009; Project Management Institute, 2004) using just a comparison against the real value (MRE).

The practitioners who participated in the experiment were using the EPCU context that was previously configured using the experience of the other people in a specific organization and who had played the role of expert in the experiment. The estimates are derived by using the EPCU context with the values assigned to the input variables by the practitioners.

By comparison, it can be observed that the EPCU model enables a systematic replication: no matter the level of skills of the people who assign the values for the input variables, the EPCU model generates estimates with less dispersion (see MMRE variations and SDMRE variation in Table 6.11).

Table 6.12 Scenario C – Min-Max Ranges for MMRE and SDMRE for the 5 projects

	Max MMRE	Min MMRE	Max SDMRE	Min SDMRE
Project 1	59%	52%	33%	30%
Project 2	18%	14%	12%	10%
Project 3	46%	39%	32%	28%
Project 4	37%	27%	32%	27%
Project 5	24%	19%	16%	11%

6.7.3 Comparing the Estimation Performance of the EPCU Model with the Expert Judgment Estimation Approach

To evaluate the performance of the EPCU model against the practitioner's "*a priori"* duration estimates (i.e. an experience-based estimation approach), the MMRE, Pred (25%), the SDMRE obtained for the sub-samples using both estimates (those generated by the EPCU model and the practitioners' estimates) were compared.

The leftmost column of Table 6.13 identifies the project ID (from P1 to P5) and the second-left column refers to the quality criteria analyzed. The third column is relative to the quality criteria data obtained from the results while using the EPCU model approach for the 5 projects. The rightmost column reports the data relative to the quality criteria from the experience-based estimation approach.

In addition, in Table 6.13, before the beginning of the data for each project, there is a row that indicates the number of practitioners who participated in the EPCU approach and in the experience-based approach.

It must be noted that the information on the estimates of the practitioners using the experience-based estimation approach was not collected using the "Electronic data collection" described in Phase 2 – Involvement of the practitioners in selecting "*a priori*" input values for each of the projects to be estimated. Only the experience-based estimates were gathered from the physical questionnaire: this is why the numbers of participants are less than 84 for the expert judgment approach. The data about the quality criteria for each project are presented in Table 6.13.

As can be seen in Table 6.13:

- MMRE criterion: the data obtained are considerably better (i.e. lower) for most projects (an improvement varying from 21% to 71%), with the exception of project 1 (with a decrease of 7% in this criterion). See Figure 6.7.
- SDMRE criterion: the data obtained for all projects by the EPCU model are much better (i.e. lower) than that obtained using Expert Judgment Estimation. The best improvement is for project 3: from an SDMRE of 112% for Expert Judgment Estimation, down to 30% for the EPCU model. See Figure 6.8.
- PRED(25%) criterion: the data from the Pred(25%) criterion, shows asimilar performance as SDMRE criterion, in the project 1, the criterion is better for the EPCU model than for the expert judgment approach.

Table 6.13 Results obtained using the EPCU model and Expert Judgment Estimation

Project ID	Quality Criteria	FULL SAMPLE using EPCU	Practitioner experience-based estimation
Practitioner Number		84	52
P1	MMRE	55%	48%
	SDMRE	31%	42%
	Pred(25%)	21%	33%
Practitioner Number		84	51
P2	MMRE	16%	57%
	SDMRE	11%	29%
	Pred(25%)	85%	24%
Practitioner Number		84	52
P3	MMRE	41%	112%
	SDMRE	30%	112%
	Pred(25%)	42%	12%
Practitioner Number		83	53
P4	MMRE	34%	55%
	SDMRE	31%	45%
	Pred(25%)	58%	17%
Practitioner Number		83	51
P5	MMRE	22%	54%
	SDMRE	13%	25%
	Pred(25%)	80%	27%

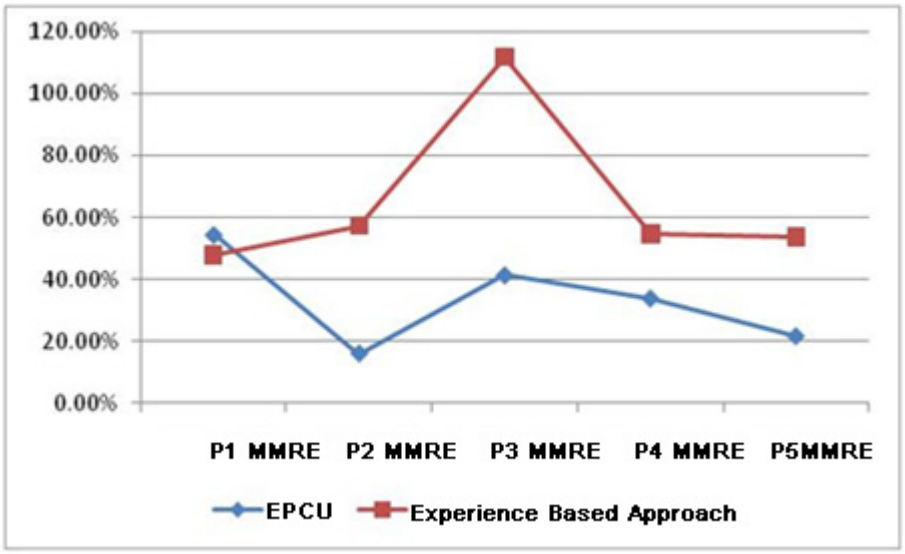


Figure 6.7 Scenario C : MMRE comparisons

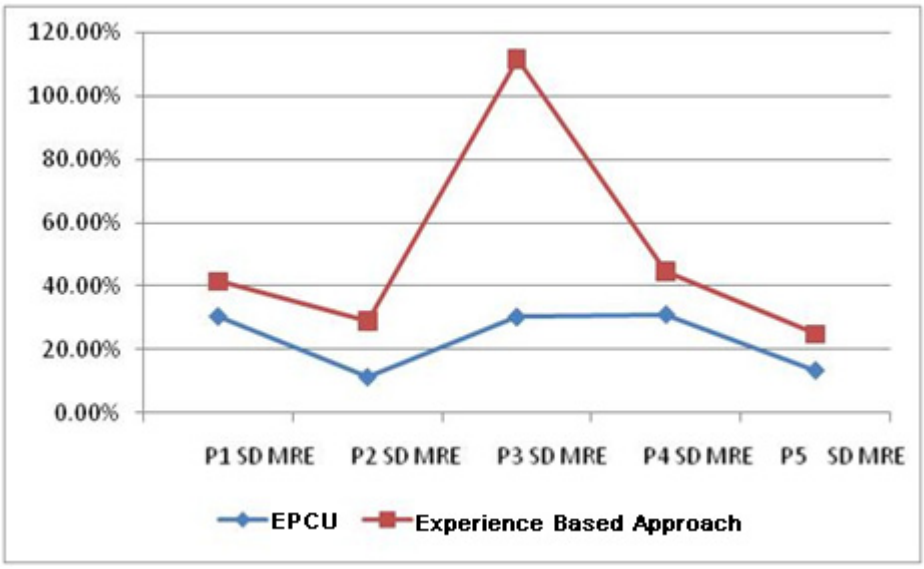


Figure 6.8 Scenario C : SDMRE comparison

In the experiment reported here, the performance of the EPCU model for most of the projects is significantly better than that of the experience-based estimation approach, based on the quality criteria used – see Figures 6.7 and 6.8. When the performance is better using the experience-based approach (i.e. project no. 1), the difference is relatively small: the performance of the two approaches can then be considered as equivalent.

Considering this, under similar experimental conditions the use of the EPCU model in the early phases is preferable to the experience-based estimation approach. This illustrates that the use of the EPCU model can contribute to addressing some of the weaknesses of the experience-based approach.

CHAPITRE 7

ADDITIONAL USES OF THE EPCU ESTIMATION PROCESS

7.1 Introduction

7.1.1 Detailing the EPCU context

For this research work, the EPCU model has been used to estimate, for most experiments, the duration of software projects (i.e. the dependent variable = duration). For this research, the experimentation was structured into 3 scenarios:

- scenario A: 16 projects were used,
- scenario B: 3 other distinct projects were used,
- scenario C: a subset of 5 projects from the 16 projects from scenario A was used.

The total number of projects used was 19 real projects from industry: therefore, 19 corresponding EPCU contexts were defined to estimate the selected output variable (most often the project duration) required to develop these software projects.

The use of the EPCU model could also be extended to define for one specific project several EPCU contexts: for example, one context for each phase of the development process - Figure 7.1.

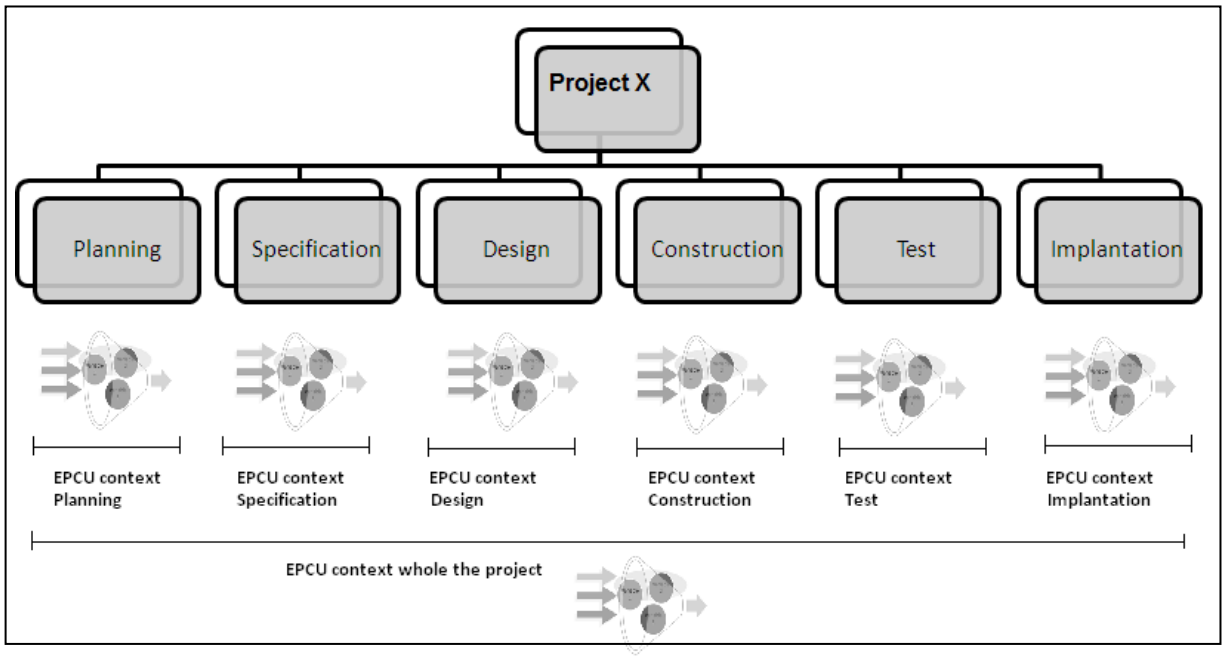


Figure 7.1 EPCU contexts by development phases

Furthermore, the EPCU model could be used to generate different levels of contexts, from the high abstract level (ideally, the software industry) to the more specific level (development of a specific software piece within a software project) - see Figure 7.2.

When the EPCU contexts are related to a more detailed level, the variables used could be more specific. For example: for the estimation of the duration for the whole life cycle of a project, several variables could be identified as having influence for similar projects (i.e. integrated team). As another example, for the specification phase, there may exist different variables related to the team integration that affect this particular phase (i.e. stakeholders commitment, knowledge of the process to implement in a software, level of leadership, and so on). This means that the scope of the variables selected can be related to the scope of the EPCU context.

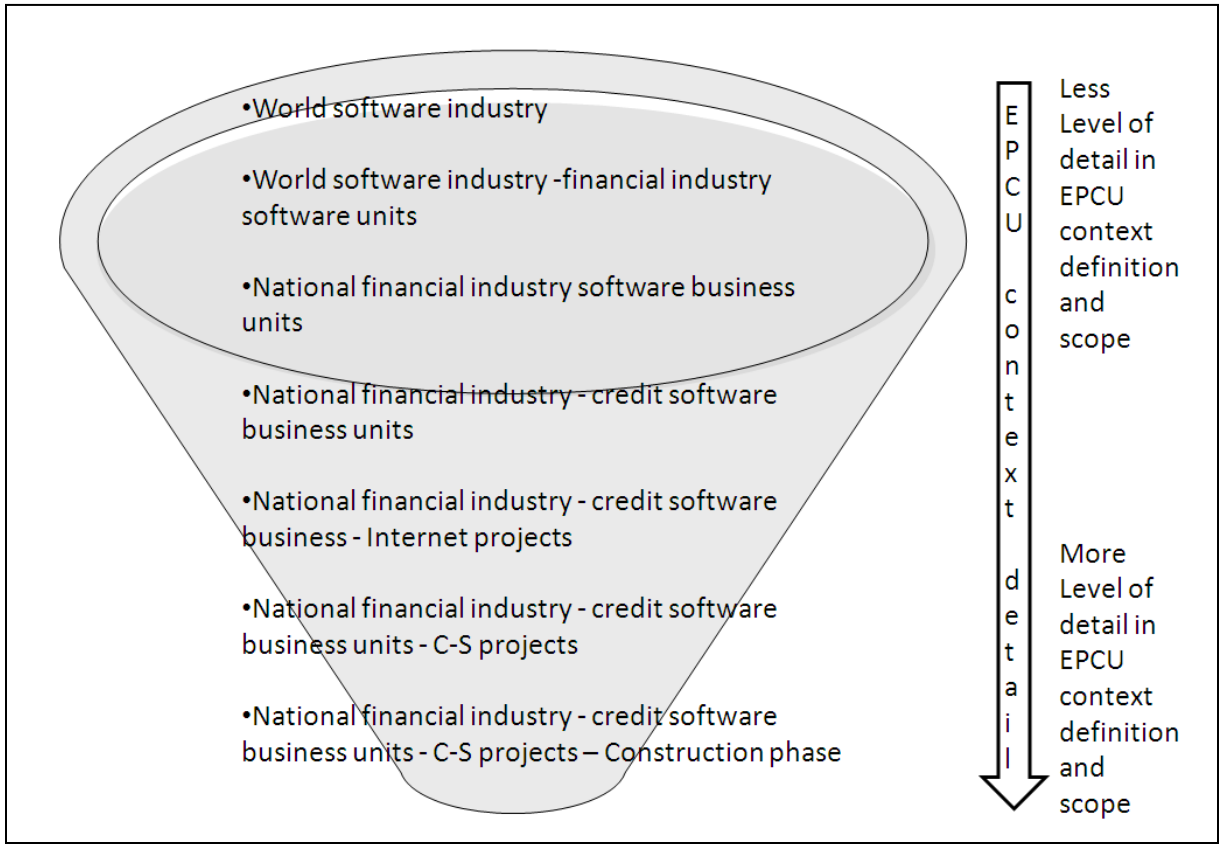


Figure 7.2 Levels of detail - examples of variables in EPCU contexts

7.2 Additional uses for the EPCU model

When the concept of the EPCU model is analyzed outside of the software estimation domain, it can be observed that the experience-based approach could be used in several other fields of human activities. This has been observed by Zadeh (2008) who mentions that the fuzzy logic can be viewed as an attempt to formalize or mechanize two human capabilities:

- the capability to converse, reason and make rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility – in short, in an environment of imperfect information, and
- the capability to perform a wide variety of physical and mental tasks without any measurement and any computation.

When the experience-based approach is used, it presents some of the advantages described in Chapter 1, like:

- Can manage the qualitative or linguistic variables.
- Can manage or work with uncertainty.
- Can create commitment for the people or team to participate.

However, there are also a number of problems in this kind of approach (described in Chapter 1, too), such as:

- the experience is specific to the expert and not to the organization;
- the estimation expertise is neither well described nor well understood;
- this estimation expertise is hard to assess;
- a human is implicit in the social context and the estimation is affected by this social factor: the estimation could be different from one day to another one;
- this estimation expertise cannot be replicated systematically.

The structure of our proposed EPCU fuzzy logic-based estimation model is not specific to software duration estimation and it could be used with different goals. Some other usages of the EPCU estimation model are mentioned next and could be explored in future research.

7.2.1 Portfolio-based selection

There are several studies about software projects portfolio (de Almeida, 2007) (Hunter, 2006) (Mendoza, 2007) (Amoribieta, 2001) (Kotler, 2001) (Barton, 2002); however, most of them are using linguistic variables, nominal classifications, ordinal at most.

Barton (2002) proposes a portfolio-based approach — a simple method that categorizes IT services and aim to enable CIOs to focus on the specific services most important to the business.

Through this approach, CIOs can go beyond aligning IT with business strategy, and can use IT to lead and innovate new ways of doing business. The main idea is to make an assessment of software applications for strategic and operational importance (Figure 7.3).

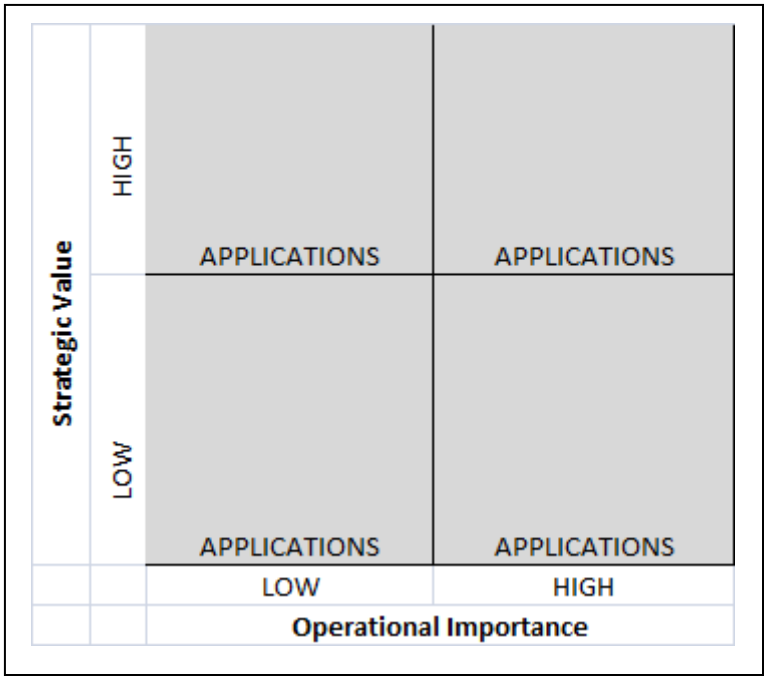


Figure 7.3 Portfolio for strategic and operational importance - Adapted from Barton (2002)

As shown in Figure 7.3, the four quadrants are of a categorical scale type: the assessment of the projects with respect to their strategic value and operational importance is made with an experience-based approach (the quadrants are used only for a classification of the type of applications in terms of strategic value and operational importance).

Hunter (2006) describes a portfolio approach related to the migration of the legacy systems and mentions that the CIO strategy to migrate legacy applications should focus on high business value and high business risk systems. This strategy consists of two parts – see Figure 7.4:

1. To define whether to migrate legacy systems or not; if yes, it should define how and when.

2. To avoid that new systems with high business value and high business risk could be incorporated into the portfolio, constantly reviewing mainly the high-value systems.

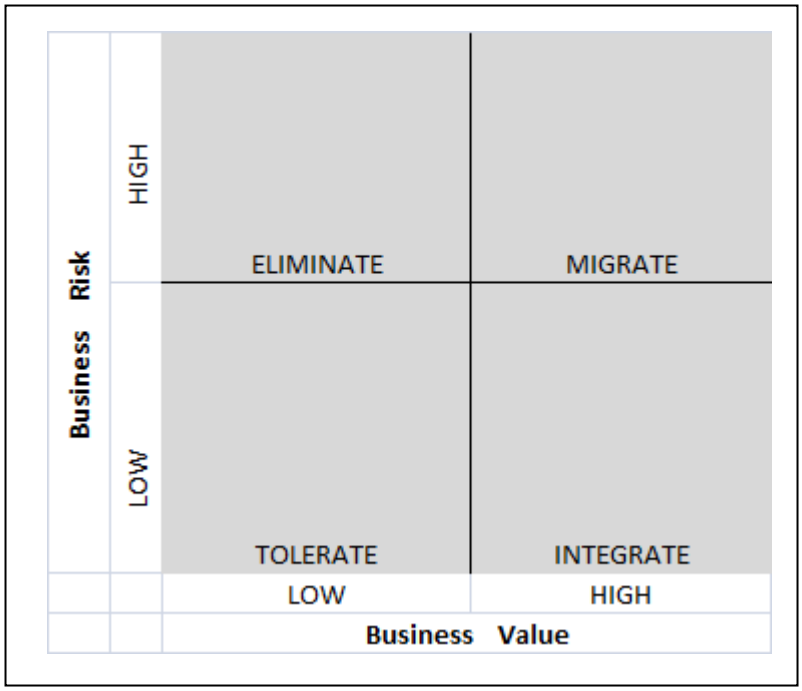


Figure 7.4 Project portfolio classification for managing migration systems - Adapted from Hunter (2006)

Amoribieta (2001) mentions that the focus on outsourcing is increasing in popularity and the primary reason are to save money; however, not all organizations can harvest the benefits from this business model because every organization has special features, information and different types of projects. The portfolio approaches that he proposes is to identify which projects are good candidates to outsourcing (Figure 7.5).

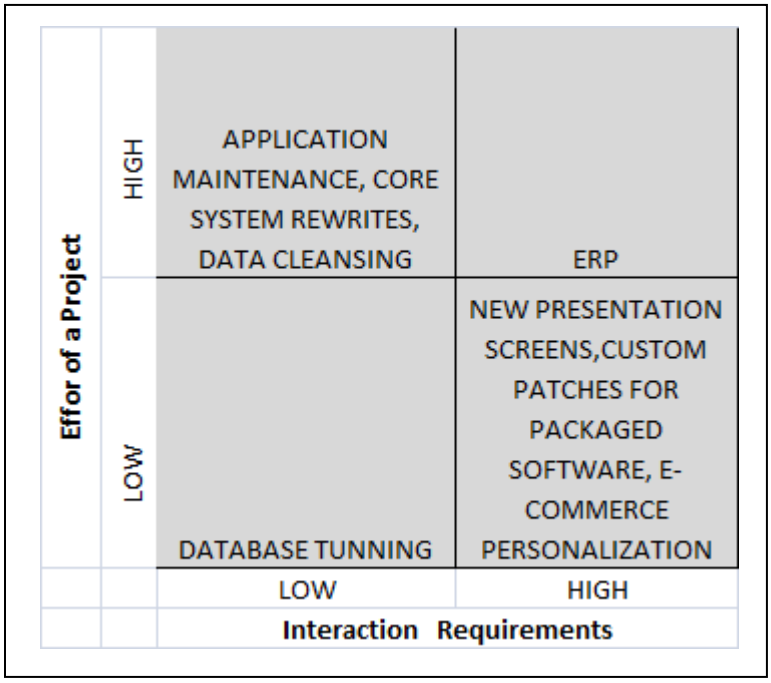


Figure 7.5 Candidates projects to be outsourced - Adapted from Amoribieta (2001)

Using the EPCU model to represent portfolio approaches like proposed by Amoribieta (2001), (Hunter, 2006) and (Barton, 2002) that were defined by the relation between two qualitative variables, can be represented as shown in Figure 7.6.

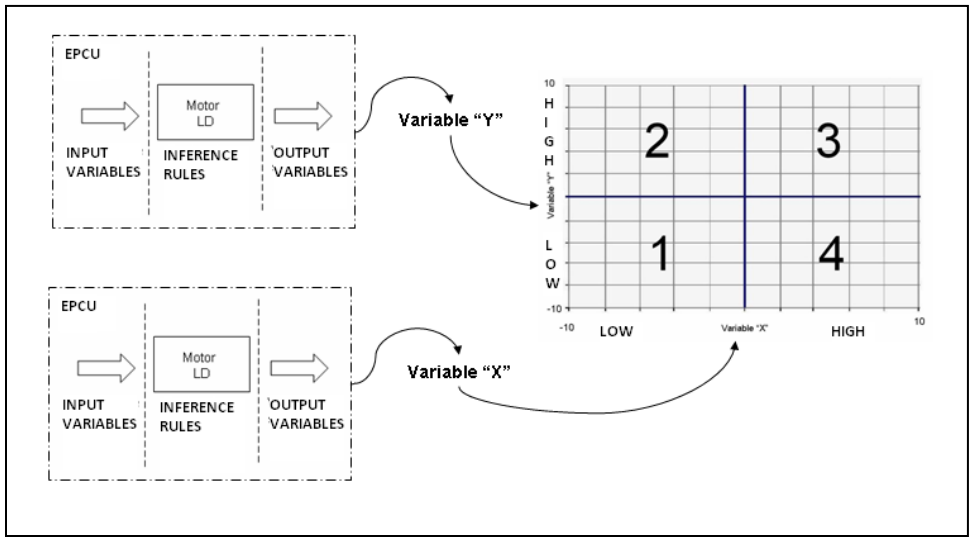


Figure 7.6 Representation of portfolio approach defined by 2 variables relationships using EPCU model

The main advantage of the use of the EPCU model in the portfolio approach is to provide a quantitative approach as a substitute to the subjective approach originally described (Hunter, 2006; Amoribieta, 2001; Kotler, 2001; Barton, 2002).

7.2.2 Projects priorization

The following section describes a case study to illustrate how the EPCU model can be used in prioritizing project initiatives. The case study is taken from a Mexican financial organization.

7.2.2.1 Prioritizing "ad hoc" Initiatives

The "*ad hoc*" approach used by the organization used in the case study begins at the end of each year when the financial institution executes its strategic planning process; in this process the projects to be developed by the organization over the next year will be defined from a list of project initiatives.

The projects initiatives at this stage are only conceptualized: this means that they do not have a detailed business plan for each initiative. However, the people who assess the feasibility of the projects initiatives can provide a perception of the impact of specific variables to each project in contrast to the other projects being assessed.

The set of people who determine the priority of the projects initiatives are the CEO, CFO, CIO and operations director. Each of these people (for research purposes, the people who determine the priorities of the projects initiatives are referred to as 'practitioners') have knowledge, skills and different information. These practitioners could have also different interests: some of the projects may impact directly their own area.

The same set of facts or information will be provided to the practitioners of each project initiative. Based on their experience and considering A) how much the project will contribute to the profits for the organization, B) how complex is the implementation of the project and

C) the cost of the project development, they will assign a priority index to the projects initiatives.

Obviously, depending on the experience and the focus of each of the practitioners and the conditions under which they determine the priority index (the social context), these indexes would give different ratings. To determine a unique consolidated index, the final value will depend on the negotiation between the practitioners: who is more convincing or who holds a higher position?

Assuming the availability of an experience-based model to determine a priority index, and that it is used to prioritize the projects initiatives for the year *n*, what could happen for the next year? In the strategic planning of the next year (*n* +1), different people will probably participate in the project initiatives assessment. In this sense, it is very likely that the considerations made in year n to determine the priority index will be different from those considered in year (*n* +1), and the following (*n* +2…*m*).

An "*ad hoc*" approach for projects initiatives assessment as described above is not like a systematic assessment approach, i.e. in the "*ad hoc*" approach, a specific project for a certain year could be considered with less priority (lower priority index) under certain constraints, while in another year it might be considered very important (high priority index) under the same assumptions since the experience of the practitioners who determine the priority index for each event could lead to the differences.

7.2.2.2 EPCU Model for Prioritizing Initiatives

The use of EPCU model enables a replicable framework of knowledge (experience systematic replication) by which consistent results could be obtained from the same facts; this is feasible with the use of fuzzy logic.

With the EPCU model, project initiatives could be assessed in the same way, but with inputs provided by different people (practitioners): in the EPCU model, the assessment model has already been defined previously without knowing the projects initiatives that have to be assessed in the strategic planning process, and it is considering how the experts make inferences and using a set of variables previously defined.

The possibility to define the inference rules and with it to preserve the knowledge for the organization is a big advantage: this will help reduce the dependency on key people.

The need expressed by the financial institution which wanted to be covered by its strategic planning process was:

- Have a formal mechanism for evaluation of project initiatives with incomplete information.
- Do not depend on the experts for the evaluation of initiatives.

The steps to configure the EPCU model were carried on in conjunction with the experts from this organization.

7.2.2.3 Identification/Definition of the input variables

Taking into consideration the above assumptions, a context was created for this financial institution in the EPCU model to evaluate the initiatives.

Considering the EPCU "context" definition (See Chapter 4), the input variables were defined in interviews with the experts, using the same procedure defined in Chapter 6 for the estimation of software projects.

This definition is important because in the use of the EPCU model, the practitioners will assign a value as input, representing their opinion about the context of the project, rather than

asses/estimate directly the output variable (in this case priority index) using an intuitive approach.

The input variables defined by the financial organization experts were:

- profit expected to the business for the execution of a specific project,
- the complexity considered while developing the project, and
- the effort estimated.

When the strategic planning is carried on, these variables are subjective and typically each project would not have its own business plan.

When the projects initiatives were assessed, they were described in linguistic terms instead of a precise quantitative value; this means they were assessed based on experience as well as on opinions provided by experienced practitioners.

The selected numerical range that represent the opinions of the practitioners for each of these variables is [0, 5], where 0 is the minimum value (lowest) and 5 is the maximum value (highest) - see Chapter 4, Step 4 - Fuzzification.

7.2.2.4 Specification of the output variable

The selected output variable, which is the priority index, can be defined as a percentage [0,100] in order to identify the projects with higher prioritization.

For the output variable (the priority index) the membership function defined has four linguistic values (LOW, AVERAGE, HIGH, and VERY HIGH). In addition, the defined range is [0,100], the unit is a percentage; the highest priority for a project initiative (i.e.the priority index) will be close to 100.

7.2.2.5 Generation of Inference Rules

Once the input variables and the output variable defined, the next step is to define the inference rules in order to link them.

These inference rules were defined by the experts of the organization, supported by the researcher.

7.2.2.6 Prioritizing the project initiatives with the EPCU model

The financial institution provided a set of 11 proposed project initiatives for prioritization using the EPCU context defined. These initiatives were assessed in the strategic planning exercise for 2009. The project initiatives list is shown in Table 7.1.

Table 7.1 Project Initiatives List

ID	**Project initiative Name**
P1	AMS SEARCH SPACE MEXICO FASE2
P2	SISPAGOS
P3	SCOTIANÓMINA
P4	CALIFICACIÓN CARTERA COMERCIAL
P5	PISCO
P6	MEJORAS AL PROCESO DE CALIFICACIÓN
P7	CÁLCULO DE CAPITAL PARA OPICS Y SIBUR
P8	INFORMACIÓN DE COSECHAS PARA CRÉDITOS A
P9	BURSATILIZACIÓN DE LA CARTERA HIPOTECARI
P10	TASA DE ACUERDO AL RIESGO
P11	FORMULARIO OPERACIONES EN DÓLARES

Table 7.2 EPCU Model for Project Initiatives Prioritization List: Input variables values and estimated priority index

ID	Project initiative	Profit expected to the business	Effort estimated	Complexity considered	Priority index
P1	AMS SEARCH SPACE MEXICO FASE2	3.5	2	3	53%
P2	SISPAGOS	2	3.5	4	15%
P3	SCOTIANÓMINA	1.5	3	2	28%
P4	CALIFICACIÓN CARTERA COMERCIAL	4	2	2	61%
P5	PISCO	2.5	3	3	29%
P6	MEJORAS AL PROCESO DE CALIFICACIÓN	5	2	2	61%
P7	CÁLCULO DE CAPITAL PARA OPICS Y SIBUR	3.5	1.5	2	68%
P8	INFORMACIÓN DE COSECHAS PARA CRÉDITOS A	2	3	3	26%
P9	BURSATILIZACIÓN DE LA CARTERA HIPOTECARI	5	2	3	57%
P10	TASA DE ACUERDO AL RIESGO	3	2.5	3.5	35%
P11	FORMULARIO OPERACIONES EN DÓLARES	3.5	2	1	69%

With the list of project initiatives, the prioritization was made in a session with the people that usually determine the priority of the projects in "*ad hoc*" manner; for this case study, the experts play the practitioners role too.

Using the Delphi method (Harold, 1975), the value assignment for the input variables was obtained for each project. Once the values assigned to the input variables for each project initiative, the EPCU model was executed and the priority indexes were obtained. Table 7.2 shows the results obtained (prioritization index) using the EPCU model, and the Figure 7.7 shows the results plotted.

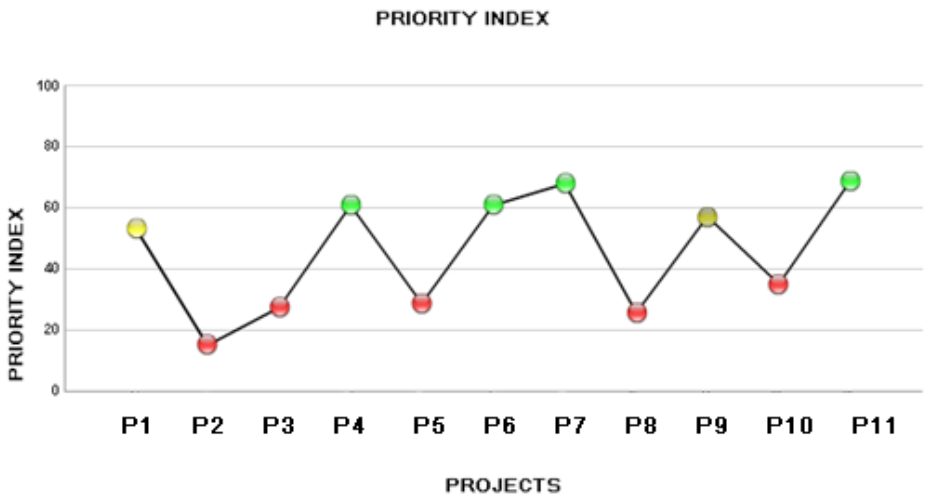


Figure 7.7 Results of the EPCU prioritization of Project Initiatives

7.3 Summary

The proposed EPCU model is an estimation process based on fuzzy logic that mimics the way experts make estimates: the EPCU model has been designed to be used in the early stages in the project life cycle when there is high uncertainty and the information about the project is vague (i.e. described usually by linguistic variables).

However, this situation (high uncertainty and a lack of information) is not only present in software estimation. In this section examples have been presented related to the projects portfolio approaches and the prioritization of project initiatives.

As in the software estimation, the use of the EPCU model could contribute to addressing some of the weaknesses of the experience-based approach and could generate benefits in those fields.

CONCLUSION

The estimation of software projects is very important: the resources for the project are assigned and managed throughout the development life cycle of the projects on the basis of such estimation.

Software project estimates often have to be made early in the project life cycle: this implies that these estimates are to be made in a highly uncertain environment on the basis of information that is vague and incomplete. In practice, the estimation approach most used at this early stage is the experience-based approach (also called: expert judgment, intuitive approach, "*ad hoc*" way, subjective way, research intuition, and so on).

However, there are a number of problems with the experienced-based approach, such as: the expertise is specific to the people and not to the organization, and this intuitive estimation expertise is neither well described nor well understood. In addition, the expertise is difficult to assess and cannot be replicated systematically.

To address some of these problems, this thesis has proposed a more formal process – the EPCU model, based on fuzzy logic, to leverage the experience-based approach to generate estimates in the early stages.

The research goal of this thesis was to design a software estimation process able to manage the lack of detailed and quantitative information embedded in the early phases of the software development life cycle.

The research strategy aimed to leverage the advantages of the experience-based approach that can be used in early phases of software estimation while addressing some of the major problems generated by this estimation approach by experienced-based judgments.

The specific research objectives to be met by this improved software estimation process were:

A. The proposed estimation process must use relevant techniques to handle uncertainty and ambiguity in order to consider the way practitioners make their estimates: the proposed estimation process must use the variables that the practitioners use (qualitative).

B. The proposed estimation process must be useful in early stages of the software development process.

C. The proposed estimation process needs to preserve the experience or knowledge base for the organization: this implies an easy way to define the experience of the experts.

D. The proposed model must be usable by people with skills distinct than the people who configure the original context of the proposed model.

The proposed EPCU model (Estimation of Projects in Contexts of Uncertainty) is an estimation process based on fuzzy logic that mimics the way experts make estimates: the EPCU model has been designed to be used in the early stages in the project life cycle when there is high uncertainty and the information about the project is vague (i.e. described usually by linguistic variables).

The next paragraphs summarize how each of these research objectives has been met, as illustrated with the outcomes of the experiments reported in Chapter 6.

Objective A

The main element of the estimation process is the use of fuzzy logic: the elements proposed by the fuzzy logic make it useful in the management of uncertainty and imprecision, such as the fuzzy sets theory which is basically a theory of classes with unsharp boundaries.

With the use of fuzzy logic the research objective A was met: as its creator mentions (Zadeh, 2008) “Basically, fuzzy logic is a precise logic of imprecision and approximate reasoning. More specifically, fuzzy logic may be viewed as an attempt at formalization/mechanization of two remarkable human capabilities: First, the capability to converse, reason and make

rational decisions in an environment of imprecision, uncertainty, incompleteness of information, conflicting information, partiality of truth and partiality of possibility – in short, in an environment of imperfect information. Second, the capability to perform a wide variety of physical and mental tasks without any measurements and any computations"

Other considerations to use the fuzzy logic were mentioned in Gray *et al.* (1997) in their comparison of modeling capabilities of distinct techniques in the software estimation field: with the fuzzy logic approach, all studied criteria are fully satisfied, to the exception of only two criteria that are met only partially (criteria: "can resist outliers" and the "can be adjusted for new data").

Objective B

In the early phases of the project, most of the variables are linguistic, or qualitative, and more often the estimates are developed within an uncertainty environment. While the objective A can be reached with the EPCU model, it is possible to state that the EPCU model can be used in such early phases of the software development life cycle: in the experiments reported in this thesis, the performance of the EPCU estimation process for most of the projects is significantly better than that of the experienced-based estimation approach, based on the quality criteria used. When the performance is better using the experienced-based estimation approach, the difference is small, so the performance can then be considered equivalent.

Considering this, the use of the EPCU model in the early phases is preferable to the experienced-based estimation approach, under similar experimental conditions - See Figure 6.7 and Figure 6.8 and Annex XI.

Objectives C and D

One of the main weaknesses of the experienced-based estimation approach is that the experience belongs to the expert, so it is hard to assess such experience, and once the expert moves out of the organization, such a valuable experience is lost.

The experience-based estimation approach can be described and stored through inference rules, such as in the EPCU model, and can become part of an organization's assets. This constitutes a valuable solution for some of the problems described with the "*ad hoc*" experience-based estimation technique.

The systematic replication of estimation experience (the use of the expert's experience by other people with distinct experience and skills) is a basic element in this research. The definition by experts of an estimation context in the EPCU model represents the experience through the inference rules that the experts use to make the estimation; these EPCU inference rules can next be used by other people who do not have the same experience level.

In scenario C, for all the practitioners' classification, it has been observed that the EPCU model performance is similar to the full sample. And all of the SDMRE values are less than 50% and most of the MMRE are lower than 50% - only project 1 has its MMRE over 50%.

An important observation is that the estimation experiment in scenario C was similar as when estimating in the very early stages with high uncertainty and little information in terms of requirements, as it often happens in an industry context.

In similar situations the variation reported in some literature is greater than 50% (Boehm, 1981; McConnell 2006; Standish Group, 2004) between the estimates and the real values (MRE).

By comparison, it can be observed that the EPCU model enables a systematic replication: whatever the level of skills of the people who assign the values for the input variables, the EPCU model generates estimates with less dispersion than the experience-based approach for the projects analyzed.

It is important to observe that while using the EPCU model does not require accurate historical data, it requires the experts' experience for the set up of the configuration of the

EPCU contexts to be used for estimation purposes. Once configured, the EPCU estimation model can be used by people who lack experience in the type of projects to be estimated under a specific EPCU context.

Publications to date

- Valdés, F., A. Abran. 2007. « Industry Case Studies of Estimation Models based on Fuzzy Sets ». In: *IWSM-Mensura 2007*, (UIB-Universitat de les Illes Baleares, Palma de Mallorca, Spain), p. 87-101. Editors: Abran-Dumke-Màs, Publisher: Proceedings of the IWSM-Mensura 2007. ISBN 978-84-8384-020-7, Nov. 5-9, 2007.
- Valdés, F., A. Abran. 2010. « Comparing the Estimation Performance of the EPCU Model with the Expert Judgment Estimation Approach Using Data from Industry ». In: *Software Engineering Research, Management and Application 2010 (SERA 2010)*. (Montreal, Canada. May 24-26, 2010), p. 227.240, chapter 15. Verlag, Berlin: 'Studies in Computational Intelligence',Vol 296, Springer. ISBN:13: 9781615209750
- Valdés, Francisco. 20011. « La Estimación de Proyectos de Software: Un Problema una Solución ». *Software Guru*, Año 2011 No. 32 Mayo Julio 20011.
- Valdés, Francisco. 20011. « La Estimación de Proyectos de Software: Un Problema una Solución ». *Software Guru*, Año 2011 No. 33 Agosto Octubre 20011.

EPCU model current limitations

The EPCU model defines the rule base considering the experience for a specific organization. The rulebase aims to represent the experience and to enable the use of that experience without the expert presence. When in an organization there is no expertise because a completely new type of project needs to be estimated, the limitations are the same as in the experience-based estimation approach: there is a lack of experience to define an EPCU context appropriate to a specific context or instantiation. This is considered as a current limitation of the proposed EPCU estimation process.

Another current limitation of the EPCU model is that the specific impact (or weight) of each input variable is the same; thist means that the conditions appearing in the antecedent portion of the production rules, all have the same degree of importance (Chen, 1993).

In the practice, any input variable could be different. If this assumption can be considered in the model, this could allocate more flexibility to the EPCU model. It could be relevant to consider factors that represent the importance of the input variables in the calculation of the final value. This needs to be investigated in further resarch work.

For the research reported in this thesis, the scope for the estimation using the EPCU context was the entire project: i.e. the total project duration. Although, intuitively with more details more precision could be obtained because more specific variables would be considered (Figure 7.2). This needs further research, and this was considered as out of the scope of the research work reported here.

The fuzzy operator used in this research was the Zadeh Operator; however it could be interesting as a further research to analize the performance of distinct fuzzy operators using the same expert's opinions.

All of the above elements illustrate how the use of the EPCU model could contribute to addressing some of the weaknesses of the experience-based approach and generates benefits to the software engineering field.

BIBLIOGRAPHY

Abran, Alain, A. Sellami and W. Suryn. 2003. «Metrology, Measurement and Metrics in Software Engineering», International Software Metrics Symposium, IEEE-METRICS 2003. pp. 2-11.Sydney, Australia: IEEE Computer Press, Los Alamitos.

Abran, Alain, J. W. Moore, P. Bourque, R. Dupuis and L. Tripp, Guide to the Software Engineering Body of Knowledge - 2004 Version – SWEBOK: IEEE-Computer Society Press, April 2005, 200 pages, isbn:0-7695-2330-7.

Abran, Alain. 2008. Software Benchmarking, Estimation and Quality Models Based on Functional Size with COSMIC – ISO 19761, Draft April 2008. MGL-841: La mesure: Concept clef en ingénierie du logiciel: Programme de Doctorat en genie.

Abran, Alain. May 2010. Software Metrics and Software Metrology, John Wiley & Sons Interscience and IEEE-CS Press, New Jersey, p. 328, ISBN:978-0-470-59720-0.

Abran, Alain. 1998. «Software Metrics Need to Mature into Software Metrology (Recommendations) ». NIST Workshop on Advancing Measurements and Testing for Information Technology (IT). (Gaithersburg Maryland, 1998).

Albrecht, Allan J. 1979. «Measuring Application Development Productivity»,IBM Applications Development Symposium. (Monterey, CA, Oct 14-17, 1979), p. 83: GUIDE Int and Share Inc., IBM Corp.

Amoribieta, Iñigo, K. Bhaunik, K. Kanakamedala, A. D. Parkhe. May 2001. Programmers abroad: A primer on offshore software development. Coll. «McKinsey Quarterly . McKinsey & Company.

Barton, N. 2002. Business Innovation Through IT A Realistic Approach to Supporting Organizational Strategy. Compass America Inc.

Boehm, Barry W. 1981. Software Engineering Economics, Englewood Cliffs. NJ: Prentice-Hall, Inc.

Booch, Grady. 1996. Análisis y Diseño Orientado a Objetos con Aplicaciones. 2ª edición en español: Addison-Wesley.

Bourque, Pierre, S. Oligny, A. Abran, B. Fourrnier. 2007. « Developing Project Duration Models in Software Engineering ». Journal of Computer Science and Technology, vol. 22, p. 348-357.

Buckley, J. J. and Y. Hayashi. 1994. « Can Fuzzy Neural Nets Approximate Continuous Fuzzy Functions », Fuzzy Sets and Systems, vol. 61, p. 43-51.

Buglione, L., J. J. Cuadrado-Gallego and J. A. Gutiérrez de Mesa. November 2008. « Project Sizing and Estimating: A Case Study Using PSU, IFPUG and COSMIC ». Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 5338/2008, ISBN: 978-3-540-89402-5.

Casals, Mas i O 1997. « Sistemas difusos dinámicos para el tratamiento de información temporal imprecisa », Escola Tècnica Superior D'enginyeria de Telecomunicació de Barcelona (UPC).

Charette, N. R. September 2005. Why software fails. Coll: «Special report», IEEE Spectrum.

Chen, Shyi-Ming. 1993. « A new Methodology for Fuzzy Control Based on Weighted Fuzzy Logics ». In: IEEE TENCON 93, (Beijing, 1993).

Chrissis, M. B., M. Konrad and S. Shrum. 2007. CMMI Guidelines for process Integration and Product Integration, 2nd Edition: Addison-Wesley.

Condori-Fernandez, N, O Pastor, A Abran and A Sellami. 2008. «Introduciendo Conceptos de Metrologia en el Diseño de Medidas de Software». In: XI Iberamerico Workshop on Requirements Engineering and Environments, IDEAS 2008. (Pernambuco, Brasil, 2008), p. 112-125.

COSMIC Measurement Practice Commitee. 2007. « The COSMIC Functional Size Method Version 3.0, Advanced and Related Topics ». En ligne <http://www.cosmicon.com/portal/public/COSMIC%20Method%20v3.0%20Advanced%20&%20Related%20Topics.pdf>. Consulté le September 4th 2010.

De Almeida, I. 2007. «Equilibrar el riesgo». InformationWeek México, septiembre 5, 2007, Núm. 169.

De Wit, A. 1998. «Measurement of Project Success». Project Management Journal, Vol.6(3).

Dekkers, C. A. 2005. «reating Requirements-Based Estimates before Requirements are Complete », CrossTalk The journal of the Defense Software Engineering, April 2005 Issue.

De Marco, Tom. 1982. Controlling Software Projects. Englewood Cliffs, N.J.: Prentice Hall.

Dickes, P. J. Tournois, A. Fieller and J-L. Kop. 1994. La psychométrie. Coll. « Le Psychologue », Paris 1994 : Presses Universitaires de France (PUF).

Fincham, R. 2002. «Narratives of success and failure in systems development». British Journal of Management. Vol. 13.

Gray, A. and S. MacDonell. 1997. «A Comparison of Techniques for Developing Predictive Models of Software Metrics». Information and Software Technology. Vol. 39, p. 425-437.

Gray, A. R. and S. G. MacDonell. 1997. «A comparison of model building techniques to develop predictive equations for software metrics». Information and Software Technology - INFSOF Journal.

Gray, A. and S. MacDonell. 1999. «Software Metrics Data Analysis – Exploring the Relative Performance of Some Commonly Used Modeling Techniques», The Information Science Discussion Paper Series, Number 99/11, June 1999.

Habra, N, A. Abran, M. Lopez, A. Sellami. 2008. «A Framework for the Design and Verification of Software Measurement Methods». Journal of Systems and Software, Elsevier 81: 5. 633-648.

Halstead, M. H. 1977. « Elements of Software Science », Elsevier, New York: North-Holland, 1977.

Harold, A. Linstone and Turoff Murray. 1975. The Delphi Method: Techniques and Applications. Adisson-Wesley, ISBN: 9780201042948.

Hill, J., L.C. Thomas and D. E. Allen. 2000. «Experts' estimates of task durations in software development projects». International Journal of Project Management. Vol. 18, n° 1, p 13-21. Feb. 2000.

Hofstede, G. 1999. «The universal and the specific in 21st-century global management». Organizational Dynamics. Vol. 28, n° 1, 1999.

Horikawa, S., T. Furnuhashi and Y. Ucikawa. 1992. «On Fuzzy Modelling Using Fuzzy Neural Networks with the Back-Propagation Algorithm», IEEE Trans. Neural Networks 3, p 801-806 (1992).

Carnegie Mellon University. Carnegie Mellon School of Computer Science, 2009. En ligne. <http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html>.

Hunter, R. 2006. High Value, High Risk: Managing the Legacy Application Portfolio. Coll. «Gartner Executive Programs (EXP) para Latinoamérica», Gartner.

Idri, A. and A. Abran. 2000. «Towards A Fuzzy Logic Based Measures for Software Projects Similarity». In: 6th MCSEAI'2000 – Maghrebian Conference on Computer Sciences. (Fez, Morocco, 2000).

Idri, A., A. Abran and T. M. Khosgoftaar. 2001. «Fuzzy Analogy: A New Approach for Software Cost Estimation». In: International Workshop on Software Measurement (IWSM'01). (Montréal, Québec, 2001).

Idri, A., A. Abran, T. M. Khosgoftaar and Robert S. 2004. « Fuzzy Case-Based Reasoning Models for Software Cost Estimation ». Soft Computing in Software Engineering: Studies in Fuzziness and Soft Computing, Springer-Verlag, 2004.

Idri, A., A. Abran, T.M. Khoshgoftaar and S. Robert. 2002. « Estimating Software Project Effort by Analogy Based on Linguistic Values ». In: 8th IEEE International Software Metrics Symposium. (Ottawa, Ontario, 2002), p. 21-30.

IFPUG. 2005. Function Point Counting Practices Manual, Version 4.2.1: International Function Points Users Group.

Information Systems Audit and Control Association (ISACA) and IT Governance Institute (ITGI). 2011. Control Objectives for Information and related Technology (COBIT). In Le site de ISACA. En ligne.< http://www.isaca.org >. Consulté February 2010.

International Function Points User Group. 2011. En ligne.< http:// www.ifpug.org>. Consulté February 2010, (http:// /).

International Organization for Standardization (ISO). 2007. International Vocabulary of Metrology – Basic and General Concepts and Associated Terms, VIM, ISO/IEC Guide 99-12:2007, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

Jacobson, I., M. Christerson, P. Jonsson and G. Övergaard. 1998. Object-Oriented Software Engineering a Use Case Driven Aprproach. 4th printing, ESSEX Inglaterra: Addison-Wesley.

Jacquet, J.P. and A. Abran.1997 « From software metrics to software measurement methods: a process model ». In: 3rd International Symposium and Forum on Software Engineering Standards, ISESS'97. (Walnut Creek (CA)).

Jacquet, J. P. and A. Abran. 1999. « Metrics Validation Proposals: A Structured Analysis In:Software Measurement - Current Trends in Research and Practice ». In: 8th International Workshop on Software Measurement - IWSM 98, (Deutscher Universität Verlag, 1999), p. 43-59.

Jang, R.J.-S. 1993. « ANFIS: Adaptive-Network-Based Fuzzy Inference System ». IEEE Trans. Systems, Man, and Cybernetics 23, p. 665-685.

Jensen, R.W. and C.C. Tonies. 1979. Software Engineering. Englewood Cliffs NJ: Prentice-Hall, Inc.

Jones, C. 2006. « Social and Technical Reasons for Software Project Failures ». CrossTalk The journal of the Defense Software Engineering. June 2006 Issue.

Jones, C. 2005. « Software Cost Estimating Methods for Large Projects ». CrossTalk The journal of the Defense Software Engineering. April 2005 Issue.

Jones, C. 2004. « Software Project Management Practices: Failure Versus Success ». CrossTalk The journal of the Defense Software Engineering. October 2004 Issue.

Kadoda, G., M. Cartwright, L. Chen and M. Shepperd. 2000. « Experiences Using Case-Based Reasoning to Predict Software Project Effort ». In: EASE Conference. (Keele , UK, 17-19 April 2000).

Kitchenham, B. 1997. « The Problem with Function Points ». IEEE Software, Vol. 14, Issue 2, Mar/Apr 1997. p.29 - 31.

Kolodner, J.L. 1993. Case-Based Reasoning. San Mateo, CA.: Morgan-Kaufmann Publishers.

Kotler, Philip. 2001. Dirección de marketing, Edición del Milenio: Prentice Hall.

Laurenz, E. J. and C. Verhoef. 2010. « The Rise and Fall of the Chaos Report Figures ». IEEE Computer Society. Vrije Universiteit Amsterdam, IEEE 2010.

Lavagnon, A. Ika.2009. « Project success as a topic in project management journals ». Project Management Journal, vol. 40 n° 4.

Lawrence, P. S., R. Jeffery, B. Curtis and B. Kitchenham. 1997. « Status Report on Software ». IEEE Software, vol. 14, Issue 2, Mar/Apr 1997, p. 33 - 43.

Mccabe, T. J., A. H. Watson and D. R. Wallace. 1996. Structured Testing : A Testing Methodology Using the Cyclomatic Complexity Metric. NIST, Special Publication 500-235, (Gaithersburg, MD 20899-0001, September, 1996): Computer Systems Laboratory, National Institute of Standards and Technology.

Mccabe, T. J. and A. H. Watson. 1995. Complexity in Software Development Systems Management Development. Auerbach. 1995.

McConnell, S. 2006. Software Estimation: Demystifying the Black Art. Microsoft Press. ISBN 0-7356-0535-1.

Milos, Manic. 1999. « Fuzzy Operators Weight Refinements ». In: Annual Reliability and Maintainability Symposium, RAMS'99. (Washington, DC USA, January 18-21 1999), p. 245-251: IEEE Reliability Society.

Normalización y Certificación Electrónica A.C. 2005. Modelo de Procesos para la Industria del Software. Norma Oficial Mexicana, NMX-I-059/01-NYCE-2005, NMX-I-059/02-NYE-2005.

Morgenshtern, O., T. Raz and D Dovir. 2007. « Factors affecting duration and effort estimation errors in software development projects », Information and Software Technology, Vol. 49, n° 8, Aug. 2007, p. 827-837.

Myrtveit I. And E. Stensrud. 1999. « A Controlled Experiment to Asses the Benefits of Estimating with Analogy and regression Models ». IEEE Transaction on Software Engineering, vol. 25, Issue: 4, p. 510-525. 25- 4 July/August, 1999.

Office of Government Commerce of United Kingdom, Information Technology Infrastructure Library (ITIL). En ligné < http://www.ogc.gov.uk/>.

Oligny, S., P. Bourque, A. Abran and B Fournier. 2000. « Exploring the Relation Between Effort and Duration in Software Engineering Projects » In: World Computer Congress 2000, (Beijing China August 21-25, 2000), p.175-178, Proceedings of the World Computer Congress 2000.

Park, R. E., W. B. Goethert and J. T. Webb. 1994. « Software cost and schedule estimating: A process improvement initiative ». Software Engineering Institute, Pittsburg, U.S.A, <http://www.sei.cmu.edu/pub/documents/94.reports/pdf/sr03.94.pdf>. Consulting in November 2009.

Peñaloza, R. E. 1996. Apuntes, Fundamentos de Programación, 2a edición, UNAM, ENEP Aragón.

Pinto, J. K. and, D. P. Slevin. 1988. « Project success: definitions and measurement techniques », Project Management Journal, vol.19, n° 1.

Ponce. C. P. 2010. Inteligencia Artificial Con Aplicaciones a la Ingeniería, México: Alfaomega, Julio 2010.

Pressman, S. R. 1993, Ingeniería de Software un Enfoque Práctico, 3a edición, España: McGraw-Hill.

Project Management Institute (PMI). 2004., PMBOK Guide - A Guide to the Project Management Body of Knowledge, 3th Edition. Project Management Institute.

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

Santillo, L. and C. Grande. 2006. « Breve storia della Misurazione del Software ». GUFPI-ISMA, Newsletter, Vol. 3, n° 1, 15 Gennaio 2006.

Shepperd, Martin, Chris Schofield, and Barbara Kitchenham. 1996. « Effort Estimation Using Analogy ». In: 18th International Conference on Software Engineering (ICSE18), (Berlin, 25-29 March 1996), p. 170-178.

Shepperd, Martin and Chris Schofield. 1997. « Estimating Software Project Effort Using Analogies », IEEE Transactions on Software Engineering, Vol. 23, n° 12, p. 736 - 743, November 1997.

Shore, Barry. 2008. « Systematic biases and culture in project failures », Project Management Journal, Vol. 39, n° 4.

Smith M. and S. Tockey. 1988. An Integrated Approach to Software Requirements Definition Using Objects, Seattle WA: Boeing Commercial Airplane Support Division, p. 132.

Mendoza, Artemio. 2007. « No todo lo que brilla es oro ». Software Guru, Año 03 No. 3 Mayo Junio 2007.

Stamey, K. 2006. « Why Do Projects Fail? », CrossTalk The journal of the Defense Software Engineering, June 2006 Issue.

Stensrud, E., T. Foss, B. Kitchenham and I. Myrtveit. 2002. « An Empirical Validation of the Relationship Between the Magnitude of Relative Error and Project Size ». In: Eighth IEEE Symposium on Software Metrics (METRICS.02). (Ottawa, Canada, 4-7 June 2002), (Washington, DC, USA) : IEEE Computer Society 2002, ISBN 0-7695-1339-5.

Sybase Inc. 1996. Building Object-Oriented Applications In Power Builder Student Guide, Powersoft, Nov 1996.

Standish Group International. 2009. CHAOS Summary 2009. Coll. « Research Reports », The Standish Group International, Inc.

Standish Group International. 2004. Extreme Chaos Report. « Research Reports », The Standish Group International, Inc.

Timothy, B. 1991. An Introduction to Object-Oriented Programming, Corrections Edition, Oregon State University: Addison-Wesley Publishing Company, april 1991.

Timothy, K. P. 2006. « Knowledge: The Core Problem of Project Failure », CrossTalk The Journal of the Defense Software Engineering, June 2006 Issue.

Valdés, F., A. Abran. 2007. « Industry Case Studies of Estimation Models based on Fuzzy Sets ». In: IWSM-Mensura 2007, (UIB-Universitat de les Illes Baleares, Palma de Mallorca, Spain), p. 87-101. Editors: Abran-Dumke-Màs, Publisher: Proceedings of the IWSM-Mensura 2007. ISBN 978-84-8384-020-7, Nov. 5-9, 2007.

Valdés, F., A. Abran. 2010. « Comparing the Estimation Performance of the EPCU Model with the Expert Judgment Estimation Approach Using Data from Industry ». In: Software Engineering Research, Management and Application 2010 (SERA 2010). (Montreal, Canada. May 24-26, 2010), p. 227.240, chapter 15. Verlag, Berlin: 'Studies in Computational Intelligence',Vol 296, Springer. ISBN:13: 9781615209750.

Valdés. F. 2010. Modelando la realidad - cuantificando lo intangible, « Essay Award 2010 ». Fundación EVERIS.

Weinberg, G. W. 1985. The Secrets of Consulting. New York: Dorset House Publishing.

Wirfs-Brock, R., B. Wilkerson and L. Wiener. 1990. Designing Object-Oriented Software, 4th printing, Englewood, NJ: Prentice-Hall.

Wong, B. K., J. A. Monaco. 1995. « A bibliography of expert system applications for business (1984–1992) », European Journal of Operational Research, Vol. 85, Issue 2, p. 416–432.

Zadeh, A., I. M. Fam, M. Khoshnoud and M. Nikafrouz. 2008. « Design and implementation of a fuzzy expert system for performance assessment of an integrated health, safety, environment (HSE) and ergonomics system: The case of a gas refinery Inform », Elsevier Science Inc. Vol. 178 Issue 22, New York, NY, USA November 2008. doi:10.1016/j.ins.2008.06.026.

Zadeh, L. A. 1988. « Fuzzy logic ». IEEE Computer, 1:83.

Zadeh, L. A. 1965. « Fuzzy sets ». Information and Control 8, p. 338–353.

Zadeh, Lotfi A. 2008. « Is there a need for fuzzy logic? ». In: Fuzzy Information Processing Society, NAFIPS 2008. Annual Meeting of the North American. (New York, New York, May 19-22, 2008).

Zadeh, Lotfi A. 2008. « Is there a need for fuzzy logic? ». Information Sciences, vol. 178, issue 13, 1 July 2008, p. 2751-2779.