DEPARTMENT OF COMPUTER SCIENCE

EXPLORING SOFTWARE COST MODELLING AND ESTIMATION WITH COMPUTATIONAL INTELLIGENCE

Ph.D. Dissertation

EFI PAPATHEOCHAROUS

2011

EXPLORING SOFTWARE COST MODELLING AND ESTIMATION WITH COMPUTATIONAL INTELLIGENCE

Efi Papatheocharous

University of Cyprus, 2011

Software cost estimation (SCE) is the art of balancing time and resources to optimally budget a project. An essential requirement is to estimate the schedule, cost and human effort required to complete the project with adequate accuracy and before the project commences, or during its life-cycle, at an acceptable point in time when such an estimation may be considered useful to project managers (i.e., at the 'early' project phases). SCE models consist of mathematical algorithms or parametric relations used to approximate the most dominant cost, the human effort, in terms of person-months for developing a software. It is considered one of the basic project management processes to support efficiently the activity of resource allocation.

Although numerous SCE models and techniques have been proposed by researchers, many problems still exist. Recent research reports 60-80% of projects overrun software cost estimates by 30-40% (Moløkken and Jørgensen, 2003). The aforementioned problem stems from the complex, intangible and unique nature of software and the inconsistent selection of the factors that affect productivity by cost estimators. Moreover, many cost factors are qualitative rather than quantitative (Boehm et al., 2000b) and hence subjective in nature. Finally, there is also lack of explicit terminology, data gathering principals, measurement rules and formal definitions of these factors. As a result, SCE is affected by a semantic vicious cycle: What constitutes a 'successful' project is hardly clearly defined; whereas, 'non-

successful' projects are those that are either cancelled, delivered with less functionality and/or with lower than the agreed quality, or exceeded resources, budget and/or schedule estimates. The real reason for the underestimations occurring is hard to contemplate. Over and under estimations may be attributed to the project going 'wrong' or to the budget estimates that were inaccurate in the first place. This dissertation aims at minimising such imprecisions in SCE.

Novel models and techniques are employed, based on the factors of People – Process – Product, for improving SCE accuracy and comprehending the risks occurring. Moreover, the essential quantitative and quanlitative factors that affect productivity are identified and explored. Thus, this thesis adopts two approaches: A quantitative and a qualitative. The quantitative approach, aims at improving SCE accuracy, reliability and generalisability, by exploring Computational Intelligent (CI) models and techniques, such as Artificial Neural Networks (ANN), Evolutionary Algorithms (EA), Fuzzy Logic (FL), and hybrid forms of the aforementioned techniques. Moreover, the main target is to develop SCE models of a practical value, i.e., dealing with the inherent uncertainty of the software engineering data and producing relatively 'early' (i.e., post specifications) estimations. The qualitative approach extends the numerical and empirical CI investigations, by employing Fuzzy Cognitive Maps (FCM) and Influence Diagrams (ID), which facilitate exploring the relationships between qualitative cost factors and effort. It also visually reveals the contribution of attributes in SCE and enhances the understanding of their cause-and-effect dependencies.

The results and observations of this diatribe reveal that considerable benefits may be gained by CI-based methods employed in cost prediction improvement and in understanding which factors are considered 'significant' in the process of SCE. The various Feature Subset Selection (FSS) methods applied assisted in identifying and excluding the less 'influential' cost factors from the models, which, in turn, lowers the model's complexity and the overall time and effort required to measure and quantify each and evey one of them. The SCE models proposed in this thesis are proven viable, practical alternatives through extensive experimentation with widely known and used benchmark data of the relevant literature.

DEPARTMENT OF COMPUTER SCIENCE

EXPLORING SOFTWARE COST MODELLING AND ESTIMATION WITH COMPUTATIONAL INTELLIGENCE

Efi Papatheocharous

A Dissertation Submitted for Fulfillment

of the

Requirements for the Ph,D, Degree

at the Faculty of Pure and Applied Sciences

of the University of Cyprus

December, 2011

2011

APPROVAL PAGE

Doctoral Dissertation

EXPLORING SOFTWARE COST MODELLING AND ESTIMATION WITH COMPUTATIONAL INTELLICENCE

Presented by

Efi Papatheocharous

Research Supervisor *(Jan. 2007-Aug. 2010)*	ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Co-Supervisor *(Sept. 2010-Dec. 2011)*	Andreas S. Andreou
Co-Supervisor *(Sept. 2010-Dec. 2011)*	ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Examining Committee Member	George A. Papadopoulos
Examining Committee Member	ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Examining Committee Member	George Pallis
Examining Committee Member	ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Examining Committee Member	Georgia Kapitsaki
Examining Committee Member	ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Examining Committee Member	Ioannis Stamelos
Examining Committee Member	ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
	Stephen G. MacDonell

University of Cyprus

December, 2011

ACKNOWLEDGEMENTS

The diatribe has been materialised during my years as a Ph.D. student and research associate at the Software Engineering and Intelligent Information Systems research lab from January 2007 to December 2011. It would have been extremely difficult to accomplish pursuing and completing this Ph.D. without the support, advice and patience from collaborators, mentors and professors, parents, family and friends.

First of all, I would like to express my profound gratitude to my supervisor Associate Professor Andreas S. Andreou who inspired me and stood by my side throughout this work. I would like to thank him for being a mentor, a teacher, a key supporter and a friend. He has provided unconditional support and encouragement and has also offered generously his knowledge, guidance and time, which facilitated in all good abilities and opportunities granted in my life and career, especially during the last nine years that we have been acquainted. I would also like to thank him for the encouragement to get involved in several national and international research projects which enriched my professional knowledge and experiences and at the same time alleviated my family from any financial burden.

I would also like to give special thanks to Professor George A. Papadopoulos and Professor Constantinos Pattichis for their academic support and caring attitude, as well as their willingness to act as my co-supervisors when Professor Andreas S. Andreou joined the Cyprus University of Technology and could no longer act as my full supervisor at the University of Cyprus. Their contributions of time, ideas, feedback and advice are greately appreciated. I am also tremendously grateful to the remaining members of my examining committee Dr. George Pallis, Dr. Georgia Kapitsaki, Assistant Professor Ioannis Stamelos and Professor Stephen G. MacDonell for their time, helpful comments and encouraging attitude.

During the years of my doctoral student experience the University and the research lab have been a source of friendships, cooperations and collaborations. Especially, I would like to express my appreciation by thanking my colleague and friend Constantinos Stylianou with whom I have been sharing ideas, experiences and facilities throughout these years. His willingness to participate in brainstorming sessions and contributing in problem-solving turning-points resulted in a highly motivating, challenging, productive and pleasurable environment to work. My gratitude also goes to Dr. Nicos H. Mateou for giving endless support, personal boost and encouragement. He has been supportive in many aspects enabling me to see the 'big picture' in cases where it was quite difficult for me to see.

Especially, I would like to express my sincerest gratitude to the first person to ever believe in me and my abilities, Assistant Professor Panagiotis Germanakos. He has been a source of abundant patience, exemplary ethics, invaluable advice and limitless emotional and spiritual support. I would like to thank him for the tremendous inspirations, intellectual challenges and endless discussions which enabled me to become more insightful and able to cope with complex circumstances.

I would also like to individually thank my co-authors, colleagues and collaborators Dr. Harris Papadopoulos, Dr. Stamatia Bibi, George Rossides – a close friend and supporter for many years, Christos Skouroumounis, Angela Iasonos, Despina Trikomitou and Pantelis Yiasemis for their contribution and professionalism. They have all contributed immensely to maintaining the level of enthusiasm and excitement for professional work growth. Finally, I am also thankful to Professor Jean-Marc Desharnais from École de Technologie Supérieure (ÉTS), Montreal for providing me after personal communication one of the datasets used in this work and published in his thesis.

Lastly, I would like to express my sincerest thanks to my mother, Maro, and to my father, Loizos, who both supported me in my whole life with invaluable life lessons, their indescribable kindness and sacrifice without asking anything for return. Also, a special thank you goes to my sister, Stalo, and her husband, Marinos, for always encouraging and supporting me throughout all my efforts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

Abbreviation	Description
AFA	Arbitrary Function Approximators
AFP	Adjusted Function Points
AMSE	Adjusted Mean Square Error
ANGEL	ANaloGy softwarE tool
ANN	Artificial Neural Networks
AQUA	Software Cost Model developed by Li et al. (2007)
ASD-SCE	Agile Software Development and Software Cost Estimation
ASMA	Australian Software Metrics Association
BANN	Backward Feature Removal with ANN using Garson's Relative Importance
BFE	Backward Feature Elimination combined with Ridge Regression
BMMRE	Balanced Mean Magnitude of Relative Error
BSWF	Backward StepWiseFit with Ridge Regression
C4.5	Extended classification algorithm of the ID3 algorithm
CART	Classification And Regression Trees
CBR	Case-Based Reasoning
CBSD	Component-Based Software Development
CC	Correlation Coefficient (Pearson)
CCE	Core Cost Estimation
CC-SCE	Clustering and Classification for Software Cost Estimation
CD	Class Diagrams
CHAID	Chi-squared Automatic Interaction Detection algorithm
CI	Computational Intelligence
ClS	Cluster Size
CMM	Capability Maturity Model
CN	Concept Node
CNFCM	Certainty Neuron Fuzzy Cognitive Map
COCOMO	COnstructive COst Model
CompTIA	Computing Technology Industry Association
COSEEKMO	Software Cost Model reported in Menzies et al. (2006)
CP	Conformal Predictors
CS	Conditional Sets
DENFIS	Dynamic Evolving Neuro-Fuzzy Inference System
DSN	Deep Space Network
DT	Decision Trees

Abbreviation	Description
EA	Evolutionary Algorithms
EBSE	Evidence-Based Software Engineering
ERD	Entity Relationship Diagrams
exCHAID	Exhaustive Chi-squared Automatic Interaction Detection algorithm
FCM	Fuzzy Cognitive Maps
FCM-SCE	Fuzzy Cognitive Maps for Software Cost Estimation
FDT	Fuzzy Decision Trees
FFS	Forward Feature Selection combined with Ridge Regression
FIS	Fuzzy Implication Systems
FL	Fuzzy Logic
FLANN	Functional Link Artificial Neural Network
FP	Function Points
FPA	Function Points Analysis
FSS	Feature Subset Selection
FSS-SCE	Feature Subset Selection and Software Cost Estimation
FSWF	Forward StepWiseFit with Ridge Regression
FUSP	Fuzzy Use Case Size Points
GA	Genetic Algorithms
GP	Genetic Programming
GRA	Grey Relational Analysis
GRACE	Grey Relational Analysis based on Software ProjeCt Effort Prediction
GUI	Graphical User Interface
HR	Hit Ratio
IBM DP	IBM's Data Processing services
IBM-FSD	IBM's Federal Systems Division
ID	Influence Diagrams
ID3	Iterative Dichotomiser 3 algorithm
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Point Users Group
IOM	Input Output Methods
IS	Input Strength
ISA	Input Sensitivity Analysis
ISBSG	International Software Benchmarking Standards Group
Jørgensen95	Dataset mentioned in Mair et al. (2005)
KLOC	Thousands Lines Of Code
k-NN	*k*-Nearest Neighbour
KSLOC	Thousands Source Lines Of Code
LOC	Lines Of Code
LS	Less Strict criterion
LSBFE	Backward Feature Elimination with Least Squares
LSFFS	Forward Feature Selection with Least Squares
LSGA	Genetic Algorithm with Least Squares
LSR	Least Squares Regression
MAE	Mean Absolute Error
MBRE	Mean Balanced Relative Error
MERMAID-2	Dataset reported in Kitchenham (2002)
ML	Machine Learning
MLP	Multi-Layer Perceptron
MLR	Multiple Linear Regression
MMRE	Mean Magnitude of Relative Error
MRW	Median Relative Width

Abbreviation	Description
MSE	Mean Squared Error
MW	Median Width
NATO	North Atlantic Treaty Organization
NF	Neuro-Fuzzy
NG	Neuro-Genetic
NRMSE	Normalised Root Mean Squared Error
OLS	Ordinary Least Squares
OS	Overall Size
PB-SCE	Phased-Based Software Cost Estimation
PE	Percentage of Errors
ph/pm	person-hours/person-months
PI-SCE	Predictive Intervals in Software Cost Estimation
POJO	Plain Old Java Objects
Pred(l)	Prediction Level
PRICE-S	Parametric Review of Information for Costing and Evaluation – Software
PSO	Particle Swarm Optimisation
RBF	Radial Basis Function
RBFN	Radial Basis Function Network
RCA	Requirements Capture and Analysis
RDBMS	Relational DataBase Management System
RI	Relative Importance
RMSE	Root Mean Squared Error
RR	Ridge Regression
RS	Relative Strength
RUP	Rational Unified Process
S	Strict criterion
SB-SCE	Size-Based Software Cost Estimation
SCE	Software Cost Estimation
SDLC	Software Development Life-Cycle
SEER-SEM	Software Evaluation and Estimation of Resources – Software Estimating Model
SL	Significance Level
SLIM	Software LIfecycle Management
SLOC	Source Lines Of Code
SOFCOST	SOFtware COST Grumman's Software Cost Estimating Model
SOM	Self-Organising Maps
SVM	Support Vector Machines
TRW	TRW Inc. - an American corporation
UCD	Use Case Diagrams
UFP	Unadjusted Function Points
USP	Use Case Size Points
WNN	Wavelet Neural Networks
WSD1	Dataset mentioned in Mair et al. (2005)

Chapter 1

Introduction

The term "*software*" was coined by John Wilder Tukey in computing context in a 1958 article (Brillinger, 2002). Since then, it has been gaining vital ground in our daily lives, contributing into providing innovative solutions for problems within organisations and assistance in management and control for many products branded in the market today. Thus, a plethora of organisations emerged producing/developing software. Software today, stands on the foundation of everything and comes in everything.

The discipline of "*Software Engineering*" emerged in the 1968 NATO Conference (Naur and Randell, 1968), but was explicitly described not long ago in the IEEE standard 610.12 (1990) as "*the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software*". The definition implies that individuals will probably be able to estimate the cost of building such controllable systems. However, in recent years, the software industry has become increasingly specialised and has resulted to a wide range of software developments and even wider application and adoption areas. Moreover, the ever-morphing, complex, multi-dimensional nature of software engineering as a discipline, as an art form and as a culture, make it intricate to comprehend and monitor the associated processes, challenges and risks and henceforth, effectively estimate software development costs.

A software project usually follows a structured/systematic approach to be engineered, i.e., adopts life-cycle models like the Waterfall (Royce, 1970), the Incremental (Mills and O'Neil,

1980), Rapid Prototyping (Naumann and Jenkins, 1982) and the Spiral (Boehm, 1988). These structured approaches may be adopted or adapted in prevalent forms of software development by organisations. There is also the trend to diversify to non-traditional software development methods and facilitate the production of high-quality software in a cost-effective way. Cost effectiveness is associated with the balance of schedule, resources and budget. Since the adoption of the classic Waterfall method (Royce, 1970) and until the more recently adopted methods like the Rational Unified Process (RUP) (Ambler, 2005) and the Agile Paradigm (Martin, 2002) in software engineering, maintaining a standard level of resource planning, scheduling and budgeting is very hard to be accomplished during the development life-cycle due to the individuality of each project. An essential task, especially from the project's inception and until its 'early' phases (post specifications), is to effectively describe the activities related to the project, reflect on the project's unique characteristics and estimate the required costs to deliver it on time. Finally, in this process, software project managers are responsible for estimating the costs, planning how resources will be allocated, scheduling and budgeting the main activities (Sommerville, 2006).

1.1 Problem Definition

Software project management involves a variety of critical activities, including planning, scheduling, monitoring and controlling the processes employed, the resources spent and the artefacts utilised for software development. Among these activities the accurate and realistic Software Cost Estimation (SCE) is considered a key task, exhibiting an increasing interest on behalf of software engineers, researchers, managers and stakeholders.

Software project costs may include the hardware and software development tools and platform costs (including maintenance), the travel and training costs and of course the effort costs, i.e., the costs of paying software professionals (Sommerville, 2006). SCE involves the overall assessment of these costs, even though for the majority of the projects the most popular metric is considered the effort cost. The development effort cost is settled as the

dominant cost in nearly all the related SCE studies performed because it is the most complex, unpredictable and substantial cost factor, compared to the rest abovementioned costs. Moreover, the development cost is related to the company's/organisation's productivity and cannot be easily or objectively calculated.

The activities performed to effectively estimate (or predict) the required development effort are highly dependent on the sizing and costing of the software product. The basic units for measuring software size are the software Lines Of Code (LOC) (Gilb, 1976) and the functional size obtained from the Function Points (FP) introduced by Albrecht (1979). Sizing and costing are considered equally important since the successful delivery of a product on time, within budget and with the anticipated functionality heavily depends, among other things, on how accurate the aforementioned estimations turn out to be. Moreover, SCE involves the activity to calculate, with certain confidence, the resources required to develop software systems. Organisations estimate the costs in terms of overhead costs for carrying out the project and divide this by the number of productive staff (Sommerville, 2006). This is expressed in terms of the effort required, which is typically measured in person-months (pm). Likewise, scheduling, including the time expected to deliver a project, is considered to be related with effort estimation (Putnam, 1978; Abdel-Hamid and Madnik, 1983). In some cases the dependency of similar parameters influencing effort are investigated to estimate schedules (Jensen, 1983; Park, 1988; Putnam and Myers, 1992; Boehm et al., 2000a). Therefore, SCE and scheduling are both considered imperative activities which seem to be mostly directed by the overall size of the project as well as other project characteristics. Nonetheless, in some cases where effort is confused with progress; assuming that the people working on a project and the months of a schedule are interchangeable (Brooks, 1995) which is not valid.

Identifying and understanding the basic parameters that affect software cost, as well as making appropriate assessments of these parameters, are extremely important requirements to determine staff allocation and scheduling issues for a project and thus could pave the way to producing better software cost estimates. These parameters, usually called "software cost drivers", are not easy to be defined and sometimes are regarded as highly ambiguous and

difficult to measure. Also, they relate to software effort, quality, scheduling, sizing and productivity (Sommerville, 2006). For example, proper sizing of a software product under development from the 'early' development phases and complexity assessment of the necessary functions may significantly enhance the estimation of the total effort required for the completion of the project. This estimation may thus constitute a useful tool in the hands of project managers for better planning, monitoring, coordinating and controlling the subsequent development phases.

It was in the late '50s and early '60s when researchers first began focusing on estimations of software cost and executed experiments with a range of estimation techniques. These experiments endeavoured to achieve accurate cost predictions and provide strong project and quality frameworks for optimising software project management (Boehm et al., 2000b). The impact of various resource estimation methods and models proposed over the last 50 years on successful software engineering practices has been tremendous, whereas SCE until today continues to attract considerable research attention (Jørgensen and Shepperd, 2007). In addition, software cost methods and models for development effort estimation are included practically in all software Capability Maturity Models (CMM), software engineering textbooks and "bodies of knowledge" (Boehm and Valerdi, 2008).

Accuracy in SCE is a critical prerequisite in the initial project phases and until the end of the project for a project manager; it may represent the main reason for deciding on whether to undertake a project. In fact, precision in these initial cost estimates has a profound effect on the success of the activities and phases carried out throughout the whole Software Development Life-Cycle (SDLC). For instance, underestimating a project will probably lead to under-staffing, causing wrong allocation of resources and eventually overworked staff. Thus, underestimation will sooner or later lead to compromises as regards the quality of the development methodology, the documentation, the testing procedure and the product itself. Ultimately, it will probably result in exceeding the agreed budget and schedule, and as deadlines will probably be missed the developers will eventually lose their credibility. On the other hand, overestimating a project is likely to cause again poor resources allocation but in

the sense expressed in Parkinson's Law, i.e., "*Work expands to fill the time available for its completion*" (Parkinson, 1957). The law expresses project slack that is a result of situations with more resources than needed that causes developers becoming less productive or decide to 'gold-plate' a system by adding features that were not required by the user (Boehm, 1981). Thus, overestimating will probably lead to undertaking the wrong projects, in terms of size and capacity of developers and might risk the success of bidding and winning any other project as well. As a consequence, loss of opportunities to win more appropriate contracts to competitors due to the prohibitive costs and misallocated staff, as well as financial loss, will probably occur. Conclusively, such inaccuracies in software cost estimations may result in huge business and economic failures which may eventually be proven catastrophic for developers and organisations.

1.2 Motivation

Software development is an intractable, multi-faceted process, which frequently encounters inherent and unforeseen difficulties. With the rapid growth of technology and the increasing need for new software applications in a wide variety of disciplines, software development today has become an essential, highly valuable and quite expensive procedure. The motivation for conducting research on the area of SCE is unfolded in this section.

According to reviews on surveys, 60-80% of projects encounter cost overruns in the range of 30-40% (Moløkken and Jørgensen, 2003) leading to quality reduction, schedule overruns or extra staff employment to realise the project goals. This phenomenon may be attributed to several contingencies occurring during software development and to the high dependence of cost and schedule estimates on accurate size estimates (approximations) of the software to be developed. However, software size approximations (in LOC or FP) is hard to determine prior to its actual implementation, but it is usually one of the main components of the cost estimation process since it seems to be affecting productivity or at least a component of it (Sommerville, 2006). This is evident in classical and recent software cost models such as the

Doty (Herd et al., 1977), COCOMO (Boehm, 1981; Boehm et al., 2000a), Walston-Felix (Walston and Felix, 1977), SLIM (Putnam, 1978), Bailey-Basili (Bailey and Basili, 1981), Albrecht and Gaffney (Albrecht and Gaffney, 1983), SEER-SEM (Jensen, 1983), Kemerer (Kemerer, 1987), PRICE-S (Park, 1988), Barnett and Mellichamp (Matson et al., 1994), ANGEL (Shepperd and Schofield, 1997), GRACE (Song et al., 2005), COSEEKMO (Menzies et al., 2006) and AQUA (Li et al., 2007) where the size estimate is a main component. In addition software cost models presuming the ability of estimating LOC early in the software life-cycle enclose a major weakness over models based on estimates of FP which can be generated reasonably accurately at early project stages (Finnie et al., 1997). Therefore, studying the aforementioned dependence of size and effort is considered a challenging issue for research and is examined in this thesis. Specifically, the first investigation of the diatribe explores prediction of effort through various Computational Intelligence (CI) methodologies and Size-Based SCE (SB-SCE) models. The proposed CI methodologies originate from nature-inspired approaches to address the intricate and challenging nature of the process of developing software systems, express the dependencies among software factors and effort, as well as, deal with the imprecise and uncertain nature of available software project data.

Apart from the software size, other cost drivers, such as software complexity, team capability, team experience and programming language used, are also investigated in the context of SCE. Therefore, the secondary research investigation of this thesis relates to CI-based cost estimations with Feature Subset Selection methods (FSS-SCE). The investigation is related with the cost driver's dependence with effort, while the subsets of these cost drivers that are considered more appropriate/influencing to increase estimation performance are selected. The main benefit of this investigation is to find out or propose a set of cost drivers on which project managers and cost estimators should focus on during the software measurement and SCE activities.

A supplementary research investigation of this thesis with regards to feature selection is related with the issue of practicality of the SCE methods proposed. The examination concerns the indication of a particular project phase based on which relatively 'early' cost estimations

may be achieved and are referred to as Phase-Based SCE (PB-SCE). A four-stage progressive cost model investigates the phase in the software life-cycle which is 'safer' to obtain estimates by adjusting the estimates at each stage. Since, the dynamic form of the software development process results to changes in the project manager's beliefs about the cost estimates over the SDLC (Pendharkar et al., 2005), conducting 'early' phase estimates is especially considered risky and uncertain (Boehm et al., 2000a). However, the issue of estimating cost using drivers that are available from the early phases of development is extremely practical and consists therefore, another important research issue worth investigating. Throughout this thesis, experiments are conducted in the majority of the models proposed, investigating the issue of 'early' SCE, i.e., utilising drivers that are measured in the initial phases of development.

Another issue is the type of project data available, typically of nominal nature, containing categorical or descriptive values, which are considered hindering factors in SCE since they cannot be easily exploited in models and are usually not clearly defined. In addition, the lack of explicit terminology for project data makes the differentiation of projects obscure. Examples of such data include the type of the organisation, culture and stability of the development environment, business area, application domain, development platform and the type(s) of language(s) utilised by a project. Therefore, an important requirement of SCE is to implement models that may accommodate many-type of data, such as numerical, nominal and descriptive data, and facilitate in performing the necessary transformations. Additionally, SCE models need to accommodate techniques or reasoning mechanisms to handle the uncertainty and vagueness of values caused by the lack of explicit terminology and measurement activities regarding software. Techniques such as Fuzzy Logic (FL), clustering and classification as well as predictive intervals (instead of crisp predictions) are associated with SCE in this work to address for example the lack of homogeneity and confidence in the project data. Thus, this thesis investigates the merits added in SCE by the implementation of such techniques, i.e., Clustering and Classification methods in SCE (CC-SCE), Predictive Intervals in SCE (PI-SCE) and Fuzzy Cognitive Maps for SCE (FCM-SCE), the latter

including models of Qualitative nature, and all of which approximate cost factors and effort in a manner closer to the way humans conceive/conceptualise and relate information.

Nevertheless, research in SCE has and will still have a long way to go, especially as the nature of the development process continues to evolve. The prevailing problem of resource misallocation is reported also in recent statistics. Figure 1.1 obtained from the Standish Group report (The Standish Group, 2007) indicates that only 35% of the projects in 2006 successfully delivered their software within their estimates. Whereas, the issue of reliable estimation is between the most imperative success factors for software projects (Grimstad et al., 2006).

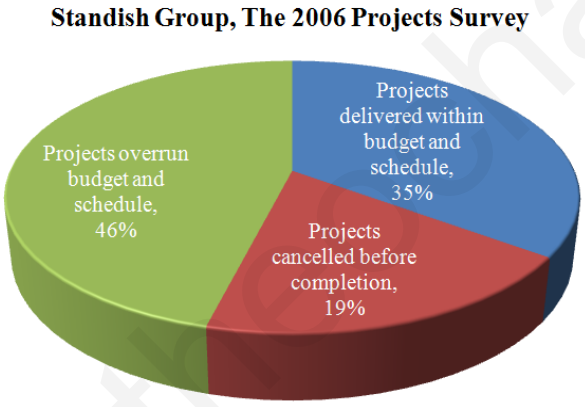


Figure 1.1: Survey results of projects from 2006 regarding 'successful' costing of software (The Standish Group, 2007)

Moreover, the emerging recent trends of software engineering contribute to the ascending horizon of SCE research as the field is being re-invented in every aspect of the triad People – Process – Product involved. Emerging paradigms include object-orientation, abstract data types, very high level programming and modelling languages, rapid and agile development processes, etc. As the different levels of customer requirements, quality and individual team's unique characteristics eventually affect the development process, SCE models are expected to be extended and include such evolutions. One of the issues investigated in the SCE approaches of this thesis, is the cost change in a newly adopted development environment, namely the Agile (Beck et al., 2000). The models proposed are referred to as Agile Software Development SCE (ASD-SCE) models.

Concluding, one of the greatest research questions still not adequately answered is: "*Which type of software cost method is appropriate to use for a particular project/case to obtain successful, reliable and practical (useful) effort estimations?*". Even though this research question most probably cannot be adequately solved for every single case, the exploration of a proposed range of solutions for software cost modelling and estimation based on enhanced/hybrid CI techniques is considered a promising endeavour.

1.3 Goals and Objectives

Taking into consideration the motivation for conducting research in the area of SCE, summarised in the previous section, this dissertation carries out an exploration of Computational Intelligent (CI) techniques and models. The goal is to find accurate, reliable and practical alternative solutions for estimating software cost taking into consideration the available information of particular circumstances. These circumstances are unique for each project. Moreover, in order to produce estimations, the cost techniques and models require information that is not known from the initiation of the project. Thus, project managers and engineers try to specify the exact values for the metrics used as inputs (MacDonell and Gray, 1997). Since for many of these metrics the actual values are never known with certainty until the project is completed, managers often assume values they anticipate (Jørgensen, 2004a). As an alternative, cost data values may be collected from past completed projects and be utilised for future cases in an analogy-based method (Chiu and Huang, 2007) according to a set of project characteristics. While the former situation suffers from subjectivity, the latter does not guarantee that the new project will require the same amount of effort with that of 'similar' projects in terms of the selected characteristics, especially considering the diversity of projects. Thus, typically the estimates are based on the subjective human judgements encoded in measured data of previous human experiences with software projects.

CI methods and techniques were chosen since several research studies appeared during the last decade with promising results (such as the work of Srinivasan and Fisher (1995),

Finnie et al. (1997), Wittig and Finnie (1997), Mair et al. (2000), Heiat (2002), Idri et al. (2004), Xu and Khoshgoftaar (2004) and Ahmed et al. (2005)) and suggesting that their application might yield significant benefits. These studies have shown that the appropriateness and comprehensibility of the algorithms coming with CI techniques have added significant advantage for project managers over traditional methods (like another popular technique, i.e., regression analysis). Moreover, according to Fenton (2000) traditional methods hardly provide adequate support for quantitative decision making.

The main goal of this diatribe is briefly summarised in the following statement: *"Investigate the effectiveness of Computational Intelligent (CI) techniques and propose novel models for accurate and reliable Software Cost Estimation (SCE) in conjunction with the identification of the most significant (appropriate) cost driver attributes/schemes and their relationships with effort."*. Among the main goals of this thesis is also to investigate both empirical and theoretical issues emerging in the field of SCE, under two approaches, the Quantitative and the Qualitative. The first refers to exploration of numerical and empirical project observations through Quantitative cost models; the second introduces empirical scenarios analysis through Qualitative cost models devised with the aid of one or more experts. In this context, existing cost models, methods and techniques, and their actual use in real-life software environments comprise the cornerstone of the initial research investigation. The first realisation was that most well-known parametric cost models, such as SLIM (Putnam, 1978), Function Points Analysis (FPA) (Albrecht, 1979), COCOMO I (Boehm, 1981), COCOMO II (Boehm et al., 2000a), PRICE-S (Park, 1988) and ANGEL (Shepperd and Schofield, 1997), require different sets and types of inputs. Therefore, this work carries out an explorative study for software cost modelling utilising a set of cost factors related with the three axons People – Process – Product. Moreover, since the majority of parametric cost models commence by associating the size of the software product with the effort required for each phase of the life-cycle and the total effort (Ghezzi et al., 2003), primarily in this work, Size-Based SCE (SB-SCE) are investigated (Papatheocharous and Andreou, 2008; Papatheocharous and Andreou, 2009c; Papatheocharous and Andreou, 2011).

Then, Feature Subset Selection for SCE (FSS-SCE) is performed to establish the appropriate set(s) of software project factors and principles for (professional) practice of cost estimation upon which schedules, resource allocation and decisions can be based. The chief cost estimation models developed are based on two widely known techniques, i.e., Artificial Neural Networks (ANN) (Papatheocharous and Andreou, 2007; Papatheocharous and Andreou, 2010; Papatheocharous and Andreou, 2012b) and Ridge Regression (RR) (Papatheocharous et al., 2010c; Papatheocharous et al., 2010b). ANN represent a dominant CI technique widely used in various fields and RR is used as a reliable estimator for comparing various FSS approaches. The basic research investigation of the thesis related to modelling and cost factor's extraction is complemented by hybrid SCE models from the area of CI, such as Evolutionary Algorithms (EA) and Fuzzy Logic (FL).

Of particular interest is the development of software cost models that also apply vertical and horizontal dataset filtering, through methods of clustering and classification. Some of the techniques proposed are used in conjunction with Fuzzy Logic (FL) for addressing the subjectivity and uncertainty of the data and comprise the Clustering and Classification SCE (CC-SCE) models. The techniques involve fuzzy clustering with the Entropy-based and Fuzzy *k*-modes algorithm (Papatheocharous and Andreou, 2009a) and classification with association rules obtained from Fuzzy Decision Trees (FDT) (Papatheocharous and Andreou, 2009b; Andreou and Papatheocharous, 2008b). Moreover, two hybrid approaches are used for both clustering and classification purposes, namely Conditional Sets (CS) coupled with a Genetic Algorithm (GA) (Andreou et al., 2007; Andreou and Papatheocharous, 2008a) and Genetic Programming (GP) yielding cost equations (Papatheocharous et al., 2010a).

Advancing the above techniques, the subsequent SCE approaches implemented have the ability to make intelligent decisions effectively for an assortment of 'similar' software projects and at the same time to handle incomplete, imprecise and fuzzy information. Thus, hybrid models consisting of Fuzzy Decision Trees (FDT) and Fuzzy Implication Systems (FIS) (Papatheocharous and Andreou, 2012a) are developed. The aforementioned models focus on improving the prediction accuracy produced by employing hybrid Computational Intelligence

(CI) techniques and by identifying rules of association between effort and the most 'influencing' cost drivers from the ones available. During the experiments conducted with the models of FSS-SCE and CC-SCE, this thesis included the study of the prediction ability of cost drivers that are available from the 'early' project phases (i.e., after the end of the specifications) thus attempting to provide models of a more practical value.

Lastly in the Quantitative approach, another hybrid model is developed using Ridge Regression (RR) and Conformal Predictors (CP) (Papadopoulos et al., 2009) to replace the production of a crisp estimation value for effort to predictive intervals, according to specific confidence levels. The Predictive Intervals for SCE (PI-SCE) are developed to yield more flexible approximations of effort and investigate maximum-minimum conditions. This flexibility is described through the predictive intervals constructed that include a minimum and a maximum value of the effort estimate. Also, the Phased-Based progressive SCE models (PB-SCE) developed perform estimations in four distinct stages of development, i.e. at an early, post planning, post specifications and a post design phase (Papatheocharous et al., 2012).

The Qualitative models produced investigate, simulate and analyse the dynamics of various cost factors (even factors that are hard to measure) through Fuzzy Cognitive Maps (FCM) and Influence Diagrams (ID). FCM utilise a group of experts for empirically weighing the set of inputs in the form of cognitive states and then estimate effort on a scenario basis (Papatheocharous et al., 2008). Finally, effort is investigated within non-conventional software development environments, i.e., when Agile Software Development (ASD) methodologies are followed, through ID. The Agile Software Development SCE (ASD-SCE) models constructed investigate and analyse the benefits of switching from traditional methodologies to agile in terms of productivity and cost (Papatheocharous et al., 2011).

The above comprise the contribution of this Ph.D. thesis and meet the following set of objectives:

- Investigate the validity of size-based software cost models.
- Model and accurately predict development costs of real-life software projects.

- Locate the strongly influencing cost drivers and examine their impact in estimating cost.
- Improve software cost models by using hybrid and evolutionary algorithms.
- Cluster and classify software projects to improve estimations.
- Handle the problematic nature of software engineering data (characterised by uncertainty) and incorporate fuzzification and predictive intervals in the estimations.
- Investigate the issue of defining which phase or time-period is the most appropriate (safer) one for producing accurate enough cost estimations, i.e., how early in the phases of the development life-cycle.
- Investigate software cost estimations within scenarios and modern development environments (e.g., Agile).

1.4 Research Approach

The scientific approach is briefly described in this section to summarise how this research work was developed: The first attempt aimed to discover new relations of cost factors according to the People – Process – Product aspects and model these relations into an assortment of SCE models. These models were divided into Quantitative and Qualitative models. The need of creating various cost models stems from the fact that no software cost model can address every project case in software engineering. It would not help, for example, to tailor the model if the technology or the process is too immature for the project in hand, or if the people involved are not skilled enough. Of primal concern was the study of the causal relationships among cost factors and an organisation's (or a person's) productivity. Productivity was accounted with work-effort estimates. Definition and measurement issues of cost factors that are quantitative (such as software size) and qualitative (such as people skills) were investigated.

New methodologies for SCE were then developed stemming from the area of Computational Intelligence (CI), such as Artificial Neural Networks (ANN), Genetic Algorithms (GA), Probabilistic Theory and Fuzzy Logic (FL). CI methods offer a

complement to conventional Artificial Intelligence in the area of Machine Intelligence and approximate in a qualitative manner human reasoning. Moreover, the introduction of new hybrid SCE models, by combining Evolutionary Algorithms (EA) and Fuzzy Logic (FL), improved the techniques and the results in terms of accuracy and comprehensiveness. In addition, the quality of the available data was investigated through Feature Subset Selection (FSS) methods to locate the strongly influencing set of software cost factors. Moreover, various clustering and classification algorithms were employed to improve the SCE obtained by identifying the most appropriate factors to group projects and optimise prediction results. Especially the effect of 'early' software effort estimations (i.e., after concluding with requirements specification) was examined. Recent software paradigms and modern development approaches such as the Agile were also examined, as regards the impact they pose on current cost models and approaches and for any necessary enhancements and/or modifications to account for the changes they introduce. This research thesis also studied qualitative software metrics such as the developer's working environment, which are harder to quantify. Such metrics were used together with expert assessments regarding the software under to model and execute hypothetical scenarios.

1.5 Significance

The impact of this dissertation is included in the benefits of the introduction of CI techniques for the improvement of SCE models. The discipline of CI has been chosen to be applied in the area of SCE combining elements of learning, adaptation, prediction, evolution and fuzzy logic to create hybrid models with some sort of intelligence. The CI-hybrid models developed extend or replace traditional SCE techniques such as Algorithmic, Expert Judgement, Analogy-based, Parkinson's Law and Price-to-win (Sommerville, 2006). CI in SCE offers the opportunity to build solutions inspired from a combination of several research disciplines, for example, computer science, economics, philosophy, sociology and biology. The solutions: (i) are practical and repeatable, (ii) are objective and realistic since the result is

based on extensive analysis of real project data and feedback from project stakeholders, (iii) provide automatic data exploration, filtering, clustering and classification for improving estimates, (iv) provide ways to handle efficiently complex, hard, NP-complete problems, where no obvious mathematical formulas can be extracted by hand to associate the independent variable(s) with the dependent one(s) and, finally, (v) add significant value of empirical evidence, something which is important in the case the discipline of software engineering turns into Evidence-Based Software Engineering (EBSE). The term adapted from Evidence-Based Medicine aims to provide ways by which current best evidence from research can be integrated with practical experience and values in the process of decision making for the development and maintenance of software (Kitchenham et al., 2004).

1.6 Contents of the Dissertation

The rest of this dissertation is organised as follows:

- Chapter 2 begins with a definition of SCE and a brief historical overview. It describes the SCE models and techniques classification and the evaluation criteria proposed in the literature. It surveys background work on models and particularly introduces the area of Computational Intelligence, and finally, it summarises the open research issues in the area of SCE.
- Chapter 3 describes the technical background relevant to this research work in two approaches, namely the Quantitative and the Qualitative. The theoretical background of the basic algorithms and techniques related with this work is summarised.
- Chapter 4 describes the models and cost estimation approaches developed in this research work aiming to accurately estimate effort and investigate the effect from various project factors on cost. The chapter also summarises the empirical experiments conducted with the CI-based models and techniques and presents the evaluation results. The experimental results include various accuracy indicators, analysis of the relationship between values of relevant cost drivers and discussion.

- Chapter 5 concludes this dissertation, provides a summary of the proposed software cost models and techniques presented in the previous chapter. The chapter summarises the main results and goals achieved with the techniques and research questions explored. In addition, the chapter provides a critical discussion of the threats to validity and concludes with the future research steps.

Chapter 2

Literature Overview

Over the years, a lot of research effort has been devoted to quantitative studies of software productivity and the factors affecting cost. Many models and techniques have been proposed for estimating the cost to develop a software product. In this chapter a historical overview along with a specification of the classical and more recent techniques and advances employed in the area of SCE are provided. Also, the evaluation criteria proposed by various researchers are described and a section of the chapter refers to the most commonly used accuracy measures for assessing cost estimations. The chapter provides also a review of related research work and concludes with CI techniques emerging in the research area. Concluding the chapter the identified open research problems are summarised.

2.1 Brief Historical Overview

The process of Software Cost Estimation (SCE) is one of the fundamental tasks performed by project managers. It involves predicting the amount of cost (or effort) required to produce a software artefact. The earliest attempts of SCE included simple rules of thumb such as "*on a large project, each software performer will provide an average of one checked-out instruction per man-hour*" (roughly 150 instructions per man-month) or "*each software maintenance person can maintain four boxes of cards*" (in those early days a box of cards held 2000 cards or roughly 2000 instructions of few comment cards) (Selby, 2007). Later on, several project

managers (e.g., Benington (1956), Norden (1958) and Nelson (1966)) began collecting quantitative data on effort and effort distribution along the Software Development Life-Cycle (SDLC). The relevant early studies of Norden (1958) which attempted to estimate the expected production costs for building a system using logistic growth curves and Nelson (1966) which provided estimates using analogy, rules of thumb and parametric models, concluded that there were too many non-linear aspects of software development for the models to work well.

Up to the early 1970s relatively little progress was made in SCE, until the 1980s and 1990s when several SCE models were developed that worked reasonably well on a restricted set of projects based on which they were calibrated. Models of that period include the TRW Wolverton model (Wolverton, 1974), the Doty Model (Herd et al., 1977), the IBM-FSD (Walston and Felix, 1977), the Function Points Analysis (FPA) (Albrecht, 1979), the RCA PRICE-S model (Freiman and Park, 1979; Park, 1988), the SLIM model (Putnam, 1978) and the COCOMO (Boehm, 1981). Some more recent models use algorithmic equations, quite similar to the COCOMO (Boehm, 1981) or to the Rayleigh curve as in SLIM (Putnam and Myers, 1992). Such examples include Baily-Basili's meta-model (Bailey and Basili, 1981), Grumann's SOFCOST model (Dircks, 1981), Tausworthe's Deep Space Network (DSN) model (Tausworthe, 1981), Jensen's model (Jensen, 1983) and COCOMO II (Boehm et al., 2000a). A basic shortcoming of such models' contribution of collecting data and fitting simple linear or exponential equations is that the independent cost drivers and their effect on effort have to be known a priori. However, the cost drivers values, coefficients and degree of relationship with effort are hard to be determined before the requirements of the project under estimation are gathered, analysed and a fairly detailed design is prepared (Stutzke, 2006). In addition, high inaccuracies occur when the aforementioned models are applied in different environments (i.e., different organisations, organisation types and projects) than the ones calibrated for (Kitchenham and Taylor, 1985; Conte et al., 1986; Kemerer, 1987).

Some successive software cost models developed showed that research has moved beyond the above models and include the ANGEL (Shepperd and Schofield, 1997), GRACE (Song et

al., 2005), COSEEKMO (Menzies et al., 2006) and AQUA (Li et al., 2007). ANGEL (Shepperd and Schofield, 1997) is a model based on analogy which produces an estimate by selecting only projects with very similar characteristics (e.g., number of interfaces, development method, or size of functional requirements documents). The model has presented several advantages, such as user acceptability, dealing with the inadequately understood domain of software projects since the estimations are based on previous project experience and the reasoning is closer to human thinking. GRACE is based on the uncertainty presented in small sample datasets and addresses feature subset selection and effort prediction based on Grey Relational Analysis (GRA) (Song et al., 2005). COSEEKMO is used to rank alternative SCE models and select the best parametric method among models in the COCOMO format (Menzies et al., 2006). AQUA combines analogy and collaborative filtering and handles non-nominal features and missing values (Li et al., 2007).

Even though the aforementioned models presented several advantages, some limitations are mentioned in this section. For example, the yielded estimations from analogy-based models (e.g., ANGEL, AQUA) will always be limited to the selected projects and thus they will present increased uncertainty. Additionally, GRACE cannot handle efficiently outliers, feature subset selection, and the weighted determinations for both features and efforts were not completely considered. Meanwhile COSEEKMO was restricted to work only with inputs having the specific COCOMO 81 or COCOMO II format, does not work well with noisy data and mostly relies on expert opinions and feedback. Other SCE models that have appeared even more recently (e.g., Huang et al. (2008), Li and Ruhe (2008), Azzeh et al., (2010), Menzies et al. (2010) and Song and Shepperd (2011)) seem to introduce and combine techniques from Machine Learning and Computational Intelligence, or combine results from multiple methods to enhance the particular approaches.

However, today's organisations and environments utilise modern development methods, such as object-orientation, component, or in general, reuse-based development and agility, which force project management methods to evolve. Current trends in software development utilise specialised tools, e.g., Integrated Development Environments (IDE), Graphical User

Interfaces (GUI), Relational DataBase Management Systems (RDBMS) and application composition languages. These support Component-Based Software Development (CBSD) and software reuse, whereas software variability and size are continuously increasing in more complex software systems. Moreover, new types of software projects prevail such as Open Source, which usually are not considered in the latest models and have been hardly analysed regarding SCE. Therefore, the literature overview shows that the current models lack of several capabilities, emerging models using combination of approaches have gained some ground during the last decade and there are many open issues (summarised in the final section of this chapter) which need to be addressed by future SCE models and techniques. The following section summarises the models classification and evaluation criteria in SCE.

2.2 Classification Schema of SCE Models and Techniques

Software cost estimation methods are distinguished generally in model and non-model-based methods (Briand and Wieczorek, 2000). Model-based methods include at least the following: (a) one or more models, most probably concerning a representation of the relationship between the value estimated (i.e., effort, productivity, etc.) and cost drivers, (b) one or more modelling method(s), concerning the required techniques and steps applied for acquiring the particular model(s), and, (c) one or more application method(s), involving the procedure for obtaining an estimate by putting into practice the model(s) of a particular context. The systematic classification of software cost methods reported in Figure 2.1 distinguishes estimation methods based on explicit resource modelling and on expert judgements (Briand and Wieczorek, 2000).

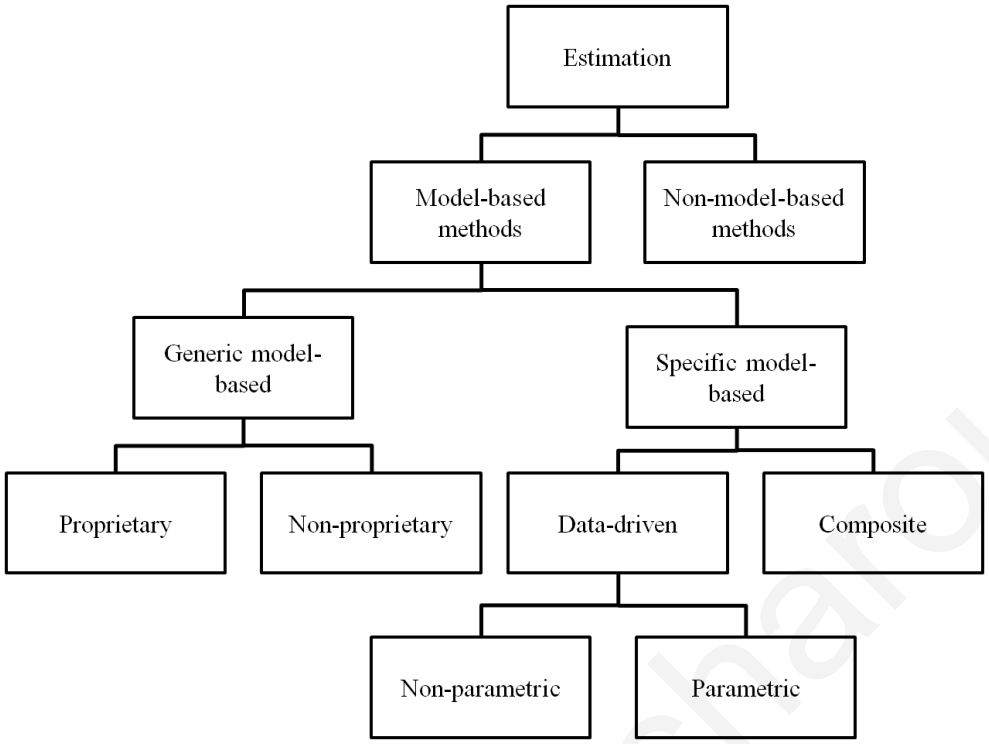


Figure 2.1: Software estimation methods classification (Briand and Wieczorek, 2000)

A further analysis of the software estimation methods follows: Model-based methods are further separated into Generic and Specific model-based methods, according to the context in which they may be applied. The case of Generic models may be public and documented (Non-proprietary), or not public and not fully documented (Proprietary), while the case of Specific models may be distinguished according to Data-driven or Composite methods. Data-driven methods are divided into Parametric, i.e., methods requiring the a priori specification of a functional relationship between project attributes and cost, and Non-parametric, i.e., methods that do not make this type of specific assumptions. Composite methods may combine expert judgement and Data-driven methods. Finally, Non-model-based estimation methods consist of one or more estimation techniques together with the specifications of how to apply them in a particular context. These methods do not build any models but just direct estimation by usually resorting to consulting one or more experts and deriving a subjective effort estimate.

Another popular classification schema of SCE methods is provided in Figure 2.2 (Myrtveit et al., 2005). The schema distinguishes the methods into Sparse-data and Many-data methods. Sparse-data methods require few or no historical data. Many-data methods need

large datasets and are subdivided to methods based on Functions, i.e., pre-defined mathematical equations describing the relationship among effort and the independent features, and methods of Arbitrary Function Approximators (AFA), i.e., functions that do not require any assumptions for the aforementioned relationships. The latter include estimation by analogy and Machine Learning techniques.

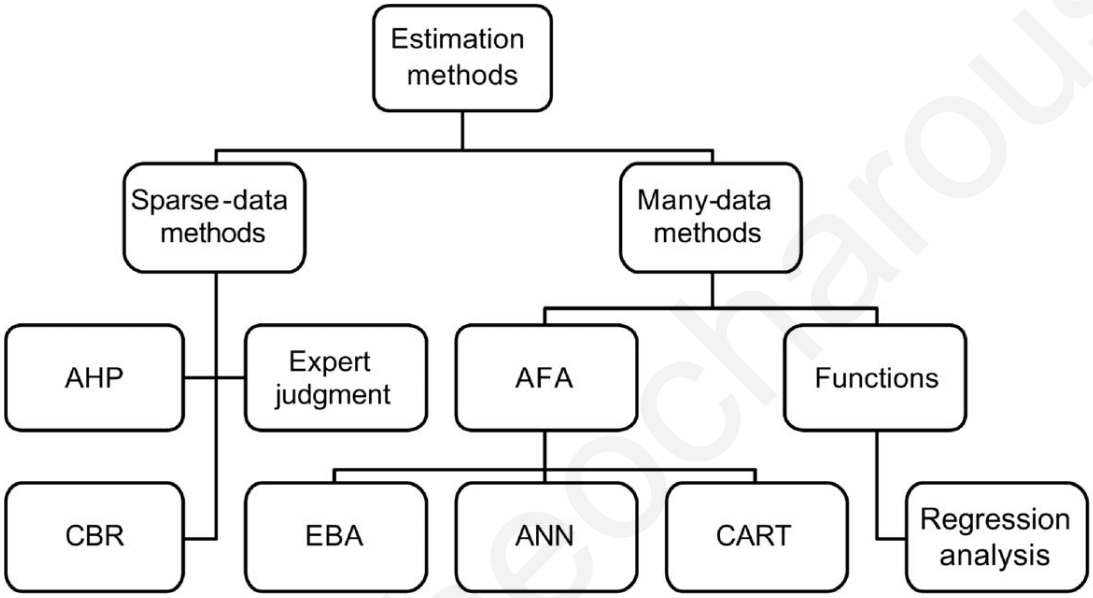


Figure 2.2: Software estimation methods classification schema (Myrtveit et al., 2005)

Function-based methods, also called Algorithmic models (Boehm, 1981; Boehm, 1984), are even today among the most popular methods in SCE. They attempt to represent the relationships between effort and one or more project characteristics with mathematical formulas derived through statistical data analysis. Expert Judgment methods (appearing within the Sparse-data methods) rely purely on the experience and knowledge of one or more experts. They highly depend on the ability of the experts to extract a sufficiently meaningful understanding of the underlying mechanisms behind the development process. As projects become increasingly large, such information extraction, understanding and unbiased estimations of costs turns to be even harder, as estimations are subjective. Machine Learning techniques mostly combine concepts and notions from the area of Soft Computing to form cost estimators or predictors. Machine learning models contrast traditional models, such as

regressions, which are typically defined with a mathematical formula, in the sense that they may hold with many different shape functions.

2.3 Evaluation Criteria of Software Cost Models

Substantial research effort is focused on discussing cost estimation methods (Briand and Wieczorek, 2000), reviewing effort estimation models (Boehm et al., 2000b), proposing best practices for effort estimations (Menzies et al., 2006) and finally, guidelines on how to advance towards skilled estimation techniques and measurement (Jørgensen, 2004a). However, software engineers are until today hindered to adopt a specific approach for SCE. The aforementioned problem may be attributed to the uniqueness of each software product, the numerous unmanageable software risks (Boehm, 1988) and the volatile conditions and contingencies occurring during the development process. In addition, a notably long list composed by Menzies et al. (2006) of effort estimation best practices is extremely difficult to follow by cost estimators and software engineers. Very little support is provided on which of the proposed SCE models, techniques or best practices are considered 'safe' to use, essential, appropriate to combine or ignored. Moreover, each project and organisation appears to have different tolerance regarding over or under estimated runs of cost, and should be treated as a unique case. For example, an organisation may be satisfied with a cost estimate that overruns by 10% the actual cost if their profit rate is around 30%, while the case for another organisation may be quite different. Therefore, the evaluation of various cost models is a highly subjective matter.

In order to evaluate the effectiveness of competing methods several global accuracy measures have been proposed, with the most dominant (in terms of most frequently reported in studies) being the *Mean Magnitude of Relative Error* (*MMRE*) and the *Prediction Level* (*Pred(l)*). The equations calculating these accuracy measures are provided at the end of this section. Researchers consider a model's accuracy acceptable if it produces an effort estimate within 25% of the actual effort 80% of the time (Conte et al., 1986; Kitchenham, 1990).

Nevertheless, if for example a model reaches the 'acceptable' accuracy (i.e., is reported to produce an effort estimate within 25% of the actual effort 75% of the time) it may yield significantly higher relative errors for the remaining 25% of the estimates, something which suggests that this model is not reliable (Nguyen et al., 2008).

Therefore, apart from the quantitative accuracy comparison of SCE models various researchers have described a list of supplementary evaluation criteria. These other desirable properties of software cost models found in relative literature are provided in Table 2.1 (Boehm, 1981), Table 2.2 (Gray and MacDonell, 1997a), Table 2.3 (Mair et al., 2000),Table 2.4 (Briand and Wieczorek, 2000), Table 2.5 (Burgess and Lefley, 2001), Table 2.6 (Ahmed et al., 2005) and Table 2.7 (Zhang and Zhang, 2009). Even though there are also many other supplementary criteria proposed in the relative literature for evaluating software cost models they are not listed in this section. The reason for this was that the most significant evaluation criteria are already covered in the tables reported below and considering more criteria resulted to overlaps in terms of definition.

Table 2.1: Software cost model evaluation criteria by Boehm (1981)

Criteria	Description
Definition	Has the model clearly defined the costs it is estimating and cost it is excluding?
Fidelity	Are the estimates close to the actual costs expended on the project?
Objectivity	Does the model avoid allocating most of the software cost variance to poorly calibrated subjective factors (such as complexity)? That is, is it hard to rig the model to get the results you want?
Constructiveness	Can you tell a user why the model gives the estimate that it does? Does it help the user understand the software job to be done?
Detail	Does the model easily accommodate the estimation of a software system consisting of a number of subsystems and units? Does it give (accurate) phase and activity breakdown?
Stability	Do small differences in inputs produce small differences in output cost estimates?
Scope	Does the model cover the class of software projects whose costs you need to estimate?
Ease of Use	Are the model inputs and options easy to understand and specify?
Prospectiveness	Does the model avoid the use of information which will not be known until the project is complete?
Parsimony	Does the model avoid the use of highly redundant factors or factors which make no appreciable contribution to the result?

Table 2.2: Software cost modelling technique capability criteria by Gray and MacDonell (1997a)

Criteria	Description
Model free	The ability of the technique to determine its own structure, rather than relying on the developer to provide the form of the relationship between inputs and outputs.
Resist Outliers	The modelling technique's robustness of estimation when faced with a dataset containing outliers.
Explains Output	The technique's capability of providing some explanation for their reasoning.
Suits small datasets	Can an accurate model still be derived with small datasets?
Adjusted for new data	Once a model has been developed, the issue of whether additional data can be added or whether the entire model must be re-generated on the combined data set must be considered.
Reasoning process is visible	Related to the explanation of a model is the capability for a user to see how a model arrived at its conclusions.
Suit complex	The suitability of a technique to incorporate complex models is related to the issue of model-free estimation and the ability to add expert knowledge.
Include known facts	The technique's capability to include known information into a model, that is, to initialise a model with known facts (expert knowledge) and then use data to improve and refine it.

Table 2.3: Software cost system evaluation criteria by Mair et al. (2000)

Criteria	Description
Accuracy	The spread of error in terms of *MMRE*.
Explanatory value	Does the system provide explicit discernible results?
Configurability	How much effort is required to build the prediction system in order to generate useful results?

Table 2.4: Software cost model/estimate, method and application evaluation criteria by Briand and Wieczorek (2000)

Criteria	Description
Quality of model and estimate	An important criterion regarding the quality of an estimation model and estimate, or its estimates, is predictive accuracy. This compares the predicted resource expenditures with actual values, e.g., in terms of the relative error. The higher the quality of an estimate, the less the risk associated with an estimate, the more likely is an estimation method to be accepted by practitioners. This criterion is often considered the most important as models or techniques have to be sufficiently accurate to be even considered as an alternative for resource estimation.
Input variables required	This category considers the kind of inputs required to develop and use a model. For generic models, we will consider whether it is possible to tailor inputs to a particular environment and the extent to which they can be objectively assessed in terms of their contribution to estimating resource expenditures. For example, COCOMO II proposes a set of up to 17 cost-drivers (or model input variables) that one has to estimate and use to produce an effort estimate.
Completeness of estimates	This category evaluates a model's capability to provide estimates for different project resource expenditures like, effort, cost, or duration. During project planning, support is needed for all these types of resources. However, effort estimation has been the focus of most research, as it is believed that cost and duration can then be derived from effort estimates.

Criteria	Description
Type of estimates	This category assesses the different possible types of estimates that a model can provide, like point estimates, interval estimates, or probability distributions. In general, the uncertainty of a resource expenditure estimate should be modelled. This is particularly important in software engineering where decisions are made within the context of large uncertainties and based on risk analysis. For example, it can be a range of values that has a given probability (e.g., 0.95) of including the actual cost.
Calibration	This category captures the extent to which a model can be calibrated based on project data and to which extent calibration is clearly supported by a modelling method. Usually, generic models such as COCOMO need to be calibrated to different environments. But proprietary models and tools do not always provide such a capability in a satisfactory form.
Interpretability	This category specifies the extent to which a model is easy to interpret by a software engineering practitioner, (e.g., project manager). A model that consists of a multivariate regression equation, for example, may not be very easy to interpret, and thus might not be accepted by practitioners in certain contexts. It is often the case that practitioners want to understand how an estimate was obtained before relying on it. Those human factors play a very important role in the adoption of a resource estimation method.
Assumptions	This criterion assesses how realistic are the underlying assumptions of the estimation model(s) in a given context. The more unrealistic the assumptions, the more risky the application of an estimation method.
Repeatability	The repeatability of an estimation method captures the extent to which the steps to use models and techniques, combine their results, and obtain a final estimate, are clearly defined. The better defined and specified an estimation method, the more independent the estimate from any specific human estimator.
Complexity	This characterises the cognitive complexity of the steps that are required to generate an estimate. The more complex an estimation method, the higher the effort invested into estimates, the more error-prone, the less likely to be adopted by practitioners.
Automation of Modelling	This criterion captures the extent of tool support that is available to apply a modelling method in order to derive estimation models. The effort to derive models and evaluate them is drastically reduced when effective tool support is available.
Transparency	This assesses the extent to which the algorithms and heuristics of the estimation method are documented and justified by a clear rationale. This is different from repeatability as the estimation process may be well defined but proprietary and invisible to the estimator. This criterion mostly applies to proprietary estimation methods embedded into commercial tools.
Application Coverage	This category evaluates the extent of possible applications of a model. Questions addressed here are the following: Can the provided models be used for prediction, benchmarking, and/or risk-assessment? Can usage scenarios be readily identified for these purposes?
Generalisability	This assesses the extent to which an estimation method is applicable across development environments. This depends on the conceptual assumptions underlying the estimation methods and its underlying models (if any) and may be supported by empirical evidence reported in existing studies.
Comprehensiveness	This tells how fine grained an estimate can be (e.g., effort predicted at the project, phase, or activity level), and what project activities can be included into an estimate (e.g., administrative and management overhead activities).
Availability of Estimates	This category captures the applicability of an estimation method during the various stages of software development. This mainly depends on the availability of the inputs required by the estimation model(s) or techniques to obtain an estimate.

Criteria	Description
	For example, COCOMO II provides ways to obtain estimates at different stages of a project, each requiring different sets of input variables are used for subsequent stages.
Automation of Method Usage	For a given estimation method, this criterion captures the extent to which the derivation of a final estimate for different purposes such as prediction, risk analysis, and benchmarking are supported by tools.

Table 2.5: Quantitative and qualitative software cost evaluation criteria by Burgess and Lefley (2001)

Criteria	Description
Accuracy	Summary statistics (such as *CC*, *AMSE*, *Pred(.25)*, *Pred25%*, *MMRE*, *BMMRE*).
Resources required	Time and memory needed to train and query.
Ease of set up	Ease of configuration and number of parameters required to be configured.
Transparency of solution or decision	Explanatory value of a solution and how it was reached.
Generality	Extent of generality.
Robustness	Sensitivity of solutions on parameter values.
Likelihood of convergence	Possibility of converging to a solution.
Prediction beyond learning data set space	Extent of prediction.

Table 2.6: Software cost model evaluation criteria aggregated by Ahmed et al. (2005)

Criteria	Description
Underlying model	The underlying model specifies whether the effort prediction soft computing approach is based on an existing algorithmic cost estimation model like COCOMO, SLIM, etc. or based on other models like expert judgment, analogy.
Trainability	Trainability is the ability of a prediction system to learn the relationships between features and adapt during training. This attribute is what has generally been referred to as adaptability in many of the approaches surveyed.
Adaptability	This attribute describes the ability and ease of the prediction system to adjust to new environments as new information and knowledge are supplied. Trainability does not translate to adaptability, as a system could be trainable but not adaptive. Adaptability subsumes trainability.
Sensitivity	This attribute refers to the responsiveness to changes in input data, and the type of input data it can handle (e.g., numeric data, categorical data). Responsiveness to changes in input data assesses the effect an imprecision in input to the model has on the effort estimate produced. For example, we desire to know how well the system can accommodate an error involving size supplied as 900 KLOC as opposed to the actual 850 KLOC.
Aspect coverage	This refers to the ability of the approach to cover wide range of aspects of the development process and environment. For example, whether the effort prediction system takes into consideration the following; reuse, capability-maturity model level, etc.

Criteria	Description
Spectrum coverage	This attribute refers to the coverage of different types (classes) of systems, e.g. organic, semidetached and embedded systems. If a prediction system is not sophisticated, it might not be able to cover the whole spectrum. A prediction system might still be able to model a software project inherently made up of different classes without necessarily breaking the project into different groups, although such systems may be too complicated.
Implementation technique	This attribute captures the implementation approach taken. An implementation approach using soft computing can be as simple as a straightforward application of an underlying model by applying a single soft computing approach, e.g. fuzzifying input/output, or a more sophisticated implementation technique that explores/combines various capabilities of the Soft Computing methods used (e.g., Fuzzy Logic (FL), ANN, Neuro-Fuzzy (NF), Neuro-Genetic (NG), etc.).
Input data	This attribute identifies the type of input data required by the effort prediction system to perform estimates, e.g., LOC. It also reflects the ease of getting the input data and the accuracy in making reasonable estimates. Input data is simply the input required to make estimates using the prediction system, but not to develop the prediction system.
Knowledge acquisition and data source	This refers to the mode of knowledge acquisition considered in developing (i.e., training/adapting) the prediction system, the source of data required and how reliant the system is on the data source. The mode of knowledge acquisition could either be manual, with users being the source of the knowledge or automatic (based on perceived relationship between the data through learning). For example, a system that relies on users to supply rules in a FL-based approach is said to exhibit manual knowledge acquisition. The data source can be either from historical or simulated data.
Complexity of the model	This attribute refers to the amount of effort or size (e.g., number of neurons, number or rules, etc.) required for building and/or using the prediction system. This attribute reflects the efficiency of the prediction system. A model that is not practical or rather difficult to use might not be a good model. For example, ANN are known to give good approximations, but they might be overly complex and require considerable effort and expertise.
Accuracy	Accuracy is the attribute of a prediction system that reflects its effectiveness. A software manager who wants to use a prediction model would desire to use an accurate one.
Transparency	Transparency of a prediction system reflects the visibility of the prediction process to the software engineer/expert. Interaction or collaboration between the prediction system and the end-user/expert is of great importance, especially for maintenance purposes. If a system is transparent, an expert can easily evaluate and add his own knowledge to improve accuracy of the model, because it would be possible to see and understand the processes involved. Empirical research has indicated that experts coupled with prediction systems outperform either prediction systems or experts alone.
Extendibility	Extendibility reflects the ability of a prediction system to accommodate changes to its model, in that it will be useful for predicting effort required for other activities of software development, e.g., maintenance, testing, etc. A prediction system that uses an underlying model in such a way that the prediction process expects a specific type of input, might not be useful on extending it to other activities for which such inputs are not defined.

Table 2.7: Estimation inputs, process and outputs evaluation criteria by Zhang and Zhang (2009)

Criteria	Description
Comprehensiveness	Whether the users can clearly understand definitions and requirements of inputs that estimation methods need.
Accessibility	Whether the inputs information needed can be received in estimation stage, not until the project completes.
Objectivity	Whether the users' subjective judgment information can be minimised.
Parsimony	Whether the input of unnecessary information and factors which have little impact on result can be excluded.
Scientificalness	Whether the analytical method or data processing method used in estimation is reasonable, and there is no objection to the basic hypothesis condition.
Repeatability	Whether the description of estimation process is detailed and clear enough, and subjective comprehension error produced in application can be avoided.
Sustainableness	Whether the estimation method can answer problems of response ability enhancement and timeliness spread by adjustment of relevant parameters and calibration within the organisation.
Information Completeness	Whether the outputs provided can satisfy the requirement of information scope users need.
Results Reliability	Whether the estimation result is credible and how can it be validated.

The problematic issue of definition regarding the evaluation criteria summarised above is apparent; even thought the researchers agree in the general concept of the evaluation criteria they appeared to be quite inconsistent in terms of definition, i.e., what each criteria means in each respective study is quite different. Most research studies summarised proposed the evaluation of different aspects of the models, introduced non-identical descriptions (using different terminology) for sometimes the same criterion, and in some cases, the definitions of some criteria overlap. This makes the decision to evaluate software cost models on specific criteria hard and the comparison of the results reported from various SCE studies and models in the literature even more difficult.

Nevertheless, if we attempt to gather the common criteria described in at least half of the aforementioned studies (even though the list is indicative and it could be expanded in a future investigation and although quite inconsistent terms were used), the following merged list is established: Accuracy (i.e., fidelity as described by Boehm (1981)), comprehensiveness (i.e., transparency as described by Briand and Wieczorek (2000)), robustness (i.e., sensitivity as described by Ahmed et al. (2005), usability (i.e., ease of use as described by Boehm (1981)), adaptability (i.e., generalisability as described by Briand and Wieczorek (2000)) and early

estimations (i.e., prospectiveness as described by Boehm (1981)). This aggregated list comprises the chief criteria that were considered in the research work conducted in this thesis. Finally, it is worth noting that none of the aforementioned software cost evaluation criteria list (not even our own merged list) claims to be complete or fully applicable for evaluating every type of cost model or technique.

Moreover, this section concludes with the most popular accuracy evaluators used in the area of SCE since accuracy is the primal concern of project managers, stakeholders and researchers. The criteria of evaluation for the performance results of the methods/models developed in this research are obtained using a combination of common error metrics found in literature (Conte et al., 1986; Foss et al., 2003; Jørgensen et al., 2004), namely the *Mean Magnitude of Relative Error* (*MMRE*), the *Correlation Coefficient* (*CC*) and the *Normalised Root Mean Squared Error* (*NRMSE*). These error metrics are employed to validate the model's forecasting ability considering the difference between the actual and the predicted cost values and their ascendant or descendant progression in relation to the actual values. However, in cases where utilising a range of accuracy metrics made the comparison of the results of a particular model difficult (especially since they measure different aspects of the model), in this work, we mainly focused on using the most popular metric in the SCE literature, i.e., the *MMRE* for comparing the results of one or more models.

The *MMRE*, given in eq. (2.1), shows the prediction error based on the sample being predicted; $x_{act}(i)$ is the actual effort and $x_{pred}(i)$ is the predicted effort of the i^{th} project.

$$MMRE(n) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_{act}(i) - x_{pred}(i)}{x_{act}(i)}\right| \tag{2.1}$$

The *CC* between the actual and predicted values, described by eq. (2.2), measures the ability of the predicted samples to follow the upwards or downwards of the original values as it evolves in the sample prediction sequence. An absolute *CC* value equal or near 1 is interpreted as a perfect follow up of the original series by the forecasted one. A negative *CC*

sign indicates that the forecasting series follows the same direction of the original with negative mirroring, that is, with a $180°$ rotation about the time-axis.

$$CC(n) = \frac{\sum_{i=1}^{n}\left[\left(x_{act}(i) - \bar{x}_{act,n}\right)\left(x_{pred}(i) - \bar{x}_{pred,n}\right)\right]}{\sqrt{\left[\sum_{i=1}^{n}\left(x_{act}(i) - \bar{x}_{act,n}\right)^2\right]\left[\sum_{i=1}^{n}\left(x_{pred}(i) - \bar{x}_{pred,n}\right)^2\right]}} \tag{2.2}$$

The *NRMSE* assesses the quality of predictions and is calculated using the *Root Mean Squared Error* (*RMSE*) as follows:

$$NRMSE(n) = \frac{RMSE(n)}{\sigma_{\Delta}} = \frac{RMSE(n)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[x_{act}(i) - \bar{x}_n\right]^2}} \tag{2.3}$$

$$RMSE(n) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[x_{pred}(i) - x_{act}(i)\right]^2} \tag{2.4}$$

If *NRMSE*=0 then predictions are perfect; if *NRMSE*=1 the prediction is no better than taking x_{pred} equal to the mean value of *n* samples.

In addition to the above, four evaluation metrics were used to evaluate the model's performance, namely the *Prediction of specific Level l* (*Pred(l)*), *Mean Balanced Relative Error (MBRE)*, *Mean Squared Error* (*MSE*) and *Mean Absolute Error* (*MAE*). These are specified in eqs (2.5), (2.7), (2.8) and (2.9) respectively. Eq. (2.5) defines the ratio of the accurate data predictions *k* to the total number of data points predicted *n.* This accuracy is measured by the *RE* metric (given in eq. (2.6)) which must be lower than level *l*. For our case the parameter *l* was set equal to 0.25. *MBRE* measures the bias in prediction accuracy and in some cases may be considered more appropriate than *MMRE* for the analysis of linear relationships, especially in datasets including projects with strongly underestimated effort. *MSE* and *MAE* are scale dependent errors and do not offer objective evaluation. Nevertheless, in some experiments these error figures are calculated as well, to facilitate the comparison with similar studies which report these metrics.

$$pred(l) = \frac{k}{n} \tag{2.5}$$

$$RE(n) = \frac{\left| x_{act}(i) - x_{pred}(i) \right|}{x_{act}(i)} \tag{2.6}$$

$$MBRE(n) = \frac{1}{n} \sum_{i=1}^{n} \left\{ \begin{array}{l} \dfrac{x_{act}(i) - x_{pred}(i)}{x_{act}(i)}, \ if \ x_{act}(i) \le x_{pred}(i) \\ \dfrac{x_{act}(i) - x_{pred}(i)}{x_{pred}(i)}, \ if \ x_{act}(i) > x_{pred}(i) \end{array} \right\} \tag{2.7}$$

$$MSE(n) = \frac{1}{n} \sum_{i=1}^{n} \left(x_{pred}(i) - \bar{x}_{pred,n} \right)^2 \tag{2.8}$$

$$MAE(n) = \frac{1}{n} \sum_{i=1}^{n} \left| x_{act}(i) - x_{pred}(i) \right| \tag{2.9}$$

The following section summarises the related work on SCE models.

2.4 Overview of Related Work in Software Cost Estimation

A huge number of empirical studies on SCE models exist. Jørgensen and Shepperd (2007) report 61% of the 304 papers published before 2005 that concerned research on software development effort or cost estimation, have also introduced and evaluated estimation methods. Although the work on developing SCE methods is vast and therefore quite difficult to survey adequately this section reports the main streams of research work.

Foremost models (49% according to Jørgensen and Shepperd (2007)) are based on regression analysis, which is usually used as a baseline to compare the performance of other models (Myrtveit et al., 2005). These models assume that by employing some independent attributes as inputs (i.e., project characteristics) and a dependent variable as the output (namely development effort) the resulted complex I/O relationships may be captured by an explicit formula. Some examples adopting the concept include the work of Miyazaki et al. (1994), Finnie et al. (1997), Angelis and Stamelos (2000), Boehm et al. (2000b), Barry et al. (2002), Benediktsson and Dalcher (2003), Mair and Shepperd (2005), Nguyen et al. (2008)

and Mittas et al. (2010). Project managers are expected to produce a preliminary software cost estimate based on (i) either solely on their experience (Jørgensen, 2007), (ii) or with the help of the cost models developed (Boehm et al., 2000b), (iii) or through statistical analysis of historical data (Liu et al., 2008), (iv) or through a combination of the above methods (MacDonell and Shepperd, 2003a; Jørgensen and Shepperd, 2007). Also, most of the aforementioned studies applied the Ordinary Least Squares (OLS) Regression method.

Substantial research (22% of the studies reviewed by Jørgensen and Shepperd (2007)) involves Function Point-based estimation approaches. These approaches mainly involve identifying and classifying the major system components such as external inputs, external outputs, logical internal files, external interface files and external inquiries. The classification is based on their characterisation as 'simple', 'average' or 'complex', depending on the number of interacting data elements and other factors. Then, the unadjusted Function Points (FP) are calculated using a weighting schema and adjusting the estimations utilising a complexity adjustment factor (Albrecht, 1979). This is influenced by several project characteristics, namely data communications, distributed processing, performance objective, configuration load, transaction rate, on-line data entry, end-user efficiency, on-line update, complex processing, reusability, installation ease, operational ease, multiple sites and change facilitation. Some examples include the work of Albrecht and Gaffney (1983), Kemerer (1987), Heemstra and Kusters (1991), Betteridge (1992), Matson et al. (1994), Horgan et al. (1998), Antoniol et al. (2003), Zivkovic et al. (2005) and Xia et al. (2008).

Another increasing stream of work involves Expert Judgements (15% of the studies from the aforementioned reference, i.e., Jørgensen and Shepperd (2007)) which includes consulting one or more experts and in which the estimation process may be directed on a non-explicit, non-recoverable reasoning processes, i.e., 'intuition' (Jørgensen, 2004c). Moreover, most expert-based estimations rely on analogy of the estimator's past experience on similar projects developed. Some examples include the work of Hughes (1996), Höst and Wohlin (1997), Hill et al. (2000), Rush and Roy (2001), MacDonell and Shepperd (2003), Jørgensen et al. (2004), Jørgensen (2004b) and Jørgensen (2004c).

In recent years, the diversity and growth of SCE approaches have increased and a tendency towards alternative approaches like, analogy (10%), neural networks (7%), Bayesian (2%) and other approaches (8%) (such as genetic programming, linear programming, soft computing, fuzzy logic modelling and bootstrap-based analogy) has been reported (the percentages reported are obtained from Jørgensen and Shepperd (2007)). Especially in recent years, research is increasing towards Computational Intelligent (CI) techniques and combinations of approaches. Machine Learning or CI research work in SCE has gained a lot of interest within researchers and has also stimulated the research work conducted and included in this thesis. Therefore, specific research work related with CI techniques is discussed in the subsequent section. Whereas also considerable benefit is achieved by combining more than one technique to obtain an estimate, or the individual results of more than one technique to attain again at the end a unique estimate. Examples of such composite methods include the hybrid expert and regression-based methods of Boehm and Sullivan (1999), Genuchten and Koolen (1991), Kitchenham et al. (2002) and MacDonell and Shepperd (2003b). Clearly, these combinations influenced the results and led to the convergence of a more 'reasonable' value of the estimation (as reported by MacDonell and Shepperd (2003), but also enhanced the substantiality of the techniques by combining their individual strengths. Finally, a common approach by organisations and/or individual estimators is to optimally tailor and apply existing methodologies according to their requirements, or device new techniques to benefit the most out of them.

Despite the vast research conducted and variety of SCE methods proposed, the need to answer the following question still exists: "*Which model is more appropriate under specific conditions?"* (Shepperd and Kadoda, 2001). Moreover, even if 'acceptable' (accurate) cost predictions are produced, they do not point to an explicit, measurable and concise set of factors affecting productivity. Also, most techniques do not account for the uncertainty which is commonly discovered in the values of software cost factors. As already mentioned, many researchers have resulted in investigating intelligent CI and hybrid techniques since promising

early indications were found on their effectiveness. The emergence of CI techniques and related work in SCE is presented in the next section.

2.5 Computational Intelligence in Software Cost Estimation

Latest trends in SCE research seems to promote the adoption of CI techniques, in combination with other approaches for building hybrid models capable of yielding more robust results (MacDonell and Shepperd, 2003a; Huang and Chiu, 2006). One of the challenges identified includes increasing the generalisability, comprehensibility and transparency of the approaches and techniques, in order to contribute in reducing the inherent uncertainties and increasing user acceptance. In addition, the investigation and understanding of the interrelated factors affecting cost and their relationship with effort and productivity is limited even for experts, and thus demands further research (Jørgensen, 2007).

A wide range of studies report research attempts to approximate effort, either directly based on a CI technique or via their combination with other CI and/or statistical, analogy or expert-based techniques. These attempts have resulted in improved, hybrid forms of SCE models. However, even if a particular method outperforms others in a comparative analysis, it may still not be of any practical use in real project management if the prediction is not based on software factors that may be measured in the early phases of the project life-cycle.

Early studies, such as the work of Serluca (1995), compared the results of three methods for effort estimation: Regression, analogy and Artificial Neural Networks (ANN). Utilising the MERMAID-2 dataset ANN achieved far more superior results compared to regression and marginally better than analogy when the dataset was fully used. However, when the dataset was separated into two more homogenous and therefore smaller clumps, the ANN performed very poorly, while the other two methods improved considerably. This led the author to conclude that ANN require large training sets before they can yield accurate predictions.

Srinivasan and Fisher (1995) compared ANN and regression trees for predicting effort reported in the Kemerer dataset, using the COCOMO dataset for training. The results of the experiments were in favour of ANN.

Jørgensen (1995) utilised four modelling approaches to estimate maintenance effort: regression, ANN, a form of pattern recognition and a simple baseline rule of thumb model according to which, "*effort is equal to size divided by the mean productivity*". The study used a Multi-Layer Perceptrons (MLP) with a back-propagation training algorithm on the Jørgensen95 dataset and the ANN was found to perform worse than the best regression model in terms of the *MMRE,* but very successfully in terms of the *Pred(.25)* metric.

Wittig and Finnie (1997) compared a back-propagation MLP ANN with Case-Based Reasoning (CBR) which is a form on analogy, using the Desharnais dataset and 136 sample observations from the Australian Software Metrics Association (ASMA). In this work the ANN yielded very encouraging results, with only utilising the attribute of system size to obtain the effort predictions. The trials conducted to test the model combining attributes other than size resulted in reduced prediction errors, which suggested that there is room for further investigation and improvement through a more systematic study of the development characteristics.

Samson et al. (1997) developed an Albus MLP to predict software effort, which operates in a similar way to a lookup table, using a generalisation mechanism so that a solution learned at one point in the input space influenced solutions at neighbouring points. Different ANN were then compared with linear regression. Although predictions made by the ANN outperformed those produced by linear regressions using the COCOMO dataset the accuracy of the prediction results obtained was rather low.

Hughes (1997) compared a wide range of approaches for effort estimation including analogy, regression, and ANN, using the WSD1 dataset. The dataset was initially divided into two homogenous groups. When the two groups were merged the *MMRE* was improved, reinforcing the fact that ANN can perform well when presented with larger datasets, while, at

the same time, performance of other techniques, including analogy and regression, deteriorated.

MacDonell and Gray (1997) proposed the combination of feedforward MLP ANN and fuzzy models to overcome the uncertainty and limitations of the measurements included in the datasets of past projects. However, the results obtained from the models on the Desharnais samples were not overly impressive.

Gray and MacDonell (1997b) compared Function Point Analysis, regression, ANN and FL using real data from the Canadian industry. The results indicated that FL achieved good performance with only ANN outperforming estimation accuracy, but with considerably more input variables. In their approach triangular membership functions were defined for small, medium and large intervals of size, complexity and effort, and, additionally, expert judgment was utilised to define an initial set of nine rules, which were later refined.

Mair et al. (2000) evaluated predictions of effort utilising regression, rule induction, CBR and ANN models. The results showed considerable variations in accuracy, unstable and inconsistent results. This was attributed to the various datasets utilised which contained different attributes, like the number of features and the number of projects, and, additionally, the series of data presented outliers, collinearity and thus convergence was difficult to be obtained.

Idri et al. (2000) employed algorithmic Fuzzy Logic (FL) models to offer fuzzification of the COCOMO cost drivers. The fuzzy Intermediate COCOMO developed showed that it can tolerate input imprecision, is less sensitive to input uncertainty and can generate more robust cost predictions.

Musilek et al. (2000) used FL concepts to represent only ‘mode’ and ‘size’ as inputs to the COCOMO model, which was called f-COCOMO. With this extension the non-numeric nature of the inputs was considered more appropriate to use by project managers and fuzzy sets provided the means for a more flexible and highly versatile development environment.

Idri and Abran (2001) in an attempt to address the need of analogy-models to handle categorical attributes, proposed the utilisation of fuzzy reasoning for measuring the similarity

between projects expressed in categorical values. Categorical data were represented by linguistic values through fuzzy sets suggesting that fuzzy analogy is able to handle such data. The validation experiments performed proved that fuzzy Intermediate COCOMO offered improved estimations.

Burgess and Lefley (2001) performed a comparative evaluation of the following techniques: ANN, CBR, *k*-nearest neighbours (*k*-NN) where *k* was set equal to 2 and 5, Linear Least Squares Regression (LSR), Genetic Programming (GP) and random to test the hypothesis of whether GP can improve software effort estimates. In terms of accuracy, GP was found more accurate than the rest of the techniques for the Desharnais dataset, but it did not converge to a good solution as consistently as ANN.

Dolado (2000) investigated GP evolving tree structures which represent cost equations compared to classical equations, like the linear, power, quadratic, etc. Different datasets were used, i.e., the Abran and Robillard, Albrecht and Gaffney, Bailey and Basili, Belady and Lehman, Boehm, Heiat and Heiat, Academic, Kemerer, Miyazaki et al., Shepperd and Schofield, Desharnais and finally, the Kitchenham and Taylor datasets. The technique yielded diverse results, classified as 'acceptable', 'moderately good', 'moderate' and 'bad' results. The diversified results were attributed to the fact that the datasets examined varied extremely in terms of complexity, size, homogeneity, or values' granularity consistent results were hard to obtain.

Heiat (2002) compared the prediction performance of an MLP and a Radial Basis Function Network (RBFN) to that of regression analysis and found that when for project data implemented with a third generation language the ANN performed equally well with regression. However, when a combined third and fourth generation languages dataset was used ANN outperformed regression.

Lefley and Shepperd (2003) presented results of various effort estimation techniques which included comparison of the following: random, LSR, *k*-NN, ANN, GP and average of all non-random estimators. GP was modelled as a symbolic regression problem in order to improve effort predictions. The so-called 'Finnish dataset' collected by a software project

management consultancy organisation was used in the context of within and beyond a specific company estimations. The results indicated that the approaches of LSR, ANN and GP yielded better predictions than the rest of the techniques.

Huang et al. (2003) combined FL with ANN and demonstrated improved results than the original COCOMO model for software development effort estimation. The neuro-fuzzy model used for input software size and 22 ratings of cost drivers, including 5 scale factors and 17 effort multipliers. The ratings could be numerical continuous values or linguistic terms such as "low", "nominal" and "high". The projects used for validation were 63 projects from the COCOMO dataset and 6 projects from the industry.

Idri et al. (2004) investigated the use and interpretation of RBFN in software cost estimation by mapping the ANN to a fuzzy rule-based system. Results on the COCOMO dataset indicated that the accuracy of the ANN depended heavily on the parameters of the middle layer and more specifically on the number of hidden neurons and the weight values.

Xu and Khoshgoftaar (2004) improved significantly the accuracy results of COCOMO using fuzzy input data rather than using the original data. The authors extracted rules and membership functions using an advanced FL technique and the three types of COCOMO models, i.e., Basic, Intermediate and Detailed.

Braz and Vergilio (2004) proposed the use of the Fuzzy Use Case Size Points (FUSP) metric in object-oriented software, in an attempt to offer 'early' estimations and showed functional size allows better effort estimation than estimations made with USP (i.e. without fuzzy logic concepts).

Several studies by Aggarwal et al. (2005a, 2005b, 2005c) evaluated various techniques for software engineering applications, serving three objectives: Find the optimal training algorithm in an ANN model (Aggarwal et al., 2005a), combine ANN and Bayesian regularization (Aggarwal et al., 2005b), and combine ANN and linear regression (Aggarwal et al., 2005c) to optimise the results. Initially, results indicated that the ensemble with 15 neurons in the hidden layer and the Bayesian regularization training algorithm yielded the best results. Then, various models were compared with regression e.g., linear, quadratic, cubic and

robust, which was found to yield the best performance. Therefore, the robust regression model was then combined with ANN to form an expert committee model, taking advantage of the two, leading to optimisation and achieving better performance.

Ahmed et al. (2005) combined FL and knowledge incorporation to approximate the effort by utilising the Intermediate COCOMO model. The authors proposed a Fuzzy Inference System (FIS) that integrates two components using the set of fuzzy rules produced: (i) training of COCOMO and generation of artificial data and (ii) adjustment (fuzzification) of the cost drivers. Utilising FL enabled the technique to model effectively and adapt well to the complex development environment. Their approach appeared to perform as good as the COCOMO model and could potentially perform better in future investigations.

Huang and Chiu (2006) developed a GA to determine the appropriate weighted similarity measures of effort drivers in analogy-based software effort estimation models. The ISBSG and the IBM DP services databases were used in the experiments and the results obtained showed that among the applied methods, i.e., unweighted analogy, weighted analogy, unequally weighted analogy, linearly weighted analogy, non-linearly weighted analogy, Classification And Regression Trees (CART), ANN and OLS, the GA produced improved estimates and the method could provide objective weights for the weighted analogy methods rather than the subjective weights assigned by experts. Finally, the non-linearly weighted analogy method produced superior prediction accuracy among the weighted analogy methods investigated.

Huang et al. (2006) embedded risk assessment into a fuzzy decision tree approach for SCE and showed that complex cost factors structures can be explained better if fuzzification was applied. The approach combined the comprehensible rules generated by the ID3 decision tree algorithm with the expressive power of fuzzy sets. For the verification projects from the COCOMO dataset were used.

Kumar et al. (2008) compared the SCE prediction of Wavelet Neural Networks (WNN) with MLP ANN, RBFN, Multiple Linear Regression (MLR), Dynamic Evolving Neuro-Fuzzy

Inference System (DENFIS) and Support Vector Machines (SVM) in terms of the MMRE. WNN seemed to outperform all other techniques.

Tronto et al. (2008) investigated the application of ANN and stepwise regression for software effort prediction. The experiments were conducted on the COCOMO dataset employing categorical variables whose impact was identified based on the work of Angelis et al. (2001) forming new categorical values. It was observed that there is a strong relationship between the success of a technique and the size of the learning dataset, the nature of the function for cost and other dataset characteristics (such as existence of outliers, collinearity, number of attributes etc.).

Aroba et al. (2008) employed fuzzy clustering to organise data samples of the ISBSG R8 into several subsets and yielded better figures of adjustment than their crisp equivalents. Although the approach presented some encouraging results and also provided higher explicative capabilities, some problems were identified regarding the use of fuzzy clusters in segmented models of parametric software estimation.

Park and Baek (2008) built and evaluated ANN effort estimation models by using regression analysis and expert interviews to select the input variables. The ANN model was compared to expert judgement and two traditional regressions. The authors found ANN to yield more accurate predictions and also emphasised that most of the existing studies focus on selecting the best estimation method without mentioning how variables are being selected, but usually refine the set of factors by a trial and error approach. In such an approach the different sets of factors are then tested repeatedly until the evaluation criteria are met. The authors also add that a method to define which factors to use as inputs in ANN does not exist yet and underline that it is critically important to identify dominant factors that should be used in these models.

Reddy and Raju (2009) used the popular COCOMO model mapped to an ANN with minimal number of layers and nodes to increase the performance of the network. They employed a feedforward backpropagation MLP and obtained improved predictions for effort using the COCOMO dataset compared to the COCOMO model.

Rao et al. (2009) used a Functional Link Artificial Neural Network (FLANN) which does not contain any hidden layers so that the network architecture becomes simple and training does not involve full backpropagation, thus reducing computational complexity. Their method provided more accurate results compared to other methods for software cost estimation on the NASA dataset.

Azzeh et al. (2010) explored the impact of Grey Relational Analysis (GRA) integrated with Fuzzy set theory for a by-analogy estimation model and also compared to ANN, CBR and MLR models using several public datasets, i.e., ISBSG, Desharnais, COCOMO, Albrecht and Kemerer. The study aimed to reduce the uncertainty in the similarity degree of different forms of measures i.e., for continuous, nominal, or ordinal types between two tuples with X features. The Fuzzy GRA appeared to produce statistically more significant results than the rest of the models. Moreover, it effectively reduced the uncertainty of attribute measurement between two software projects and improved the way to handle both numerical and categorical data in similarity measurements.

Attarzadeh and Ow (2010) showed that the fuzzification of the scale factors, cost drivers and size metrics from the COCOMO and an artificial dataset improved the performance of the traditional COCOMO II model. The validation was performed using the projects of the COCOMO dataset and 100 artificial projects.

Mittal et al. (2010) fuzzified software's size metric and used FL to tune the parameters of the COCOMO model. Their results showed improvement compared to the results of Bailey-Basili, Doty and Halstead models in terms of prediction accuracy.

Prasad Reddy (2010) combined Particle Swarm Optimisation (PSO) to further optimise estimates of the FL technique for developing 10 NASA software, 18 NASA and 63 COCOMO projects. Two models were built which yielded better results than the rest models compared.

Bhatnagar et al. (2010) attempted to estimate cost in the early stages of development by using Entity Relationship Diagrams, Use Case Diagrams and Class Diagrams of student projects. The authors identified that parameters used in cost estimation models contain some

degree of fuzziness and this requires uncertainty to be introduced in the models. Thus, they included uncertainty in the metrics measured by the students at the requirement, analysis and design phases of development by applying fuzzification, rule-based inference and defuzzification to obtain crisp effort values.

Kaur et al. (2010) proved the effectiveness of ANN models for the NASA dataset compared to the Halstead, Walston-Felix, Bailey-Basili and Doty models, all of which are popular legacy models used in software cost estimation. Backpropagation ANN were used and reported as the most generalised networks currently in use that present good estimation capabilities.

Summarising the above, the current SCE literature is quite rich in studies reporting the use of CI techniques and comparing the prediction results obtained from various models. Moreover, various hybrid forms of techniques and cost models attempting to improve intuitiveness, accuracy and robustness emerged in an increasing number of studies. In several cases the CI techniques employed were found to outperform the techniques they were compared to and in addition, offered considerable improvements. In addition, CI techniques offered the advantage to automatically detect complex relationships taking into consideration historical empirical data and derive estimates independent from the subjective opinion of domain experts. However, the heterogeneous sampling, measurement and reporting techniques used for software data (Mair and Shepperd, 2005), the inconsistency and inappropriateness of the performance measures and statistical tests used for the comparison of alternative models (Kitchenham et al., 2001; Kitchenham and Mendes, 2009) and other problems discussed in the subsequent section lead in many cases to incomparable and non-conclusive results.

Particularly, the use of ANN in most cases showed some promising results, although one may argue that they mainly lack transparency on the way they work and on how their results are interpreted. ANN also present high data dependence and may fail to generalise if they are not properly trained and calibrated. In addition, hybrid and evolutionary forms of SCE techniques were found by researchers to improve the solutions obtained and reach to moderate

levels of accuracy. Other popular learners, such as decision trees and association rules, since they learn by examples, in the presence of outlying data or too few data samples, it is quite possible that they may fail to generalise well. Therefore, popular alternatives for learning and reasoning, such as fuzzy hybrid forms, offer the advantage of adaptive learning, increasing the comprehensibility of the results obtained and the ability to deal with inexact and uncertain information expressed with fuzzy representations. Many of the aforementioned relative research studies have identified that many parameters used in SCE models contain some degree of uncertainty. Thus, this implied that Fuzzy Logic introduced in such models could lead to considerable improvements in the SCE prediction results of the techniques applied.

2.6 Open Research Problems in Software Cost Estimation

This section summarises the main open research problems in SCE. Considering the track record of the software industry from a survey performed in 2001 by the Standish Group[1], it seems that 23% of all software projects are cancelled before completion. From those projects that are actually completed only 28% are delivered on time, within budget and with all originally specified features. Additionally, the average percentage of software project budget overruns is 45%. Recent survey reviews indicate that most projects (60-80%) encounter cost overruns of the range of 30-40% (Moløkken and Jørgensen, 2003). This shows that there is still ample room for improvement regarding cost estimation accuracy. The rest of this section summarises the main reasons why this phenomenon occurs.

Recent performance figures of 8,000 projects (listed in Table 2.8) gathered from 350 organizations and reported in the Chaos survey from 2000-2006 (The Standish Group, 2007)

[1] Older reports mention that for every 100 projects that start there are on average 94 restarts. An average of 189% of projects exceed the original cost, time or schedule estimates by 239%, and more than one quarter of the projects are completed with only 25%-49% of the originally-specified features and functions (The Standish Group, 1994). The subsequent year's figures report an average of more than 50% of the completed projects have less than 50% of the original requirements (The Standish Group, 1995).

show that a large percentage of projects overrun budget and schedule estimates. According to a Web poll with 1000 respondents released by the Computing Technology Industry Association (CompTIA) the primary reason most IT projects fail is poor communication, insufficient resource planning was found to be the second reason, while unrealistic deadlines was the third (Rosencrance, 2007).

Table 2.8: The performance of 8,000 projects in 350 organizations (The Standish Group, 2007)

Year	2000	2002	2004	2006
Percentage of projects delivered within budget and schedule	28	34	29	35
Percentage of projects cancelled before completion	23	15	18	19
Percentage of projects overrun on budget and schedule	49	51	53	46

Such statistics are usually included in case studies on project failures, articles and estimation surveys, as well as reports produced by project management consultants. These resources characterise effort estimation as a key research area in software engineering and project management. Even though the situation has been somewhat improved since the first references on the problems in producing software (Brooks, 1995; De Marco and Lister, 1999), these numbers have not changed much, compared to the aforementioned recent reported statistics. These statistics reveal the complex underlying problems concerning the development procedure and emphasise the difficulties of the SCE process.

The ultimate objective of the research conducted in the area is to develop a repeatable process for creating and empirically validating SCE models. Jones (2007) mentioned that there is lack of theoretical grounds to support the process of assessing the effect of development on software cost: "*Measurements, metrics, and statistical analysis of data are the basic tools of science and engineering. Unfortunately, the software industry has existed for more than 50 years with metrics that have never been formally validated and with statistical techniques that are at best questionable.*". Therefore, a primary need is to investigate existing foundations of research and the form of available historical data (metrics and measurements) in software engineering.

A recent review of Jørgensen and Shepperd (2007) reports a disconnection between research and the actual use of effort estimation methods, such as the ones described in the

previous section. More specifically, it seems that studies in real-life estimation situations are rare as the authors describe: "*there seems to be a lack of in-depth studies on the actual use of estimation methods and real-life evaluations published as journal papers*". Moreover, it is worth noting that throughout the material reviewed in the aforementioned study (304 software cost estimation papers in 76 journals), the authors could not locate any study with in-depth data collection and analysis of how estimation methods were actually applied, even for well-distinguished models such as the COCOMO (Boehm, 1981).

Despite the vast research background of SCE the need for better costing in software projects is obvious and until today is considered a highly difficult task (Briand and Wieczorek, 2000; Moløkken and Jørgensen, 2003; Jørgensen and Shepperd, 2007; Menzies at al., 2010). The main challenges relate to: (a) establishing basic and consistent terminology (Grimstad et al., 2006), (b) obtaining objective measurements of software cost drivers (Fenton and Pfleeger, 1997), (c) modelling and estimating, as well as other practical issues related with the SCE models and methods proposed thus far (Kan, 2003; Laird and Brennan, 2006).

In essence, accuracy and robustness (in terms of stability) of SCE are of primal concern and are one of the most difficult merits to acquire. The reason is that the experimental studies performed are governed by unstable factors since different (a) datasets, (b) pre-processing and transformation methods, (c) evaluation criteria, (d) design of experiments, (e) variations of parameters, are used and thus inconsistent and inconclusive comparisons are presented (Mair and Shepperd, 2005; Kitchenham and Mendes, 2009).

The main reason that cost estimations are usually inaccurate is that some software cost drivers affecting development effort relate to highly subjective characteristics. These project characteristics are purely qualitative, or subjective and thus hard to measure. Most of them are not standardised, heterogeneous sampling and reporting techniques are used and exhibit complex influencing inter-relationships that cannot be easily modelled. In addition, due to the fact that software development is driven by a large number of factors that are consequently used in SCE models (refer for example to the popular COCOMO which uses 16 cost drivers (Boehm, 1981) and its successor COCOMO II which uses 23 cost drivers (Boehm et al.,

2000a)), on the one hand they can be quite hard to calibrate to specific environments and on the other, they may result in strong collinearity, heteroscedasticity and unstable prediction accuracy (Li et al., 2010; Nguyen et al., 2008).

Also, as previous research identified, there is lack of standard definitions in the software terminology (Grimstad et al., 2006), causing the presence of inconsistencies in the empirical data samples (Miyazaki et al., 1994). In order to be able to estimate more accurately software effort, researchers need to understand, accept, and manage these inherent inconsistencies in estimation and change the existing terminology to include uncertainties as proposed by DeMarco (1982), that is, include the effect from other fields and estimating upper and lower bounds (Kitchenham et al., 2003).

Apart from the aforementioned complexity of the influencing cost factors, the constant pressure on standardisation and lack of software data, many other open problems have been identified as equally important. These refer to the need to systematically address missing data values, detect and eliminate outliers, perform feature subset selection and facilitate the continuous evolution/adjustment of predictions as the project unfolds (Song and Shepperd, 2011). Moreover, as recent studies report, the quality and appropriateness of the datasets utilised in most effort estimating techniques are key factors to obtaining better results (Kitchenham and Mendes, 2009). Furthermore, finding an appropriate software cost model for every dataset is practically impossible, since several datasets have been proven to exhibit unstable results and other datasets have been proven unsuitable to distinguish the performance of different methods (Keung et al., 2012).

Another important research issue of practical importance is the time-period that the cost estimate is performed. This time-period is related to when the system is proposed and when it is delivered. According to Boehm et al. (2000a) a manager may accurately estimate the resources required to develop a system according to how much information is available at the time of the estimate. Therefore, throughout most of the software development life-cycle, estimations regarding size and effort can be quite uncertain. As the software process proceeds this level of uncertainty is lowered and thus the estimation accuracy is increased. This

dependency became widely known as the "cone of uncertainty" and is illustrated in Figure 2.3. Kitchenham and Linkman (1997) attribute this uncertainty to the following factors: (a) measurement error of input variables, (b) model error, (c) assumption error of input parameters, and finally, (d) scope error.

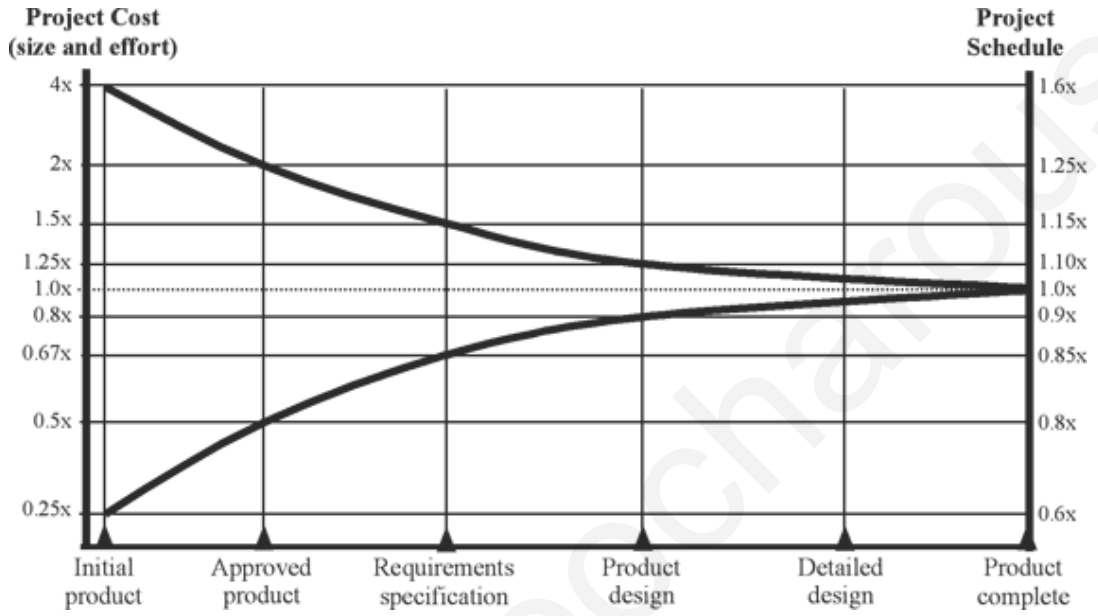


Figure 2.3: The cone of uncertainty (Boehm et al., 2000a)

Kitchenham et al. (2003) proposed the representation of effort not only as a crisp value, but as a distribution of values in order to deal with these uncertainties of SCE. Grimstad (2006) also proposed to view effort in a probabilistic manner, and thus, for example, the 'most likely effort', the 'planned effort' and the 'budgeted effort' will be values (with different probabilities of being exceeded by actual effort) on an effort probability distribution. Gruschke and Jørgensen (2008) introduced the notion of prediction interval as "*a minimum-maximum range of values for the effort estimates, attached with a confidence level that the actual value of the effort will be included in the range*". Accordingly, such estimates extend the typical single-value estimates that do not associate the estimation with the degree of 'how good' an estimation is, which would have been more informative.

In the rest of this section, which also concludes this chapter, the open problems tackled by this diatribe are summarised. All aforementioned issues were taken into consideration for the research work conducted. The thesis focused on devising novel hybrid techniques taken from

the field of CI. The issues of primal concern involved were: (a) performance accuracy in terms of prediction, and (b) robustness in terms of responsiveness to data and parameter changes. The SCE models investigated these two factors through size and other cost drivers used as inputs, and additionally, their degree of influence was examined. Particularly, to the best of our knowledge the contribution of the variables within a model is rarely measured in the SCE context. Most studies focus on the accuracy of the models and few on their explanatory value, even though both are considered very important attributes (Bibi et al., 2008). Thus, the models developed employed feature selection based on the input's significance and accuracy optimisation through hybrid techniques.

Moreover, this thesis handled one of the main obstacles of constructing empirical SCE models, i.e., the subjectivity of the available data of the software development domain. The subjectivity is addressed through pre-processing (filtering) activities, including outlier's removal, fuzzification and other transformations carried out, as well as clustering and classifications to yield interval effort predictions instead of crisp predictions. Thus, the models developed incorporated uncertainty in the techniques employed for prediction. The models produced are repeatable, useful in practice, may be easily understood by the end-users and overcome several limitations like the uncertainty. The next chapter includes the technical background of the research conducted in this dissertation.

Chapter 3

Technical Background

This chapter presents briefly the technical background behind the two types of Computational Intelligent (CI) models proposed and explored in this thesis for SCE. The models developed concern (a) Quantitative models and (b) Qualitative models.

3.1 Introduction to Computational Intelligence

Computational Intelligence (CI), a term coined in the early '90s, is a branch of Artificial Intelligence (AI) relying on coherent heuristic algorithms, including Fuzzy Systems, Artificial Neural Networks and Evolutionary Computation providing important advantages for global optimisation. In addition, CI embraces techniques that use Swarm Intelligence, Fractals and Chaos Theory, Artificial Immune Systems, Wavelets, etc. (Engelbrecht, 2007). It is an internationally well-established scientific field for which many journals and conferences provide in-depth research contributions and that were applied in many domains.

CI combines elements of learning, adaptation, evolution and fuzzy logic (rough sets) to create programs that are, in some sense, intelligent. CI research does not reject statistical methods, but often gives a complementary view. In this diatribe CI is utilised in devising software cost models and improving them for achieving practicality and effectiveness.

3.2 Quantitative Models Technical Background

The data-driven quantitative models proposed are built on soft-computing notions such as Artificial Neural Networks (ANN), Fuzzy Systems and combinations of techniques from Machine Learning (ML) and Evolutionary Computing. In this research work, two non-linear techniques, namely ANN and Ridge Regression (RR) are mainly used as software cost predictors. ANN provide the advantage of modelling complex, non-linear relationships from noisy domains and RR is an improvement of the classical Ordinary Least Squares (OLS) Regression, which is a promising solution in cases where high correlations exist between variables. Both of the aforementioned techniques are used for modelling and prediction. Another form of software cost estimator utilised in this thesis, is the Fuzzy Implication System (FIS) which is used to obtain crisp effort estimations by aggregating rule-based inferences. The details of these techniques are summarised in this section.

In addition, the method of analogy is used as a predictor for three cases namely, in the case of clustered projects obtained from the Entropy Fuzzy *k*-modes Algorithm (Tsekouras et al., 2005) and classified projects obtained from the association rules of Fuzzy Decision Trees (FDT) (Papatheocharous and Andreou, 2009b; Andreou and Papatheocharous, 2008b) and the Genetic Programming equations (Papatheocharous et al., 2010a). Analogy-based estimations are also synonymous to Case-Based Reasoning (CBR) which uses past cases of similar projects to determine the value of effort of the target project (Li et al., 2007). Li et al. (2007) mention that effort estimation with analogies outperforms the rest estimation methods in 60% of the cases of published studies, while in 30% of the cases it yields the worst predictive accuracy. This observation suggested some instability of the method (Ruhe et al., 2003). It therefore can be concluded that effort estimation by analogy is promising, but needs further improvement in terms of better accuracy and broader applicability. Therefore, analogy is used in this thesis for estimating the effort for the target project, based upon the mean effort and standard deviation values of the clustered and the classified projects, to investigate the suitability of the clusters and/or classes obtained (Papatheocharous and Andreou, 2009a;

Papatheocharous and Andreou, 2009b). Finally, for the case of the association rules that satisfy specific clusters of projects and extracted from FDT in (Andreou and Papatheocharous, 2008b) the mean value of the fuzzy range effort is used for predicting the effort of these projects.

Moreover, the aforementioned techniques are extended to hybrid models for SCE. The techniques combined with ANN, include Input Sensitivity Analysis (ISA) and Genetic Algorithms (GA), from which the former combines ways to illuminate the 'black box' nature of ANN and optimally select the most 'significant' or influential feature subset and the latter evolves the structure of ANN in order to improve predictions (Papatheocharous and Andreou, 2007; Papatheocharous and Andreou, 2010; Papatheocharous and Andreou, 2012b; Papatheocharous and Andreou, 2009c). The techniques combined with RR include Feature Subset Selection (FSS) approaches (Papatheocharous et al., 2010b; Papatheocharous et al., 2010c) and Conformal Predictors (CP) (Papadopoulos et al., 2009), from which the first identifies the most appropriate subset of features for SCE and the second complements the predictions with software cost prediction intervals.

Finally, clustering and classification approaches are included as pre-processing or filtering steps in SCE to produce improved prediction results. Therefore, Fuzzy Logic (FL), Genetic Programming (GP), Conditional Sets (CS), Decision Trees (DT) and the Entropy Fuzzy *k*-modes Algorithm (Papatheocharous and Andreou, 2012a; Papatheocharous et al., 2010a; Andreou et al., 2007; Andreou and Papatheocharous, 2008a; Papatheocharous and Andreou, 2009a) are used and some of them are combined to produce better figures of adjustment (as regards to accuracy performance). The rest of this section summarises the technical background of the aforementioned techniques.

3.2.1 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) is a branch of CI that is closely related to Machine Learning. ANN have the ability to model complex linear and non-linear relationships and

have attractive prospects for solving pattern recognition tasks, classification and categorisation problems. They also present two main advantages, firstly, they avoid the costly, time consuming and error-prone task of trying to directly extract knowledge for a problem domain from an expert and secondly, they handle efficiently noisy, incomplete and distorted data with high degree of uncertainty.

An ANN may be viewed as a directed graph, composed of a number of basic computational elements called neurons or nodes and connections (weights or synapses) between them, forming layers. McCulloch and Pitts (1943) provided the model of a neuron similar to the biological neurons in the human brain called 'Perceptron'. Later on, Rosenblatt (1957) developed a model consisting of three layers and Rumelhart et al. (1986) proposed the multilayer ANN with an effective training algorithm and non-linear but differentiable transfer functions.

3.2.1.1 Artificial Neuron: The Basic Computational Element

The basic computational model of a neuron (illustrated in Figure 3.1) encloses a set of inputs x_i, where $i=1,...,n$ indicating the input signal source. Each x_i input is weighted by the connection strength or weight factor w_{ij} before inserted into the main body of the processing element. The result inserted into the main body may also be optionally affected by a bias term w_o (intentionally not shown in Figure 3.1). The bias term allows the effective control of the result from a layer of neurons and it may constitute a critical component for successful learning. The neuron also encloses a threshold value θ_j above which the neuron produces a signal, a non-linear function φ that acts on the yielded signal net_j and an output O. The output O constitutes input to other subsequent neurons. The transfer function of the basic model is described in eq. (3.1), where j takes values from 1 to n and denotes the source of the input signal for the i^{th} neuron under investigation (Kartalopoulos, 1996).

$$O_i = \varphi\left(\sum_{j=1}^{n} x_i \; w_{ij} + \theta_j\right) \tag{3.1}$$

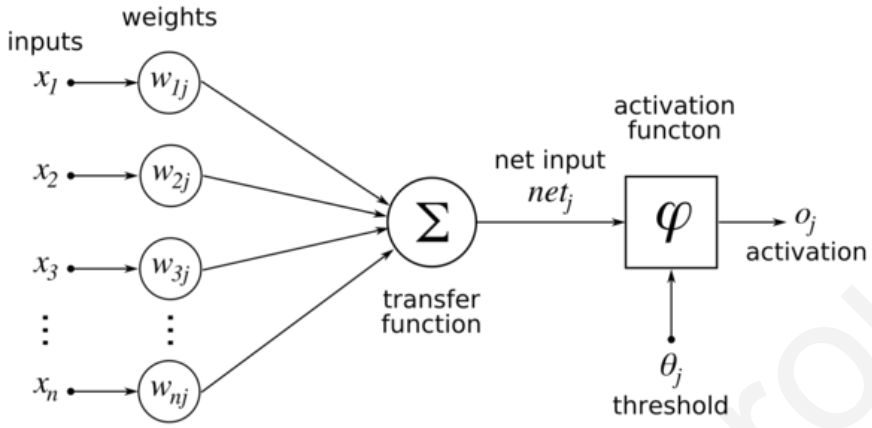


Figure 3.1: Basic computational model of a neuron

The most popular activations functions φ include the hard limiter, the log-sigmoid, the tan-sigmoid and the pure linear transfer function (summarised in Figure 3.2 (a)-(d) respectively).

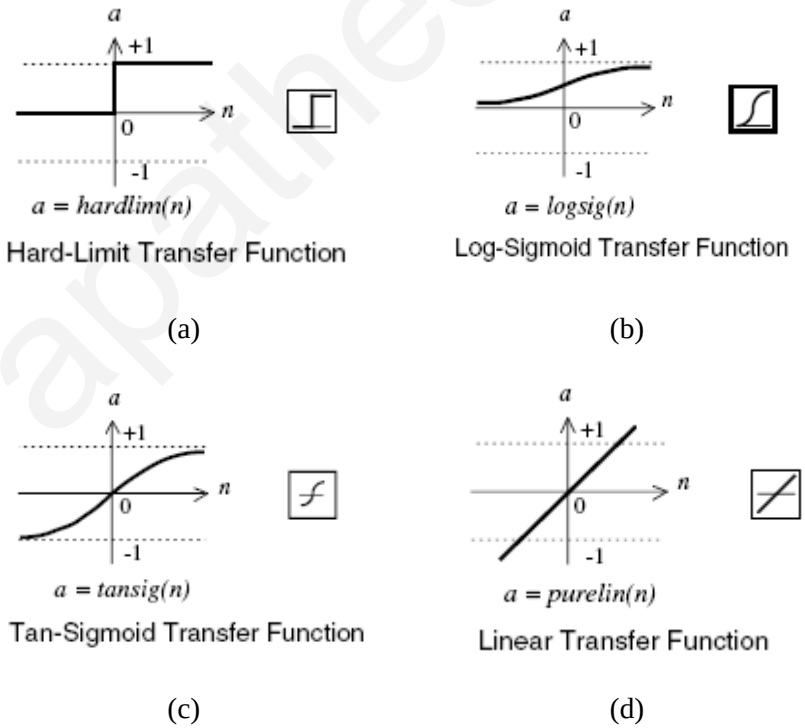


Figure 3.2: Transfer functions (Beale et al., 2011) of (a) hard limiter, (b) log-sigmoid, (c) tan-sigmoid and (d) pure linear

3.2.1.2 Feedforward ANN and Supervised Learning

ANN with neurons organised in multiple layers form the widely known Multi-Layer Perceptrons (MLP) (McCulloch and Pitts, 1943; Rumelhart et al., 1986; Haykin, 1999; Karray and Silva, 2004). Typically, the layer of neurons that the input patterns are applied is the 'Input Layer', the layer of neurons that the output is obtained is the 'Output Layer'; the rest of the layers between the Input and the Output layers are called the 'Hidden Layers'. The feedforward Multi-Layer Perceptron (MLP) is considered the most popular topology and is especially suited for the kind of problem solving domains that comprise very noisy, distorted, or incomplete sample data (Prechelt, 1994).

In the supervised-type of ANN learning, the available data is structured as a set of vectors with input and output sample values which are presented to the ANN during a training process. The goal of an ANN is to characterise the relationship between the inputs and the output(s) for the whole set of the training vectors. During the training process of an ANN, inputs from a training vector propagate throughout the network and are multiplied, as explained previously in section 3.2.1.1, by the appropriate weights. These products are then summed up and if the summation exceeds some specified threshold for a node, then the node 'fires' and its output serves as input to another node in a subsequent layer. This process repeats until the network generates an output value for the corresponding input vector. The calculated output value is then compared to the desired output and an error value is determined for the particular input vector. The goal is to minimise the total error (i.e., the mean error of the set of input vectors) by modifying the weights of the connections. Finally, processing continues, until a total low error value is achieved, or training ceases to converge (Haykin, 1999). This describes the backpropagation learning algorithm which is defined in detail in the following section.

3.2.1.3 The Backpropagation Learning Algorithm

Backpropagation is a supervised learning technique that was first described by Werbos (1974) and gained interest through the work of Rumelhart (1986). The algorithm is based on calculating the derivatives of performance of the neurons. Each subsequent layer uses the weights coming from the previous layers and adjusts them accordingly so that the accuracy performance error of the output is diminished. Backpropagation is an iterated gradient method of optimisation that updates the weights within the implicit bounds of a search weight space. The algorithm is adjusted by enforcing a learning rate (the gain) and a momentum term (the damping factor) in the model.

The backpropagation training technique for ANN is summarised as follows:

(1) Present a training sample to the ANN.

(2) Compare the ANN's output to the desired output from that sample. Calculate the error in each output neuron.

(3) For each neuron, calculate what the output should have been, and a scaling factor, which defines how much lower or higher the output must be adjusted to match the desired output. This is the local error of the neuron.

(4) Adjust the weights of each neuron to lower the local error.

(5) Assign 'blame' for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.

(6) Repeat the steps (1)-(5) on the neurons at the previous level, using each one's 'blame' as its error.

The challenge is to find a good algorithm for updating the weights and thresholds in each iteration (step (4)) to minimise the error. These continuous adjustments of the values of weights - so that the difference of error (delta) between the desired and the actual output is reduced - are also based on the delta learning rule (Kartalopoulos, 1996, p. 70), or the Least Mean Square algorithm, also known as the Widrow-Hoff learning rule (Widrow and Hoff, 1960).

3.2.1.4 Input Sensitivity Analysis Algorithms

The contribution of independent variables within an ANN may be measured with many methods (Belue and Bauer, 1995; Glorfeld, 1996; Satizábal and Pérez-Uribe, 2007). However, most of them are rarely used and especially in the SCE context they have never been used as they are considered to add complexity to the already complicated SCE process. Garson's algorithm (Garson, 1991) and another variant (which serves similar purpose) utilised in this work are described in this section. The algorithms aim to perform Input Sensitivity Analysis (ISA) on ANN by partitioning the ANN connection weights in order to determine the *Relative Importance* (*RI*) and the *Relative Strength* (*RS*) of each input variable in the network.

An example of the application of Garson's Algorithm in a single hidden layer feedforward MLP network with two neurons (shown in Figure 3.3) is described below.

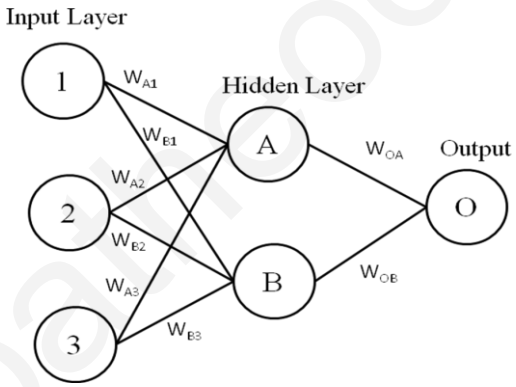


Figure 3.3: Example of network structure

The basic steps in determining the relative weights of the inputs are as follows:

Step 1: Construct a matrix containing the input-to-hidden and the hidden-to-output neuron connection weights. This matrix is shown in Table 3.1.

Table 3.1: Matrix with ANN connection weights

Layers	Input 1	Input 2	Input 3	Output
Hidden A	W_{A1}=-2.61	W_{A2}=0.13	W_{A3}=-0.69	W_{OA}=1.11
Hidden B	W_{B1}=-1.23	W_{B2}=-0.91	W_{B3}=-2.09	W_{OB}=0.39

Step 2: Calculate the contribution of each input neuron to the output (e.g., C_{A1} in eq. (3.2)) via each hidden neuron as the product of the input-hidden connection weights (e.g., W_{A1}) and the hidden-output connection (e.g., W_{OA}) weights as shown in Table 3.2.

$$C_{A1} = W_{A1} \times W_{OA} = -2.610 \times 1.110 = -2.90 \tag{3.2}$$

Table 3.2: Matrix with the calculated contribution of each input neuron

Layers	Input 1	Input 2	Input 3
Hidden A	C_{A1}=-2.90	C_{A2}=0.14	C_{A3}=-0.77
Hidden B	C_{B1}=-0.48	C_{B2}=-0.35	C_{B3}=-0.82

Step 3: Calculate the relative contribution of each input neuron to the outgoing signal for each hidden neuron (e.g., R_{A1} in eq. (3.3)) and the sum of all input neuron contributions (e.g., S_1 in eq. (3.4)) as shown in Table 3.3.

$$R_{A1} = |C_{A1}| / \left(|C_{A1}| + |C_{A2}| + |C_{A3}|\right) = 2.90/(2.90 + 0.14 + 0.77) = 0.76 \tag{3.3}$$

$$S_1 = R_{A1} + R_{B1} = 0.76 + 0.29 = 1.05 \tag{3.4}$$

Table 3.3: Matrix with the calculated relative and sum input neuron contribution

Layers	Input 1	Input 2	Input 3
Hidden A	R_{A1}=0.76	R_{A2}=0.04	R_{A3}=0.20
Hidden B	R_{B1}=0.29	R_{B2}=0.21	R_{B3}=0.50
Sum	S_1=1.05	S_2=0.25	S_3=0.70

Step 4: Calculate the relative importance (e.g., RI_1 in eq. (3.5)) of each input variable as shown in Table 3.4.

$$RI_1 = \left(S_1/(S_1 + S_2 + S_3)\right) \times 100 = \left(1.05/(1.05 + 0.25 + 0.70)\right) \times 100 = 52.5\% \tag{3.5}$$

Table 3.4: Matrix with the calculated relative importance of inputs

Inputs	Input 1	Input 2	Input 3
Relative Importance	RI_1=52.5%	RI_2=12.5%	RI_3=35%

For a more detailed step-by-step example of the calculations the interested reader may refer to (Garson, 1991). In this example, a relatively small ANN was used for demonstration purposes.

The variant of the algorithm, according to Azoff (1994), alternatively one can sum up only the absolute values of the weights fanning from each input attribute to all nodes in the

successive hidden layer and estimate the overall connection strength of each attribute (e.g., *Input Strength* (*IS*), IS_1 in eq. (3.6)) as shown in Table 3.5. This method, in contrast to techniques like the Garson's algorithm (Garson, 1991) which makes use of the entire hidden structure for calculating the effect of a certain input on the output, takes into consideration only the first level of neurons without loss of generality, as demonstrated in (Azoff, 1994). Thus, the relative input strength may also be calculated (e.g., RS_1 in eq. (3.7)).

$$IS_1 = \left(|W_{A1}| + |W_{B1}|\right)/2 = 1.92 \tag{3.6}$$

$$RS_1 = \left(IS_1 \times 100\right)/\left(IS_1 + IS_2 + IS_3\right) = \left(1.92 \times 100\right)/\left(1.92 + 0.52 + 1.39\right) = 50.13\% \tag{3.7}$$

Table 3.5: Matrix with the calculated overall connection strength of inputs

Inputs	Input 1	Input 2	Input 3
Input Strength	IS_1=1.92	IS_2=0.52	IS_3=1.39
Relative Strength	RS_1=50.13%	RS_2=13.58%	RS_3=36.29%

3.2.2 Regression

Regression is not a technique belonging to the CI domain. However, it is used as a predictor in SCE in conjunction with other techniques, as mentioned before, thus forming in some cases hybrid-CI cost models. Also, Regression-based techniques are utilised in this thesis for comparison purposes. Two types of regressions are utilised, namely linear and non-linear regressions.

3.2.2.1 Multiple Linear Regression (MLR)

Multiple Linear Regression (MLR) assumes a functional form or equation relating a dependent variable with more than one independent variables. In case one significant independent variable is used then it is reduced to a Simple Linear Regression (SLR). The goal of SLR is to achieve a model expressed in eqs (3.8), (3.9) given *i* observations:

$$y_i = f(x_i) + \varepsilon_i \tag{3.8}$$

$$f(x_i) = \beta_0 + \beta_1 x_i \tag{3.9}$$

where y is the dependent variable and each vector $\{(x_1, y_1), \ldots, (x_l, y_l)\}$ represents the data samples. Assuming that the errors ε_i are independent and have a zero mean, the aim is to discover the polynomial coefficients β_0 and β_1 representing the constant and the slope of the regression linear function $f(x_i)$ respectively (the latter defined in eq. (3.9)). In MLR accordingly more independent variables appear in eq. (3.9). Previous experience from the use of regression models showed that Regressions can be difficult to use in cases where the dataset is not large, there are no missing values or outliers and the predictor variables are not correlated (Boehm and Sullivan, 1999).

In the Ordinary Least Squares (OLS) Regression form the model specified in eq. (3.9) tries to minimise the overall sum of squared errors. One of the main assumptions of OLS Regression is that the error variation (or residual) is on average constant on the dependent variable range. Therefore, this assumes that there is no difference between the actual and the predicted values of projects and it is referred to as the homoscedasticity assumption (Briand and Wieczorek, 2000).

3.2.2.2 Ridge Regression (RR)

Ridge Regression (RR) is an improvement of the classical OLS Regression technique. RR is used in this thesis as a predictor to compare various Feature Subset Selection (FSS) approaches and to evaluate a hybrid form of predictors based on Conformal Predictors (CP). RR approximates a set of sample data $\{(x_1, z_1), \ldots, (x_n, z_n)\}$, where $x_i \in \Re^n$ is the vector of the independent variables for sample i and $z_n \in \Re$ is the dependent of that sample. The RR procedure recommends finding the w which minimises the function:

$$a\|w\|^2 + \sum_{i=1}^{n}(z_i - w \cdot x_i)^2, \tag{3.10}$$

where *a* is a positive constant, called the *ridge parameter*. Notice that RR includes Least Squares as a special case (by setting *a* = 0). The RR prediction $\hat{z}_t$ for an input vector x_t is then $\hat{z}_t = w \cdot x_t$.

The dual variables formula, derived in Saunders et al. (1999), for the prediction of an input vector x_t is:

$$\hat{z}_t = z(K + aI)^{-1}k, \tag{3.11}$$

where $z = (z_1, ..., z_n)$ is the vector consisting of the dependent variables' outputs of the samples, *K* is the $n \times n$ matrix of dot products of the input vectors $\{x_1, ..., x_n\}$ of those data samples defined as:

$$K_{i,j} = \mathrm{K}(x_i, x_j), \quad i = 1,..., n, \quad j = 1,...., n \tag{3.12}$$

and *k* is the vector of dot products between x_t and the input vectors given as:

$$k_i = \mathrm{K}(x_i, x_t), \quad i = 1,..., n, \tag{3.13}$$

and $\mathrm{K}(x, x')$ is the kernel function, which returns the dot product of the vectors *x* and x' in some feature space. Finally, the Radial Basis Function (RBF) kernel function, which is the typical choice of kernel in Machine Learning literature is selected, defined as:

$$\mathrm{K}(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\gamma^2}\right). \tag{3.14}$$

3.2.2.3 Classification and Regression Tree (CART)

A third form of Regression utilised as a hybrid technique in this thesis mainly for classification purposes, is the Classification and Regression Tree (CART). CART, introduced by Breiman et al. (1984), is a Machine Learning technique that decides the best out of a range

of variables to split the data into as homogeneous as possible groups according to a dependent variable. The most important features of CART include the ability to remain unaffected by outliers and to deal with missing values easily.

Two main procedures executed in CART are tree growing and pruning. The process of tree growing is based on splitting the data based on the aforementioned dependent variable. This splitting procedure is applied iteratively to each child node until each partition of the tree created is 'pure' (i.e., all cases classified in the particular node have the same value for the dependent variable), or sufficiently small (i.e., under a specific threshold of minimum number of cases for nodes), or the maximum number of levels of growth specified for the tree have been reached.

In cases where an overly large tree is produced, the branches are pruned to avoid overfitting. The process of pruning is based on a specified maximum difference in risk (expressed in standard errors). With this process the smallest sub tree is obtained that has an acceptable risk value.

For a set of values $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ representing the data samples, where y is the dependent variable, the CART algorithm calculates the best split s^* so that over all the possible set of splits S eq. (3.15) is minimised (where ΔR in eq. (3.15) is specified in eq. (3.16)).

$$\Delta R(s^*, t) = \arg\max_{s \in S} \Delta R(s, t) \tag{3.15}$$

$$\Delta R(s, t) = R(t) - R(t_L) - R(t_R) \tag{3.16}$$

Eq. (3.16) represents the improvement in the resubstitution estimate for split s of t and the variables t_R and t_L represent the left and right values of split t. The resubstitution estimate $R(t)$ is defined in eq. (3.17) where $y(t)$ is defined in eq. (3.18) and $N(t)$ is the total number of cases in t.

$$R(t) = \frac{1}{N} \sum_{x_n \in t} (y_n - y(t))^2 \tag{3.17}$$

$$y(t) = \frac{1}{N(t)} \sum_{x_n \in t} y_n \tag{3.18}$$

Finally, the same process is used to proceed growing the tree until a node is reached that does not significantly decrease the resubstitution estimate, and thus this node will represent the final (terminal) node of the tree (Pendharkar et al., 2005).

3.2.3 Genetic Algorithms (GA)

A Genetic Algorithm (GA) is an optimisation technique based on the principles of evolution and inheritance. GA is used in this thesis for optimising the topologies of ANN, the selection of the optimal features assessed by RR and the choice of ranges for the Conditional Sets (refer to section 3.2.5, pg. 85).

The algorithm aims to maintain the best selection from a population of solutions (individuals) based on the fitness of individuals and a set of genetic operators. The basic structure of an evolutionary algorithm is described in Figure 3.4 (Michalewicz, 1994).

```
i=0;
initialise (population(Pi));
evaluate (population(Pi));
while (not termination condition occurs)
    t=t+1;
    select P(i) from P(i-1);
    alter (Pi);
    evaluate (Pi);
endwhile
```

Figure 3.4: Genetic Algorithm pseudo code (Michalewicz, 1994)

The evolutionary computing process described in Figure 3.4 is domain-independent and it may play an important role in the development of optimal and self-improving intelligent

techniques according to Karray and Silva (2004). The evolutionary-based computational approach of the GA aims to find approximated solutions in complex optimisation and search problems (Holland, 1992). In order to achieve this, pruning a population of individuals based on the Darwinian principle of reproduction and 'survival of the fittest' is performed (Koza, 1992).

The fitness of each individual is based on the quality of the simulated individual in the environment of the problem investigated. The process is characterised by the fact that the solution is achieved by means of a cycle of generations of candidate solutions that are pruned by using a set of biologically inspired operators. According to evolutionary theories, only the most suited solutions in a population are likely to survive and generate offspring, and transmit their biological heredity to the new generations. Thus, GA are much superior to conventional search and optimisation techniques in high-dimensional problem spaces due to their inherent parallelism and directed stochastic search implemented through recombination operators. The GA operators are selection, crossover and mutation. The stochastic search procedure of GA usually does not locate the exact location of the optima identified by some other gradient-based optimisation techniques, but effectively avoid the entrapment to local minima (Karray and Silva, 2004).

3.2.4 Genetic Programming (GP)

Genetic Programming (GP) comprises an evolutionary technique similar to GA where the potential solutions of the population are represented as parse trees of computer programs and the fitness value is calculated by executing these programs. GP may be conceived as a subset of Genetic Algorithms, with the main difference being the solution's representation. GP is used to examine in a stochastic search approach extensive Regression and other mathematical models for SCE.

The following steps are usually followed to implement a GP algorithm:

- **Step 1:** Initialise the population of solutions comprising functions and terminals that are considered the alphabet of the computer programs.
- **Step 2:** Execute each solution and obtain a fitness value according to how well each solution approximates the problem.
- **Step 3:** Generate new solutions of offsprings by applying:
    - Replication of the computer program that has the best fitness value.
    - Mutation of the computer program using the mutation operator. Two types of mutation may be applied; replacing randomly a function or replacing a terminal, with the latter offering the ability to replace an entire sub tree of the parse tree with another sub tree.
    - Crossover of two probabilistically chosen parents based on their fitness values to create new computer programs. After a crossover point is randomly determined for both parents, a sub tree from the first one is substituted with a sub tree from the second.
- **Step 4:** Obtain the computer program with the highest fitness value as the result of the GP. Repeat Steps 2-4 if a termination criterion does not occur. Otherwise stop.

3.2.5 Conditional Sets (CS)

The Conditional Set (CS) theory is a very simple idea that refers to a set of boundary conditions. These sets of conditions may be used in a combination with an algorithm such as GA to optimise their form (as described in Adamopoulos et al. (1998)). In this work CS are used to specify appropriate set of conditions enrolling as minimal as possible ranges for SCE. The main concept is to represent conditional knowledge as a collection of classical logic statements and conditions involving bounds (Packard, 1990).

Consider a dataset of N samples x_i, $i=1, ..., N$ and a condition C_i of the form:

$$C_i : (lb_i < x_i < ub_i),\ i = 1,...,N \tag{3.19}$$

where lb_i and ub_i are the lower and upper bounds of C_i respectively.

A conditional set S of length L entails L conditions that are coupled with the logical operator ^ (Boolean AND) as follows:

$$S_{AND} = C_1 \wedge C_2 \wedge ... \wedge C_L \tag{3.20}$$

and a numerical example is the following:

$$S_{AND} = (340 < \mathrm{x}_1 < 470) \wedge ..... \wedge (580 < \mathrm{x}_N < 610). \tag{3.21}$$

The theory of CS is based on a set of such conditions with respect to boundaries representing an upper and a lower boundary value. It may be used to disengage the model from the need of an expert, past experience or a technique to define the conditions.

3.2.6 Fuzzy Logic (FL)

Fuzzy Logic (FL) is particularly useful for representing human knowledge and making inferences in reasoning with that knowledge. It is generally conceived as qualitative, descriptive and subjective, while it may contain some overlapping degree of neighbouring quantity, i.e., some degree of a particular quantity (Karray and Silva, 2004). FL is used in this thesis as a pre-processing step for project data with uncertain and vague values.

FL allows a way of processing data through partial set membership rather than crisp set membership or non-membership (Zadeh, 1965). It is a problem solving methodology that provides a simple way to arrive at a definite conclusion based on vague, ambiguous, imprecise, noisy or missing information.

In FL, a membership function is used to express the distribution of truth of a variable in the context of a given (fuzzy) set. An example of a membership function of a particular quantity is shown in Figure 3.5. A fuzzy descriptor may thus be represented by a membership function. The membership function provides a membership grade between 0 and 1 for each possible value of the fuzzy descriptor it represents.

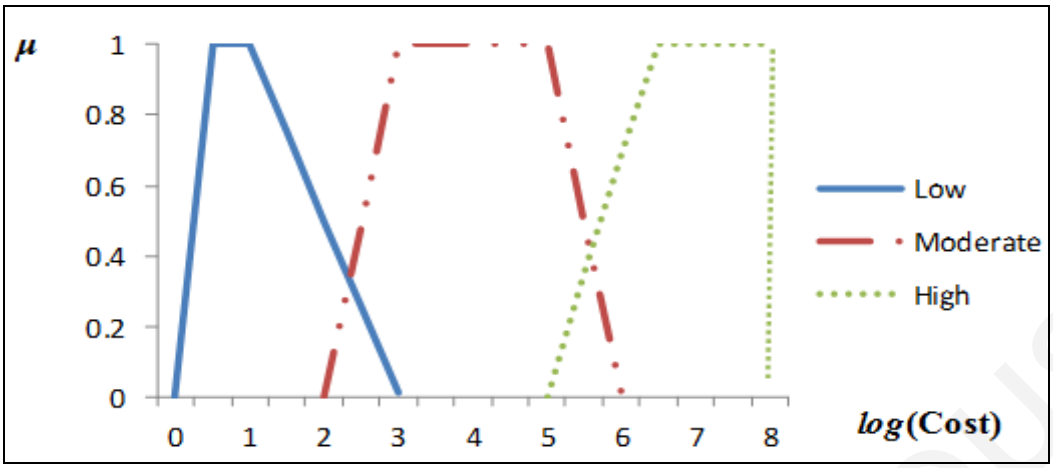


Figure 3.5: Membership function of cost example for Low, Moderate and High quantity.

A fuzzy set *A* is represented by a membership function defined in eq. (3.22) in which each element of *A*, as denoted by a point *x* on the real line $\Re$, is mapped to a value μ which may lie anywhere in the real interval of 0 to 1. This value represents the grade of membership of *x* in *A*. If the membership grade is greater than 0 but less than 1, the membership is not crisp, i.e., it is fuzzy and thus the element has some possibility of being within the set and some complementary possibility of being outside the set (Karray and Silva, 2004).

$$F_z[x \in A] = \mu_A(x) : \Re \longrightarrow [0,1] \tag{3.22}$$

In most fuzzy problems rules are generated through past experience and based on the defuzzification process an output is deduced. Particularly, defuzzification is the process where the membership functions are sampled to find the grade of membership, then the membership is used in the FL equations and an outcome region is specified (Kartalopoulos, 1996).

3.2.7 Fuzzy Implication Systems (FIS)

In Fuzzy Implication Systems (FIS) knowledge is based on '*if-then*' linguistic rules expressed through fuzzy terms. The knowledge base is formed by aggregating several such fuzzy rules. FIS are used in this thesis as predictors.

Considering fuzzy set *A* defined in universe *X* and fuzzy set *B* defined in another universe *Y*, the fuzzy implication "*If A then B*" is denoted by $A \rightarrow B$. While *A* represents the situation, condition, or the antecedent, *B* represents another fuzzy situation and is the action or the consequence in this fuzzy rule (Karray and Silva, 2004).

Karray and Silva (2004) mention the following two (mostly used) definitions of fuzzy implications (even though several others exist):

Method 1 (Mamdani implication):

$$\mu_{A\rightarrow B}(x,y) = \min\left[\mu_A(x), \mu_B(y)\right] \quad \forall x \in X, \forall y \in Y \tag{3.23}$$

Method 2 (Lukasiewics implication):

$$\mu_{A\rightarrow B}(x,y) = \min\left[1, \left\{1 - \mu_A(x) + \mu_B(y)\right\}\right] \quad \forall x \in X, \forall y \in Y \tag{3.24}$$

Thus, these two methods are used to obtain the membership function of the particular fuzzy relation of an *if-then* rule (implication). The first method defines a symmetric expression to *A* and *B* while the second method gives an upper-bounded membership function to 1.

3.2.8 Decision Trees (DT)

Decision Trees (DT) comprise non-parametric approaches for decision analysis, classification, conditional probabilities calculation, data mining and predictive modelling. DT are used in this thesis in conjunction with FL, thus forming Fuzzy Decision Trees (FDT) to obtain fuzzy association rules, which may be utilised either as classifiers and then as predictors, or for constructing FIS and applying an implication method, as described in the previous section.

DT represent structures with flow-chart-like nodes. The top-most node is called the 'root' and the terminal nodes are called 'leaves'. The internal nodes represent the attributes and each

branch corresponds to an outcome of the test performed on an attribute. DT provide a hierarchical representation of the feature space and they are considered popular choices for learning, reasoning and increasing understandability in specific contexts (Maimon and Rokach, 2005; Han et al., 2006). The feature space is allocated to classes according to the result of a decision made at the sequence of nodes, where the branches of the tree diverge.

The popular DT extensively used in data mining mainly comprise the following:

(a) Classification trees to predict an outcome in the class to which the data belongs, and,

(b) Regression trees to predict an outcome as a real number.

The term Classification and Regression Tree (CART) refers to both the above types of processes, introduced by Breiman et al. (1984). The CART algorithm has been previously described in section 3.2.2.3. Other popular DT algorithms include the ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) and CHAID (Kass, 1990). These algorithms share some similarities but also present several differences, such as the splitting conditions.

The CHAID algorithm is selected to be briefly described below since it was utilised in this thesis. The Chi-Squared Automatic Interaction Detection (CHAID) (Kass, 1990) splits the data in an iterative manner using multi-way splits on nominal and ordinal data. Each split is executed if the level of significance in a chi-square test of independence between the target value and the branch exceeds a threshold. CHAID uses a significance test to determine the number of branches and the Bonferroni adjustment for nominal values to mitigate the bias towards inputs with many values.

The following steps are executed:

(1) For each value x_i of the data sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where y is the dependent variable, assign a different branch.

(2) For each pair of branches form a two-way table and count the number of cases in each branch according to the target value (dependent variable).

(3) Locate the pair of branches which has the lowest chi-square independence measure.

(4) If the significance level is below a threshold, merge the branches and repeat steps (2)-(4). Otherwise, consider re-splitting the branches that contain more than three input values and continue with step (5).

(5) If a binary split is located that exceeds a threshold significance level, split the branch in two and go back to step (4) and merge branches. When no more merges or re-splits are significant continue with step (6).

(6) The procedure stops and the last split on the input is chosen as the candidate split for the input.

The CHAID algorithm does not select the most significant split examined. Biggs et al. (1991) introduced the algorithm "exhaustive CHAID" which selects the most significant split and produces more branches than the original CHAID. The exhaustive CHAID has also been used in the experiments conducted in this work.

3.3 Qualitative Models Technical Background

A major weakness of quantitative software cost estimation models is observed when attempting to take into consideration factors that are purely qualitative, immeasurable and subjective, such as development group dynamics, cohesiveness and experience. Consequently, the interactions among such factors and their effect on development effort are hard to understand and model. In this section, two forms of Qualitative models are presented, Fuzzy Cognitive Maps (FCM) and Influence Diagrams (ID), both being used in this diatribe.

3.3.1 Fuzzy Cognitive Maps (FCM)

A Fuzzy Cognitive Map (FCM) model, initially proposed by Axelrod (1976), provides a graphical representation of the knowledge used to describe a given real-world problem in the form of an acyclic graph comprising of cognitive states (concepts) (Kosko, 1995). Each

concept node is characterised by a numeric state, which denotes the qualitative measure of its presence in the conceptual domain. For example, a high numerical value indicates that the concept is strongly present in the analysis, while a negative or zero value indicates that the concept is not currently active or relevant to the conceptual domain.

The FCM works in discrete steps (Kosko, 1986). When a strong positive correlation exists between the current state of a concept and that of another concept in a preceding period, we say that the former positively influences the latter, indicated by a positively weighted arrow directed from the causing to the influenced concept. By contrast, when a strong negative correlation exists, it reveals the existence of a negative causal relationship indicated by an arrow charged with a negative weight. Two conceptual nodes without a direct link are, obviously, independent.

The effect of a causal relationship between two nodes is represented as follows:

- Positive (+) causality, in cases in which a node promotes, enhances, or is a benefit to another node, etc. An increase in the cause variable will result to an increase in the effect variable; a decrease in the cause concept will result to a decrease in the effect concept.
- Negative (-) causality, in cases in which a node retards, prevents, or is harmful to another node, etc. In such cases an increase in the cause variable will end up with a decrease of the effect variable and vice-versa.
- No effect (0), when a node has no effect on, or is not affected by another node.

The activation level of each node of the FCM and the weighted arrows are set to specific values. These values for example may be based on the beliefs provided by a group of experts. Then, the FCM is free to progress and react until it (Taber, 1987):

(i) Reaches equilibrium at a fixed point, with the activation levels, being decimals in the interval [-1, 1], stabilising at fixed numerical values.

(ii) Exhibits a limit cycle behaviour, with the activation levels falling in a loop of numerical values under a specific time-period.

(iii) Exhibit a chaotic behaviour, with the activation levels reaching a variety of numerical values in a non-deterministic, random way.

A simple FCM structure is described in (Tsadiras and Margaritis, 1996) to build a Certainty Neuron Fuzzy Cognitive Map (CNFCM). The activation level for each concept C_i is defined by a function coming from the area of Expert Systems (Andreou et al., 2004) used to update each concept after receiving new evidence concerning previous beliefs based on the present certainty factor (eq. (3.25)), where A_i is the activation level for concept C_i at some time *(t+1)* or *(t)*. Eq. (3.26) defines the sum of the weighted influences that concept C_i receives at time step *t* from all other concepts, while d_i is a decay factor.

$$A_i^{t+1} = f\left(S_i^t A_i^t\right) - d_i A_i^t \tag{3.25}$$

$$\text{where} \quad S_i^t = \sum_{\substack{j=1 \\ j \neq i}}^{n} A_j^t W_{ij} \tag{3.26}$$

Finally, equation (3.27) describes the function used for the aggregation of certainty factors. The meaning of this function is that the external influence can affect the activation of a concept just to a certain degree.

$$f_m\left(A_i^t, S_i^t\right) = \begin{cases} A_i^t + S_i^t(1 - A_i^t) = A_i^t + S_i^t - A_i^t S_i^t, \; if \;\; A_i^t \geq 0, S_i^t \geq 0 \\ A_i^t + S_i^t(1 + A_i^t) = A_i^t + S_i^t + A_i^t S_i^t, \; if \;\; A_i^t < 0, S_i^t < 0, \left|A_i^t\right| \leq 1, \left|S_i^t\right| \leq 1 \\ \left(A_i^t + S_i^t\right) \Big/ (1 - \min(A_i^t, S_i^t)), otherwise \end{cases} \tag{3.27}$$

3.3.2 Influence Diagrams (ID)

Influence Diagrams (ID) represent a way to model a decision problem and are very useful for probabilistic and decision analysis problems (Howard and Matheson, 1984; Shachter, 1988). The models are represented by interconnected nodes, directed arcs and contain no

cycles. They rely on probabilistic dependence, reveal the information flow and contain uncertain variables and decisions. ID contain four types of nodes, described as follows:

- Decision Node: represents the variables that are under the control of the decision maker and models the available decision alternatives.
- Chance (or Uncertainty) Node: represents the random variables and uncertain quantities that are relevant to the decision problem. This means that each chance node is associated with a random variable; for each of these nodes there is an underlying probability distribution for all the random variables.
    - Deterministic Node: is a special type of chance node whose outcome is defined in a deterministic manner, i.e., represents either constant values or values that are algebraically determined from the states of their parent nodes.
- Value Node: represents a measure of desirability of the outcomes from the decision process. The value node is quantified by the utility of each of the possible combinations of outcomes from the parent nodes defined in the ID.

ID contain two types of arcs, described as follows:

- Influence (or Conditional) Arc: Indicates a node's direct influence to another node. The influence can be directed to chance and value nodes.
- Informational Arc: Indicates one node's influence into decision nodes.

The arcs indicate the probabilistic dependence of uncertain quantities (questions in the decision problem).

The algorithm of Cooper (1988) is used for obtaining the outcome of a decision by solving an ID initially transforming it into a Bayesian Belief Network (BBN) and then estimating the expected utilities for each of the decision alternatives through a repetitive process of inference.

For example, consider the ID of Figure 3.6 representing the decision to bring an umbrella to work or not (Shachter and Peot, 1992). The goal is maximise Satisfaction depending on the Weather and whether we bring an umbrella. The latter represents the decision and the key uncertainty here is the Weather, which we will not observe until we

reach to a decision. However, prior knowledge exists regarding Forecast before we reach to the decision. Forecast depends on the Weather since otherwise it would not offer any valuable information to the decision problem modelled.

The data for the ID is stored within the nodes and each node can take a set of possible values: (i) outcomes or alternatives for random variables, for which there is a conditional probability distribution giving chances to different outcomes dependent on the outcomes of the variable parents, and, (ii) value variables, for which there is a function $v(A)$ giving the expected value as a function of its parents and denoted by the set A.

Decision variables do not have a distribution but when the optimal choices are determined, the decision is replaced by a random variable, called the 'optimal policy'. Regarding this optimal policy the choices can be indicated by a probability distribution or a deterministic function. Finally, for evidence nodes a likelihood function is sufficient, since the outcome is already known.

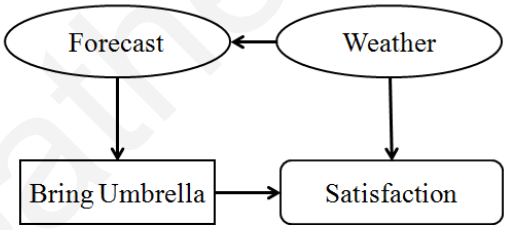


Figure 3.6: A simple Influence Diagram (ID) example

The main advantage of ID is that they provide clear insight into a decision problem including the qualitative structure of the problem and distinguish the negative or positive dependence of each node on each other and on the decision (Shachter, 1988). Therefore, they are very useful in cases of investigating the cause-and-effect relationships and in the distinction between informational and probabilistic relations within a particular context (Mateou et al., 2005).

In the next chapter, the proposed research as conducted is described.

Chapter 4

Proposed Software Cost Modelling and Estimation Methodologies

This chapter provides details on the research contribution of this dissertation and emphasises on the hybrid Computational Intelligent (CI) Software Cost Estimation (SCE) models developed. The models are distinguished in Quantitative and Qualitative models.

4.1 Overview of Software Cost Estimation Models and Datasets

The models proposed and developed in the context of this research thesis are illustrated in Figure 4.1. The figure shows the basic steps followed which include dataset pre-processing, software cost modelling and estimation which is at the final stage evaluated (validated). The pre-processing activities involve data cleaning for ensuring that the models proposed will utilise qualitative data. The pre-processing activities carried out are described thoroughly in this section. In cases where it was considered necessary, pre-processing involved the transformation of nominal and multi-valued nominal attributes to binary. In some other cases, normalisation was applied on numerical and binary (dummy) attributes. The normalisation aims to achieve rescaling in a way to distribute the data evenly within a particular range. In addition, for models based on Fuzzy Logic (FL), fuzzification was applied on real-valued (numerical) data to limit the underlying uncertainty of the datasets. Finally, several techniques to cluster or classify the data considering the project's features were applied as a filtering step. This chapter includes the detailed description of the necessary pre-processing activities to

carry out exploratory experiments and investigations to effectively model and accurately estimate software cost. The SCE component of Figure 4.1 includes two types of approaches: the Quantitative and the Qualitative. The Quantitative SCE models, on the one hand, examine the use of various techniques to model and forecast software development effort. The techniques utilised comprise Artificial Neural Networks (ANN), Ridge Regression (RR), Genetic Programming (GP) and hybrid systems. The hybrid systems include ANN combined with Genetic Algorithms (GA) or Input Sensitivity Analysis (ISA), Fuzzy Decision Trees (FDT) combined with Fuzzy Inference Systems (FIS), Conditional Sets (CS) combined with GA and RR combined with Conformal Predictors (CP). On the other hand, the Qualitative models identify the vital cost factors in software engineering environments and utilise techniques of Fuzzy Cognitive Maps (FCM) and Influence Diagrams (ID), by representing the factors that affect cost as nodes with certain interrelationships, to investigate the problem of SCE in real-case scenarios. Finally, each design of experiments used in this thesis aims to answer specific questions which helped to structure the contents of this chapter.

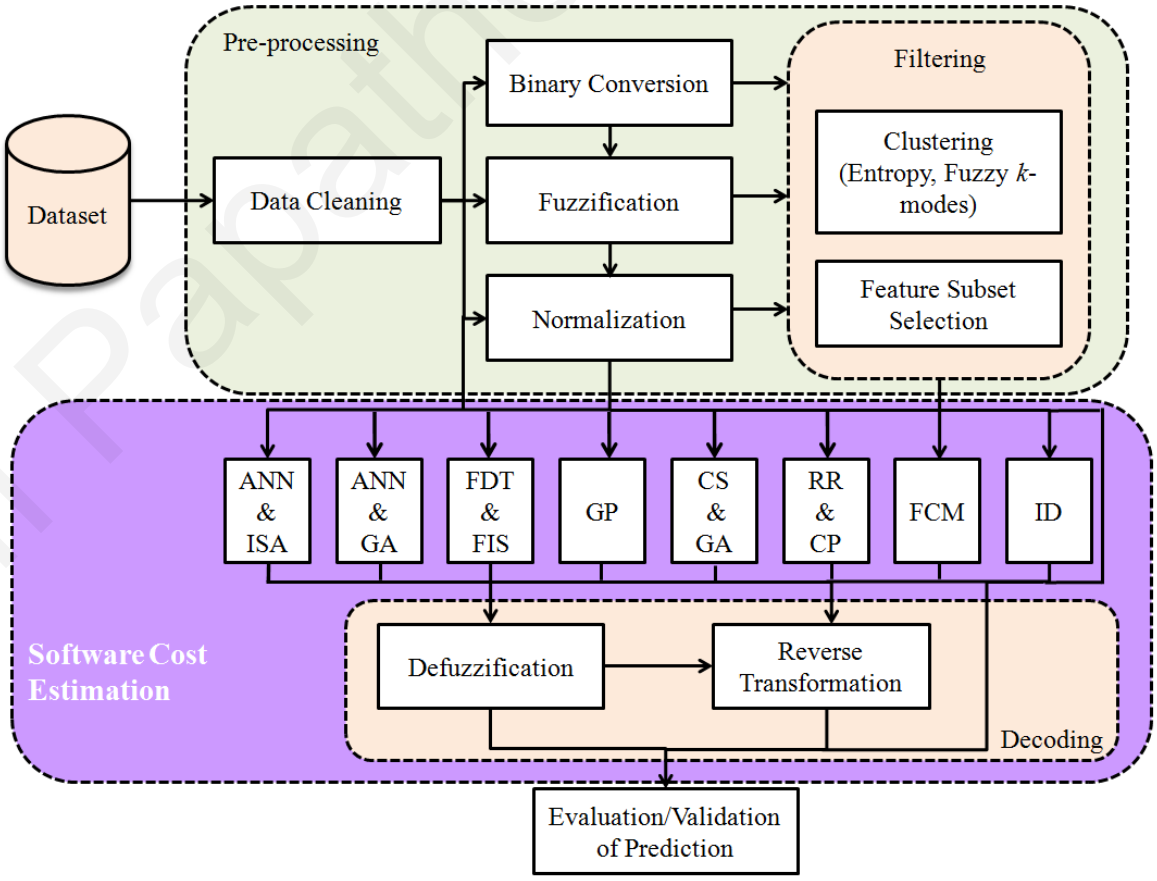


Figure 4.1: Software Cost Modelling and Estimation Research Components

4.1.1 The Datasets Utilised

In this section the datasets used in the experiments conducted are described. They incorporate a rich volume of empirical, past, completed projects from the software industry.

4.1.1.1 The COCOMO Dataset

The COCOMO dataset (Boehm, 1981, pg. 496-497) contains information of 63 software projects from different applications developed at TRW Inc. Each project is described by the following 17 cost attributes: reliability, database size, complexity, required reusability, documentation, execution time constraint, main storage constraint, platform volatility, analyst capability, programmer capability, applications experience, platform experience, language & tool experience, personnel continuity, use of software tools, multi-site development and required schedule. Also, for the projects the Source Lines of Code (SLOC) are measured.

4.1.1.2 The Albrecht and Gaffney Dataset

The Albrecht and Gaffney dataset (Albrecht and Gaffney, 1983) contains information of 24 projects developed by the IBM DP service organisation. The datasets' characteristics correspond to the actual project effort, the SLOC, the number of inputs, the number of outputs, the number of master files, the number of inquiries and the Function Points' (FP) count.

4.1.1.3 The Kemerer Dataset

The Kemerer dataset (Kemerer, 1987) contains 15 software project records gathered by a single organisation in the USA, which constitute business applications written mainly in COBOL. The attributes of the dataset include the actual project's effort measured in man-

months, the project duration, the thousands of Source Lines of Code (KSLOC), the unadjusted and adjusted FP's count.

4.1.1.4 The Desharnais Dataset

The Desharnais dataset (Desharnais, 1989) includes observations for 81 systems developed by a Canadian software development house at the end of 1980. The basic characteristics of the dataset account for the following: development effort measured in hours, team's experience, project manager's experience, number of transactions processed, number of entities, unadjusted and adjusted FP, development environment and year of completion.

4.1.1.5 The ISBSG R9 Dataset

The ISBSG R9 dataset (ISBSG, Repository Data Release 9) is obtained from an international non-profit organisation, the International Software Benchmarking Standards Group (ISBSG), and contains an analysis of characteristics and costs for a group of software projects. The ISBSG organisation establishes and maintains a database of historic IT industry project data to assist and improve IT project management globally. The projects come from a broad cross section of industry and range in size, effort, platform, language and development technique data. The release R9 of the dataset contains 92 variables for 3,024 projects and contains multi-organisational, multi-application domain and multi-environment data. Some of the attributes collected and stored in the ISBSG repository were mandatory and some were optional.

4.1.1.6 The ISBSG R10 Dataset

The ISBSG R10 dataset (ISBSG, Repository Data Release 10) is a newer version of ISBSG R9 dataset that was released on January 2007 with contributions of data from many industries all over the world. The dataset includes 4,106 projects which have different characteristics, are developed following different methodologies and techniques. The data originate from 25 countries with 60% of the projects being less than 10 years old. The database consists of 106 number of attributes recorded, but only a small portion of records contains values for all of these attributes (the same applies for ISBSG R9 dataset). The list of attributes includes, among others, the application domain, programming languages used, language types, development techniques, resource levels, functional size of the software produced, etc.

4.1.2 Pre-processing Activities

The generic pre-processing steps the datasets went through are summarised in Table 4.1. However, the appropriate and required combinations of these pre-processing steps were applied in each particular methodology according to the research question examined, the technique used and the requirements of the methodology.

Table 4.1: Summary of the pre-processing steps applied for the datasets

Code	Description
(a)	Projects with null values in numerical (real value) data were excluded.
(b)	Projects with null values in categorical data were excluded.
(c)	Projects with data quality and Unadjusted Function Point (UFP) rating from the ISBSG reviewers as 'C' or 'D' Category were excluded and projects with 'A' or 'B' Category quality were maintained.
(d)	Projects that used a technique to measure the size of the project that was different than the unadjusted functional size measurement method IFPUG (ISO/IEC 20926, 2003) were excluded.

Code	Description
(e)	Projects reporting partial work effort for a limited set of phases in the project life-cycle even if the effort was adjusted to represent the full-cycle work effort were excluded.
(f)	Attributes containing more than 40% of projects with null values for that specific attribute were excluded.
(g)	Attributes that were considered not to have a direct or apparent impact on software cost (irrelevant) were excluded.
(h)	Attributes of numerical and binary form were normalised in the range *[-1, 1]*. The normalisation was based on eq. (4.1), where *y'* is the normalised value of *x*, y_{max} and y_{min} are the bounds of the ranges we are normalising to (i.e., *[-1, 1]*) and x_{max} and x_{min} are the maximum and minimum elements of the original values. It is assumed that *x*'s values are real and that the elements in each row of *x* are not equal. $$y' = \frac{(y_{\max} - y_{\min}) * (x - x_{\min})}{x_{\max} - x_{\min}} + y_{\min} \quad (4.1)$$
(i)	Attributes with categorical (but not ordinal) attributes were excluded.
(j)	Attributes that were not considered relevant to software size, complexity, or productivity, or constituted derived attributes from transformations of other attributes, were excluded.
(k)	Project attributes that could not be measured before the project completion and therefore would not be practical for the cost estimation model under creation were excluded.
(l)	New binary columns for different values of categorical attributes were created. In these columns the value of 1 was reported for each project that belonged to the new categories created, and 0 otherwise.
(m)	Attributes of numerical form were normalised in the range *[0, 1]* according to eq. (4.1), i.e., where the bounds of the ranges we are normalising to y_{max} and y_{min} are 1 and 0 respectively.
(n)	New categories to describe different but logically similar sample values of categorical columns were created, thus merging and homogenising similar pieces of information into new categories (e.g., 'Oracle v7', 'Oracle 8.0' into 'Oracle').

Code	Description
(o)	Numerical values were fuzzified under a fuzzy linguistic representation proposed in (Braz and Vergilio, 2004). Each numerical cost attribute was fuzzified based on the trapezoidal membership function by calculating variables m_i, n_i, a_i and b_i (see Figure 4.2) where $1 \leq i \leq n$, and n is the number of linguistic terms in the classification table being analysed and using the following eqs: $$m_i = \min \textit{ value of linguistic term } T_i \textit{ in classification table} \quad (4.2)$$ $$n_i = \frac{m_i + m_{i+1}}{2} \quad (4.3)$$ $$a_i = n_{i-1} \quad (4.4)$$ $$b_i = m_{i+1} \quad (4.5)$$ Figure 4.2 illustrates an example of a three linguistic values encoding (or fuzzification) scheme.
(p)	New categorical sample bins were created of similar size to group data into classes.
(q)	Attributes of numerical form were transformed using the natural logarithm (i.e., the logarithm to the base-e).

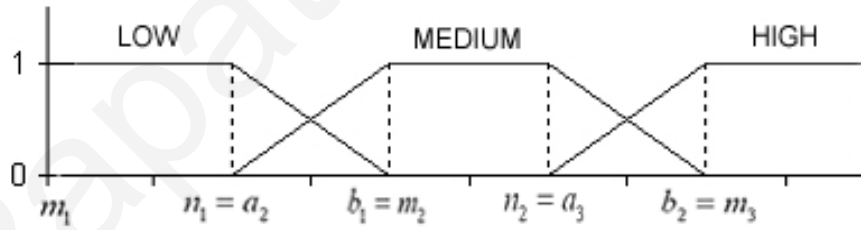


Figure 4.2: Membership functions of three fuzzy sets for the linguistic values *LOW, MEDIUM and HIGH*

4.2 Quantitative Software Cost Estimation Models

Typically the process of Software Cost Estimation (SCE) requires the identification and utilisation of a set of attributes, factors or cost drivers. These attributes may be of numerical, nominal or even non-quantitative nature. The latter, being qualitative and thus quite subjective, are difficult to measure with certainty. Examples of quantitative attributes include

the project manager's experience (in years/projects), the software size, the development technique followed, etc. factors that both describe the project and influence its cost. This section describes the Quantitative software cost models developed for estimating a software project's effort and examining the complex interrelations between cost factors and effort. Several techniques are employed from the domain of CI in order to build software cost models, improve forecasting ability and analyse the form of the input variables used, the latter being rarely adequately conducted in SCE models (Park and Baek, 2008).

4.2.1 Size-Based Software Cost Estimations (SB-SCE)

Many software engineers have focused on measuring internal product attributes and building software cost models based on such attributes; an example of these is software size (Fenton and Pfleeger, 1997; Schach, 2004). Software size is commonly recognised as one of the most important factors affecting the amount of effort required to complete a project (Fenton and Pfleeger, 1997; Boehm et al., 2000b; Sommerville, 2006). Even though size measurements reported do not usually account for all the amount of effort required to create various artefacts of the product, we may assume that they represent estimates of the size for all forms of products produced for some required software, including requirement specifications, design, source code and associated documentation. In addition, size measurements are assumed to represent size estimates which in reality are subjective and thus suffer from some degree of uncertainty.

Therefore, in this preliminary research work the main goal is to investigate and confirm the relation of software effort with size estimates, even though according to Jørgensen (2007) in the domain of SCE the stability of this relationship is controversial, since the technology, the types of software produced and the development methodologies followed change regularly. The modelling is carried out through ANN and the performance of the Size-Based SCE (SB-SCE) model is investigated through a series of experiments given that it depends on the properties of the relationships it attempts to model. Once a robust relationship between

size of the deliverables of a software project and effort is established by means of a model, then it can allow the accurate evaluation of effort estimations (Kemerer, 1993).

Moreover, since the most popular size-based metrics used in practice are the number of Lines Of Code (LOC) and the number of Function Points (FP) of the software (Fenton, 2000), these are used in the SB-SCE methodologies and series of experiments executed which are described subsequently. Software size metrics (either in LOC, FP, or both) represent the only common metrics measured in all the datasets, previously described in section 4.1.1. Particularly, the size-based metrics used in the experiments were obtained from the datasets COCOMO, Kemerer, Albrecht and Gaffney, Desharnais and ISBSG R9. The latter includes projects from many organisations while the rest datasets from specific companies. The details of the pre-processing applied for the attributes summarised in Table 4.2 can be found in Table 4.1 (a). Thus, measurements of Source Lines of Code (SLOC) and/or Function Points (FP) found in each of the datasets were utilised for estimating development effort in all SB-SCE experiments conducted.

Table 4.2: Size-based software attributes description for each dataset

Dataset	Size-Metric Name	Description
COCOMO	SLOC	Source Lines of Code
Kemerer	AFP	Adjusted Function Points
	KSLOC	Thousands of Source Lines of Code
Albrecht and Gaffney	FP	Function Points
	SLOC	Source Lines of Code
Desharnais	AFP	Points ajust. (Adjusted Function Points)
ISBSG R9-1	AFP	Adjusted Function Points

4.2.1.1 Single Hidden Layer MLP ANN for SB-SCE

Initially, the employment of ANN for SCE utilising size-related software attributes (such as LOC and FP), aims to investigate the levels of predictive performance of simple, single hidden layer Multi-Layer Perceptron (MLP) architectures. More specifically, the goal is to inspect whether sufficient estimates of software development effort using only size-related metrics on different datasets of empirical cost samples may be achieved. Since the estimates of software size suffer from subjectivity, the ANN technique was selected because it is

especially suited for adapting estimations to the type of problem-solving domains that comprise very noisy and complex data samples. In particular, the nature of software size metrics presents tremendous restrictions, such as, the absence of widely accepted definitions, counting rules, as well as subjectivity and complexity in calculating Lines Of Code (LOC) and/or Function Points (FP). These restrictions result to high size estimates' variations and uncertainty.

The filtered datasets initially used consisted of 63 projects in the COCOMO, 15 projects in the Kemerer, 24 projects in the Albrecht and Gaffney and 77 projects in the Desharnais case. The data was randomly separated into three subsets, i.e., the training set consisting of 60% of the original samples, the validation set 20% and the testing set 20%, the latter being called the 'holdout sample' as it is used to examine the generalisation ability of a trained model. The holdout sample was 'unseen' to the training process. This process of validation is known as 'holdout' (Weiss and Kulikowski, 1991).

The ANN models were developed in Matlab 2010b and comprised a single hidden layer MLP architecture, as shown in Figure 4.3. The number of nodes in the hidden layer was empirically defined (from 2 to 10 hidden nodes), the input layer utilised the *netsum* function (defined in eq. (4.6) and in which the calculation of output h_1^1 is based on the summation of the product of x_i and weights w_{i1}^1, where $i=1,\ldots,n$ for each input weight and bias θ_1^1 variables), the hidden layer utilised the *hyperbolic tangent sigmoid transfer* (*tansig)* function (eq. (4.7)) and finally, the output layer utilised the *pure linear* (*purelin*) function.

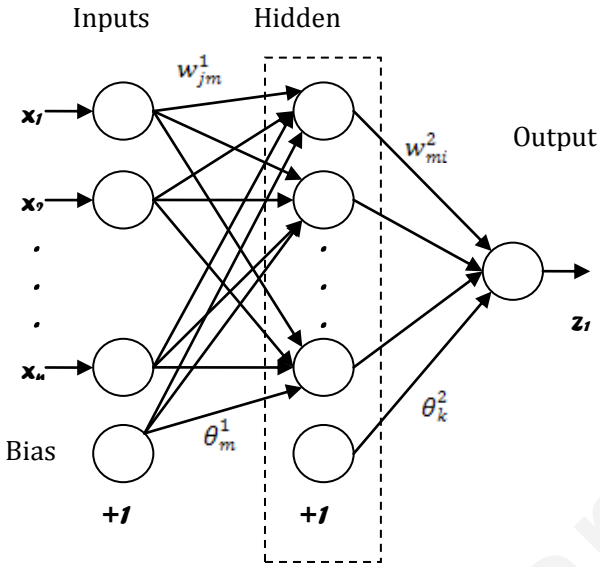


Figure 4.3: A Feedforward Multi-Layer Perceptron (MLP) Artificial Neural Network (ANN)

$$h_1^1 = f\left(\sum_{i=1}^{n} x_i w_{i1}^1 + \theta_1^1\right) \tag{4.6}$$

$$f(y) = (1-\exp(-by))*(1+\exp(-by))^{-1} \tag{4.7}$$

The ANN were trained using the Levenberg-Marquardt (Levenberg, 1944; Marquardt, 1963) backpropagation algorithm (*trainlm*) which is widely accepted due to its wide range of problem applicability and generally good performance (Demuth and Beale, 2000; Shukla and Misra, 2008; Koivo, 2008). The maximum number of epochs and the initial momentum constant were set equal to 100 and 0.001 respectively. Training was repeated 20 times (as proposed by Prechelt (1994)) and also, validation was used as a pseudo-test to ensure that the network was indeed trained and to impose 'early stopping'. Early stopping is considered a powerful method to apply cross-validation in ANN (Prechelt, 1994). In each iteration of training random initialisation was performed, i.e., the weights and momentum coefficients (or bias) were randomly initialised with the Nguyen-Widrow initialisation method (Nguyen and Widrow, 1990). The performance function was the *Mean Squared Error* (*MSE*). The best trained ANN, i.e., the ANN that yielded the lowest *Mean Magnitude of Relative Error* (*MMRE*) figure, was used for evaluation, i.e., was used for the final prediction using the testing samples. The selection of *MMRE* for comparing models was made since it is the most widely used criterion to assess SCE model's performance in literature, even though it has been

proven not suitable for comparisons across datasets or all kinds of model types (Myrtveit et al., 2005).

The indicative best results in terms of effort prediction obtained from the single hidden layer SB-SCE ANN models are summarised in Table 4.3 (Papatheocharous and Andreou, 2011). The term 'best result' is used to discern the lowest in *MMRE* result one can get using a particular method. The ANN were also constructed a number of times before reaching to the models summarised in Table 4.3, with variations of activation functions in the hidden layer, performance function, number of hidden layer neurons, etc. and then picking the type of network that yielded the best result. The results reported were obtained with minimum experimentation, i.e., most of the parameters altered at the beginning (for model calibration) did not affect the result and thus many ANN parameters were kept constant (and same as the default ones) such as the performance function and the training parameters.

Table 4.3: Selected experimental results from single hidden layer ANN MLP of SB-SCE models

DATASET	INPUT	TOPOLOGY	TESTING PHASE			
			MMRE	***CC***	***NRMSE***	***Pred(.25)***
COCOMO	SLOC	1-4-1	1.629	0.597	2.890	0.154
Kemerer	AFP	1-3-1	0.282	0.943	0.614	0.333
	KSLOC	1-3-1	0.257	0.792	0.527	0.333
Albrecht and Gaffney	FP	1-2-1	0.324	0.987	0.149	0.600
	SLOC	1-4-1	0.469	0.985	0.691	0.200
Desharnais	AFP	1-3-1	0.348	0.712	0.696	0.400

The overall accuracy performance indicates that the ANN approach can reach to quite promising effort approximations with very simple architectures as the level of *MMRE* obtained is in some datasets is near to the acceptable rate for software prediction models set by Conte et al. (1986), i.e., ≤ 0.25. The datasets whose effort is better approximated is the Kemerer, followed by the Albrecht and Gaffney (using FP as input) and then followed by Desharnais. This means that the optimal SB-SCE models with ANN cannot be constructed for every case (dataset). Moreover, the accuracy level obtained across the datasets and across the accuracy metrics reported varies considerably, thus inconsistent ordering of the various accuracy measures used is evident. This also means that for some datasets and some accuracy

metrics the predicted value obtained for effort is considered successful (since it is very close to the actual) and for others it is not.

Commenting the above results, we observe that for the Albrecht and Gaffney dataset when FP are used as inputs instead of LOC, the prediction is better which shows that there is an indication of high association between the two (i.e., FP and effort). Metrics such as LOC, even though seem to affect the value of effort (e.g., in the Kemerer case), they do not affect the model in yielding better predictions. This may be attributed to the subjectiveness of LOC estimates as a metric and its high dependence on factors not considered in the experiments, such as complexity of the code, code design, programming skills and style of the developers, degree of code re-use, programming language and type (if it is object-oriented or not) and the use of tools for generating code. For example, application generators such as Eclipse IDE generate automatic code for POJO (Plain Old Java Objects) (a term coined by Fowler et al., (2000)), e.g., constructors, getters, setters, as well as comments, without any effort spent on actual coding. Whereas FP estimates are considered less subjective as a metric by definition, because even though they are affected by factors such as the software complexity in the functional aspect, the latter remains unaffected during the project, and thus, is independent from technical factors, such as the ones affecting LOC and the actual implementation. However, since FP are affected by the team's and project manager's domain experience, requirements, changing requirements, etc., it is worthwhile to investigate such factors in subsequent research analyses.

Additionally, for comparison purposes, Table 4.4 summarises the prediction results over the same project samples and data separation utilising a popular method in the literature, i.e., Ordinary Least Squares (OLS) Regression. Regression was used to capture the relationship between size and effort of projects in the form of an exponential function which was represented by a polynomial transformed to linear using the natural logarithm. Polynomial curve fitting was used for finding coefficients β_0 and β_1 (as specified in eq. (3.9) of section 3.2.2.1, pg. 80 representing the constant and the slope of the regression curve using the

training set in a least-squares sense. Then, the coefficients were utilised for estimating effort of the testing set.

Table 4.4: Selected experimental results from OLS Regression of SB-SCE

DATASET	INPUT	TESTING PHASE			
		MMRE	***CC***	***NRMSE***	***Pred(.25)***
COCOMO	SLOC	1.022	0.639	1.045	0.154
Kemerer	AFP	0.196	0.974	0.289	0.667
	KSLOC	0.234	0.825	0.566	0.667
Albrecht and Gaffney	FP	0.248	0.969	0.453	0.600
	SLOC	0.397	0.970	0.556	0.200
Desharnais	AFP	0.311	0.708	0.819	0.467

The aim here was to investigate as a starting point the performance of two well-known, established and simple techniques in SCE (Papatheocharous and Andreou, 2011). Particularly the experimentation was made only with projects developed by single companies whereas projects developed within various organisations will need to be also examined. The experimental results showed that Regression outperformed the simple single hidden layer ANN using the exact same data partitions in all the respective datasets. The rest of the training and generalisation (testing) performance figures of the models are provided in Appendix B (pg. 291-294).

Taking into consideration all the experiments conducted, it became evident that several parameters calibrated, such as the internal activation function in the ANN model, did not differentiate the performance results. However, both models were found to be particularly sensitive in all the datasets used to the random subdivision of training, validation and testing sets, and particularly to the type of inputs used (i.e., FP or LOC in cases where both were available). In most datasets, both models performed better when FP instead of LOC were used. Thus, the high dependence between FP and effort was confirmed by both models examined (ANN and Regression).

Nevertheless, further investigation needs to be carried out, especially regarding the network architecture, the optimum number of internal hidden neurons (computational elements) and combination of types of size-related metrics to yield even better software cost predictions in these benchmark datasets.

4.2.1.2 Multiple Hidden Layer MLP ANN for SB-SCE

The previous experimentations provided evidence that there is need to develop and examine perhaps more complicated ANN topologies that could optimise prediction accuracy, reliability and generalisability. Therefore, the models subsequently developed (Papatheocharous and Andreou, 2008) were based on multiple hidden layer MLP networks. Moreover, a dedicated method was applied to combine size-related measures and actual effort expended to develop these products. The coupling method used a variable window of *i* past projects, where *i=1,...,5*, so that the ANN could become 'aware' of this information and assimilate it as a pattern. These patterns of completed projects of size and effort were expected to increase the learning capacity of the ANN and optimise effort prediction of impending new projects. The size of the project's window was considered sufficient based on the rather small project samples available in the datasets utilised.

The ANN models were developed in the Neuroshell tool Release 2.0. (Ward System Group, 2008). The architecture employed is illustrated in Figure 4.4. The ANN input layer utilised the *linear* function (and thus the inputs were normalised in the range [*-1, 1*]) and the output layer used the *logistic* function. The networks were built connecting each input neuron with hidden layers consisting of three parallel slabs activated by different functions, i.e., the *hyperbolic tangent*, the *Gaussian* and the *Gaussian complement* as specified in eqs (4.7) pg. 105, (4.8) and (4.9) respectively.

$$f(y) = \exp\left(-x^2\right) \tag{4.8}$$

$$f(y) = 1 - \exp\left(-x^2\right) \tag{4.9}$$

The *linear* and *tanh* functions were selected because they are considered useful when the output is a continuous variable, the *logistic* function because it is considered the most useful function for most ANN applications, the *Gaussian* is especially useful in data-starved domains while both the *Gaussian* and the *Gaussian complement* bring out meaningful characteristics in

the extremes of the data (Ward Systems Group, 2008) which is useful in the case of the size-based data examined.

Empirical variations of architectures were examined regarding the number of neurons in the internal hidden layers (Papatheocharous, 2004). The backpropagation algorithm was used for training with performance error the difference of the sum of squares and the number of maximum iterations (epochs) was 10,000. The learning rate, momentum and initial weights were set to 0.1, 0.1 and 0.3 respectively (Papatheocharous, 2004; Papatheocharous and Andreou, 2008).

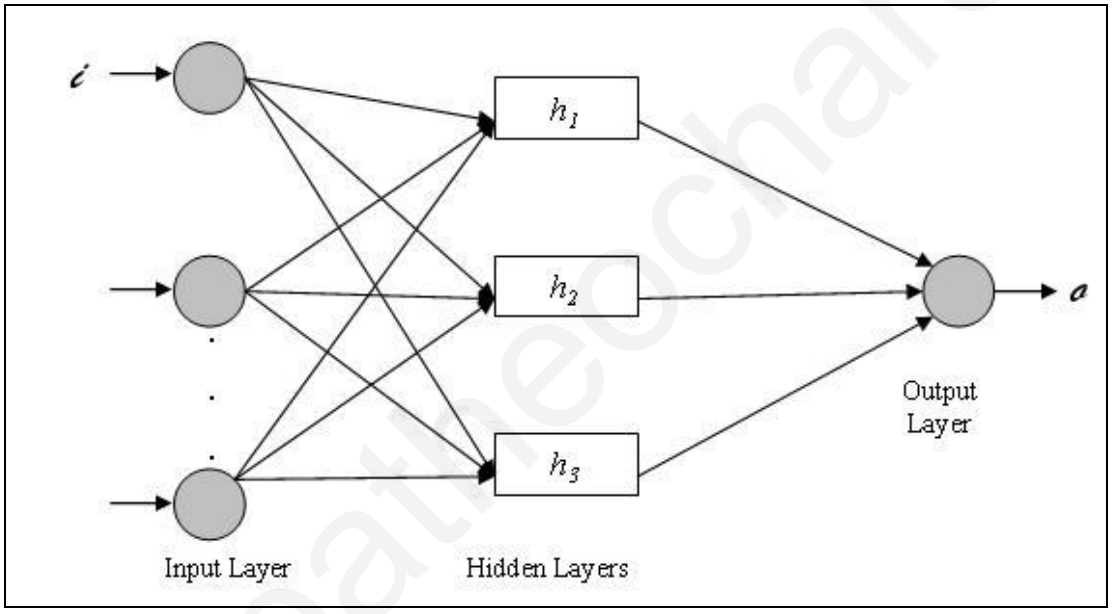


Figure 4.4: Feedforward MLP Artificial Neural Network (ANN) consisting of an input and an output layer and three slabs of hidden layers neurons.

The varying size sliding-window technique applied on the data aimed to provide a specific sample sequence order (t_i). This order specified couples of size-effort data of *i* projects, which were fed to the ANN models using this particular organisation in patterns, to estimate the effort of the $(t+1)^{th}$ project. These combinations allow ANN to employ pattern matching and discovery regarding the dependent variable (effort) in each coupling of size-related cost drivers and their effort. Again, as a starting point, the ability of only size-related software metrics (LOC and FP) was investigated to approximate effort. The window-sliding size *i*

varied, from 1 to 5, as defined in Table 4.5, covering the following Input Output Methods (IOM):

- IOM1-IOM2: Using the Lines of Code or Function Points of *i* projects attempt to estimate the effort of the i^{th} project;
- IOM3-IOM4: Using Lines of Code or Function Points with effort of the i^{th} project to estimate the effort required for the next project $(i+1)^{th}$ in the series sequence;
- IOM5-IOM6: Using Lines of Code or Function Points of the i^{th} and $(i+1)^{th}$ projects and effort of the i^{th} project we estimate the effort required for the $(i+1)^{th}$ project.

In each IOM the number of past samples included in the sliding-window was increased as *i* increased and all variations were examined.

Table 4.5: Sliding-window technique to determine the ANN inputs

METHOD	SOFTWARE METRICS*	Output*
IOM1	$LOC(t_i)$	$EFF(t_i)$
IOM2	$FP(t_i)$	$EFF(t_i)$
IOM3	$LOC(t_i)$, $EFF(t_i)$	$EFF(t_{i+1})$
IOM4	$FP(t_i)$, $EFF(t_i)$	$EFF(t_{i+1})$
IOM5	$LOC(t_i)$, $LOC(t_{i+1})$, $EFF(t_i)$	$EFF(t_{i+1})$
IOM6	$FP(t_i)$, $FP(t_{i+1})$, $EFF(t_i)$	$EFF(t_{i+1})$

(*where $i=1...5$)

The same pre-processed datasets described in section 4.2.1.1 along with the cross-organisational projects obtained from the ISBSG dataset were also used. The filtered version of the ISBSG R9, named ISBSG R9-1, was used consisting of 961 projects. The details of the pre-processing applied are described in Table 4.1 (a), (b)-(h). From the attributes only the Effort (EFF) and size-related factors, i.e., the Adjusted Function Points (AFP), were selected.

In these multiple hidden layer MLP ANN models the number of projects utilised in the training process was increased due to the sliding-window technique applied. Therefore, 70% of the original data samples, instead of 60%, were used for training, 20% for validation of training and finally, only 10% of holdout samples were used for testing the generalisability of the model. Some indicative performance results (Papatheocharous and Andreou, 2008) are summarised in Table 4.6.

Table 4.6: Selected experimental results from multiple hidden layer MLP ANN sliding-window

DATASET	METHOD	WINDOW SIZE *i*	TOPOLOGY	TESTING PHASE		
				MMRE	***CC***	***NRMSE***
COCOMO	IOM5	1	3-15-15-15-1	0.551	0.407	0.952
	IOM1	2	2-9-9-9-1	0.525	0.447	0.963
Kemerer	IOM1	1	1-15-15-15-1	0.256	0.878	0.830
	IOM5	2	5-20-20-20-1	0.232	0.988	0.503
Albrecht and Gaffney	IOM6	2	5-3-3-3-1	1.142	0.817	0.649
	IOM2	2	2-20-20-20-1	1.640	0.936	0.415
Desharnais	IOM4	2	4-9-9-9-1	0.481	0.970	0.247
	IOM4	3	6-9-9-9-1	0.051	1.000	0.032
ISBSG R9-1	IOM2	4	4-15-15-15-1	0.843	0.577	2.617
	IOM6	1	3-3-3-3-1	0.809	0.601	2.435

The figures show that deploying a mixture of architectures and IOM yields various accuracy levels based on the intrinsic characteristics of the datasets. The performance accuracy obtained compared to the previous experiments (summarised in Table 4.3) is improved in the COCOMO but worsened in the Albrecht and Gaffney case. This means, that for the latter dataset, the effort of projects cannot be accurately calculated (or at least approximate the acceptable rate for software prediction models set by Conte et al. (1986), i.e., ≤ 0.25)) with the multiple hidden layer neuron MLP ANN developed and using the patterns of LOC and FP from the rest projects contained in the dataset. This means that at least for the Albrecht and Gaffney projects, there are other factors that drive effort to divergent levels and thus, the addition of the values from other projects to the model does not contribute positively in obtaining better prediction results.

Also, it is worth mentioning that similar performance compared to the performance of Table 4.3 was obtained in Table 4.6 for the Kemerer case, whereas in the COCOMO and only in one of the Desharnais cases accuracy was considerably improved to successful levels. However, in the other case from the Desharnais projects the ANN accuracy again deteriorated. The main observation obtained is that effort predictions appear quite divergent even across similar datasets (i.e., datasets containing similar small number of projects, developed in the same decade, by single-companies) and even in predictions within the same datasets. The same applies for the multi-organisational dataset of the ISBSG R9-1.

The same observation can again be made as with the previous experiments conducted, i.e., that FP and LOC metrics alone cannot always produce accurate effort estimations for every

dataset and thus other factors affecting effort might need to be considered. Also, the results obtained for the cases thus far, infer that for the projects in the COCOMO and Desharnais datasets the addition of past historical pairs of size-effort to the model increases performance accuracy. This may be attributed to some pattern existence between the size-related metrics and effort of the projects within these datasets, which is discovered by the ANN models.

Another interesting observation here is that the majority of the optimal ANN models employ a large number of internal neurons. Therefore, it seems that the optimal type of ANN architecture in SCE is an important issue for investigation and relying on simple empirical rules to determine it is not the best approach. Finally, since considerable accuracy performance variation was observed in the results, the models proposed are probably not the optimum ones for each case, and thus, further investigation is needed with respect to the selection of a 'closer to optimum' ANN topologies in each dataset case as well as considering factors other than size (which is addressed in a subsequent section, i.e., 4.2.2 (pg. 121)).

4.2.1.3 Hybrid Multiple Hidden Layer MLP ANN Coupled with GA

Early investigations of ANN emphasised on the importance of the inputs used, the architecture (size, topology/structure, connectivity) and the existence of a deterministic relationship among input and target (Wittig and Finnie, 1997), so that, accurate predictions can be obtained. The lack of existence or appropriate selection of any of the above may cause inadequate learning. The SB-SCE models developed next were based on a hybrid technique, i.e., combining ANN with Genetic Algorithms (GA) for optimising the networks utilised for predictions in the previous section. The main objective was to exploit the benefits of Evolutionary Algorithms (EA) in software cost modelling of ANN and provide better effort predictors for impending new projects by calibrating the number of hidden layers and nodes (Papatheocharous and Andreou, 2011). Since manually determining the structure of the ANN is a tedious task and common trial-and-error methods entail the risk of overlooking promising optimisations, a dedicated GA was constructed to calibrate the model's architecture. The

investigation included whether the combination of ANN and GA, may contribute in choosing an 'ideal' ANN architecture for the particular set of inputs that meet some performance evaluation criteria, i.e., improve performance accuracy and minimise the complex structure of the ANN as much as possible.

The GA developed aimed to obtain the ideal network structural settings by means of a cycle of generations including candidate solutions that were pruned by the criterion 'survival of the fittest'. This criterion was specified for ANN as the 'best performing' (in terms of effort prediction accuracy in the evaluation phase) and the 'less complex' structure (in terms of number of hidden neurons). The aforementioned criterion was selected after conducting some preliminary experiments (Papatheocharous, 2004) that showed that the GA favoured overly complex networks with large number of internal neurons in the multiple hidden layers. This may cause 'overfitting' of the ANN, which in this case was avoided by using validation mechanisms of the generalisation ability of the ANN. The GA therefore replaced the trial-and-error method previously applied to select the optimum architecture for the aforementioned IOM of the SB-SCE ANN models with the risk of overlooking models that would potentially lead to optimisation of performance.

The following steps describe the way the genetically evolved SB-SCE models (also referred as hybrid ANN & GA in Figure 4.1) were implemented (Papatheocharous and Andreou, 2008):

(1) The initial population of individuals was created randomly containing an encoding of the necessary pieces of information, that is, the number of internal hidden neurons for each hidden layer and the Input Output Method (IOM) – see Table 4.5.

(2) From each individual of the generation the information regarding the network architecture and the structure of the input vector was extracted. Then the corresponding network was initialised, trained for a number of epochs and finally, tested. From the testing results obtained, all individuals were evaluated and the network state and performance results were stored. The performance of each

individual was calculated based on the fitness function (see eq. (4.10)) an equation based on common forecasting performance errors.

(3) Once all individuals of the respective generation were trained and tested on generalisation, the generation was evaluated as a whole. This was carried out by summing the fitnesses of each individual in the generation.

(4) The top 5% of best individuals were forwarded to the next generation unchanged (elitism) and the rest individuals required to complete the size of the next generation were obtained through generic evolution steps applying the selection, crossover and mutation operators. The offsprings produced through these steps replaced their parents in the original population.

(5) Steps (2), (3) and (4) were repeated until finally a predefined number of generations had been reached. Each generation evolved one or more near to optimal solution(s), that is, the best ANN architectures in terms of internal neurons for the IOM selected for predicting effort with the highest possible accuracy within the current population.

More specifically, the first task for implementing the hybrid model was to determine a type of encoding so as to express the potential solutions. The encoding used was a binary string representing the ANN structure, the internal hidden neurons following the various methods of inputs' coupling, i.e., effort and size-related attributes. The inputs were inserted into the ANN models created within the hybrid algorithm following the IOM specified earlier (Table 4.5). The number of neurons used in the hidden slabs was restricted not to exceed 20 neurons to avoid building ANN models that would lead to overfitting. The space of all feasible solutions (i.e., the set of solutions among which the desired solution resides) is called the 'search space'. Each point in the search space represents one possible solution. Each possible solution was 'marked' by its fitness value, which in our case was expressed by eq. (4.10), minimising the *MMRE* (as defined in eq. (2.1) pg. 50) and the overall size of the network, i.e., the total number of internal neurons.

$$fitness = \frac{1}{1 + MMRE + size} \tag{4.10}$$

The hybrid GA searched the problem space to locate the best solution among a number of possible solutions. Searching for a solution is then equal to looking for some extreme value (minimum or maximum) in the search space. The number of generations was set equal to 30 and the population of each generation equal to 50.

The GA developed included three types of operators: selection (roulette wheel), crossover (with crossover rate equal to 0.25) and mutation (with mutation rate equal to 0.01). Selection chooses members from the population of chromosomes proportionally to their fitness and elitism was used to ensure that the best members of each population were always selected for the new population. Crossover adapted the genotype of two parents by exchanging parts of them and creating new chromosomes with modified genotypes. Crossover was performed by selecting a random gene along the length of the chromosomes and swapping all the genes after that point. Finally, the mutation operator simply changed a specific gene of a selected individual in order to create a new chromosome with a different genotype.

The hybrid ANN utilised in the input layer the *netsum* function (defined in eq. (4.6), pg. 105), the hidden layer the *hyperbolic tangent sigmoid transfer* (*tansig)* function (defined in eq. (4.7), pg. 105) and finally, the output layer utilised the *purelin* function. The backpropagation training and learning function that updates weight and bias values according to gradient descent with momentum (*traingdm*/*learndgm*) was used. The maximum number of epochs was set to 100, early stopping was applied and the learning rate and momentum constant were both set to 0.3. The training process was repeated 3 times and each time the network weights and bias were randomly initialised with the Nguyen-Widrow method (Nguyen and Widrow, 1990). The performance function used was the *Mean Squared Error* (*MSE*) with reg performance function (*msereg*), which takes into account the weight sum of two factors, the mean squared error and the mean squared weight and bias values and is referred as an alternative performance function to improve generalisation (Beale et al., 2011).

The indicative 'best' results obtained from the hybrid multiple hidden layer MLP genetically evolved SB-SCE ANN models for a constant window size (see Table 4.5, i=1) are summarised in Table 4.7 (Papatheocharous and Andreou, 2009c; Papatheocharous and

Andreou, 2011). The estimation of *MMRE, CC* and *NRMSE* was carried out on the normalised actual and predicted effort values (as specified in Table 4.1 (h)).

Table 4.7: Indicative experimental results of the constant sliding-window size hybrid model (ANN &GA) with multiple hidden layer MLP ANN coupled with GA for SB-SCE

DATASET	METHOD*	ANN ARCHITECTURE	TESTING PHASE		
			MMRE	***CC***	***NRMSE***
COCOMO	IOM1	1-9-17-10-1	0.003	1.000	0.014
	IOM3	2-20-18-3-1	0.075	0.961	0.278
	IOM5	3-19-20-4-1	0.044	0.981	0.199
Kemerer	IOM1	1-17-13-16-1	0.009	1.000	0.019
	IOM3	2-18-14-18-1	0.211	0.822	0.550
	IOM5	3-19-15-20-1	0.028	0.997	0.081
	IOM2	1-17-20-11-1	0.009	1.000	0.006
	IOM4	2-19-15-20-1	0.062	0.993	0.122
	IOM6	3-19-9-16-1	0.031	0.999	0.051
Albrecht and Gaffney	IOM1	1-13-20-6-1	0.005	1.000	0.024
	IOM3	2-19-20-8-1	0.163	0.977	0.210
	IOM5	3-19-11-10-1	0.173	0.975	0.218
	IOM2	1-9-17-10-1	0.014	1.000	0.018
	IOM4	2-18-15-11-1	0.112	0.985	0.171
	IOM6	3-20-19-10-1	0.084	0.984	0.177
Desharnais	IOM2	1-3-18-20-1	0.016	0.998	0.075
	IOM4	2-20-19-20-1	0.589	0.437	1.022
	IOM6	3-20-20-19-1	0.381	0.674	0.750
ISBSG R9-1	IOM2	1-16-18-11-1	0.004	0.998	0.073
	IOM4	2-19-14-20-1	0.952	0.030	1.141
	IOM6	3-19-15-26-1	1.312	0.705	0.742

*sliding-window size *i* was equal to 1.

The hybrid model developed shows ability to optimise the architecture and the accuracy performance of estimating effort in all the datasets, even though the results cannot be directly compared to the previous set of experiments. Table 4.8 summarises the results obtained for a varying sliding-window size (Papatheocharous and Andreou, 2008; Papatheocharous and Andreou, 2009c). The larger sliding-window size employed, aims to feed the model with more pairs of historical samples of size-effort so that more information is used to train the model. Moreover, the number of generations and number of individuals within the generations were altered to 200 and 30 to allow the algorithm to converge, as some preliminary experiments trying a few variations on these parameters indicated that the algorithm needed to explore more generations. Increasing the number of generations led to vast increase of the time the GA required to reach to a solution and therefore, to minimise the execution time fewer solutions (individuals) were examined in each generation. Moreover, the ANN

developed were trained using the scaled conjugate gradient backpropagation (*trainscg*) over a maximum of 500 epochs and early stopping was applied.

Table 4.8: Indicative experimental results of varying sliding-window size hybrid model (ANN &GA) with multiple hidden layer MLP ANN coupled with GA for SB-SCE

DATASET	METHOD	WINDOW SIZE *i*	ANN ARCHITECTURE	TESTING PHASE		
				MMRE	***CC***	***NRMSE***
COCOMO	IOM1	3	3-25-2-9-1	0.431	0.838	0.549
	IOM3	1	2-16-21-18-1	1.967	0.942	0.556
	IOM5	3	7-0-9-14-1	0.981	0.708	0.725
Kemerer	IOM1	2	2-20-14-2-1	0.572	-0.552	1.521
	IOM3	1	2-11-4-5-1	0.474	-0.500	1.593
	IOM5	2	5-6-19-6-1	0.572	-0.551	1.521
Albrecht and Gaffney	IOM2	6	6-20-6-11-1	0.083	0.141	1.109
	IOM4	4	6-3-7-3-1	0.113	0.061	1.075
	IOM6	1	3-0-2-5-1	0.083	0.141	1.109
Desharnais	IOM2	2	2-20-19-19-1	0.285	0.899	0.443
	IOM4	3	6-17-15-20-1	0.327	0.835	0.550
	IOM6	2	5-20-19-20-1	0.493	0.461	1.058
ISBSG R9-1	IOM2	3	3-19-9-3-1	0.070	0.241	1.000
	IOM4	2	4-19-20-20-1	1.201	0.005	1.738
	IOM6	2	5-19-20-19-1	0.892	-0.053	1.648

The performance of the various ANN architectures constructed with the aid of the GA and for the IOM examined, indicates that quite large differences in behaviour are observed, i.e., very high and mediocre learning ability is indicated by the error figures obtained in the evaluation of all the datasets. In regards the results of Table 4.7 (listing the performance obtained from the hybrid model with a constant sliding-window size) quite complex architectures are yielded as the optimum models. In fact, the ANN architectures comprised of many hidden layers and many neurons in these hidden layers.

In regards the results listed in Table 4.8 the main observation is that for almost all of the datasets at least one optimal model of ANN is found to maximise the prediction accuracy compared to the previous performance summarised in Table 4.3 and Table 4.6. In fact, considerable effort prediction accuracy gain is obtained for the cases of the COCOMO, the Albrecht and Gaffney, and the ISBSG R-1 datasets respectively. This means that at least for

these cases the GA improved the performance results and also automated and optimised the process of ANN construction, resorting to less complex architectures this time.

Some additional comments are provided regarding the vast and less homogeneous dataset (ISBSG R9-1) which across the experiments conducted exhibits higher variations in performance than the other datasets (i.e., Table 4.7 IOM2 and IOM6/*i=1* and Table 4.8 IOM2/*i=3* and IOM4/*i=2* refer to two extreme cases).

Figure 4.5 also presents the actual versus the predicted normalised effort sample values during the training and testing phases of the ANN juxtaposed from an indicative experiment using the ISBSG R9-1 dataset of the IOM2, *i*=1 scheme with the ANN architecture 1-16-18-11-1 which is the best yielded prediction.

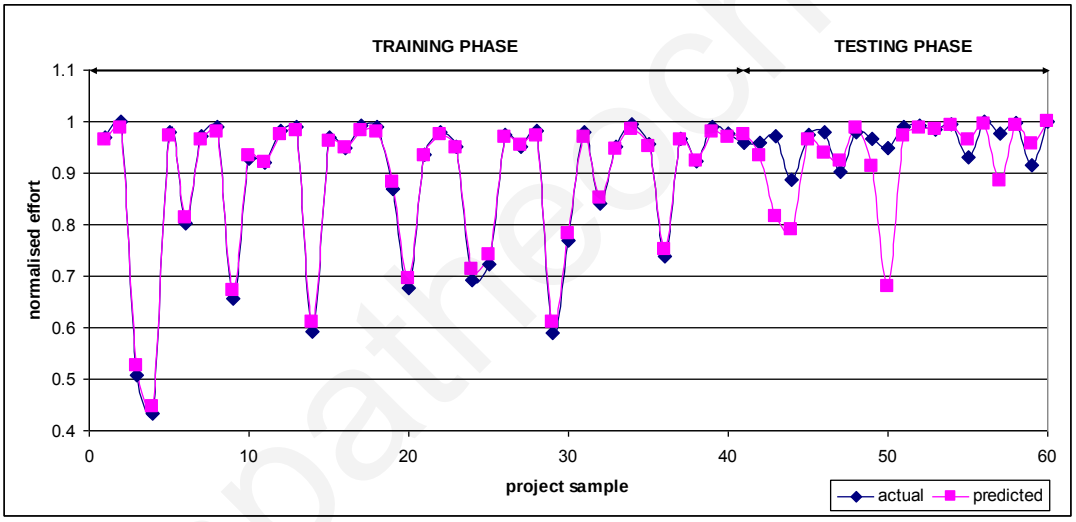


Figure 4.5: Actual vs. Predicted normalised effort estimation values with ANN architecture 1-16-18-11-1 and IOM2 on the ISBSG R9-1 dataset

The above leads us to conclude that solving the problem of SCE for large and multi-organisation datasets, even though some promising results have been obtained, is quite difficult and requires further research. Future research work within this context could be directed on establishing measures for the sensitivity of the models and intelligent selection of the input parameters/data separation methods/model parameters' calibration, etc.

Lastly, commenting on the window of projects fed to the ANN it was logical to expect that providing a larger number of coupled data of size-effort to the model regarding past projects (i.e., larger window size), would inherently improve the pattern matching ability and

consequently the prediction results of the ANN, compared to the experiments conducted with a constant and smaller window size (i=1), but this was not observed. This of course might hold if the records came from a single population, which may not be the case here since a randomisation process for data separation was applied.

Nevertheless, it may be assumed that there are lower correlations between the patterns of effort-size across the various projects reported in the datasets than the ones expected. This is perhaps an indication of the lack of explicit pattern relationship between the couples of size-effort and thus inadequate learning is many times observed (refer also to some additional experiments and results in Appendix B).

Moreover, the algorithm proposed may not be the optimal solution for this problem since conflicting goals were included in the internal fitness function (i.e., low number of processing elements – neurons – and improvement in performance). This leads us to conclude that multi-objective Evolutionary Algorithms (EA) may comprise more appropriate solutions.

Summarising this section, the experiments conducted showed that the hybrid model of genetically evolved ANN is a promising approach for SCE that yields improved results compared to the Ordinary Least Regression (OLS) and the empirically designed ANN, even though some limitations exist.

These limitations include the high execution time and computation power required to find the optimal ANN architectures. Other limitations are classical situations appearing in any other SB-SCE approach, and involve the LOC-size estimates. These estimates must be known in advance to provide accurate enough effort estimations, which is never the case. Also, size-related estimates anticipated by the project manager before the software is actually developed are usually different than the size of the actual delivered code. Moreover, there is generally the belief that the same project counted twice will not give exactly the same size (LOC or FP) or even effort measurements. This basic limitation and other measurement validation issues are usually recognised between practitioners (Abran and Robillard, 1994; Poels, 1996; Kitchenham et al., 1997). In addition, differences in measurements of size and effort should be taken into consideration in any approach used for SCE and especially when a large

discrepancy between the actual and estimated size, is occurring in the estimations made at the early project phases.

Another limitation is the lack of a satisfactory volume of homogeneous data which has lowered the performance accuracy of effort predictions in many cases in our experiments. There is also lack of a clear definition and measurement rules for size units, such as LOC and FP, which result in uncertainty to the whole estimation process. The software size is also affected by other factors that are not investigated by the models of this approach, such as the programming skills, language and platform used during development. This section of this diatribe has focused only on coding effort, which accounts for only a percentage of the total effort in software development.

Another limitation related to the technologies used is that the ANN have many candidate solutions as architectures which cannot be optimally trained every time and so even the GA developed requires to conduct extensive search of the solution space, something which is very time-consuming. The coupling of size-related attributes with effort fed to the ANN in the form of training patterns are considered a viable solutions for efficiently modelling the relation of size and effort, since in some cases the ANN converged to very accurate results. Finally, as already mentioned, further research needs to be carried out in this context, investigating the degree of influence of other project attributes on development effort (such as programming language used, personnel skills etc.) and locate such interdependencies. The following section deals with this matter.

4.2.2 Feature Subset Selection and Software Cost Estimation (FSS-SCE)

SCE models make use of a wide range of project attributes, also called ‘cost drivers’, for approximating development effort as accurately as possible. Some cost drivers relate to the actual size (code length in Lines of Code (LOC) or Function Points (FP)), complexity, duration of the project, experience of the people involved in developing the system, etc. The relevant SCE research conducted thus far utilises models such as ANN commonly only as

predictors - and very rarely quantitative analysis is conducted on the influence of the inputs to the model's output. Obtaining the influence degree of inputs within a modelling technique that estimates effort, means that possibly some inputs will be regarded as more significant than others. Moreover, identifying the dominant cost drivers from a wider pool of available data has been recognised as a crucial step in SCE (Park and Baek, 2008).

The hybrid models presented in this section for identifying the optimal subset of cost drivers are based on two popular SCE techniques, namely Artificial Neural Networks (ANN) and Ridge Regression (RR). The methodologies involve Feature Subset Selection in SCE (FSS-SCE) to serve the following twofold aim: (i) identify and investigate the leading cost drivers that decisively influence software effort, and (ii) provide sufficiently accurate cost approximations (i.e., near or close to the acceptable rate for software prediction models set by Conte et al. (1986), i.e., ≤ 0.25).

4.2.2.1 ANN and Input Sensitivity Analysis (ISA) for FSS-SCE

The ANN developed thus far consist of variations of structures of basic computational elements that aim at measuring or predicting software development effort. ANN models combined with supportive tasks, such as Input Sensitivity Analysis (ISA), can assess quantitatively and rank the significance of any set of attributes used as inputs for these models (Papatheocharous and Andreou, 2007; Papatheocharous and Andreou, 2010; Papatheocharous and Andreou, 2012b). Isolating or extracting a subset of the most 'significant' or 'influencing' project attributes and investigating whether this FSS improves, or, at least maintains the same levels of estimation accuracy is a critical exploration (Papatheocharous et al., 2010b).

This section describes novel FSS methods for ANN to qualify the most 'significant' project features, i.e., those that seem to highly affect the estimation. The cost drivers selected are then utilised for SCE. The proposed methodology comprises of a practical way to reduce the model's input space (and also its computational complexity) while maintaining high levels of effort prediction accuracy. The basic idea is to describe and apply a simple yet reliable

automatic approach that leads to a subset of project attributes which may be more usable, i.e., more feasible to measure, collect and maintain, thus reducing human error and uncertainty.

The datasets employed were the Desharnais and the ISBSG R9 which went through the pre-processing steps described in Table 4.1 (a), (b) and (h). The ISBSG R9 additionally went through steps (f) and (g) and thus, 738 projects measuring the attributes summarised in the second part of Table 4.9 consisted the filtered ISBSG R9-2, while 77 projects measuring the attributes summarised in the first part of the same table were used in the Desharnais case. The attributes of Table 4.9 in italics represent dependent variables.

Table 4.9: Attributes used in the experiments of Desharnais and ISBSG R9-2

Dataset	Attributes	Abbreviation
Desharnais	*Development Effort (hours)*	*DEff*
	Team Experience (years)	TE
	Manager Experience (years)	ME
	Duration (months)	DU
	Transactions	TR
	Entities	EN
	Function Points Adjusted	FPA
	Scale of project	SC
	Function Points Non-Adjusted	FPNA
ISBSG R9-2	*Summary Work Effort (hours)*	*SWE*
	Functional Size	FS
	Adjusted Function Points	AFP
	Project PDR (afp)	PDRA
	Project PDR (ufp)	PDRU
	Normalised PDR (afp)	NPDRA
	Normalised PDR (ufp)	NPDRU
	Project Elapsed time	PET
	Project Inactive time	PIT
	Resource Level (ordinal)	RL
	Input count	INC
	Output count	OC
	Enquiry count	EC
	File count	FC
	Interface count	IFC
	Added count	AC
	Changed count	CC
	Deleted count	DC

The preliminary investigation employed MLP ANN of various topologies, i.e., with varying number of hidden neurons in the single internal hidden layer, starting from being equal to the number of inputs and increased by one until the number of hidden neurons reached twice the number of the inputs. The activation function used was the *hyperbolic tangent sigmoid transfer* (*tansig)* function (defined in eq. (4.7), pg. 105) in the input and

hidden layer and the pure linear (*purelin*) function was used in the output layer. Each ANN was trained to learn and predict the behaviour of the available datasets and characterise the relationship among the inputs and the output, the latter being the development effort. Training was based on the backpropagation algorithm and learning was based on the gradient descent with momentum weight/bias learning function (*learngdm*). The network training function updated the weight and bias values based on the scaled conjugate gradient method (*trainscg*). The initial weights and biases of each ANN were randomly set by the Nguyen-Widrow initialisation method (Nguyen and Widrow, 1990). The number of epochs was set to 100 and early stopping was applied. The data was randomly divided into 70% of the samples for training, 20% for validation and 10% for testing, and the whole process was repeated 10 times. This process is also known as holdout cross validation (Weiss and Kulikowski, 1991). During the process of training, the data was presented to the various ANN developed in patterns (inputs/output) and corrections were made on the weights of the network according to the overall error in the output and the contribution of each node to this error. After training was completed the 'best' ANN in terms of performance accuracy (*MMRE*) were extracted.

On these ANN the following Input Sensitivity Analysis (ISA) process was performed as initially described in (Azoff, 1994): Each weight connecting neurons from a particular input to the hidden layer was summed up, thus developing an order of significance for the inputs based on this sum. The higher the sum of weights for a certain input is the higher is its contribution in defining the final ANN output. Using these values of 'significance' the following filtering was applied (to select the most 'influential' inputs and reject the rest): This filtering was carried out according to two empirically defined thresholds, namely the *Strict* (*S*) (eq. (4.11)) and the *Less Strict* (*LS*) (eq. (4.12)) criteria, which promote inputs in each trained ANN whose sum of weights is above a certain threshold.

$$w_i^S = \frac{\max\{\mathbf{W}_{ki}\} - \min\{\mathbf{W}_{ki}\}}{2}, \text{ where } k \in \{A, B, ..., Z\} \text{ and } i \in \{1,2,...,n\} \quad (4.11)$$

$$w_i^{LS} = \max\{\mathbf{W}_{ki}\} * 0{,}25, \text{ where } k \in \{A, B, ..., Z\} \text{ and } i \in \{1,2,...,n\} \quad (4.12)$$

Particularly, the *Strict* (*S*) criterion threshold considers as 'significant' the inputs whose sum of weights was higher than half the difference between the corresponding maximum and minimum weight sums among the inputs of the ANN (defined in eq. (4.11), where *k* represents the hidden layers and *i* the input neurons). Whereas, the *Less Strict (LS*) criterion threshold was more flexible, conceiving as 'significant' those inputs whose summed weights value was higher than the 25% of the ANN's maximum weight value (as defined in eq. (4.12)).

The final criterion to decide which inputs to select was based on eq. (4.13), which calculated for each input parameter *i* the percentage to which it was rated 'significant' using each (*S*) or (*LS*) criterion respectively:

$$rate\_total_i = \frac{N}{T} \ \% \tag{4.13}$$

where *N* is the number of the ANN satisfying the threshold for the *Strict* criterion (w_i^S) or the threshold for the *Less Strict* criterion (w_i^{LS}) and *T* is the total number of ANN created. In this way, not only the weights define which inputs to consider as 'important' factors to define predictions, but also the assessment of whether these variables were considered 'important' to more than half of the best ANN was investigated. Subsequently, after identifying and selecting the most 'influential' attributes validation experiments were executed.

During the 10 experimental repetitions performed the attributes suggested as significant by the *Strict* threshold for the Desharnais dataset were: the FPA and FPNA and for the ISBSG R9-2 dataset: the NPDRA, FC and AC. The attributes suggested by the *Less Strict* criteria case were for the Desharnais dataset: the TE, TR, FPA, SC and FPNA and for the ISBSG R9-2 dataset: the NPDRA, EC, FC, AC and CC. Moreover, a third subset of attributes was empirically selected so as to investigate the prediction of attributes that can be measured in the 'early' (end of specifications) project phases. In this case, the attributes for the Desharnais case were: the TE, ME, FPA and FPNA and for the ISBSG R9-2 were: the FS, AFP and NPDRU. These subsets of features were finally selected to repeat the same series of

experiments with ANN, as described before, to examine under the same conditions whether the exclusion of insignificant attributes from the datasets improves accuracy performance.

Table 4.10 lists the indicative prediction results obtained from the *Initial* phase using the whole spectrum of attributes with ANN and the selected significant attributes, called the *Final* phase. Table 4.10 lists also the number and the abbreviation of selected attributes based on each threshold used (i.e., *none:* all attributes at the Initial phase; *Strict*: selected attributes by the *Strict* (*S*) criterion at the *Final* phase; and *Less Strict* (*LS*): selected attributes by the *Less Strict* (*LS*) criterion at the *Final* phase; *Early*: attributes measured from the early SDLC phases at the *Final* phase). The error figures were calculated on the normalised data as previously described in Table 4.1 (h) (Papatheocharous and Andreou, 2007).

Table 4.10: ANN results obtained for the FSS-SCE using empirical ISA thresholds

Dataset	Phase/ Threshold	TESTING PHASE				Number/Selected attributes
		Architecture	*MMRE*	*CC*	*Pred(.25)*	
Desharnais	*Initial/none*	9-20-1	0.047	0.955	1.000	9/all
		9-16-1	0.061	0.933	1.000	
		9-10-1	0.063	0.922	1.000	
		9-19-1	0.074	0.900	1.000	
	Final/Strict	2-3-1	0.196	0.700	1.000	2/FPA, FPNA
		2-4-1	0.209	0.676	1.000	
		2-5-1	0.243	0.580	1.000	
		2-6-1	0.203	0.676	1.000	
	Final/Less Strict	5-7-1	0.184	0.833	1.000	5/TE, TR, FPA, SC, FPNA
		5-8-1	0.188	0.772	1.000	
		5-11-1	0.172	0.841	1.000	
		5-12-1	0.178	0.804	1.000	
	Final/Early	4-6-1	0.246	0.485	1.000	4/TE, ME, FPA, FPNA
		4-7-1	0.204	0.741	1.000	
		4-8-1	0.210	0.677	1.000	
		4-10-1	0.237	0.508	1.000	
ISBSG R9-2	*Initial/none*	17-22-1	0.064	0.938	1.000	17/all
		17-23-1	0.054	0.951	0.993	
		17-35-1	0.053	0.948	0.993	
		17-19-1	0.041	0.968	1.000	
	Final/Strict	3-4-1	0.037	0.852	0.993	3/NPDRA, FC, AC
		3-5-1	0.026	0.977	1.000	
		3-6-1	0.042	0.909	0.993	
		3-7-1	0.039	0.939	1.000	
	Final/Less Strict	5-6-1	0.038	0.957	1.000	5/NPDRA, FC, AC, EC, CC
		5-7-1	0.045	0.862	0.993	
		5-11-1	0.039	0.806	0.993	
		5-12-1	0.050	0.718	0.993	
	Final/Early	4-6-1	0.039	0.956	1.000	3/FS, AFP, NPDRU
		4-7-1	0.026	0.973	1.000	
		4-8-1	0.023	0.981	1.000	
		4-9-1	0.028	0.960	1.000	

The results using the selected attributes (*Final* phase) compared to the *Initial* cost estimates, with the whole spectrum of the available parameters acting as inputs, indicate an overall improvement of predictions in the ISBSG R9-2 case only, whereas in the Desharnais case accuracy appears to be worsened. Moreover, the case of 'early' estimation is considered quite successful again only for the ISBSG R9-2 dataset. Conclusively, the ISA applied for FSS in ANN has contributed to model improvement only in the ISBSG R9-2 case. In addition, the high *Pred*(*.25*) metric results show that through the predictions very few cases (projects) were not accurately predicted, i.e., prediction error was most cases lower than 0.25 in terms of *MMRE*. The less accurate predictions obtained for the Desharnais case and the selected attributes (compared to the *Initial*) may be attributed to the lower contribution of these attributes to the ANN prediction result.

Overall, the prediction results obtained with the ANN models are considered quite successful in terms of accuracy and at the *Final* experimental phase the slightly worse error figures (compared to the *Initial*) may be considered minimal trade-off with respect to the overall gain of the approach. This slight deterioration in terms of predictive power may be regarded for example acceptable by project managers since the approach managed to identify a very small number of features and may disengage the SCE process from the difficult and time-consuming need of gathering values for a large variety of metrics. Also, obtaining accurate estimations with attributes measured early is a significant advantage.

Subsequently, the abovementioned empirical ISA methodology applied was enhanced in (Papatheocharous and Andreou, 2012b). The idea was quite similar since the aim again was to eliminate less influential input parameters by computing the sensitivity level of each connection (weight) to the internal ANN structure and extending the idea of Azoff (1994). A more detailed and complete methodology was proposed (illustrated in Figure 4.6) to: (i) Isolate a set of ANN providing accurate estimates of software development effort, in terms of low prediction error and consistent performance, (ii) Identify a set of the most significant attributes through an enhanced ISA, and (iii) Reduce the size of input vector dimension of the ANN and not significantly compromise the accuracy of the results?

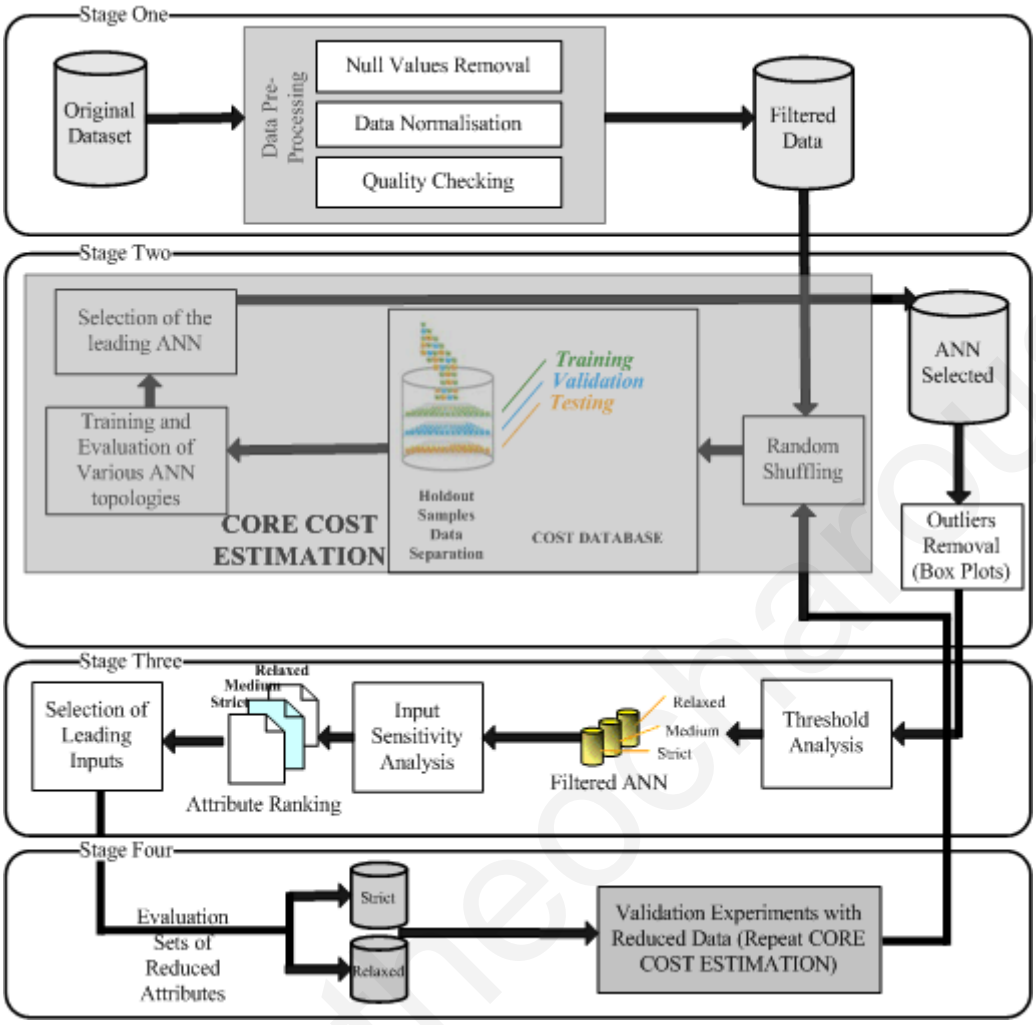


Figure 4.6: The stages of the ANN methodology with ISA for SCE (Papatheocharous and Andreou, 2012b)

A brief description of the complete methodology follows: In Stage One the necessary pre-processing tasks were carried out. In the Desharnais case the same pre-processing was carried out as described in Table 4.1 (a), (b) and (h), and thus the same 77 projects and attributes mentioned in Table 4.9 were used. In addition to these steps, for the ISBSG R9 case, the pre-processing involved the steps of Table 4.1 (c)-(f), (j) and (i), which were considered important to complete since they contribute to obtaining a qualitative and homogeneous filtered sample. The sample, called ISBSG R9-3 consisted of 113 projects and its attributes are summarised in Table 4.11. The attribute in italics represents the dependent variable.

Table 4.11: ISBSG R9-3 Attributes used in the enhanced ANN and ISA cost estimations

Dataset	Attributes	Abbreviation
ISBSG R9-3	*Summary Work Effort (hours)*	*SWE*
	Functional Size	FS
	Adjusted Function Points	AFP
	Project Elapsed time	PET
	Project Inactive time	PIT
	Resource Level (ordinal)	RL
	Maximum Team Size	MTS
	Input count	INC
	Output count	OC
	Enquiry count	EC
	File count	FC
	Interface count	IFC
	Added count	AC
	Changed count	CC
	Deleted count	DC

In Stage Two, the Core Cost Estimation (CCE) module was executed 250 times. The process executes random shuffling and data separation into 70%, 10%, 20%, percentages that refer to training, validation and testing subsets. In regards to previous experiments, here the testing percentage was increased to examine the ANN's generalisability on more testing samples. The process of validation is called random holdout samples cross validation (Weiss and Kulikowski, 1991; Maimon and Rokach, 2005). The same logic as before was applied in producing various feedforward ANN architectures of MLP with one hidden layer, i.e., the number of neurons in the internal hidden layer was modified, starting from the number of inputs for each dataset and increasing by one in each step until it reached to twice the number of inputs. The activation function used was the *hyperbolic tangent sigmoid transfer* (*tansig)* function (defined in eq. (4.7), pg. 105) in the input and hidden layer and the pure linear (*purelin*) function was used in the output layer. The MLP were trained with the gradient descent backpropagation (*learngdm*) algorithm. The number of training epochs was set to 100 and the training function updated the weight values according to the scaled conjugate gradient method (*trainscg*). The performance function was the *Mean Squared Error* (*MSE*), the learning rate and mutation constant were set to 0.3 and 0.6, whereas the Marquardt adjustment parameter, Marquardt decrease and Marquardt increase factors were set to 1, 0.8 and 1.5. 15% of the 'best' performing ANN were selected and Box Plots were used to exclude outlying networks, with extreme performance. Box Plots were used to examine the distribution of the

most popular performance metric (namely the *MMRE*) because it is a scale-independent metric. In this stage the examination did not target the exact error figures in terms of accuracy obtained from the normalised predicted values of the ANN, but targeted consistent and coherent (i.e., around-the-mean) networks that may provide a reliable input ranking. Thus, the ISA technique did not consider insufficiently trained and 'overfitting' ANN, which led to analysing ANN of consistent performance accuracy.

In Stage Three, the order of significance for the input attributes was examined using the value of the *Input Strength* (defined in the previous chapter in Table 3.5, pg. 79) with respect to three different thresholds (specified in Table 4.12 namely the *Strict*, *Medium* and *Relaxed*). These thresholds were defined to isolate groups of top-performing ANN (in terms of prediction accuracy), i.e., within 15%, 20% and 25% of the whole population of ANN produced as these were expressed through the *MMRE*, *CC* and *NRMSE* performance metrics. The error figures were calculated on the normalised data as described in Table 4.1 (h).

Table 4.12: Threshold specification for selecting ANN of various performing levels

	Strict	***Medium***	***Relaxed***
MMRE	≤ 0.15	≤ 0.20	≤ 0.25
CC	≥ 0.85	≥ 0.80	≥ 0.75
NRMSE	≤ 0.15	≤ 0.20	≤ 0.25

In the same Stage, right before executing the final validation experiments, the following FSS algorithm was employed which utilised the thresholds specified in Table 4.12 in order to select two levels of validation sets. The algorithm aimed to reduce the quantity of selected features to more than 50%:

(i) The first *trunc[n/2]* leading inputs according to their weights were isolated based on each filtered level of networks (i.e., the networks filtered by the *Strict*, *Medium* and *Relaxed* threshold were the *StrictANN*, *MediumANN*, *RelaxedANN* respectively), where *n* is the number of available attributes (e.g., *n=8* for the Desharnais and *n=14* for the ISBSG) and *trunc[]* denotes the integer part of the quotient.

(ii) The attributes ranked among the leading ones by all filtering levels were placed in the so-called *'Strict* evaluation set'.

(iii) If the number of elements in the *Strict* evaluation set was equal to *trunc[n/2]*, which essentially meant that each filtering level indicated the same leading inputs, the process was terminated. Otherwise, the process continued with step (iv).

(iv) The rest of the attributes that were promoted by any of the filtered levels of networks were further examined to create the *'Relaxed* evaluation set' as follows:

a. One by one the attributes promoted by the *Strict* level networks were examined first, followed by those of the *Medium* and finally by those suggested by the *Relaxed* level networks. For each attribute the following two steps were executed.

a.1. If the attribute was suggested also by at least one of the other two filtering levels then it was selected and together with the attributes of the *Strict* evaluation set, formed the *Relaxed* evaluation set.

a.2. If the elements in the evaluation set reached *trunc[n/2]* the process was terminated. Otherwise, it continued with the rest of the attributes of step a. above.

In Stage Four validation experiments were carried out using the selected project attributes by the methodology, i.e., the *Strict* and *Relaxed* evaluation sets.

The proposed methodology (Papatheocharous and Andreou, 2012b), after the Box plot filtering (illustrated in Figure 4.7) that eliminated the extreme and mild performing ANN (considered as outliers), grouped the ANN according to the thresholds (specified in Table 4.12) in the *Relaxed*, *Medium* and *Strict* networks, each of which represented a superset of the next in the order they are mentioned (i.e., *RelaxedANN* ⊇ *MediumANN* ⊇ *StrictANN*). For the Desharnais dataset the *Strict* filtering level retained a total of 5 ANN, the *Medium* level 14 ANN and finally the *Relaxed* level 35 ANN. For the ISBSG R9-3 dataset, the *Strict* filtering level kept 5 ANN, the *Medium* level 28 ANN and lastly, the *Relaxed* level 69 ANN.

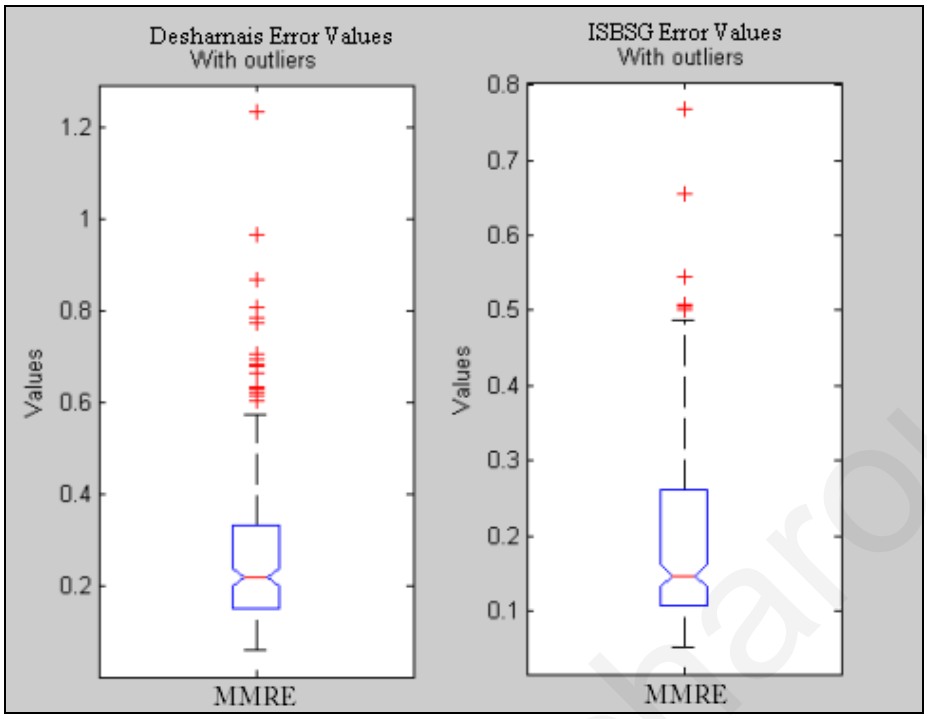


Figure 4.7: Outlying ANN identified by Box Plots based on their *MMRE* performance

The attributes selected for the Desharnais in the *Strict* evaluation set were the DU and SC, and in the *Relaxed* evaluation set the DU, SC and TR. For the ISBSG R9-3 dataset in the *Strict* evaluation set were the FS, AC and OC and in the *Relaxed* evaluation set were the FS, AC, OC, CC, DC, EC, IFC and RL. The attributes are listed in Table 4.13 that also juxtaposes the best and median results of ANN and Multi Linear Regression (MLR) models developed using all the attributes (*Initial/Original* case) and the selected 'significant' attributes (*Final case*) for comparison purposes (Papatheocharous and Andreou, 2012b).

In the Desharnais case the best ANN yielded *MMRE* equal to 0.062 and in the ISBSG R9-3 the best *MMRE* was equal to 0.052, which are very successful prediction results, but utilising in both cases all attributes. The results of the *Final* phase indicate that with the reduced attribute sets (*Strict* or *Relaxed* evaluation set) some increase in terms of the *MMRE* is observed. However, this performance decrease was expected considering the immense decrease in the number of attributes used in the evaluation subsets compared to the original experiments conducted.

Table 4.13: Indicative experimental results of the proposed methodology of ANN and ISA

Dataset	Phase/Subset	TESTING PHASE				#Selected attributes/Abbreviations
		Model[a]	***MMRE***	***CC***	***Pred(.25)***	
Desharnais (all: TE, ME, DU, TR, EN, FPA, SC, FPNA)	*Initial/Original*	8-13-1_{74}	0.105	0.987	0.933	8/all
		8-15-1_{199}	0.085	0.992	1.000	
		8-15-1_{87}	0.062	0.995	1.000	
		Median	0.214	0.931	0.933	
	Final/Strict evaluation set	2-6-1_{58}	0.133	0.964	0.933	2/DU, SC
		2-5-1_{146}	0.154	0.974	1.000	
		2-5-1_{23}	0.185	0.974	1.000	
		Median	0.396	0.806	0.933	
	Final/Relaxed evaluation set	3-8-1_{81}	0.130	0.971	1.000	3/DU, SC, TR
		3-6-1_{3}	0.230	0.895	1.000	
		3-5-1_{237}	0.245	0.957	1.000	
		Median	0.366	0.832	0.933	
	Initial/Original	MLR	0.880	0.897	0.350	8/all
	Final/Strict evaluation set	MLR	0.781	0.889	0.350	2/DU, SC
	Final/Relaxed evaluation set	MLR	0.779	0.901	0.350	3/DU, SC, TR
ISBSG R9-3 (all: FS, AFP, PET, PIT, RL, MTS, INC, OC, EC, FC, IFC, AC, CC, DC)	*Initial/Original*	14-16-1_{22}	0.052	0.989	1.000	14/all
		14-19-1_{172}	0.060	0.99	1.000	
		14-26-1_{145}	0.059	0.986	1.000	
		Median	0.150	0.955	1.000	
	Final/Strict evaluation set	3-8-1_{226}	0.131	0.973	1.000	3/FS, AC, OC
		3-7-1_{16}	0.132	0.953	0.955	
		3-8-1_{27}	0.138	0.965	0.955	
		Median	0.344	0.657	0.955	
	Final/Relaxed evaluation set	7-10-1_{109}	0.100	0.961	1.000	8/FS, AC, OC, CC, DC, EC, IFC, RL
		7-15-1_{150}	0.105	0.967	1.000	
		7-15-1_{57}	0.116	0.969	0.955	
		Median	0.281	0.776	0.955	
	Initial/Original	MLR	0.790	0.952	0.276	14/all
	Final/Strict evaluation set	MLR	1.083	0.581	0.172	3/FS, AC, OC
	Final/Relaxed evaluation set	MLR	0.979	0.284	0.167	8/FS, AC, OC, CC, DC, EC, IFC, RL

Model[a] *refers to the model employed. In ANN the topology is mentioned, i.e., x-y-1 refers to architecture with x nodes in the input layer, y nodes in the hidden layer and 1 output node. In MLR the Ordinary Least Regression method comprises the model. The subscript indexing scheme is used to differentiate experiments in the respective experiment repetition of the Core Cost Estimation component (refer to Figure 4.6) with similar ANN topologies but different training and testing sets.*

Since the performance figures of MLR were obtained after the data were transformed (using step (q) described in Table 4.1) and then transformed back to the original values in the reverse manner, while the ANN performance figures were obtained utilising the normalised values of the datasets (i.e., as specified in the pre-processing step (h) in Table 4.1) the results of the two models cannot be directly compared.

The graphical representation of the actual values versus the predicted values (transformed back to their original scale) of two indicative ANN is presented in Figure 4.8 and Figure 4.9.

The predicted samples are quite close (with very few exceptions), to the actual values in the testing phase of the forecasting process.

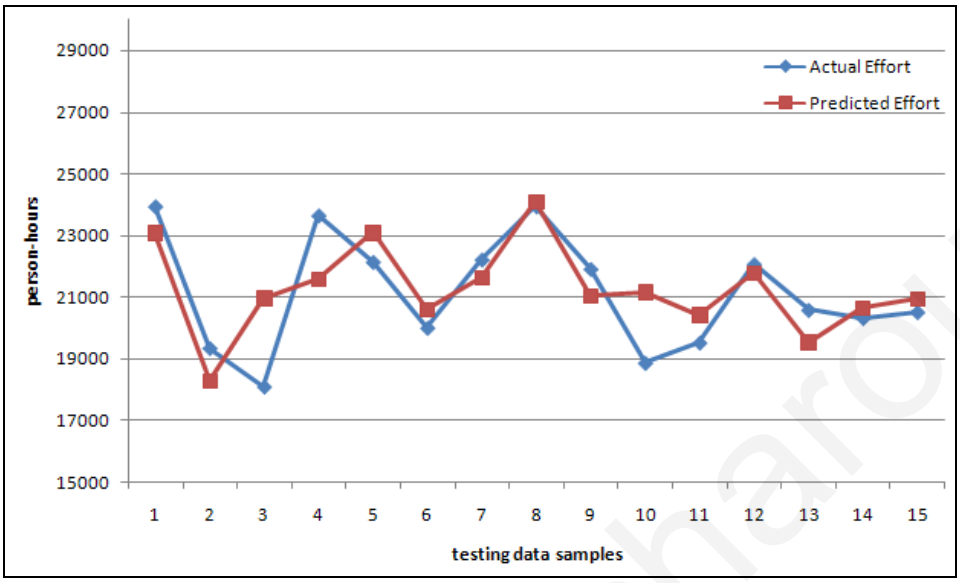


Figure 4.8: Partial data samples of Actual vs. Predicted Effort during validation (testing) experiments for the Desharnais dataset using a 3-8-1 ANN topology

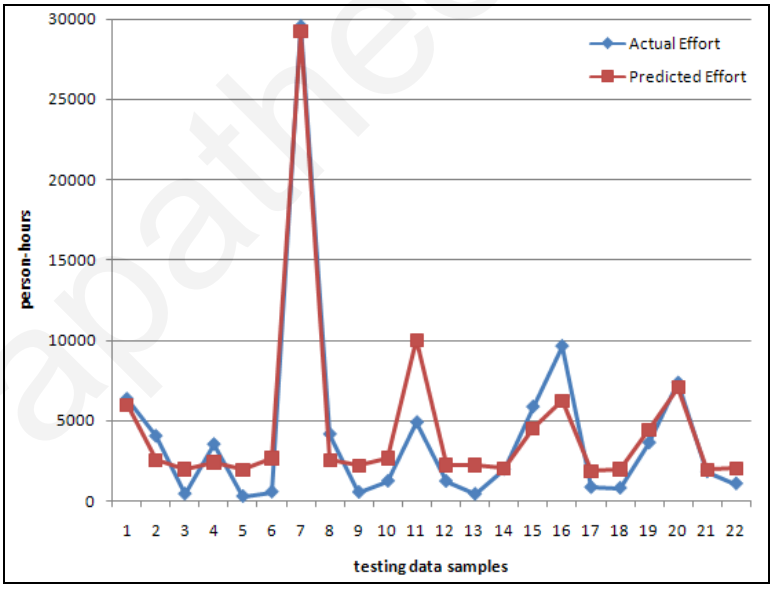


Figure 4.9: Partial data samples of Actual vs. Predicted Effort during validation (testing) experiments for the ISBSG R9-3 dataset using a 7-10-1 ANN topology

Since the groups of predictions obtained are independent, the non-parametric test of Mann-Whitney is usually employed to examine the chances of one population having greater observations over another population (Mann and Whitney, 1947). The null hypothesis in the Mann–Whitney test is that the two samples are drawn from a single population, and therefore

their probability distributions are equal. The alternative hypothesis is that the probability distribution of one sample is stochastically greater than the other. The test is used to evaluate and compare the results of two techniques. Therefore the Mann-Whitney test is performed to compare the significance of the results obtained using the *Original*, *Strict* and *Relaxed* evaluation sets.

The results of the Mann-Whitney test are listed in Table 4.14. The *U* value is the sum of the numbers of scores from the experimental group that are less than each of the control group scores (or the other way round), whichever gives the smaller value of *U*. The *z* is the associated *z*-approximation and the *p*-value refers to the significance value of the test, which gives the two-tailed probability that the magnitude of the test statistic is a chance result. The important part of the test is the significance value of the test, which gives the two-tailed probability that the magnitude of the test statistic is a chance result. Results are highly significant if $p<0.001$.

Comparing the results of '*Dataset A'* over '*Dataset B'*, the indication of the mean rank shows which one outperforms the other and the *p*-value indicates its significance. For example, the indication of mean rank '*Dataset A*' > '*Dataset B*' shows the outperforming results of '*Dataset A'* over '*Dataset B'* for all the set of predictions obtained.

From the experiments conducted, the *MMRE* error values obtained from the ANN with the *Original* set of attributes and the *Strict* or *Relaxed* evaluation subsets indicate that the former outperforms the rest. This means that the results obtained with the features selected do not outperform the experiments with the *Original* dataset, while the performance of the *Relaxed* evaluation set outperforms the *Strict* evaluation set. Since statistical significant loss of accuracy was observed for the reduced attribute models this might mean that the FSS may not be acceptable. In reality, however, project managers may well be willing to accept this level of reduction if the number of variables removed is sufficient, i.e., so that the overall costs and effort of data collection and analysis are reduced to a substantial extent. That is, managers may not be especially worried about statistical significance.

Table 4.14: Mann-Whitney signed-rank test results

Dataset	Subset: Model[b]	Mean Rank	U	z	p
Desharnais	*Original*: 8-15-1_{87} *Strict*: 2-5-1_{23}	*Original* > *Strict*	51	-2.551	0.011
	Original: 8-15-1_{87} *Relaxed*: 3-8-1_{81}	*Original* > *Relaxed*	56	-2.344	0.019
	Strict: 2-5-1_{23} *Relaxed*: 3-8-1_{81}	*Strict* < *Relaxed*	87	-1.058	0.290
ISBSG R9-3	*Original*: 14-26-1_{145} *Strict*: 3-8-1_{226}	*Original* > *Strict*	129	-2.652	0.008
	Original: 14-26-1_{145} *Relaxed*: 7-15-1_{150}	*Original*> *Relaxed*	196	-1.080	0.280
	Strict: 3-8-1_{226} *Relaxed*: 7-15-1_{150}	*Strict* < *Relaxed*	215	-0.634	0.526

Model[b] *refers to the model employed. In ANN the topology is mentioned, i.e., x-y-1 refers to architecture with x nodes in the input layer, y nodes in the hidden layer and 1 output node. The subscript indexing scheme is used to differentiate experiments in the respective experiment repetition of the Core Cost Estimation component (refer to Figure 4.6) with similar ANN topologies but different training and testing sets.*

Conclusively, the SCE accuracy obtained, using the *Final* subsets of selected attributes, is at lower levels compared to the results of the *Original* ANN models. However this loss of accuracy might be beneficiary, since some of the excluded project attributes may not be required to be measured, collected and maintained. Thus, the process of data collection might be substantially less complex, costly and time consuming. Practically, the proposed methodology achieved to minimise the number of independent attributes of ANN, emphasising on scheduling and sizing attributes for predicting effort.

The methodology described in the previous paragraphs that used Azoff's ISA (Azoff, 1994) to calculate the overall connection strength of each input to the output, did not consider any impact of the weights connecting the hidden nodes to the output, and this may be considered a limitation. According to the relative literature (Refenes et al., 1995; Belue and Bauer, 1995; Glorfeld, 1996; Olden and Jackson, 2002) there are other saliency measures of input variables that calculate the impact of the input vector on the output by using the whole set of connection weights between neurons (e.g., Garson's algorithm (Garson, 1991)). Thus the exploration of alternative methods for FSS implemented are summarised subsequently.

The next methodology implemented (Papatheocharous and Andreou, 2010) aimed to complement the previous investigations in two ways: (i) examine accuracy performance of ANN by gradually removing the least significant attribute, and (ii) compare one of the most

popular saliency measures of ANN weights, namely the value of *Relative Importance* (*RI*) (Garson, 1991) with the measures already utilised in the previous investigation (Papatheocharous and Andreou, 2012b). The logic of Garson's Algorithm is used for FSS because it is considered a good trade-off example between complexity and effectiveness. The motivation of the work conducted was that the contribution of the independent variables within the ANN models in SCE is rarely measured or taken into account, especially using complicated algorithms. Many techniques are proposed (Belue and Bauer, 1995; Glorfeld, 1996; Satizábal and Pérez-Uribe, 2007) but to the best of our knowledge none of them has been used.

The methodology followed the same rationale for producing various topologies of ANN MLP architectures, as explained in previous methodologies (i.e., starting with a topology which contains a number of neurons in the hidden layer equal to the number of attributes used as inputs in each experiment and continuing with topologies resulting from increasing the number of hidden neurons by one until their number becomes twice the size of the input attributes). The activation function used was the *hyperbolic tangent sigmoid transfer* (*tansig)* function (defined in eq. (4.7), pg. 105) in the input and hidden layer and the pure linear (*purelin*) function was used in the output layer. The initial weights and biases of each ANN were randomly set by the Nguyen-Widrow initialisation method (Nguyen and Widrow, 1990). The same pre-processed datasets of Desharnais and ISBSG R9-3 summarised in Table 4.9 and Table 4.11 were used.

The datasets were randomly divided into three subsets, training, validation and testing, with the percentages of 60%, 20% and 20% of the total available samples respectively, where each sample participated in only one subset. The scaled conjugate gradient training function was used which is based on the derivative functions of weights, net inputs and transfer functions (*trainscg*). The learning process used was based on the gradient descent with momentum weight and bias learning function (*learngdm*). The performance function was the *Mean Squared Error* (*MSE*) and the number of epochs, learning rate and momentum coefficient were set to 1000, 0.01 and 0.9 respectively. The training process was repeated 10

times so that the optimal network that minimises the prediction error (in terms of *MMRE*) was identified and the weights of each input-hidden-output path were stored for further processing by Garson's Algorithm. The experiments conducted involved the gradual evaluation of ANN inputs (Papatheocharous and Andreou, 2010). Particularly, the ANN's performance was traced as the number of input attributes was gradually reduced by one. The reduction was based on the rank of the inputs regarding the *Relative Importance* (*RI*) (refer to Table 3.4, pg. 78) and was performed until the inputs were reduced to half the initial size. Moreover, the attribute that would be removed according to the *Input Strength* (*IS*) (refer to Table 3.5, pg. 79) utilised in the previous experiments was also estimated.

Table 4.15 lists the inputs that made the smallest contribution to the final output of the indicative ANN experiments (as this was reflected through the weight connections). Even though each experiment of ANN evaluation was repeated 10 times to investigate the stability of the technique, here, only 5 indicative experiments with their specific experiment-id (Exp.Id) are reported. The complete experiments are provided in Appendix B (Table B. 12 and Table B. 13).

Table 4.15: Indicative experiments of backward attribute elimination using ISA on ANN

Dataset	Exp. Id	Importance Measure	Order of Attributes Removed	Initial *MMRE*	Initial *Pred(.25)*	Final *MMRE*	Final *Pred(.25)*
Desharnais	4	*RI*	TE,TR,FPA,SC	0.364	1.000	0.387	1.000
		IS	TE,FPNA,FPA,SC				
	5	*RI*	ME,FPA,TE,DU	1.264	0.867	1.060	0.867
		IS	ME,SC,TE,DU				
	6	*RI*	TE,SC,DU,ME	0.386	0.933	0.346	0.933
		IS	TE,SC,DU,ME				
	7	*RI*	ME,SC,FPNA,FPA	0.569	0.800	0.512	0.800
		IS	ME,SC,FPNA,FPA				
	10	*RI*	TE,SC,EN,DU	0.312	0.933	0.293	0.933
		IS	TE,SC,EN,DU				
ISBSG R9-3	3	*RI*	INC,CC,DC,RL,OC,AC,FC	0.255	0.955	0.265	0.909
		IS	INC,CC,DC,RL,OC,FC				
	4	*RI*	MTS,CC,RL,INC,FC,FS,AC	0.199	0.955	0.191	0.955
		IS	MTS,CC,RL,AFP,OC,FS,AC				
	7	*RI*	CC,OC,RL,FC,INC,AFP,IC	0.377	0.909	0.257	0.909
		IS	CC,OC,RL,FC,INC,AFP,IC				
	9	*RI*	FC,IC,RL,AFP,AC,EC,PIT	0.662	1.000	0.500	1.000
		IS	FC,IC,RL,AFP,AC,EC,PIT				
	10	*RI*	FS,IC,PIT,OC,AFP,RL,FC	0.249	0.955	0.270	0.955
		IS	FS,IC,PIT,RL,AFP,RL,FC				

The backward input elimination methodology showed that in most cases the prediction accuracy in terms of error metrics comparing the *Initial* (using all the attributes) with the *Final* (using only the selected attributes, i.e., after removing the attributes mentioned in the column "Order of Attributes Removed" of the Table 4.15) was positively affected, meaning accuracy was increased. In the Desharnais case the highest accuracy improvement obtained in testing was 20% and the highest accuracy decrease obtained was 16%. In the ISBSG R9-3 dataset the highest accuracy improvement during the testing phase was 16% and the highest accuracy decrease obtained was 24%. The experimental results in Table 4.15 imply that since the *Pred(.25)* values are very high (near the unit) while the *MMRE* values reported contain some degree of variability, there were only a few poor predictions per experiment.

Throughout the experiments similar or consistent attributes were selected to being the 'weakest' effort contributors. These were in the Desharnais case TE, ME, DU and SC and in the ISBSG R9-3 case RL, FC, CC, INC and OC. Therefore, the most influential attributes in the Desharnais dataset were the TR, EN, FPA and FPNA and in the ISBSG R9-3 dataset the FS, AFP, PET, PIT, EC, INF, AC, DC and MTS. Also, comparing the attributes filtered out by both measures of weakness (*RI* and *IS*) only in very few exceptional cases the attribute exclusion would not be in agreement (these exceptional cases were underlined in Table 4.15). Therefore, comparing the ISA methodologies utilised thus far for FSS, the most recent ISA applied (based on Garson (1991)) provided the best prediction accuracy with the yielded 'significant' features. In terms of simplicity and practicality all methods are similar and as shown from the results of Table 4.15 agree to a high degree on the promoted attributes. Some attributes however in the Desharnais case, such as DU and SC, have appeared in the previous ISA methodologies as significant attributes whereas in the last method they are listed among the weakest contributors, which implies that a vulnerability issue exists regarding the method utilised, dataset and data partitions used to select features.

Summarising the CI-based FSS techniques on ANN applied thus far, we conclude that all of the aforementioned variations of ISA have in common the measurement of saliency measures of network weights, which provide insight on the interpretation of the ANN and

'illuminate' their 'black-box' nature. Moreover, they all represent feasible and simple solutions for ISA. ISA may be used for obtaining the interactions between the variables of a complex environment and effectively evaluating the contribution of each variable on the prediction result. In addition, the ISA methods described provide different levels of empirical forms of experimentation, since various thresholds and processes are followed. The ISA variations may be also compared in terms of simplicity, practicality and effectiveness.

Slight deterioration in terms of performance accuracy in SCE was usually obtained using the features selected for prediction. However, only in the Garson's Algorithm case of ISA the prediction results were slightly improved compared to the rest. The main advantage of the techniques developed and results obtained is that it may disengage the SCE process from the hindering and extremely time and effort consuming process of measuring values for a large set of metrics. The features commonly selected by all the ISA techniques applied on ANN were for the Desharnais dataset related with the number of transactions (TR) and size-based metrics (FPA and FPNA) and for the ISBSG the attributes the functional size (FS) and, since the projects were measured using the IFPUG standard (ISO/IEC 20926, 2003), with the number of additions (AC) and enquiries (EC) of unadjusted FP. The AC is related to the count of new or added functions and EC to the count of external enquiries, i.e., the number of reports created by the applications and where the report does not include any derived data.

The consideration of the aforementioned variables as most significant indicates that the basic components of software size, number of transactions and enquiries performed by the software, account to a decisive degree, for the amount of effort to develop the software. Moreover, the unadjusted FP metric of software change to the initial software specifications (i.e., adding new functions) has a decisive effect on effort. This however raises a question regarding the availability of such estimates in a development phase where it is also useful, in terms of enabling early SCE. Thus, making changes after the specification of the requirements at the initial phases seems to add a considerable burden to the overall effort accounted during the development process. This finding is rather rational for traditional software development processes where the cost of changing specifications at the design and implementation phases

increases as development proceeds to later phases. It is important to also emphasise that these attributes related to FP software size, number of transactions and reports can be measured at the early project phases, in the sense that at least a rough estimation may become available at the beginning of the development process.

4.2.2.2 Ridge Regression (RR) and FSS-SCE

A significant limitation of ANN is related to the type of attributes the structures can accommodate. In SCE many cost factors are of nominal nature and were in the aforementioned experiments with ANN disregarded. Since the descriptive aspects of software might considerably reflect on the amount of development effort, the models of SCE are extended to investigate techniques that may utilise many-type of data. With the appropriate pre-processing (described in this section) a range of models based on Ridge Regressions (RR) are developed that may handle nominal and numerical data types and investigate a wide range of CI and non-CI techniques for Feature Subset Selection approaches is SCE (FSS-SCE).

The approach aimed initially to further investigate classical techniques, such as Regressions, and particularly for selecting the appropriate set of features for SCE. However, some initial investigations on Multiple Linear Regression (MLR) based on the Least Squares technique, (Papatheocharous and Andreou, 2012b) indicated quite mediocre curve fitting and accuracy performance in approximating effort.

RR is considered an improvement of the classical Least Squares technique, which is one of the dominant methods applied in SCE and one of the most widely used algorithms in cases where there is high correlation among predictor variables. RR is considered a promising solution for addressing the abovementioned issues and it has also been already successfully applied in the area of software engineering by various researchers yielding promising results. Specifically, RR has been used to estimate the coefficients for the COCOMO model (Nguyen et al., 2008), to produce classification scores and remove unnecessary features (Parsa et al.,

2008) and to improve the performance of regressions on multi-collinear datasets (Li et al., 2010).

Nevertheless, previous SCE research did not investigate systematically or to any significant extent the issue of identifying the optimum attribute subset for estimating effort more accurately. The models proposed in this thesis involve nine different Feature Subset Selection (FSS) approaches which are combined with RR. Particularly, RR is used as the evaluator and the FSS techniques are used as filtering methods (Papatheocharous et al., 2010b). Selecting the most informative subset of features from a pool of available cost drivers stemmed from the hypothesis that reducing the dimensionality of datasets will significantly minimise the complexity and time required to reach to an estimation using any particular modelling technique. Especially, using a search approach for selecting the most appropriate or relevant set of attributes in high dimensional datasets, i.e., datasets with a large number of available attributes is a tedious task. The exhaustive search of all possible subsets of features is impractical and extremely time-consuming but produces the most predictive features (Kirsopp and Shepperd, 2002; Azzeh et al., 2008). Based on previous investigations, in data-intensive estimation methods, such as ANN or Regressions, a significant task is the pre-processing performed to remove irrelevant data which may lead to less complex and equal or more accurate effort approximations. Moreover, accurate SCE are not the result of a purely blind process that takes any number of inputs found in empirical software databases and outputs work effort, but seem to be influenced by various factors to different degrees.

Therefore, the models proposed in this work investigate effort estimation via various FSS approaches for reducing the number of cost drivers required in successful SCE, under the assumption that some cost drivers are more informative than others. The main goal of the methodology was to assess the appropriateness of the available cost drivers for SCE and investigate the number and type of attributes selected by various FSS approaches.

Typically, feature selection approaches may belong to one of three categories: Wrappers, Filters or Embedded algorithms. Wrappers utilise the machine learning algorithm as a black box to rank feature subsets under examination according to their accuracy prediction. Filters,

as a pre-processing step, filter the feature subsets independently of the machine learning algorithm. Embedded methods are included as part of a specific machine learning technique and through training they provide subset selection for the specific technique.

The following FSS approaches were examined: *Filter approaches* (i) Backward StepWiseFit (BSWF) with RR, (ii) Forward StepWiseFit (FSWF) with RR, (iii) Backward Feature Elimination with Least Squares (LSBFE), (iv) Forward Feature Selection with Least Squares (LSFFS), (v) Genetic Algorithm with Least Squares (LSGA), and (vi) Backward Feature Removal with Artificial Neural Networks, Garson's Relative Importance (BANN) with RR; *Wrapper approaches* (combined with RR) involved (vii) Forward Feature Selection (FFS), (viii) Backward Feature Elimination (BFE) and finally, (ix) Genetic Algorithms (GA).

The datasets of Desharnais (filtered in the same way as described before, i.e., applying the pre-processing steps described in Table 4.1 (a), (b) and (m)) and ISBSG R9 were used. The 77 projects were described by the attributes summarised in upper part of Table 4.9. The attributes of ISBSG R9 went through the pre-processing steps of Table 4.1 (a), (c), (f), (k)-(m). The filtered dataset, namely ISBSG R9-4 comprised 467 projects and 82 attributes and is summarised in Table 4.16.

Table 4.16: ISBSG R9-4 attributes description

Abbreviation	ISBSG R9-4 Attribute Description	Number of values
CA1-4	Count Approach	4
AFP	Adjusted Function Points	1
PET	Project Elapsed Time	1
IY	Implementation Year (extracted from Implementation Date)	1
DTY1-4	Development Type	4
OT1-12	Organization Type	12
DT1-15	Development Technique	15
FST1-3	Functional Sizing Technique	3
DP1-5	Development Platform	5
LT1-6	Language Type	6
PPL1-11	Primary Programming Language	11
DBS1-9	Database System	9
RM1-7	Recording Method	7
RL	Resource Level	1
MTS	Max Team Size	1
ATS	Average Team Size	1

For each FSS approach the same data partition was used. Particularly, 10 fold cross-validation was used (Maimon and Rokach, 2005), where 80% of the total projects were

allocated to the training set and the remaining 20% to the testing set. The training samples were used to select the optimum features with each technique and then evaluation of the optimum features was carried out with the testing samples. Specifically, in the training phase the 10 fold cross-validation procedure split the training set into 10 parts of almost equal size and applied the FSS approaches (with only the selected cost drivers) 10 times, each time evaluating its performance on one part after training on the remaining nine. Performance evaluation included calculating the *MRE* for each fold and then the *MMRE* over the whole set was calculated (i.e., the mean value of the *MRE*s of all projects) at the end of the process.

Initially, Stepwise Regression was utilised both in a backward (BSWF) and a forward (FSWF) manner, i.e., in the former case starting with a full attribute set, attributes were removed if the prediction was improved in terms of *MMRE* and in the latter case starting with an empty set of attributes, attributes were added again if the prediction was improved. Figure 4.10 shows the generic logic of the algorithm followed.

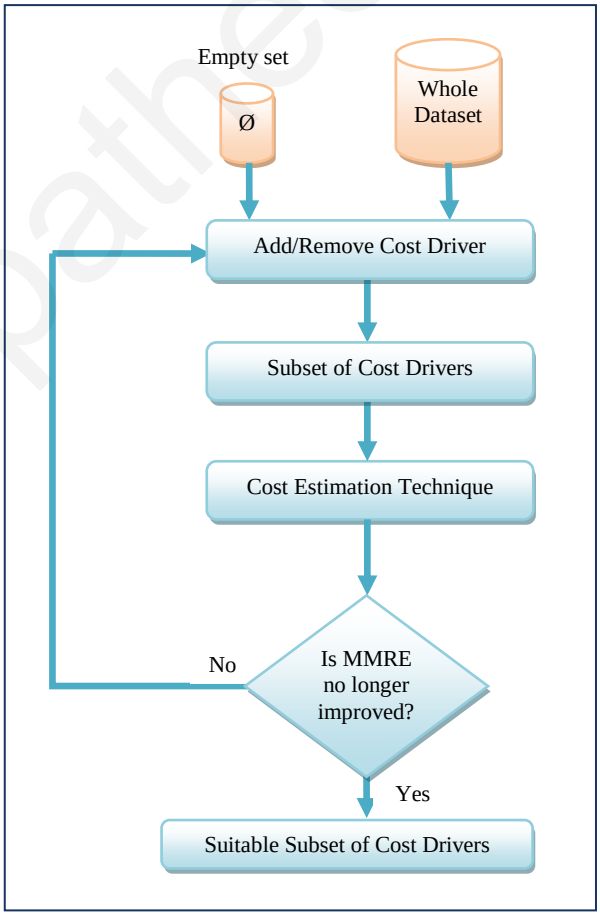


Figure 4.10: Stepwise selection of project attributes

Matlab's function *stepwisefit* (Matlab R2009a) was utilised in which, the null hypothesis is defined as "*if a variable is not currently included in the model it would have a zero coefficient if it was added to the model*" (Beale et al., 2011). If there is sufficient evidence to reject the null hypothesis, then the variable is added to the model. Conversely, the null hypothesis is that "*if a variable is currently in the model, the variable has a zero coefficient*". If there is insufficient evidence to reject the null hypothesis, then the variable is removed from the model as the coefficient is zero. The method continues with the following steps:

- *Step 1*: Fit the initial model.
- *Step 2*: Add the variable that has the smallest *p-value*, from the candidate variables that do not currently participate in the model and have *p-values* less than an inclusion threshold (that is, it is unlikely that they would have zero coefficient if they were added to the model) and repeat this step; otherwise, go to step 3.
- *Step 3*: Remove the variable that has the largest *p-value*, from the variables included in the model that have *p-values* greater than a removal threshold (that is, it is unlikely that the hypothesis of a zero coefficient can be rejected) and proceed to step 2; otherwise, terminate.

The method terminates when no single step improves the model (Draper and Smith, 1998). The maximum *p-value* for a predictor to be added was set to 0.05 and the minimum *p-value* for a prediction to be removed was set to 0.10.

Basically, the aforementioned method examines each variable and its value to be entered in a model in a stepwise sequential process. If adding the variable contributes to the model then it is retained, but all other variables in the model are then re-tested to see if they are still contributing to the success of the model. If they no longer contribute significantly to the success of the model they are removed. Thus, the method ensures that we end up with the minimal set of predictor variables included in the model.

In addition, two forward selection and backward elimination algorithms were implemented both as Filter combined with linear Least Squares (LSFFS and LSBFE respectively) and as Wrapper combined with Ridge Regression methods (FFS and BFE).

Also, a dedicated Genetic Algorithm (GA) was developed for feature selection in MatlabR2009a using the Genetic Algorithm and Direct Search Toolbox (MathWorks, 2009). The algorithm is an extension of a preliminary work (Papatheocharous et al., 2010c) which had the target of employing a GA for identifying the optimal subset of cost drivers participating in the cost estimation process of the dual variables RR technique. The dedicated algorithm had the following goal: Select the subset of cost drivers that minimises the overall relative error rate obtained with two machine learning techniques, in this case Ridge Regression (RR) and Least Squares (LS). Thus, two versions of the algorithm were implemented called GA and LSGA. The main advantage of using Evolutionary Algorithms (EA) is that they can search the vast space of possible combinations of cost drivers efficiently and reach to a near-to-optimal outcome.

The GA evolved a population of individuals encoded as bit strings of the size of the cost drivers contained in the dataset. The cost drivers represented by the bits set to 1 were taken into account as inputs, while all the others (set to 0) were not. The individuals at each step, called a generation, were evaluated in the same 10 fold cross-validation process previously described. For each individual we assigned a score, called its fitness value, indicating how good the solution it represented was. The fitness of each individual defined its likelihood of being selected for the next generation. Until a new generation was completed, individuals from the current generation were selected probabilistically based on their fitness to generate offspring for the new generation. There were also a few individuals, the fittest ones, which were carried to the new generation unchanged, that is, without the application of any genetic operation on them. The same process was repeated until an optimal solution was reached or a stopping criterion was met, which in many cases was a maximum number of generations. The solution represented by the fittest individual in the last population was the one adopted as the resulting solution of the algorithm.

Firstly, to create each generation, the 10% fittest individuals were isolated and placed in the new generation. Then, until the new generation was complete (that is, it reached the maximum size) individuals were selected and recombined and/or mutated to generate new individuals for the new population. The selection of individuals was performed with the stochastic uniform function applied on the rank of the individuals in the current population. In effect this selection function laid out a line in which each individual corresponded to a section of the line of length proportional to its rank. The function then moved along this line in steps of equal size selecting the individual from the section it landed on. The size of the first step was a uniform random number less than the fixed step size.

The crossover operator used the uniform function, with probability of being applied to a selected pair (called crossover rate) being set to 0.8. This function creates a random bit string of the same size as the two parents, called crossover mask, and generates the first child by copying the parts of the first parent at the points where the crossover mask has a 1 and the parts of the second parent at the points where the crossover mask has a 0. For the second child the same process was repeated with the parents reversed. Also, the uniform mutation was used, which flips each bit of an individual with a given probability, which in this case was set to 0.01. The number of generations and individuals in each generation were both set equal to 100.

Finally, the methodology explained in section 4.2.2.1, pg.136-138 using ANN and Input Sensitivity Analysis was followed to remove the 'weakest' attribute (identified by the *Relative Importance* using Garson's Algorithm) from the sample until the inputs were reduced to half the initial size. The approach is called (BANN) as backward feature elimination was performed through ANN.

An initial investigation of RR on the full Desharnais and ISBSG R9-4 datasets (Papadopoulos et al., 2009) showed that the following algorithm configurations were the most appropriate and thus were applied in the experiments conducted: the *ridge parameter* α and the *kernel gamma* γ were set to 0.1 and 3.5 for the ISBSG R9-4 dataset, and to 0.05 and 5 for the Desharnais dataset respectively. In order to evaluate the quality of the selected feature

subsets, and therefore their appropriateness, we compared their accuracy performance to that of the full feature set using all the aforementioned specific estimation techniques (RR or LS).

The results obtained using the various FSS approaches in the Desharnais dataset are summarised in Table 4.17 and in the ISBSG R9-4 dataset in Table 4.18 (Papatheocharous et al., 2010b). For the complete experimental results refer to Appendix B (Table B. 14 and Table B. 15). The column #F indicates the number of selected features by each technique. The fact that each technique resulted in the selection of a different cost driver subset, with some cost drivers appearing in the majority of subsets and others appearing very rarely or not at all, triggered us to examine both the quality of the different subsets as well as their similarities. Thus, the final column in Table 4.17 and Table 4.18 list the features selected by all (i.e., 100%) and by most (i.e., 80% of the executions of each method over the 10 data partitions.

Table 4.17: Software cost estimations across various FSS with the Desharnais dataset

FSS	RESULTS	TESTING PHASE				#F	Common Features Selected in a) 100% and b) 80% runs	
		INITIAL		FINAL				
		MMRE	PRED	MMRE	PRED			
BFE	MIN	0.334	0.400	0.337	0.333	4	a)	None
	MAX	0.815	0.267	0.916	0.267	5	b)	DU, EN, SC
	AVG	**0.582**	**0.387**	**0.622**	**0.347**	**4**		
FFS	MIN	0.334	0.400	0.337	0.333	4	a)	FPNA
	MAX	0.815	0.267	0.903	0.267	6	b)	DU, EN, SC, PNA
	AVG	**0.582**	**0.387**	**0.622**	**0.340**	**5**		
BSWF	MIN	0.334	0.400	0.345	0.400	4	a)	DU
	MAX	0.779	0.333	0.859	0.333	2	b)	DU, TR, FPNA
	AVG	**0.582**	**0.387**	**0.623**	**0.413**	**4**		
FSWF	MIN	0.387	0.467	0.416	0.400	2	a)	None
	MAX	0.815	0.267	0.911	0.533	3	b)	TR, FPNA
	AVG	**0.582**	**0.387**	**0.653**	**0.413**	**3**		
LSBFE	MIN	0.389	0.400	0.377	0.400	4	a)	None
	MAX	1.264	0.067	1.170	0.200	3	b)	None
	AVG	**0.648**	**0.293**	**0.699**	**0.320**	**3**		
LSFFS	MIN	0.339	0.400	0.388	0.333	2	a)	None
	MAX	1.264	0.067	1.096	0.267	3	b)	TR, FPNA
	AVG	**0.648**	**0.293**	**0.639**	**0.333**	**3**		
LSGA	MIN	0.387	0.467	0.416	0.400	2	a)	None
	MAX	0.779	0.333	0.867	0.267	3	b)	FPNA
	AVG	**0.582**	**0.387**	**0.626**	**0.320**	**3**		
GA	MIN	0.334	0.400	0.337	0.333	4	a)	None
	MAX	0.779	0.333	0.836	0.400	5	b)	DU, EN, SC
	AVG	**0.582**	**0.387**	**0.614**	**0.327**	**4**		
BANN	MIN	0.334	0.400	0.382	0.333	4	a)	None
	MAX	0.779	0.333	0.792	0.200	4	b)	FPA
	AVG	**0.582**	**0.387**	**0.606**	**0.447**	**4**		

Table 4.18: Software cost estimations across various FSS with the ISBSG R9-4 dataset

FSS	RESULTS TESTING PHASE					#F	Common Features Selected in a) 100% and b) 80% runs	
		INITIAL		FINAL				
		MMRE	PRED	MMRE	PRED			
BFE	MIN	0.434	0.398	0.456	0.390	34	a)	AFP, PET, OT12, FST3, RM4, MTS, ATS
	MAX	0.657	0.415	0.851	0.423	36	b)	AFP, PET, OT4, OT9, OT12, DT1, DT3, FST3,
	AVG	**0.585**	**0.349**	**0.592**	**0.377**	**32**		LT2, PPL2, RM4, MTS, ATS
FFS	MIN	0.434	0.398	0.439	0.455	33	a)	AFP, PET, DT1, LT3, RM4, MTS, ATS
	MAX	0.657	0.415	0.771	0.463	29	b)	AFP, PET, OT9, OT12, DT1, DT3, DT5, DT7,
	AVG	**0.585**	**0.349**	**0.577**	**0.382**	**29**		LT2, LT3, PPL2, RM4, MTS, ATS
BSWF	MIN	0.434	0.398	0.434	0.447	32	a)	CA1, CA2, AFP, PET, RM4, MTS, ATS
	MAX	0.763	0.309	0.880	0.407	30	b)	CA1, CA2, CA3, AFP, PET, OT8, LT1, PPL2,
	AVG	**0.585**	**0.349**	**0.618**	**0.386**	**27**		RM4, MTS, ATS
FSWF	MIN	0.434	0.398	0.425	0.447	11	a)	AFP, PET, LT2, RM4, MTS
	MAX	0.763	0.309	0.817	0.382	11	b)	AFP, PET, LT2, LT3, PPL2, RM4, MTS, ATS
	AVG	**0.585**	**0.349**	**0.600**	**0.373**	**12**		
LSBFE	MIN	0.434	0.398	0.399	0.439	39	a)	AFP, PET, MTS, ATS
	MAX	0.815	0.317	0.869	0.301	27	b)	AFP, PET, DTY2, DT5, DT12, LT1, DBS4,
	AVG	**0.585**	**0.349**	**0.592**	**0.372**	**32**		RM2, RM4, MTS, ATS
LSFFS	MIN	0.434	0.398	0.447	0.407	37	a)	AFP, PET, LT3, RM4, MTS
	MAX	0.815	0.317	0.812	0.350	31	b)	CA4, AFP, PET, IY, DT1, DT12, FST1, LT2,
	AVG	**0.585**	**0.349**	**0.596**	**0.363**	**34**		LT3, LT5, PPL2, RM4, MTS, ATS
LSGA	MIN	0.434	0.398	0.432	0.407	36	a)	AFP, PET, LT2, RM4, MTS
	MAX	0.763	0.309	0.867	0.374	29	b)	CA4, AFP, PET, DT5, DT12, FST1, LT2, LT5,
	AVG	**0.585**	**0.349**	**0.607**	**0.355**	**31**		PPL2, DBS2, RM4, MTS, ATS
GA	MIN	0.434	0.398	0.445	0.415	35	a)	AFP, PET, DT1, DT3, PPL2, RM4, MTS, ATS
	MAX	0.815	0.317	0.748	0.382	30	b)	AFP, PET, OT4, OT12, DT1, DT3, DT5, LT2,
	AVG	**0.585**	**0.349**	**0.570**	**0.376**	**33**		LT3, PPL2, PPL8, DBS1, RM4, MTS, ATS
BANN	MIN	0.434	0.398	0.478	0.325	41	a)	None
	MAX	0.815	0.317	0.954	0.317	41	b)	PET, OT11, DBS1, DBS8, DBS9
	AVG	**0.585**	**0.349**	**0.686**	**0.320**	**41**		

Comparing the initial error figures obtained using the full attribute set with the final errors obtained using the optimum feature subset of each method on average a minor accuracy shift is observed in both datasets. This means that a large subset of software cost drivers has a small or insignificant influence in estimating software development effort accurately. Therefore, feature selection for the specific datasets has been found particularly useful for maintaining accuracy and in some cases (i.e., LSBFE, LSFFS in the Desharnais dataset and FSS, FSWF, LSBFE, GA in the ISBSG dataset) even increasing performance. The best selection was obtained with the BANN and the GA feature selection method in the Desharnais and the ISBSG case respectively, among the ones examined, as it yielded the lowest *MMRE* on average in the testing phase. It is also interesting to point out that the FSWF method promoted the smallest in size subset of features among the methods investigated in the ISBSG case and more importantly, it was able to provide highly accurate development effort

estimations, similar to the rest of the FSS methods; FSWF on average selected around less than half the original features. Considering the average number of optimal features obtained from all the techniques explored, there are 4 only for the Desharnais dataset and 30 for the ISBSG R9-4 dataset. The average number of features is estimated by the average (AVG) number of features selected by each technique. In both datasets there was a quite important attribute reduction achieved without any significant compromise on the prediction accuracy of the SCE model based on the RR technique.

In general, we observe that the commonly promoted attributes of all FSS methods presented in the last column of Table 4.17 and Table 4.18 are quite similar in the first case (i.e., the 100% case). This picture, though, is not retained in the 80% case probably due to the fact that more attributes are allowed to enter the pool of significant attributes as the threshold is less strict than the 100% case. In the Desharnais dataset, there are no attributes consistently promoted by all random splits (100% case) and the various FSS methods. However, there are a few attributes more consistently promoted than the rest (in 80% of the data splits) and these were FPNA and DU. Also, in some of the random data splits there were a few less frequently promoted attributes, namely TR, EN and SC. The significant attributes that seem to have direct influence on development effort in the Desharnais case also relate with software size and project duration. In the ISBSG case the attributes promoted in all of the 10 random data splits and all FSS methods, except BANN's filtering, are the following: AFP, PET, RM and MTS. Additionally, the attributes ATS, LT and DT are also considered important by most of the methods. Even though the features selected in the pre-processing stage comprise the most relevant attributes found in the vast ISBSG dataset, it seems that particularly the size of software and the duration of the project significantly affect effort estimates as they are consistently promoted. These two factors, as expected, drive development effort value and probably, if used as inputs in SCE models could lead to more accurate results. The comparison of the results obtained showed consistency among the results of the FSS methods, while the findings are in agreement with other related research studies that also identified significant project features on the same datasets (Azzeh et al., 2008; Li et al., 2009a; Keung et

al., 2008). In addition, an important finding is that a rather small number of features is actually required as input for the particular SCE technique investigated to yield successful results. Therefore, the subset of features leads in reducing the model's complexity and training time.

Summarising the observations obtained through the investigation of various FSS approaches in SCE described in this section, there are some commonly selected attributes in each dataset. This observation was more obvious in the case of the Desharnais dataset where the same pre-processing steps were performed before the application of each approach, whereas in the case of the ISBSG dataset this observation was harder to make. However, in both datasets size-related features were commonly selected, i.e., the FPNA and FPA attributes from the Desharnais dataset and the FS and AFP attributes from the ISBSG dataset. This direct effect and contribution of the size of the software on the effort required to develop it was presupposed in the previous section 4.2.1, where size-based variables were investigated for effort estimation.

The experiments showed that in single company datasets, such as the Desharnais, apart from FP-related variables (FPNA and FPA) which were found to significantly affect effort, the attributes of project duration (DU), scope (SC), number of transactions (TR) and entities (EN) were also considered influential to the value of effort. Factors related with human features such as the team's and manager's experience were not found significant and moreover, factors related to the domain experience, team size, application type, technologies utilised etc. were not measured in the Desharnais dataset possibly because among the projects reported these factors remained unaltered in the context of the same company. However, in the FSS experiments conducted with multi-organisational datasets where projects from many companies are reported, such as the ISBSG, apart from the functional size (FS) and the AFP which as expected were found to influence effort, other factors of technical nature were identified as 'significant'. These factors relate for example to properties that change among different projects developed by dissimilar organisations such as the developers' team size (MTS), the development type (DT), the language type (LT) used, etc. Finally, the existence of

such dissimilarities, especially in large datasets calls for the need of clustering and classification techniques for conducting SCE on more homogeneous data samples. The following section deals with this matter.

4.2.3 Clustering and Classification for Software Cost Estimation (CC-SCE)

The Clustering and Classification methodologies for SCE (CC-SCE) proposed in this section aim to complement the previously mentioned research work that emphasised attribute selection (Papatheocharous et al., 2010b), and filter software data based on explicit project characteristics. Thus, the Clustering and Classification methodologies intend to approximate the issue of SCE by performing horizontal (attribute-based) and/or vertical (project-based) filtering. The motivation is that the successful clustering and/or classification of past project data into homogeneous clusters and/or classes may provide better cost estimates within each cluster and/or class. Moreover, fuzzification and cost estimation within value ranges are executed, that transform the amount of the underlying uncertainty in software project data into useful information that can lead to better software cost estimations. The latter is considered an extension to the typical clustering and classification approaches to Fuzzy clustering and/or classifications for SCE. In addition, the estimations are based on predictive intervals and are reported as interval instead of a crisp estimate (a subject which is also addressed in a later section 4.2.4 (pg. 203) namely with the Predictive Intervals for SCE (PI-SCE)).

Initially, clustering methodologies are proposed by employing genetically evolved bounds of cost attribute value ranges are created based on Conditional Sets (CS) theory (Packard, 1990) to cluster software data and investigate software cost estimation performance. Moreover, an Entropy-based Fuzzy *k*-modes clustering algorithm, to identify clusters of similar projects that are sufficiently close to each other. Then, the descriptive characteristics (i.e., cost drivers' values) of the projects are used for classifying each new project in a certain cluster.

The classification methodologies of software projects are developed based initially on Genetic Programming (GP) for investigating genetically evolved typical algorithmic models and then Fuzzy Decision Trees (FDT) and Fuzzy Implication Systems (FIS) are employed. The main target of the methodologies was to classify project data and, at the same time, extract association rules between software projects that describe the multifaceted nature of the software development environments. The methodologies developed conceptually interpret the subjective, encrypted information, especially coming from large heterogeneous datasets, in such a way that can be comprehensively understood by individuals. Classification of software project data aims to produce and examine groups of projects that share similar characteristics and obtain association rules that can be used to perform improved cost estimations. Each association rule isolates a number of project samples which satisfy a number of cost parameters in terms of similar values, thus forming clusters of known project data. In case a new (under development) project is properly classified in a cluster (i.e., it satisfies the corresponding rule) then the estimated effort value can be expressed in terms of the mean value and standard deviation of the group's (clustered projects') effort. Alternatively, a group of association rules can be used to build a decision support Fuzzy Implication System (FIS) aiming to provide improved SCE.

4.2.3.1 Genetically Evolved Conditional Sets (CS) for CC-SCE

The methodology proposed in this section targets clustering and is based on the combination of the theory of Conditional Sets (CS) with Genetic Algorithms (GA) (Andreou et al., 2007). The idea was inspired by (Meyer and Packard, 1992; Packard, 1990) that employed Evolutionary Algorithms (EA) to improve CS. The main philosophy was to evolve value ranges (or CS) that describe the determinant relationships among project attributes and effort in a given dataset. This entails exploring a vast space of solutions expressed in ranges. These ranges utilise values that are within the range of values included in available software project data located in benchmark datasets. The ranges produced may be used to replace the

crisp/actual data values from the datasets which usually contain high levels of uncertainty and lower the dependency of the SCE process on the quality of the data gathered.

The term Conditional Sets (CS) refers to a set of boundary conditions. Some definitions and notations of the CS theory (Adamopoulos et al., 1998; Packard, 1990) are provided in the context of the SCE paradigm. Consider a set of *n* cost attributes $\{A_1, A_2, ..., A_n\}$, where each A_i has a corresponding discrete value x_i. A software project may be described by a vector of the following form:

$$L = \{x_1, x_2, ..., x_n\} \tag{4.14}$$

Let also consider a condition C_i of the form:

$$C_i : (lb_i < x_i < ub_i), \ i = 1...n \tag{4.15}$$

where lb_i and ub_i are the lower and upper bounds of C_i respectively for which:

$$\forall C_i : |lb_i - ub_i| < \varepsilon \tag{4.16}$$

that is, the lb_i and ub_i have minimal difference in their value, under a specific threshold ε.

Consider also a conditional set *S*. *S* is of length *l* ($\leq n$) if it entails *l* conditions of the form described by eqs (4.15) and (4.16), which are coupled via the logical operators of *AND* and *OR* as described in eqs (4.17) and (4.18).

$$S_{AND} = C_1 \wedge C_2 \wedge ... \wedge C_l \tag{4.17}$$

$$S_{OR} = C_1 \vee C_2 \vee ... \vee C_l \tag{4.18}$$

Each conditional set *S* is considered as an individual in the population of the dedicated GA, which will be thoroughly explained next and eqs (4.17) and (4.18) were used to describe CS representing cost attributes or cost drivers. The main purpose of the algorithm is the definition of a set of software projects, *M*, the elements of which are vectors as in eq. (4.14) that hold the values of the specific cost attributes used in relation with a CS. More specifically, the set *M* can be defined as follows:

$$M = \{L_1, L_2, ..., L_m\} \tag{4.19}$$

$$L_i = \{x_{i,1}, x_{i,2}, ..., x_{i,l}\}, \quad i = 1...m \tag{4.20}$$

where l denotes the number of cost attributes of interest.

A conditional set S is related to M according to the conditions in eqs (4.17) or (4.18) that are satisfied according to the following:

$$\forall L_i : \quad x_{i,k} \ satisfies \ C_k \ , \ i = 1...m, \quad k = 1...l \ \text{(AND)} \tag{4.21}$$

$$x_{i,1} \ satisfies \ C_1 \ OR \ x_{i,2} \ satisfies \ C_2, ..., \quad OR \ x_{i,l} \ satisfies \ C_l, \ i = 1...m, \ \text{(OR)} \tag{4.22}$$

The goal of the proposed algorithm is to identify the exact value ranges for the attributes of cost drivers and determine the attributes that have a high influence on development effort by providing associated weights together with effort predictions. These weights represent the notion of ranked importance of the associated attributes and if used for predicting effort could possibly result in a more efficient and practical solution than the solutions offered by the previously proposed SCE approaches.

The dataset used was the ISBSG R9 from which only the attributes that were suggested by a previous technique (the *Less Strict* (*LS*) criterion of ISA on ANN (pg. 126 using ISBSG R9-2 dataset in (Papatheocharous and Andreou, 2007)) were selected. The data went through the pre-processing steps summarised in Table 4.1 (c)-(e) and was named ISBSG R9-5. In addition, ISBSG R9-6 was obtained after performing Box Plots and excluding the extreme effort values observed in the previously mentioned subset (i.e., suggested by the *Less Strict* (*LS*) criterion of ISA on ANN (pg. 126)). In this dataset a variation was also made, instead of the AFP attribute the NAFP was preferred to investigate the difference in the results utilising more homogeneous values (since values are normalised and outliers were also eliminated). Finally, the third dataset utilised, namely ISBSG R9-7, consisted only attributes that could be measured 'early' in the software life-cycle and for which Box Plots were used to eliminate outlying effort values. The subsets ISBSG R9-5, ISBSG R9-6 and ISBSG R9-7 included 386, 458 and 333 projects respectively whose attributes are summarised in Table 4.19.

Table 4.19: Attributes and abbreviations used in the genetically evolved Conditional Sets

Dataset	Attributes	Abbreviation
ISBSG R9-5	Adjusted Function Points	AFP
	Enquiry Count	EC
	File Count	FC
	Added Count	AC
	Changed Count	CC
ISBSG R9-6	Normalised Adjusted Function Points	NAFP
	Enquiry Count	EC
	File Count	FC
	Added Count	AC
ISBSG R9-7	Adjusted Function Points	AFP
	Project delivery rate (productivity) in functional size units	PDRU
	Project Elapsed Time	PET
	Resource Level (ordinal)	RL
	Average Team Size	ATS

Figure 4.11 summarises the methodology proposed and the steps followed for evolving CS and providing effort range predictions (Andreou and Papatheocharous, 2008a). At the initiation step, a random set or initial population of conditions (individuals) was created. The individuals were then evolved through specific genetic operators and evaluated internally using the fitness functions. The evolution of individuals continued while none of the termination criteria was satisfied. Among the termination criteria were a maximum number of iterations (called generations or epochs) or no improvement in the maximum fitness value occurred. The top 5% individuals resulting in the higher fitness evaluations were accumulated into the optimum range population, which then were advanced to the next algorithm generation (repetition). In the end, at the evaluation step, the final population was produced that satisfied the criteria and which was used to estimate the mean effort. Also, at the evaluation step, the methodology was assessed through performance metrics. The most successful CS evolved by the GA that had small assembled effort ranges with relatively small deviation from the mean effort, were then used to predict the effort of new, unknown projects.

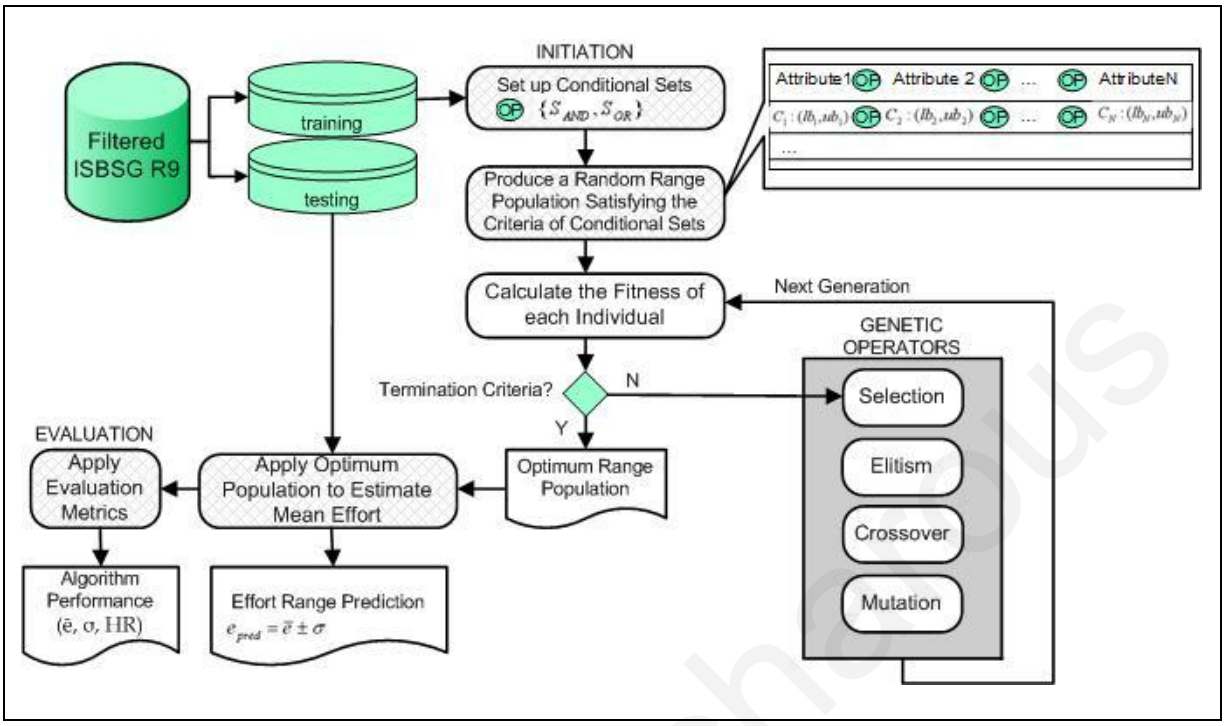


Figure 4.11: Methodology of genetically evolved Conditional Sets (CS & GA)

A dedicated GA utilising CS was implemented executing the following steps:

Step 1: Randomly create an initial population of individuals *P*, which represent solutions to the given problem (in this particular case, ranges of values in the form expressed in eqs (4.17) or (4.18)).

Step 2: Perform the following steps for each generation:

2.1. Evaluate the fitness of each individual in the population using eq. (4.23) or eq. (4.24) specified below, and isolate the best individual(s) of all preceding populations.

$$F_{AND} = k + \frac{1}{\sigma} + \frac{1}{\left(\sum_{i=1}^{I}(ub_i - lb_i) * w_i\right)} \tag{4.23}$$

$$F_{OR} = \sum_{i=1}^{I}\left(k_i + \frac{1}{\sigma_i} + \frac{1}{ub_i - lb_i}\right) * w_i \tag{4.24}$$

where k represents the number of projects satisfying the conditional set, k_i the number of projects satisfying only condition C_i, and σ, σ_i are the standard deviations of the effort of the k and k_i projects, respectively.

2.2. Create a new population by applying the following genetic operators:

2.2.1. *Selection*; based on the fitness select a subset of the current population for reproduction by applying the roulette wheel method. This method of reproduction allocates offspring values using a roulette wheel with slots sized according to the fitness of the evaluated individuals. It is a way of selecting members from a population of individuals in a natural way, proportional to the probability set by the fitness of the parents. The higher the fitness of the individual is, the greater the chance it will have to be selected, even though it is not guaranteed that the fittest member goes to the next generation. So, additionally, elitism is applied, where the top best performing individuals are copied in the next generation.

2.2.2. *Crossover*; two or more individuals are randomly chosen from the population and parts of their genetic information are recombined to produce new individuals. Crossover with two individuals takes place either by exchanging their ranges at the crossover point (inter-crossover) or by swapping the upper or lower bound of a specific range (intra-crossover). The crossover takes place on one (or more) randomly chosen crossover point(s) along the structures of the two individuals.

2.2.3. *Mutation*; randomly selected individuals are altered randomly and inserted into the new population. The alteration takes place at the upper or lower bound of a randomly selected range by adding or subtracting a small random number. This mutation number is based on the median range value of the population. Mutation intends to preserve the diversity of the population by expanding the search space into regions that may contain better solutions.

2.3. Replace the current population with the newly formed population.

Step 3: Repeat from step 2 unless a termination condition is satisfied. Output the individual with the best fitness as the near to optimum solution.

Each loop of the aforementioned steps is called a generation. The entire set of iterations from population initialisation to termination is called a run. At the termination of the process the algorithm promotes the ‘best-of-run’ individual.

By using the standard deviation in the fitness evaluation (eqs (4.23) and (4.24)) the algorithm promotes the evolved individuals that had their effort values close to the mean effort value of either the k projects satisfying S (AND case) or either the k_i projects satisfying C_i (OR case). Additionally, the evaluation rewards individuals whose difference among the lower and upper range is minimal. Finally, w_i in eqs (4.23) and (4.24) is a weighting factor corresponding to the significance given by the estimator to a certain cost attribute.

These particular fitness functions were used to define the appropriateness of the value ranges produced for a particular dataset. More specifically, when a conditional set is evaluated the dataset was used to define how many records of data (a record corresponds to a project with specific values for its cost attributes and effort) was within the ranges of values of the individual according to the conditions used and the logical operator connecting these conditions. Note that in the OR case the conditional set is satisfied if at least one of its conditions is satisfied, while in the AND case all conditions in S must be satisfied. Hence, k (and σ) is unique for all ranges in the AND case, while in the OR case k may have a different value for each range i. This requirement specified the need of having two different fitness functions for each of the two logical operators. The total fitness of the population in each generation was calculated as the sum of the fitness values of the individuals in P.

Once the GA terminates, the best individual was used to perform effort estimation. More specifically, in the AND case the individual's projects that satisfy the conditional set, while in the OR case the projects that satisfy one or more conditions of the set were distinguished. Next we calculated the mean effort value ($\bar{e}$) and standard deviation (σ) of those projects. For a new project for which an estimate of its development effort is required, the algorithm check whether the values of its attributes lie within the ranges of the best individual and that it satisfies the form of the pre-defined CS AND or OR. If this holds, then the effort of the new project is estimated to be:

$$e_{pred} = \bar{e} \pm \sigma \tag{4.25}$$

where e_{pred} is the mean value of the effort of the projects satisfying the conditional set S.

Until the phase of conducting the final set of experiments with the genetically evolved CS a series of initial setup experiments was carried out to define and tune the main parameters of the GA (as summarised in Table 4.20).

Table 4.20: Genetic Algorithm main parameters of the Conditional Sets utilised for SCE

Category	Value	Details
Attributes set	{ S_{AND}, S_{OR} }	-
Solution representation	L	-
Generation size	1000 epochs	-
Population size	100 individuals	-
Selection	-	Roulette wheel based on fitness of each individual
Elitism	-	Best individuals are forwarded (5%)
Mutation	Ratio 0.01-0.05	Random mutation
Crossover	Ratio 0.25-0.5	Random crossover (inter-, intra-)
Termination criterion	-	Generations size is reached or no improvements are noted for more than 100 generations

The indicative experimental results summarised in Table 4.21 (Andreou et al., 2007), show average performance in terms of *HR*, even in cases where the lower and upper bound values included in the conditions were broad. The most promising results were obtained with the coupling of OR-based CS. In this case, overly wide range intervals were produced and many projects (191 out of 386) satisfied the interval ranges evolved. In addition, 83 projects out of the 191 have their predicted value satisfying eq. (4.25).

Table 4.21: Performance results of genetically evolved Conditional Sets with Weights

Dataset (CS-case)	Attribute Weights /Ranges										Evaluation Metrics		
	AFP	**PDRU**	**PET**	**RL**	**ATS**	**NAFP**	**AC**	**FC**	**EC**	**CC**	$\bar{e}$	σ	***HR***
ISBSG R9-5 (**OR**)	-	-	-	-	-	-	0.4 [1, 1391]	0.3 [11,242]	0.2 [14,268]	0.1 [206,1735]	3204	1879	81/187
ISBSG R9-5 (**OR**)	-	-	-	-	-	-	0.4 [1,1377]	0.2 [3, 427]	0.2 [1, 347]	0.2 [46, 579]	3254.5	1857	83/191
ISBSG R9-6 (**AND**)	-	-	-	-	-	0.25 [1,152]	0.25 [22, 859]	0.25 [58, 3192]	0.25 [20, 563]	-	3188.6	2470.9	3/221
ISBSG R9-6 (**AND**)	-	-	-	-	-	0.25 [1, 156]	0.25 [34, 443]	0.25 [122, 2084]	0.25 [37, 469]	-	3151.6	2377.9	4/139
ISBSG R9-7 (**AND**)	0.2 [173, 1131]	0.2 [1, 20]	0.2 [2, 20]	0.2 [2, 4]	0.2 [1, 7]	-	-	-	-	-	2380.0	434.5	2/3
ISBSG R9-7 (**AND**)	0.25 [189, 1301]	0.25 [2, 26]	0 0	0.25 [1, 3]	0.25 [2, 11]	-	-	-	-	-	2477.8	838.2	5/5

In the OR CS case for the ISBSG R9-5 dataset, the best obtained indicative results included higher weight value on the attribute of AC in relation to the rest attributes that increased its corresponding significance in the fitness equation (as specified in eq. (4.24)). The level of *HR* obtained was quite good but the high value of the standard deviation compared to the mean effort (measured in person days) indicated that the CS attained were rather dispersed (too wide) and not of high practical value. The mean effort of the best 100 experiments conducted was found equal to 2929 and the total standard deviation equal to 518. From these measures the total standard error was estimated at 4.93.

In the AND CS case for the ISBSG R9-6 dataset the results appear to be very poor, since very low *HR* was obtained indicating that the CS satisfied in terms of ranges a large number of projects but the prediction range produced was not overly successful, i.e., it did not include the actual effort values of those projects. The AND case was based on a stricter method than the OR case and is considered more practical since it requires all yielded ranges to be simultaneously satisfied, instead of just one range (like in the OR case).

Finally, in the AND CS case for the ISBSG R9-7 dataset (i.e., which utilises the subset of attributes that can be measured 'early' in the development life-cycle) the best obtained standard deviation of effort falls to 74.9. The overly narrow effort prediction interval produced is considered quite successful, but unfortunately the CS obtained satisfied only 2 projects. However, both of these 2 projects' effort is included in the prediction interval obtained. This leads us to conclude that either the careful removal of outliers performed, or the utilisation of attributes measured early have contributed to obtaining optimised results with the proposed methodology. The results attained for the AND cases demonstrated that there is high individuality regarding the projects, especially the projects included in the multi-organisational dataset ISBSG R9. In general, the results may be regarded as achieving consistently successful predictions, yielding optimum ranges that are adequately small and suggesting effort estimations that are within reasonable mean values and perfectly acceptable deviation from the mean. However, due to the two goals utilised in the fitness functions, i.e.,

narrow predictions and satisfying large number of projects, further calibrations need to be performed since there are cases where these two goals can be conflicting.

Figure 4.12 depicts the total fitness value of a sample population through generations, which, as expected, rises as the number of epochs increases. A plateau was observed in the range 50-400 epochs which may be attributed to a possible trapping of the algorithm to a local minimum. The algorithm seems to escape from this minimum with its total fitness value constantly being improved along the segment of 400-450 epochs and then stabilising. Along the repetitions of the algorithm execution, the total population fitness improves showing that the methodology performs consistently well.

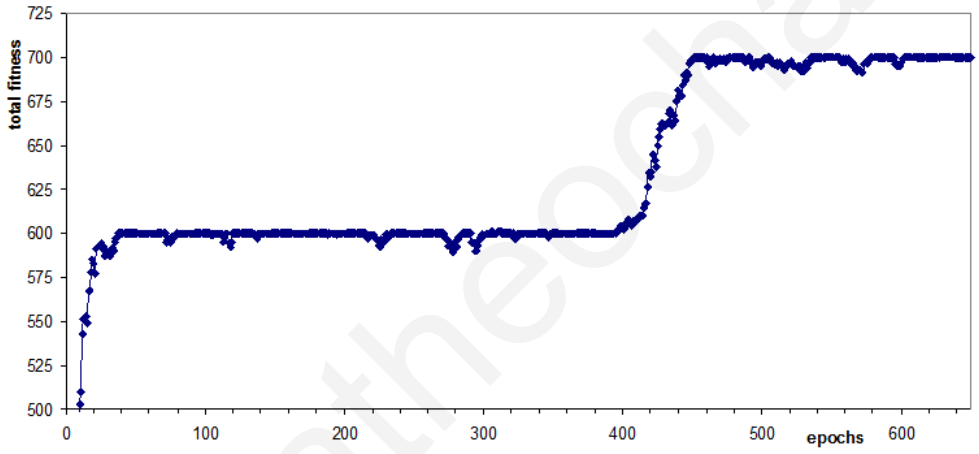


Figure 4.12: Total Fitness Evolution of the Genetically evolved Conditional Sets for SCE

The difficulty of obtaining genetically evolved CS in all the experiments described in this section suggested that large dissimilarities and high 'individuality' exist within the values reported in software projects, at least regarding the specific attributes investigated. This also suggests that perhaps selecting the appropriate features to obtain clustered groups of projects that conform to specific rules and which, if successfully isolated, e.g., through a clustering technique, and effort is estimated using classification or by-analogy techniques, then improvements may be achieved. The subsequent techniques emphasise on ways to obtain more intelligent clusters of projects that share similar characteristics utilising the theory of Fuzzy Logic.

4.2.3.2 Fuzzy Clustering in CC-SCE

Fuzzy clustering algorithms seek to organise data samples into several subsets taking into consideration the degree of membership of each object to a certain subset. Therefore, fuzzy clustering may deal with the possible lack of homogeneity, objectivity and quality of data. In previous related research work (Aroba et al., 2008) the technique of fuzzy clustering yielded better figures of adjustment (better results) than their crisp equivalent (i.e., the Expectation Maximization clustering algorithm). Aroba et al. (2008) supported that in some cases the projects under estimation cannot be assigned to a cluster in a sharp (hard) way and one may consider clustering them in more than one cluster and improve the overall prediction result. Therefore, the proposed approach for Fuzzy Clustering employs a hybrid algorithm, namely the Entropy-based fuzzy *k*-modes clustering algorithm (Tsekouras et al., 2005), which is a combination of entropy (Yao et al., 2000) and the *k*-means algorithm for categorical data (Huang, 1998). The approach applies a threshold to obtain clustered projects and ultimately estimates effort of a new project which is assigned to a specific cluster by taking into account the mean effort and standard deviation values (Papatheocharous and Andreou, 2009a).

Classical clustering algorithms work under the assumption that well-defined boundaries exist between clusters and they assign each object to one and only one cluster, with a membership degree equal to 1. In fuzzy clustering objects may belong to more than one cluster and for each association a membership level exists. The set of membership levels indicates the strength of association between the objects in each cluster and it is used to assign objects to one or more clusters.

The ISBSG R9 dataset was used in the experiments conducted. The dataset went through the pre-processing steps described in Table 4.1 (a), (c), (e), (f), (j)-(l), (n) and (o). This pre-processing led to the ISBSG R9-8 which comprised of 424 projects and 49 attributes, summarised in Table 4.22.

Table 4.22: ISBSG R9-8 attributes description

Abbreviation	ISBSG R9-8 Attribute Description	Number of values
CA	Count Approach	1
AFP	Adjusted Function Points	1
PET	Project Elapsed Time	1
IY	Implementation Year (extracted from Implementation Date)	1
DT	Development Type	1
OT1-12	Organization Type	12
DT1-15	Development Technique	15
FST	Functional Sizing Technique	1
DP	Development Platform	1
LT	Language Type	1
PPL	Primary Programming Language	1
DBS1-9	Database System	9
RM	Recording Method	1
RL	Resource Level	1
MTS	Max Team Size	1
ATS	Average Team Size	1

The fuzzy clustering methodology was performed on the training data which comprised the 75% of the initial data samples. Initially, the Entropy-based clustering computed the number of clusters (k) in the software projects datasets and identified the candidate initial cluster centres based on the entropy value (defined in eq. (4.30)) for the set of n projects described by m attributes $X = [x_1, x_2,..., x_n]$. The definition of the entropy value for each pair of data projects i and j (eq. (4.26)) is dependent on the fuzzy exponent α and the distance measure D.

$$H_{ij} = -E_{ij}\log_2(E_{ij}) - (1-E_{ij})\log_2(1-E_{ij}),\ \textit{where } i \neq j \tag{4.26}$$

E_{ij} is the similarity measure between two projects, estimated using a distance function (D_{ij}) and α (defined in eqs (4.27)-(4.28) respectively, where $\overline{D}$ is the mean distance among the pairs and D is calculated based on the Hamming distance (Hamming, 1950) of eq. (4.29), (i.e., for binary strings it includes the number of ones in an XOR).

$$E_{ij} = e^{-aD_{ij}} \tag{4.27}$$

$$a = -\ln(0.5)/\overline{D} \tag{4.28}$$

$$D_{ij} = \sum_{j=1}^{m} \begin{cases} 0, & \textit{if } X_{1j} = X_{2j} \\ 1, & \textit{if } X_{1j} \neq X_{2j} \end{cases} \tag{4.29}$$

As already mentioned, the total entropy value of a project X_i with respect to all other projects is estimated in eq. (4.30). The yielding $Z_1 = \{Z_{11},...Z_{1m}\}$ represents a cluster centre, to which

each project is assigned to, as it achieves the minimum entropy value after each algorithm iteration.

$$H_{ij} = -\sum_{\substack{j=1 \\ i \neq k}}^{n} [E_{ij} \log_2(E_{ij}) - (1 - E_{ij}) \log_2(1 - E_{ij})] \tag{4.30}$$

The Entropy-based clustering algorithm as described in (Stylianou and Andreou, 2007) executes the following steps:

1. Based on a selected threshold of similarity (β) set the initial number of clusters to zero ($c=0$).
2. Estimate the total entropy values (H) for each project in the dataset X using eq. (4.30).
3. Set $c=c+1$.
4. Select the project X_{min} with the smaller entropy and set $Z_c=X_{min}$ as the c^{th} cluster centre.
5. Remove X_{min} and all projects having similarity with $X_{min} > \beta$ from the dataset X.
6. If X is empty then stop; otherwise continue with step 3.

This means that projects with many surrounding projects will have total entropy values relatively lower than the rest, and will be thus considered as strong candidates for representing a cluster. The project with the lowest entropy value is selected as the cluster centre and the algorithm uses parameter β, which represents the similarity threshold, to exclude projects with very high similarity degree to the recently selected cluster centre, i.e., prevent these projects from being considered as potential cluster centres (Yao et al, 2000). The number of clusters is increased and the project with the next lowest entropy value is selected and the algorithm continues until zero projects are left in the dataset.

The *k*-modes algorithm, introduced by (Huang, 1998) and later on extended in (Huang, 1999) to include fuzzy elements so as to account for the uncertainty observed in data samples, uses an altered dissimilarity function based on a simple matching of the attributes (i.e., the Hamming distance instead of the Euclidean distance). In addition, in the fuzzy version of the algorithm the cluster centres are defined by the modal value of each attribute

instead of the mean value and their computation relies on the assignment of the most frequent category of each attribute as the representative of the cluster.

Consider $X = [x_1, x_2, \ldots, x_n]$ a set of n projects described by m attributes $A=[A_1, A_2, \ldots, A_m]$. An A_j attribute can take any value from the domain $D(A_j) = \{a_j^1, a_j^2, \ldots, a_j^{nj}\}$ where n_j is the possible number of category values for attribute A_j ($1 \leq j \leq m$). Thus, data objects may be represented by attribute-value pairs of the form: $[A_{i1} = x_{i1}] \wedge [A_{i1} = x_{i1}] \wedge \ldots \wedge [A_{i1} = x_{il}]$ (Kim et al., 2004) and therefore a project X_i may be represented by a vector of the form $[x_{11}, x_{12}, \ldots, x_{1m}]$ for $1 \leq i \leq m$. $Z_l = \{Z_{11}, \ldots Z_{lm}\}$ represents a cluster center for $1 \leq l \leq n$ and finally, the aim is to minimise the cost function of eq. (4.31) subject to eqs (4.32)-(4.34) (Bezdek, 1980).

$$F(W, Z) = \sum_{l=1}^{k}\sum_{i=1}^{n} w_{li}^{\alpha} d(Z_l, X_i) \tag{4.31}$$

$$0 \leq w_{li} \leq 1, \quad 1 \leq l \leq k, 1 \leq i \leq n \tag{4.32}$$

$$\sum_{l=1}^{k} w_{li} = 1, \quad 1 \leq i \leq n \tag{4.33}$$

$$0 \leq \sum_{i=1}^{n} w_{li} \leq n, \quad 1 \leq l \leq k \tag{4.34}$$

k ($\leq n$) represents a predefined number of clusters, $W = [w_{li}]$ is a $k \times n$ partition matrix, $Z = \{Z_1, Z_2, \ldots, Z_k\}$ is the set of cluster centres, and $d(\cdot, \cdot)$ is some measure of distance between two objects. The fuzziness exponent, $\alpha \in [1, \infty)$ in eq. (4.31) identifies whether hard ($\alpha = 1$) or fuzzy ($\alpha > 1$) k-modes clustering is performed.

Consider $X_1 = [x_{11}, x_{12}, \ldots, x_{1m}]$ and $X_2 = [x_{21}, x_{22}, \ldots, x_{2m}]$ two data samples of a dataset described by m attributes. The dissimilarity between these two samples, $d(X_1, X_2)$, is given by eq. (4.35)

$$d(X_1, X_2) = \sum_{j=1}^{m} \delta(x_{1j}, x_{2j}) \tag{4.35}$$

where δ is defined in eq. (4.36).

$$\delta\left(x_{1j},\, x_{2j}\right) = \begin{cases} 0, & x_{1j} = x_{2j} \\ 1, & x_{1j} \neq x_{2j} \end{cases} \tag{4.36}$$

The dissimilarity function (eq. (4.35)) is then used to (re)assign a data sample to a cluster. Accordingly, in the case of the hard *k*-modes algorithm, if object X_i yields the shortest distance with centre Z_l in a given iteration, this is represented by setting the value at the nearest cluster to 1 and the values at the rest of the clusters to 0 in the partition matrix *W*. Formally, for $\alpha = 1$:

$$\hat{w}_{li} = \begin{cases} 1, & \text{if } d\left(Z_l,\, X_i\right) \leq d\left(Z_h,\, X_i\right), \quad 1 \leq h \leq k \\ 0, & \text{otherwise} \end{cases} \tag{4.37}$$

In the case of the fuzzy *k*-modes algorithm, for $\alpha > 1$, the partition matrix *W* is given by:

$$\hat{w}_{li} = \begin{cases} 1, & \text{if } X_i = Z_l \\ 0, & \text{if } X_i = Z_h,\, h \neq l \\ \dfrac{1}{\sum_{h=1}^{k}\left[\dfrac{d\left(Z_l,\, X_i\right)}{d\left(Z_h,\, X_i\right)}\right]^{1/(\alpha-1)}}, & \text{if } X_i \neq Z_l \text{ and } X_i \neq Z_h,\, 1 \leq h \leq k \end{cases} \tag{4.38}$$

for $1 \leq l \leq k$, $1 \leq i \leq n$. This means that if a data sample has exactly the same attribute values with a particular cluster centre, then it will be assigned fully to that cluster and not at all to the rest. Otherwise, the data sample will be characterised by a membership degree for each cluster denoting its partial membership in the cluster (Tsekouras et al., 2005).

The Fuzzy *k*-modes clustering algorithm as described in (Stylianou and Andreou, 2007) executes the following steps:

1. Set *k* random initial clusters $Z^1 = \{Z_1^1.\, Z_2^1, ..., Z_k^1\}$.
2. Calculate W^1 by using eq. (4.32) or (4.33) for applying hard or fuzzy clustering, so that a cost function $F(W, Z^1)$ (eq. (4.31)) is minimised.
3. Set $t = 1$.
4. Calculate Z^{1+1} by finding the update of every cluster based on the frequency of categories of attributes (modes of attributes) for hard *k*-modes or fuzzy *k*-modes clustering, so that cost function $F(W^t, Z^{1+1})$ is minimised.
5. If $F(W^t, Z^{t+1}) = F(W^t, Z^t)$ then stop; otherwise go to step 6.

6. Calculate W^{t+1} using the same equation as in step 2, such that $F(W^{t+1}, Z^{t+1})$ is minimised.
7. If $F(W^{t+1}, Z^{t+1}) = F(W^{t}, Z^{t+1})$ then stop; otherwise set $t = t + 1$ and go to step 4.

Thus, the application of the entropy-based fuzzy *k*-modes clustering algorithm yielded the *k* clusters with respect to the fuzzy exponent α and the threshold of similarity β. At the validation step, using the rest 25% of the original data, each project was matched against the final cluster centres and using the partition matrix *W* (eq. (4.88)) the nearest centre was obtained. Then, a cut-off limit was used to further isolate projects responding to a similarity measure φ. The cut-off limit was constructed by defining an upper and lower bound (set to ±10%) based on the value of the new project's membership degree to the closest cluster centre. Thus, if a new project was assigned a membership degree of 60% in the search cluster, then the projects retrieved would have membership degrees between 50%-70%. The similarity measure φ ensures that only the closest projects falling between the ±10% radius distance from the new project, are selected. The value of the similarity measure φ was specified equal to 75% and 85%. Then, the mean effort value ($\bar{e}$) and standard deviation (σ) of the actual effort values of the projects isolated were estimated. The predicted effort value of new projects was estimated to lie within the range $[\bar{e} \pm \sigma]$ (also refer to eq. (4.25)). In this particular case however if the standard deviation was greater than the mean (i.e., $\sigma > \bar{e}$), the lower bound of the abovementioned interval was set equal to zero (i.e., the range would be $[0, \bar{e} + \sigma]$). Thus, the methodology proposed bounded estimation intervals of the minimum possible width, rather than single point value predictions, yielding more 'flexible' estimates on one hand, as it encapsulates the inherent estimation uncertainty, but of a more informative nature on the other (Papatheocharous and Andreou, 2009a).

The evaluation of the predictions was carried out through the following four performance metrics:

a. If the actual effort fell within the range $[\bar{e} - \sigma, \bar{e} + \sigma]$ then the Hit Ratio (*HR*) of the estimation was increased by one.

b. Width was calculated by subtracting $[\bar{e} - \sigma]$ from $[\bar{e} + \sigma]$ in case the result of $[\bar{e} - \sigma]$ was positive; otherwise the width was considered equal to $[\bar{e} + \sigma]$.

c. Overall Size (*OS*) was calculated dividing the Width with the subtracted minimum effort value from the maximum effort value contained in the original samples.

d. Cluster Size (*ClS*) was calculated dividing the Width with the subtracted minimum effort value and maximum effort value contained in the clustered samples.

The last two metrics represent how 'good' or how shrinked the estimation intervals are.

The best fuzzy clustering results of the algorithm and the associated effort prediction intervals of minimum width, based on the mean and standard deviation values of the effort of the clustered projects are summarised in Table 4.23. The clustering results were obtained using a variation of the dataset (namely ISBSG R9-8.5) which included the attributes of ISBSG R9-8 plus the effort attribute as a classifier, and for which the following steps were additionally executed: (i) the outlying projects based on Box Plots on the effort sample values were excluded, and (ii) an additional weight value on the effort variable (to reach the dominant significance level of 51% in the clustering process compared to the rest of the attributes) was applied.

Initially, experimentation was carried out with the Entropy algorithm to locate the cluster centres *k* and subsequently hard clustering was applied. Then, the fuzzy *k*-modes algorithm was executed to obtain the fuzzy clustering results. The best clustering carried out using all respective variations of the ISBSG R9-8 dataset are included in Appendix B. In the experiments the parameter β and fuzzy exponent α were varied, taking values from the sets {0.3, 0.4, 0.5, 0.55, 0.6, 0.7, 0.8, 0.9} and {1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8}.

Table 4.23: Entropy-based fuzzy *k*-modes clustering results (effort values in person-hours) for the ISBSG R9-8.5 dataset

β	α	*k*	OS (%)	ClS (%)	*HR*(%)	mean effort ($\bar{e}$)	std effort (σ)	width
0.8	1.7	25	1.60	37.92	76.60	2030.93	1198.76	2397.53
0.8	1.5	25	1.88	39.69	76.60	2294.23	1406.38	2812.76
0.8	1.8	25	1.92	38.74	45.74	2206.57	1433.75	2867.49
0.9	1.7	104	1.93	50.20	62.77	2625.31	1440.06	2880.12
0.9	1.5	104	1.94	77.48	62.77	2647.77	1452.07	2904.14

From the results obtained the algorithm estimated in all cases a standard deviation (σ) value consistently lower than the mean effort value ($\bar{e}$) reported. This means that the prediction intervals yielded were consistently narrow, i.e., at least lower than the initial value reported for the interval [$\bar{e}$ - σ, $\bar{e}$ + σ] before clustering was applied (refer to Appendix A, Table A. 17). The assumption formulated here was that the narrower a predictive interval is the more useful will be to project managers. Additionally, as reported by other researchers (Jørgensen and Moløkken, 2002) previous studies suggest that software development effort prediction intervals are, on average, much too narrow to reflect high confidence levels, i.e., the uncertainty is under-estimated. Therefore, a balance needs to be found on the acceptable interval estimates and the confidence regarding this estimate (i.e., the expected probability that the real value is within the predicted interval). For example, Jørgensen et al. (2004) report that on average, if a software professional is 90% confident or 'almost sure' to include the actual effort in a minimum-maximum interval, the observed frequency of including the actual effort is only 60-70%.

Therefore, the results obtained from fuzzy clustering indicate 'relatively narrow' widths, (even though this might be considered an assumption) and is not reflected by a confidence level; whereas the best results achieved a spread of approximately 2398 person-hours (ph), with a mean effort value of 2031ph and a corresponding standard deviation of 1199ph.

The *HR* degree in relation to both *OS* and *ClS* degrees reported suggested that clustering data in small segments had been achieved: The derived interval in the best case was 17% of the initial and 38% of the clustered one. The best results consistently suggested k=25 as the 'optimal' number of clusters, while parameters β and α assumed the values of 0.8 and 1.7 respectively.

Finally, the methodology identified clusters of similar projects and then assigned each new project successfully according to its resemblance with the cluster centres. The approach utilised both nominal and numerical attributes for clustering the projects and has identified as suitable and as homogeneous clusters of projects as possible. The best cost estimates yielded *HR* up to 77% which is quite promising, i.e., the estimations were within the calculated width

in nearly 77% of the cases. However, in order to obtain these estimates, 51 projects with extreme effort values were disregarded (were considered as outliers) and additionally, weighted effort values were used as described above. The relatively low *HR* rates obtained without the aforementioned modifications (included in Appendix B, Table B. 16) indicated that the modelling was not entirely successful and that there is a need for further exploration on other fuzzy clustering and classification techniques for obtaining improved predictive intervals for software projects. Towards this aim, the subsequent explorative approaches for SCE in the rest of this chapter are described, starting from the issue of obtaining algorithmic approximations using Genetic Programming are explored in the subsequent section.

4.2.3.3 Genetic Programming (GP) in CC-SCE

In this section, the development of an automatic tool that uses Genetic Programming (GP) to examine possible candidate solutions to algorithmic cost estimation is described (Papatheocharous et al., 2010a). The methodology based on GP aims to seek and locate appropriate equations consisting of cost factors which characterise the dependent variable (development effort) in the best possible way according to specific grammars (i.e., a set of syntactical constraints, operands, operators, etc.). The methodology addresses two goals: (i) To yield symbolic representations of development effort by analysing a large set of variables and constructing optimised sets of solution-equations, (ii) To generate logical expressions for attributes of categorical nature.

The datasets used consisted of the COCOMO, the Desharnais and the ISBSG R9 dataset. The datasets went through only one pre-processing step as described in Table 4.1 (a) and thus, the COCOMO consisted of 63 and the Desharnais of 77 projects. Additionally, the ISBSG R9 dataset went through the pre-processing steps described in Table 4.1 (c), (f), (k)-(m) which formed dataset ISBSG R9-4 consisting of 467 projects. The attributes of the COCOMO dataset are summarised in Table 4.24, the attributes of the Desharnais are summarised in the

first part of Table 4.9 (pg. 123) and finally, the attributes of the ISBSG R9-4 are summarised in Table 4.16 (pg. 143).

Table 4.24: The COCOMO cost attributes

Code	Attribute name
LOC	Lines of Code
RELY	Required Reliability
DATA	Database Size
CPLX	Product Complexity
TIME	Execution Time Constraint
STOR	Main Storage Constraint
VIRT	Virtual Machine Volatility
TURN	Computer Turnaround Time
ACAP	Analyst Capability
AEXP	Applications Experience
PCAP	Programmer Capability
VEXP	Virtual Machine Experience
LEXP	Programming Language Experience
MODP	Modern Programming Practices
TOOL	Use of Software Tools
SCED	Required Development Schedule

The SCE methodology based on the GP technique produced candidate solutions to algorithmic cost estimation by experimenting with a set of representations that utilised operators and operands (Papatheocharous et al., 2010a). These solutions essentially represented regression equations that made use of software cost factors to effectively describe the dependent variable in a software development environment, that is, the effort spent in software projects. The reason for using evolutionary models and more specifically structures that serve as candidate solutions through GP that combine different cost factors in simple mathematical equations close to regression models is that they may find better solutions than conventional regression or other methods.

In the GP context, potential solutions in the population are represented as parse trees (an example is illustrated in Figure 4.13) and usually utilise a customised pool of arithmetic operators in the case of input factors of numerical nature. In the case of categorical input factors parse trees are built with logical operators.

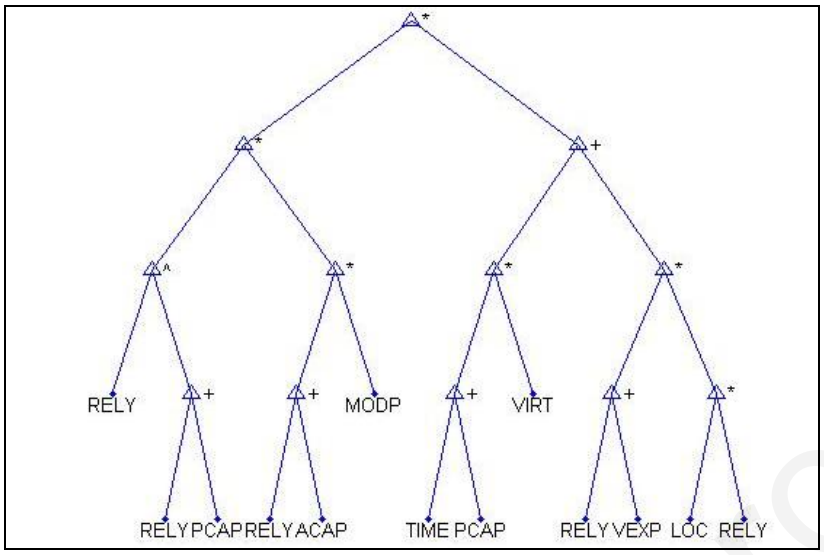


Figure 4.13: An example of parse tree for the COCOMO dataset

The GPLab toolbox (Silva, 2007) was used and extended with new, custom-built functions (developed in Matlab R2007b). The extensions may execute the steps summarised in this section. From each of these steps specific parameters and selections were utilised in the experiments conducted and presented, as these are summarised in Table 4.25 (pg. 177):

Step 1: Employ software cost attributes from a dataset to design and build a random population of potential solutions (random parse trees of regression equations).

Step 2: Execute each solution (parse tree) and assign a fitness value to it according to how well the solution describes the dependent variable (effort).

Step 3: Generate new offspring in the population using the following procedure:

- Sampling

It defines the method for selecting a parent based on which new individuals are created. The following five variations are supported: *Roulette*, where a roulette with random pointers is spun and each individual owns a portion of this roulette. *Sus*, which is based on the roulette process but here the pointers are equally spaced. *Tournament*, where the parent is chosen with a random draw of individuals and then the best of them is selected. *Lexictour*, which is based on tournament but in case of equality the shortest individual wins. *Double tournament*, where two tournaments are performed one for fitness and one for parsimony. The sampling methods

select the best individuals according to a certain fitness function (see eqs (4.39) and (4.40)) to apply the genetic operators listed below.

- Crossover

  Random nodes from two parents are chosen and swapped creating two new individuals.

- Mutation

  A random node is chosen from a parent and replaced with a randomly created tree. Several different types are utilised here: *Shrink Mutation*, a random sub-tree *S* is chosen from a parent and replaced with a random sub-tree of *S*. *Swap Mutation*, two random sub-trees are chosen from a parent and swapped. *Replace Mutation*, follows the concepts of the normal mutation but with the difference that for a certain mutation point performed on a terminal node (i.e., a cost factor) the terminal node is replaced by another random operand from the available pool, whereas in the case mutation is performed on an arithmetic operator, the operator will be replaced by another randomly selected operator from the available pool.

- Survival

  The selection of a number of individuals from the current population and the newly created children forms the new population.

- Elitism

  The best individuals of each generation are passed to the next one unchanged. This is known as the full (or total) elitism operator, according to which the more fit individuals get the chance to be part of the reproduction process throughout generations.

Step 4: Repeat steps 2 and 3 until the maximum number of generations is reached. The solution with the highest fitness value from the final population is considered to be the result of the GP.

Initially, the data is randomly split in two sets, the training and testing set, according to a percentage defined by the user. In the experiments, a random separation of the data samples into percentages of 80% for training and 20% for testing was executed and this random holdout cross-validation process (Weiss and Kulikowski, 1991) was repeated 100 times. The rationale behind this was to produce cost functions (solutions) valid within a variable and

random set of training samples and then evaluate their generalisation performance with the rest of the testing samples.

The cost equations in the case of numerical attributes make use of the following arithmetic operators: add (+), minus (-), divide (/), multiply (*), power (^), natural logarithm (*log*), base-2 logarithm (*log*2) and base-10 logarithm (l*og*10). In the case of categorical attributes the following pool of logical operators is used: less than (<), less than or equal (≤), greater than (>), greater than or equal (≥), logical AND (&&), logical OR (||), exclusive OR (XOR), negative OR (NOR), negative AND (NAND) and IF-THEN-ELSE. The specification of any of the two variations of operators (i.e., arithmetic, logical) is according to the type of input variables and results in equations in the form of either algebraic or logical expressions. Each equation-solution in the population for arithmetic attributes may be evaluated according to the default fitness equation (*regfitness*) shown in eq. (4.39) or to the modified *mrefitness* (defined in eq. (4.40)).

$$regfitness = \sum_{i=1}^{n} | x_{act}(i) - x_{pred}(i) | \tag{4.39}$$

$$mrefitness = \frac{1}{n} \sum_{i=1}^{n} | \frac{x_{act}(i) - x_{pred}(i)}{x_{act}(i)} | \tag{4.40}$$

The cost equations evaluation in the case of the logical-solutions produced by the GP is performed using eq. (4.41), where *k* represents the number of projects satisfying the equation, σ is the standard deviation of the actual effort of the *k* projects and *TreeNodes* represents the size of the rule obtained. Using eq. (4.41) for the logical equations of the GP the individuals promoted satisfied the largest possible number of projects, thus being more representative for the dataset, had effort values closer to the mean effort value of those projects and had a relatively small length.

$$\log fitness = k + \frac{1}{\sigma} + \frac{1}{TreeNodes} \tag{4.41}$$

The experiments conducted with the GP methodology (Papatheocharous et al., 2010a) aimed to obtain the optimal equation-solutions for estimating effort and to provide a relative importance ranking for the set of input variables utilised. Another important criterion for the

tree-structured equations constructed was to avoid bloating, that is, yielding relatively simple equations (avoiding overly deep trees) by imposing tree depth and node restrictions. Therefore, careful selection of several values for the parameters was performed to avoid bloating.

Bloating is the phenomenon in GP where the tree structures (i.e., program code or regression equations) expand excessively without the analogous improvement in fitness. Several techniques were applied to control bloating that seem to have promising results to other problems, like for example defining tree depth size and node number restrictions on the evolved trees (Koza, 1992), setting dynamic maximum tree depth (Silva and Almeida, 2003), applying *Lexictour* and *Double tournament* (*doubletour*) sampling methods and survival methods based on *Resources* (Silva and Costa, 2005). Variations of these parameters as well as for the rest of the GP parameters were tried out through a large series of initial simulation experiments in order to obtain the configurations reported in Table 4.25.

Table 4.25: GP parameters configurations used in the experiments for SCE

Parameter	Value
Cross-Validations	100
Training Set Size	80%
Testing Set Size	20%
Population Size	100
Number of Generations	350
Tree Population Type	Balanced and Unbalanced
Survival	Resources
Elitism	Total Elitism
Sampling	Double Tournament
Fitness Function	Regfitness (eq. (4.39))
Fitness Improvement	Best-of-Mean Population Fitness
Expected Number of Children	Absolute
Genetic Operators	Crossover (probability=0.3), Replace Mutation (probability=0.2) and Crossover & Mutation (probability=0.6)
Tree Size	Based on Depth (3, 4, 5, 6 or 7)/Based on Nodes (16, 20 or 28)
Dynamic Max Tree Depth	On
Resources	Static
Stop Condition	Generations=350/Fitness Hit
Fitness Hit Percentage	60
Fitness Hit Tolerance	10

The Tree Population Type relates to the type of the trees created in the initial population, variations of which can be: *fullinit*, *growinit* and *rampedinit*. A common approach is usually to construct 'Ramped Half and Half' trees. This means that half trees are constructed based on

the maximum tree level and the other half may have a random form. This was a way to allow the GP algorithm to produce random tree types in the initial population. The Fitness Improvement describes the way the generation can be improved considering the fitness value of either the best-of-run mean population fitness or the mean population fitness of the previous generation. The Expected Number of Children determines the method used for calculating the expected number of children of each individual. With the *absolute* method the expected number of children of each individual is proportional to its fitness value, whereas with the *rank85* the expected number of children of each individual is based on its rank in the population. The Dynamic Maximum Tree Depth is a method to control bloating by setting a maximum depth on trees being evolved, so that when a genetic operator produces a tree that outruns this limit, one of the parents enters the new population instead. The Stop Condition specifies when the algorithm stops. It is set to either a maximum generation size, or it is activated if the best individual produces exact results within a Fitness Hits Tolerance percentage of the expected results. The percentage of expected results is specified within a least Fitness Hits Percentage.

Table 4.26 lists an indicative set of cost equations obtained by applying the GP algorithm using the parameters above for each dataset. The first two equations of the COCOMO and Desharnais datasets are expressed with regression equations derived using only the numerical attributes in each dataset. The third equation was obtained using logical operators and attributes of categorical nature (operands) of the ISBSG R9-4 dataset.

Table 4.26: Indicative cost functions using arithmetic and logical operators obtained with GP

Dataset	Indicative Expressions
COCOMO	(((LOC*SCED)^(TIME/TOOL))+((LOC^VEXP)^(TIME*MODP)))
Desharnais	((((DU* DU)+ SC)+((ME+FPA)+(TR+ME)))*((FPNA-(TR+DU))))
ISBSG R9-4	((OT7='0') ||((((DT15='0') NAND(OT4='0')) NORIF ((DBS2='0') THEN ((DT1='0')) ELSE ((OT2='0')))) NOR(((DBS6='0') ||(MTS='0')) &&((DT1='1') NOR(DBS3='0')))))

The yielded equations are presented in a relatively simple form and are considered easy to be used by project managers, even though their rationale might be considered harder to understand. This however needs to be confirmed through practical investigations with real project managers via collaboration with an industrial partner.

The main advantage for a project manager will be the execution of these forms of equations obtained by the algorithm to attain in the arithmetic case effort values and in the logical case clustered groups of projects that conform to them. Therefore, regression equations, such as the ones obtained from the COCOMO and Desharnais data, may be directly used for calculating the effort of new projects. The predicted effort values using some indicative regression equations yielded by the GP are summarised in Table 4.27. Logical equations, such as the one obtained from the ISBSG R9-4, may offer an indication of the relation of specific type of projects, for example those developed within specific types of organisations (like 'Banking'), or using data modelling as a development technique, or database systems other than ORACLE or SYBASE. Such rule-based equations, once executed, may constitute a tool for clustering projects described with linguistic values, like the ISBSG database, into smaller and more homogeneous groups of projects which can be further analysed.

Table 4.27: Indicative cost estimation performance of GP cost functions execution

Dataset	Tree Depth	No. Of Nodes	TESTING		
			MMRE	***CC***	***NRMSE***
COCOMO	4	-	0.469	0.945	0.322
	4	-	0.497	0.962	0.349
	5	-	0.497	0.962	0.354
	5	-	0.494	0.979	0.243
Desharnais	5	-	0.485	0.768	0.722
	4	-	0.541	0.733	0.702
	-	20	0.521	0.731	0.684
	-	28	0.574	0.744	0.710

The performance results of the indicative equations show relatively adequate predictions for the COCOMO and Desharnais datasets. Finally, for the ISBSG R9-4 dataset the execution of the logical expression led to the classification of a number of projects which may be used for effort prediction by estimating the mean effort value ($\bar{e}$) and standard deviation (σ) of the actual effort values of the projects satisfying the equation. Then, the predicted effort value of a new project was estimated to lie within the range $[\bar{e} \pm \sigma]$ (also refer to eq. (4.25)). Based on this predictive interval of effort the Hit Ratio (*HR*) is estimated. For example, using the

function reported in Table 4.26 the mean effort, standard deviation and the *HR* value was 7069, 14072 and 87/93 (i.e., satisfied 87 out of 93 projects of the testing set).

In addition, a reduced set of features was identified by further analysing the best performing solution-equations obtained from the GP approach. The attributes identified to appear more frequently in the best performing equations were the following: for the COCOMO dataset project the attributes of size (LOC), required development schedule (SCED), complexity (CPLX) and applications experience (AEXP), for the Desharnais dataset the attributes of project duration (DU), scope (SC), number of transactions (TR) and function points (adjusted) (FPA) and for the ISBSG R9-4 the attributes of organisation type (OT), development technique (DT), database system (DBS) used and maximum team size (MTS). For the more details refer to the complete experimental results in Appendix B (pg. 300). The average performance results obtained with the GP approach on datasets that were previously examined (e.g., comparing the results of the Desharnais dataset and the various FSS approaches previously mentioned – see section 4.2.2.1 pg. 138 – refer again to Appendix B for the mean value of *MMRE*=0.511) and section 4.2.2.2 pg. 148) appear to be slightly improved.

The commonly selected features included in the majority of the regression equations obtained by the application of GP for the COCOMO and Desharnais datasets, concern attributes that are mostly project-specific and moreover, one would expect them to considerably affect the overall development effort. Whereas, the commonly selected features for the ISBSG dataset relate mostly with domain experience and organisational-related aspects of the developing organisations. The most influential attributes placed in the top nodes of the GP trees produced are quite common and in agreement with the attributes proposed by previous FSS methods. Particularly, for the Desharnais case these are DU and FPA, thus confirming the direct relation of project duration and software size with effort. Also the attributes of scope (SC) and number of transactions (TR) were commonly considered significant. For the ISBSG R9-4 case the common features selected by the majority of the FSS approaches employed thus far, and of GP, are rather difficult to discern due to the nature of

the pre-processed dataset which lead to overly complex equations in the GP case. Finally, the need for an automated mechanism to handle the complexity and subjectivity in both nominal/ordinal categorical and numerical data is identified as a prerequisite for project datasets containing this form of values. Thus, in the subsequent methodologies described we have worked towards this direction.

4.2.3.4 Fuzzy Decision Trees (FDT) in CC-SCE

The methodology described in this section focuses on Fuzzy Decision Trees (FDT) to handle the amount of uncertainty and the associated risks of software cost estimation by incorporating fuzzy logic in regression, classification and interaction detection models. Initially, SCE is approximated by classifying information describing a plethora of past projects and deriving association rules for cost drivers associated with effort. Then, these rules are used to predict the actual development effort. Particularly, the Chi-squared Automatic Interaction Detection (CHAID) (Kass, 1990) and Classification and Regression Trees (CART) (Breiman et al., 1984) algorithms are used for deriving a finite set of association rules.

The dataset initially used was the ISBSG R9, which went through the pre-processing steps described in Table 4.1 (a), (c)-(g) and (o) and was named ISBSG R9-9. The dataset included 961 projects and 12 attributes also summarised in Table 4.28. Three subsets of the dataset were used in the experiments, namely ISBSG R9-9.1, ISBSG R9-9.2 and ISBSG R9-9.3. ISBSG R9-9.1 included all the variables of ISBSG R9-9, ISBSG R9-9.2 included only the ordinal and numerical attributes, excluding all categorical attributes (i.e., PET, PIT, PDRU, AFP and ATS) and ISBSG R9-9.3 included from the ordinal and numerical attributes of ISBSG R9-9.2 only the attributes that may be measured ‘early’ (i.e., after specifications are defined) in the project life-cycle (i.e., AFP, PDRU and ATS). In addition, for the target (dependent) variable (i.e., development effort) the transformation step described in Table 4.1 (q) was performed to include the integration of a function. This transformation is usually used

for solving equations in which the unknown variable appears as the exponent of some other quantity.

Table 4.28: ISBSG R9-9 cost factors selected for the classification experimentation

Code	Factor Name	Description
EFF[a]	Full Cycle Work Effort	Total effort (in hours) recorded against the project
PET	Project Elapsed Time	Total elapsed time for the project (in calendar months)
PIT	Project Inactive Time	The number of calendar months in which no activity occurred
PDRU	Project PDR (ufp)	Project delivery rate (in hours per function point) which equals to the quotient of effort and functional size
AFP[c]	Adjusted Function Points	Functional size of the project at the final count
ATS[c]	Average Team Size	Average number of people that worked on the project
DT[b,c]	Development Type	Description of whether the project is a New Development, Enhancement or Re-development
AT[b,c]	Application Type	Description of the application addressed by the project
DP[b,c]	Development Platform	Description of the primary development platform (e.g., PC, Multi-Platform etc.)
LT[b,c]	Language Type	Definition of the language type used by the project (e.g., 3GL, 4GL, etc.)
RL[b,c]	Resource Level	Describes the four levels about the people whose time is included in the work effort data reported
ID[d]	Implementation Date	Describes the actual implementation date of the software
PPL[d]	Primary Programming Language	Description of the primary programming language used (e.g., JAVA, C++, etc.)

a Dependent variable
b Attribute participating in the Categorical Driver Scheme (*DS=Cat*)
c Attribute participating in the Early Driver Scheme (*DS=Erl*)
d Attribute excluded after the preliminary experimentation (Andreou and Papatheocharous, 2008b)

The methodology was initiated by devising instances of FDT that yielded robust rules, which could then be used for obtaining a hierarchy of significant project attributes participating in the rules and estimating effort using the mean effort values of the fuzzy range (Andreou and Papatheocharous, 2008b). The solutions are represented as a tree (an example is shown in Figure 4.14 for the ISBSG R9-9.2 dataset) which is interpreted by rules of the form "***If*** *(condition 1 AND condition 2 AND … AND condition N)* ***then*** *Z*", where the conditions are extracted from the nodes and *Z* is the root. Each path from the root node to a terminal node corresponds to a fuzzy rule.

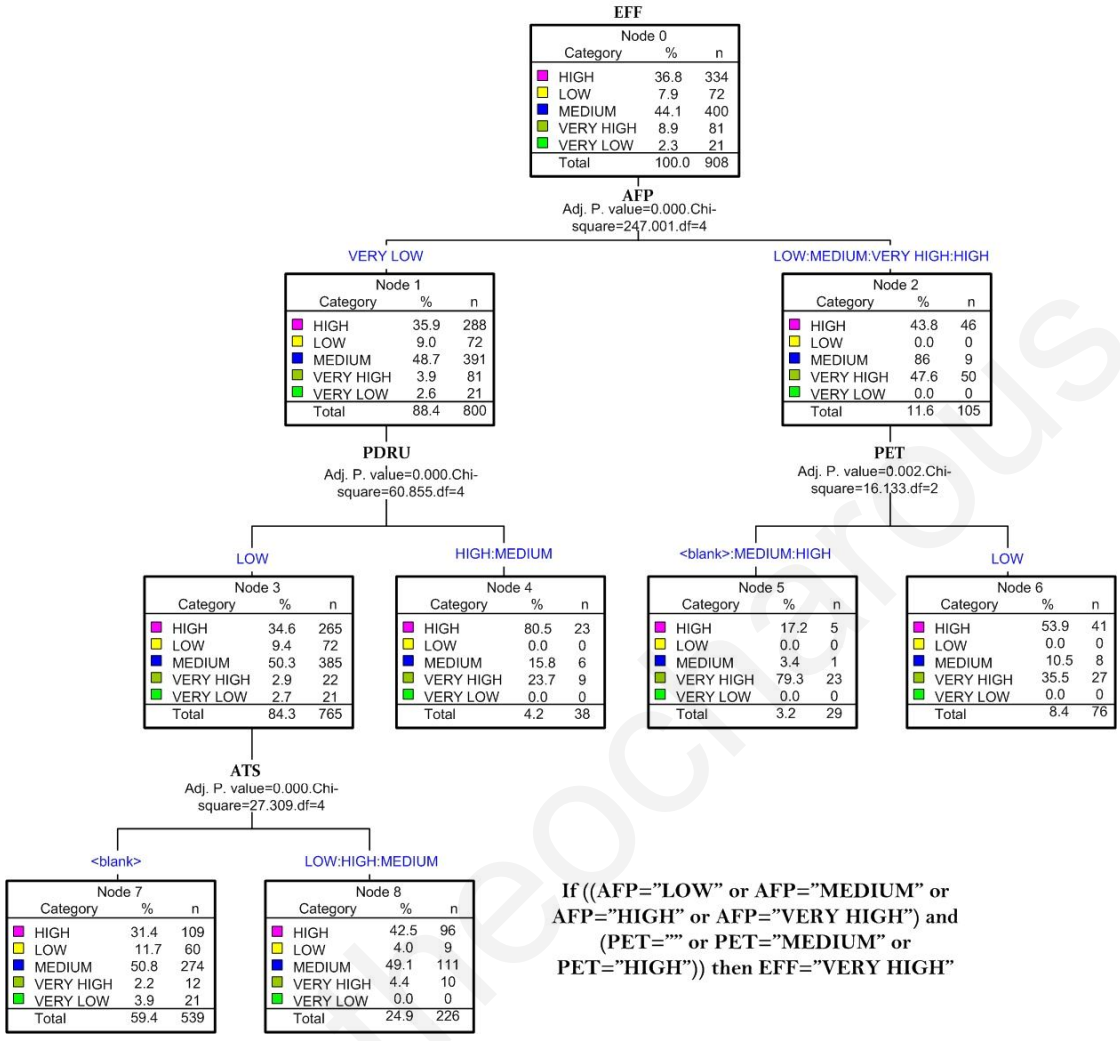


Figure 4.14: A Fuzzy Decision Tree (FDT) example with Association Rule

Figure 4.15 summarises the FDT approach employed (Andreou and Papatheocharous, 2008b). The fuzzification enhances the benefits of decision trees offering them significant advantages compared to other machine learning techniques regarding their ability to produce accurate predictive tools and extract self-descriptive rules in a way that is easier to interpret by individuals. In the experimental exploration performed, the following empirical parameter settings were set for the CHAID algorithm: the significance level value was set to 0.05 and the chi-squared test statistic used was the Pearson and for the CART algorithm: the minimum change in improvement was set to 0.0001 and both Twoing and Gini splitting methods were used. Twoing groups the splitting variable's categories into two subclasses and the best possible splits are found that separate these two groups. Whereas, Gini also finds appropriate

splits to maximise the homogeneity of child nodes with respect to the value of the dependent variable and is based on the probabilities of membership for each category of the dependent variable.

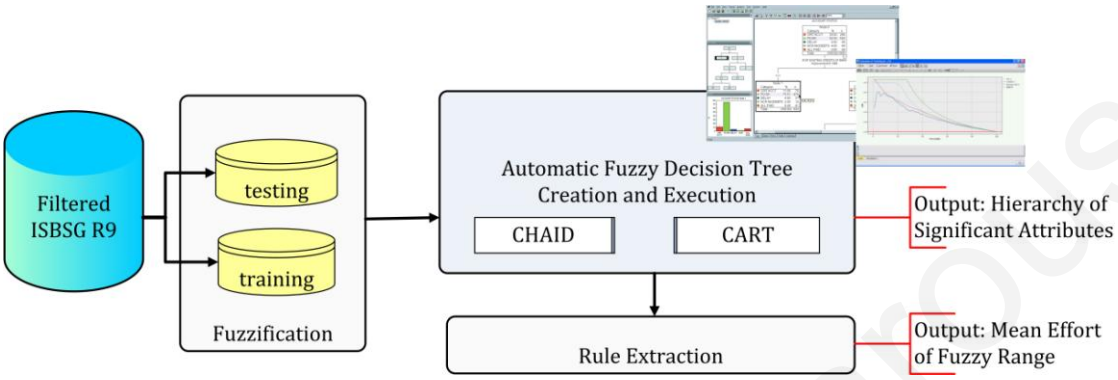


Figure 4.15: SCE using the mean of the Fuzzy Range of Fuzzy Decision Trees

In the preliminary approach (Andreou and Papatheocharous, 2008b) the set of rules with high significance level (over the value of 50%) indicated in the ISBSG R9-9.1 case that the attributes participating more often in most of the rules presented the following order: firstly AFP, secondly, PET, PDRU, ID, DT, LT and thirdly PIT and PPL. In the ISBSG R9-9.2 case, only the attributes AFP, PDRU and ATS participated in the trees (i.e., the rest of the attributes were pruned). Taking into consideration these observations, the set of rules obtained from the attribute subsets ISBSG R9-9.1-3 were used to calculate the indicative preliminary effort predictions (listed in Table 4.29 and used the attributes listed in each case) using the mean effort of the fuzzy effort values of the testing projects that satisfied the rules. Thus, the estimation was based on the mean effort values of the fuzzy range for those projects. The estimation was performed on the transformed values.

Table 4.29: Indicative preliminary FDT prediction results using the mean fuzzy range of effort

Dataset	***MMRE***	***CC***	***NRMSE***
ISBSG R9-9.1={AFP, PET, PDRU, PIT, ATS}	0.13	0.53	1.04
ISBSG R9-9.2={AFP, PET, PIT, ATS}	0.11	0.65	0.87
ISBSG R9-9.3={AFP, PDRU, ATS}	0.17	0.45	1.32

The experimental results showed that sufficiently accurate cost predictions in terms of *MMRE*, i.e., the approach may achieve estimations close to the actual development costs.

However, the levels of the *CC* and *NRMSE* metrics achieved by the rules were not so encouraging. The rules extracted promoted the linguistic representation of the attributes' associations and provided added value to the SCE process. The approach also has optimised accuracy performance and robustness in relation to the rest previously described approaches. The FDT technique is highly applicable something which suggests that the generation of fuzzy association rules is a fairly good solution in classifying projects and extracting relations that describe the nature of the software development environment. The classification rules obtained are expressed in a comprehensive manner using gradually-defined linguistic terms which are especially practical to project managers.

Since the estimation results obtained were quite encouraging, the aforementioned approach was expanded to an enhanced Fuzzy Decision Tree (FDT) approach (Papatheocharous and Andreou, 2009b). The approach described thereafter improved both the methodology's parameters and the evaluation process so that enhanced classification rules are obtained.

Particularly, the enhanced FDT classification approach, illustrated in Figure 4.16, includes the following stages: (i) Data pre-processing, quality checking and fuzzification under a fuzzy linguistic representation, (ii) Training with Fuzzy Decision Trees (FDT), including creation and evaluation, and (iii) Prediction with the classification rules obtained and class resemblance prediction enhancement, along with validation activities.

Data records were split into two subsets; the first one (containing 70% of the samples) was used to construct the FDT, thus called the training set. The second subset (contained 30% of the samples), called the testing set, was used to test the produced FDT and to assess the efficiency and generalisation of the corresponding rules of the tree. The experiments conducted were iteratively executed, changing several internal parameters of the algorithms in each iteration, to create robust FDT and produce a set of unique strong rules.

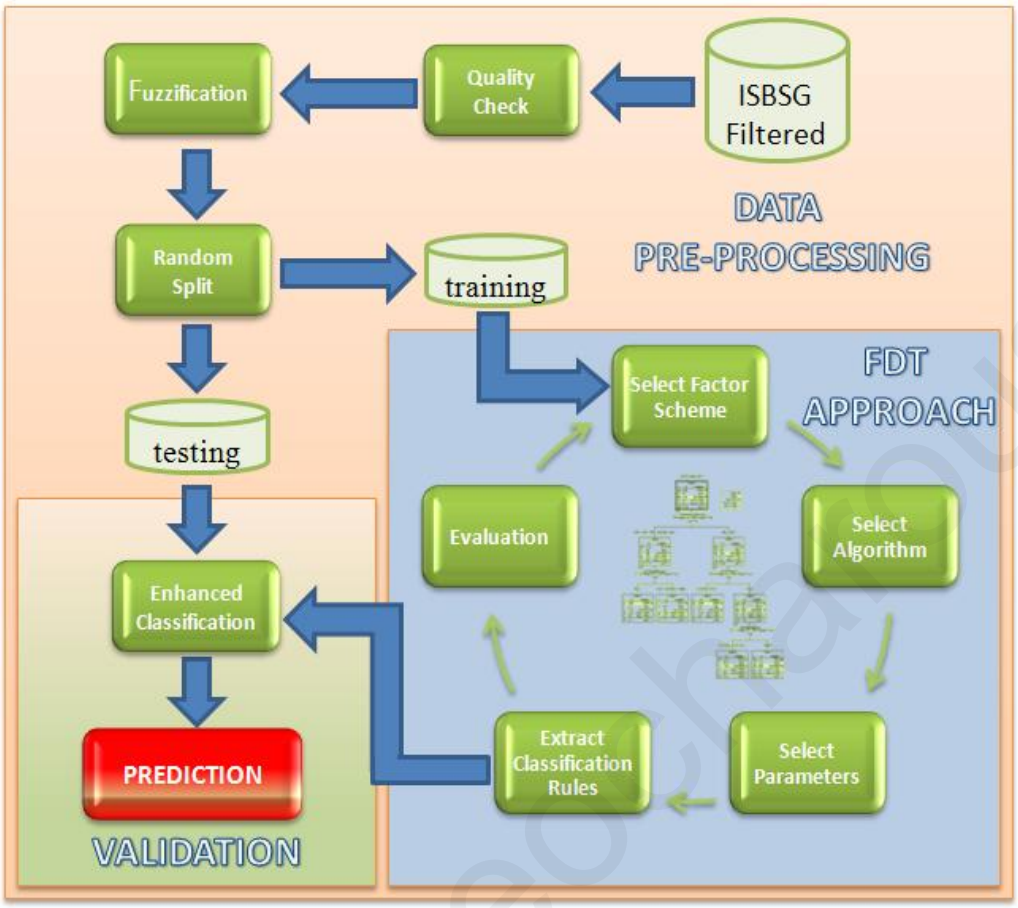


Figure 4.16: Enhanced FDT Approach (Papatheocharous and Andreou, 2009b)

The CHAID, CART and exhaustive CHAID algorithms were executed and in order to find the optimal algorithm parameters that produce the best rules, the rules obtained in each case were evaluated according to their significance level and degree of occurrence among the set of experiments performed. Also, the minimum number of cases per parent and child node for both CHAID and CART were varied and different splitting criteria were examined. The maximum tree depth was confined to the maximum number of variables found in each cost Driver Scheme (*DS*) which is described in this section. For CHAID the significance value was initially adjusted by the Bonferroni method to produce the simplest trees, but also varying values were examined to allow for more splitting nodes. Additionally, the Pearson chi-squared test statistic was used. For CHAID and CART the minimum change in improvement was set to 0.001 and 0.0001 respectively. For CART both the Twoing and Gini splitting methods were tested to either maximise the homogeneity of the child nodes with respect to the value of the

target variable, or to create binary splits. Finally, cross-validation (Maimon and Rokach, 2005) for both CHAID and CART was used with 10 sample folds.

The ultimate aim in this endeavour was to reach to stable rules that include a large number of project characteristics and yield specific clusters of projects improving the overall prediction accuracy. To this end, the deepest trees were finally selected for further experimentation and evaluation. Therefore, a representative set of rules was extracted for further processing.

The evaluation of the FDT created at the training phase was based on a combination of measures. Due to the fact that the technique is data-driven, several FDT generated were identical and thus the rules that appeared more frequently were considered to better describe the distribution of the samples and thus were selected and presented in Table 4.30. Each leaf of the FDT indicates a class or effort value range according to the distribution (Significance Level (*SL*)) and is represented by a classification rule as already mentioned. The variable that classifies the majority of the training samples is placed at the top of the tree and exhibits the most significant relationship with the dependent variable. Also, besides the statistical significance, the 'goodness' of each rule is evaluated on the number of factors participating in the rule (Number of Factors (*NF*)), thus reaching to a more homogeneous cluster of data. The promoted rules are then used for classification and validation.

Some definitions follow (Papatheocharous and Andreou, 2009b). The numbers of the train and test project samples that satisfy a rule *r* are defined as:

$$n_{train,r} = |N_r| \tag{4.42}$$

$$n_{test,r} = |L_r| \tag{4.43}$$

where N_r = *{train project samples that satisfy rule r}* and L_r = *{test project samples that satisfy rule r}*. For each rule *r* satisfied we calculate the mean effort range $\overline{eff_r}$ using eq. (4.44), with σ_{n_r} being the standard deviation of the respective project samples satisfying rule *r*. Essentially, eq. (4.44) takes the mean effort value of the projects that were classified in a certain cluster according to rule *r* as the predicted effort value of the new project (provided

that the new project is also classified in the same cluster), with a deviation tolerance threshold equal to the standard deviation of the projects in the cluster.

$$\widehat{eff_r} = \frac{1}{n_r}\left(\sum_{i=1}^{n_r} eff_{i,r}\right) \pm \sigma_{n_r} \tag{4.44}$$

For the test samples that satisfy rule *r* we define an additional threshold measure, namely Prediction Measure (*PM*) and define the number of samples that satisfy it as:

$$n_{PM,r} = |C_r| \tag{4.45}$$

where C_r = *{test project samples in* L_r *that satisfy inequality (4.46)}.*

$$\frac{1}{n_{train,r}}\left(\sum_{i=1}^{n_r} eff_{i,r}\right) - \sigma_{n_{train,r}} \leq \widehat{eff_r} \leq \frac{1}{n_{train,r}}\left(\sum_{i=1}^{n_r} eff_{i,r}\right) + \sigma_{n_{train,r}} \tag{4.46}$$

The Hit Ratio (*HR*) of *PM* is defined in eq. (4.47).

$$HR_{PM,r} = \frac{|C_r|}{n_{testing,r}} \tag{4.47}$$

So far the predicted effort value has been estimated according to eq. (4.44). This estimation is further enhanced by considering additional factors than those participating in a rule thus improving the homogeneity of the associated cluster. Therefore, for each sample *p* in the test set that satisfies rule *r* the following definitions are considered: Let a Resemblance Threshold (*RT*) for each data scheme (where *j* is the name of the dataset utilised, i.e., takes the values within the set {ISBSG R9-9.1, ISBSG R9-9.4, ISBSG R9-9.5}) as follows:

$$RT_j = \left\lfloor \frac{ds_j}{2} \right\rfloor \tag{4.48}$$

where $ds_{\text{ISBSG R9-9.1}}$=10, $ds_{\text{ISBSG R9-9.4}}$=5 and $ds_{\text{ISBSG R9-9.5}}$=7 and *ds* represents the number of attributes included in each Driver Scheme (*DS*). Let $S_{r,p}$ be a subset of L_r such that $S_{r,p}$ = *{train project samples in* L_r *that have a number of cost factors* $NF \geq RT_j$ *whose values are equal to those of sample p}* which has $n_{r,p}$ elements. Then the enhanced effort estimation is calculated as:

$$\widetilde{eeff}_{r,p} = \frac{1}{n_{r,p}}\left(\sum_{i=1}^{n_{r,p}} eff_{i,r,p}\right) \pm \sigma_{n_{r,p}} \quad (4.49)$$

The Hit Ratio (*HR*) for the enhanced *RM* of the K_r test samples is defined in eq. (4.50), where K_r = *{test project samples in* C_r *that satisfy inequality (4.52)}*.

$$HR_{RM,r} = \frac{|K_r|}{n_{testing,r}} \quad (4.50)$$

$$n_{RM,r} = |K_r| \quad (4.51)$$

$$\frac{1}{n_{train,r,p}}\left(\sum_{i=1}^{n_{r,p}} eff_{i,r,p}\right) - \sigma_{n_{train,r,p}} \leq \widetilde{eeff}_{r,p} \leq \frac{1}{n_{train,r,p}}\left(\sum_{i=1}^{n_{r,p}} eff_{i,r,p}\right) + \sigma_{n_{train,r,p}} \quad (4.52)$$

In the enhanced FDT classification approach (Papatheocharous and Andreou, 2009b) three cost Driver Schemes (*DS*) were employed on the ISBSG R9-9 dataset each consisting of different attributes named *All* (containing all the cost attributes, except the ones marked with the superscript letter [d] in Table 4.28), *Cat* (containing only the categorical attributes, also marked with the superscript letter [b] in Table 4.28) and *Erl* (containing only attributes that are available and can be measured from the early stages of project development, also marked with the superscript letter [c] in Table 4.28). Particularly ISBSG R9-9.1 included all the variables of ISBSG R9-9, ISBSG R9-9.4 included only the categorical attributes DT, AT, DP, LT and RL and ISBSG R9-9.5 included only the attributes that may be measured 'early' (i.e., after specifications are defined) in the project life-cycle namely, AFP, ATS, DT, AT, DP, LT and RL. The logic behind creating these three schemes of data inputs was to assess the power of these cost factor types on effort as this is reflected on the available empirical data samples.

The experimental approach described previously (pg.186) and the most significant rules obtained for each *DS* were extracted. Table 4.30 lists an indicative set of rules selected for further processing (i.e., see the enhanced classification method described above (pg.186). The analysis is based on the hypothesis that the most reliable rules are the ones that perform well during training (high significance levels are observed) and maximise the homogeneity of the set of projects satisfying the rules, i.e., classify appropriately the project samples with similar characteristics.

Table 4.30: Indicative Classification Rules obtained from the ISBSG R9-9 dataset with FDT

DS	Algorithm	NF	SL	'If' Part of Rule and 'Then' Part of Rule
All	CHAID	2	0.974	**IF** ((ATS != "HIGH" AND ATS != "MEDIUM") AND (PET != "MEDIUM" AND PET != "HIGH") AND (DT != "New Development")) **THEN** EFFORT="MEDIUM"
Cat	CART	4	1.000	**IF** (RL != "R4" AND RL != "R3") AND (((DT = "Enhancement" OR DT = "Re-development") OR (DT != "New Development") AND ((DP = "" OR DP = "MF") OR (DP != "Multi" AND DP != "MR" AND DP != "PC") AND ((LT = "3GL" OR LT = "" OR LT = "4GL" OR LT = "2GL") OR (LT != "ApG") AND (RL != "R2"))))) AND (((LT = "4GL" OR LT = "ApG" OR LT = "2GL") OR (LT != "3GL" AND LT != "") AND (DP = "Multi"))) **THEN** EFFORT="MEDIUM"
Erl	exCHAID	7	0.991	**IF** ((DT != "New Development") AND (DP = "MF" OR DP = "Multi") AND (AT!= "" AND AT!= "Stock control & order processing;" AND AT!= "Transaction/Production System;" AND AT!= "Maintenance;")) **THEN** EFFORT="MEDIUM"

The prediction accuracy obtained from validating the rules is shown in Table 4.31.

Table 4.31: Indicative experimental results from enhanced FDT classifications in ISBSG R9-9

DS	Algorithm	SL	TESTING			PM		RM	
			n(288)	$\widetilde{eff}$	σ_n	*HR*	*HR(%)*	*HR*	*HR(%)*
All	CART	0.953	262	7.265	1.354	166/262	63.35	170/262	64.89
All	CART	0.959	261	7.255	1.348	166/261	63.60	169/261	64.75
Cat	CHAID	0.969	153	6.899	1.396	97/153	63.39	99/153	64.71
Cat	CART	0.873	41	6.711	1.169	21/41	51.21	21/41	51.22
Erl	CHAID	0.973	152	6.904	1.399	96/152	63.15	98/152	64.47
Erl	exCHAID	1.000	22	7.420	1.198	10/22	45.45	11/22	50.00

For the 288 test project samples the mean effort and standard deviation values of the respective *n* number of samples that satisfy each rule (displayed in every row of Table 4.31) were calculated. The *Significance Level* (*SL*) obtained from the classified projects is consistently close or equal to the unit, indicating with high confidence that the classification was successfully performed. For the same projects the small standard deviation reports that the range effort and mean effort prediction obtained is near to the actual effort value, indicating relatively good prediction accuracy. Additionally, the best *HR* levels obtained in the testing phase with respect to the *PM* and *RM* thresholds set earlier are 63% and 64% respectively. The performance is not considered particularly high, even though in some experiments some higher figures were obtained (refer to Appendix B, Table B. 16) where for example the level of *HR* is equal to 75% but for only satisfying a small number of projects. In fact, the best in accuracy prediction rule obtained satisfied only a very small number of samples, and therefore we concluded that as more homogeneous clusters of data are created

the more prediction accuracy will probably be improved. Finally, the approach may adequately express the complex relationships among the project attributes under investigation and effort, and within the enhanced classification clusters produced we may achieve overall average estimation accuracy within the range of the values involved for nearly 65% of the cases.

Finally, the main conclusion is that the quality of the prediction results is highly affected by the quality of the data, as well as the homogeneity of the samples. The FDT approach applied formed the basis to discover the interrelations among vital project variables and suggested that categorical cost factors, such as the ones examined, are of high importance in determining the evolution of the final effort spent during development. The contribution of the FDT approach is that such categorical attributes may be taken into consideration in the SCE models created, even though great deal of attention should be given on the quality of especially nominal categorical features. Finally, an interesting next research step is to exploit more 'sophisticated' intelligent mechanisms for the rule exploitation activity.

4.2.3.5 FDT and Fuzzy Implication Systems (FIS) in CC-SCE

Fuzzy Implication Systems (FIS) are developed in this thesis for complementing the successfully created FDT for SCE described in the previous section. The aforementioned classification methodology with FDT is expanded by extracting the classification association rules generated in a systematic manner (Papatheocharous and Andreou, 2012a). The rules are merged in FIS systems which subsequently defuzzify the association rules obtained through the Mamdani-Sugeno method (Mamdani, 1977). The methodology addresses the amount of underlying uncertainty in software data introduced by the measurement activities, the categorical nature (i.e., linguistic) and the non-existence of clear definitions of the software data samples by incorporating fuzzy logic concepts. Moreover, the result from the approach offers to the estimator (and usually to the project manager) the ability to understand the rationale behind the estimate and interpret the result, before adopting it. The approach has

three objectives: (i) To approximate the problem of software cost estimation as accurately as possible, (ii) To generate comprehensible results and tackle, to the best possible extent, the inherent uncertainty of the development process and the uncertainty introduced by the measurement activities of project sample data, as well as (iii) develop Intelligent Systems for carrying out the cost estimations. The main contribution is overcoming limitations like the uncertainty of the metrics and the estimate itself i.e., impreciseness and vagueness, and additionally, generating an interpretable output that is easily understood by the end-users.

The proposed fuzzy cost estimation system involves four stages as illustrated in Figure 4.17 (Papatheocharous and Andreou, 2012a): (i) Pre-processing of the cost driver data, (ii) Creation and evaluation of FDT, (iii) Implementation of the FIS, and (iv) Effort Estimation. The final output of the supporting methodology consists of estimations obtained by defuzzifying the FIS predictions with specific techniques. In addition, the methodology attempts to determine a plausible way to derive a hierarchy of significant attributes that are engaged in the rules used for prediction with the FIS.

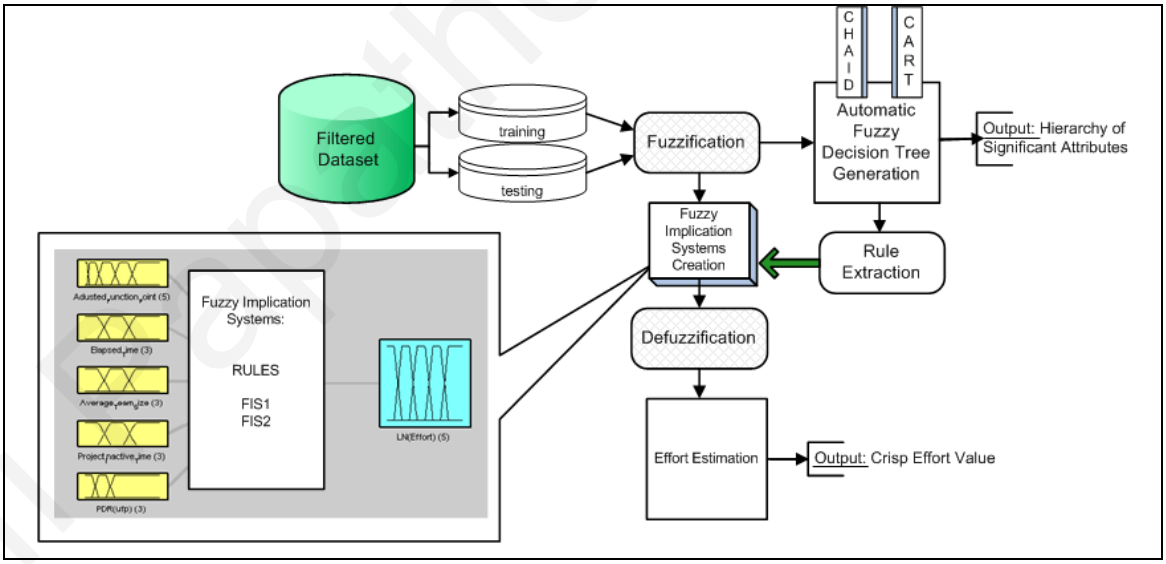


Figure 4.17: Methodology of FDT combined with FIS (Papatheocharous and Andreou, 2012a)

More details on the steps performed by the Fuzzy Cost Implication System are provided subsequently in this section. In the hybrid FDT and FIS methodology the attributes were pre-processed by the steps of Table 4.1 (a) for the COCOMO, Desharnais and ISBSG R9 datasets

and are summarised in Table 4.32 for the COCOMO dataset and in Table 4.33 for the Desharnais dataset. The COCOMO consisted of 63 and the Desharnais of 77 projects. The ISBSG R9 went through additionally the pre-processing steps described in Table 4.1 (c)-(g), (i) and (o) and was named ISBSG R9-9.2. The ISBSG R9-9.2 was also used for 'Early' SCE, and the first five cost drivers described in Table 4.28 (after the dependent variable) were used reporting data for 466 projects. The attributes used in the 'early' cost driver scheme of the last experiments conducted were the PDRU, AFP and ATS. All dependent variables (marked with [a] superscript in the respective tables) were transformed according to the step (q) in Table 4.1.

Table 4.32: Summary of the COCOMO software cost attributes

Code	Attribute name	Description
EFF[a]	Development Effort	Development effort (in man-months)
LOC	Lines of Code	Project size (in delivered source instructions)
RELY[b]	Required Reliability	Extent to which a software product is expected to satisfactory perform its intended functions
DATA	Database Size	Relative database size to be developed (size refers to the amount of data to be assembled and stored in non-main storage: D/P = (Database size in bytes or characters)/(Program size in SLOC))
CPLX[b]	Product Complexity	Subjective average of four types of functions: control, computation, device-dependent, or data management operation
TIME[b]	Execution Time Constraint	Degree of execution time constraint imposed upon a software product (expressed in terms of available execution time expected to be used)
STOR[b]	Main Storage Constraint	Percentage of main storage expected to be used by the software product and any subsystems consuming the main storage resources
VIRT[b]	Virtual Machine Volatility	Level of volatility of the virtual machine underlying the software product to be developed. The virtual machine is defined as the complex of hardware and software the product will call upon to accomplish its tasks
TURN	Computer Turnaround Time	Level of computer response time experienced by the project team developing the software product
ACAP[b]	Analyst Capability	Ratings for analyst capability (expressed in terms of percentiles with respect to the overall population of software analysts)
AEXP[b]	Applications Experience	Level of equivalent applications experience of the project team developing the software product
PCAP[b]	Programmer Capability	Capability of the programmers working on the software product. The ratings are expressed in terms of percentiles with respect to the overall population of programmers
VEXP[b]	Virtual Machine Experience	Experience of the project team with the complexity of hardware and software that the software product requires to accomplish its tasks, e.g. computer, operating system, and/or database management system
LEXP[b]	Programming Language Experience	Level of programming language experience of the project team developing the software project (defined in terms of the project team's equivalent duration of experience with the programming language used)
MODP[b]	Modern Programming Practices	Degree to which modern programming practices are used in developing the software
TOOL[b]	Use of Software Tools	Degree to which software tools are used in developing the software product
SCED[b]	Required Development Schedule	Level of constraint imposed on the project team developing a software product. Ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort

a Dependent variable
b Attribute participating in the 'Early' cost driver scheme

Table 4.33: Software cost drivers description for the Desharnais dataset

Code	Attribute name	Description
DEFF[a]	Effort	Actual development effort (in person-hours)
FPNA[b]	Function Points Non-Adjusted	Unadjusted Function Points count
FPA	Function Points Adjusted	Adjusted Function Points count
TE[b]	Team Experience	Team experience (measured in years)
ME[b]	Manager Experience	Manager's experience (measured in years)
DU	Duration	Duration of development (measured in months)
TR[b]	Transactions	Count of basic logical transactions in the system
EN	Entities	Number of entities in the systems data model
ENV	Envergure	Scale of the project
LAN[b]	Language	Development language

a Dependent variable
b Attribute participating in the 'Early' cost driver scheme

The basic procedure followed is explained using one of the aforementioned datasets, i.e., the ISBSG R9-9.2, as the example-guide. Initially, each cost driver of the dataset was associated with a different fuzzy set described by the trapezoidal membership function. The shape of a membership function essentially defines how well an adaptive fuzzy system approximates each cost factor. The membership function determines the degree to which each dataset attribute belongs to a certain fuzzy set through ordinal transformation of each variable to its fuzzy counterpart.

The separation of each variable into bins of similar size was performed by carrying out some preliminary classification trees with the raw dataset data (before any type of transformation/fuzzification took place). The splitting point for the numerical values of attributes whose value ranges were relatively large (i.e., of the order of thousands) was set to 5 ordinal intervals, whereas those that had smaller value ranges (i.e., of the order of hundreds) was set to 3 ordinal intervals This conversion to ordinal intervals was clearly empirical following common practices adopted in similar cases in literature (Braz and Vergilio, 2004) and Table 4.34 summarises the specific continuous ranges estimated.

Table 4.34: Fuzzy Interval Values for the ISBSG R9-9.2

Cost Factors	VERY LOW	LOW	MEDIUM	HIGH	VERY HIGH
FCWEFF	≤ 3.90	3.91 - 5.73	5.74 - 7.55	7.56 - 9.37	≥ 9.38
AFP	≤ 3506	3507 - 7010	7011 - 10514	10515 - 14017	≥ 14018
PET	-	≤ 17.23	17.24 - 34.65	≥ 34.66	-
PROD	-	≤ 129	129 - 258	≥ 258.10	-
PIT	-	≤ 4.0	4.10 - 7.90	≥ 8.0	-
ATS	-	≤ 26.29	26.30 - 51.62	≥ 51.63	-

Before moving on the dependent variables (FCWEFF in the ISBSG case, EFF in the COCOMO case and DEFF in the DESHARNAIS case) were normalised using the natural logarithm transformation. The aforementioned process caused the FCWEFF and AFP attributes obtained from the ISBSG dataset to be encoded using an ordinal scale of five values ('*VERY LOW*', '*LOW*', '*MEDIUM*', '*HIGH*', '*VERY HIGH*') and the rest four attributes using a scale of three linguistic values ('*LOW*', '*MEDIUM*', '*HIGH*') according to the variance of each attribute.

Figure 4.18 until Figure 4.23 illustrate the trapezoidal functions formed for the attributes of (a) Full-Cycle Work EFFort (*ln*(FCWEFF)), (b) Adjusted Function Points (AFP), (c) Project Elapsed Time (PET), (d) Project Delivery Rate (PROD), (e) Project Inactive Time (PIT) and (f) Average Team Size (ATS). The corresponding numerical boundaries are listed in Table 4.35.

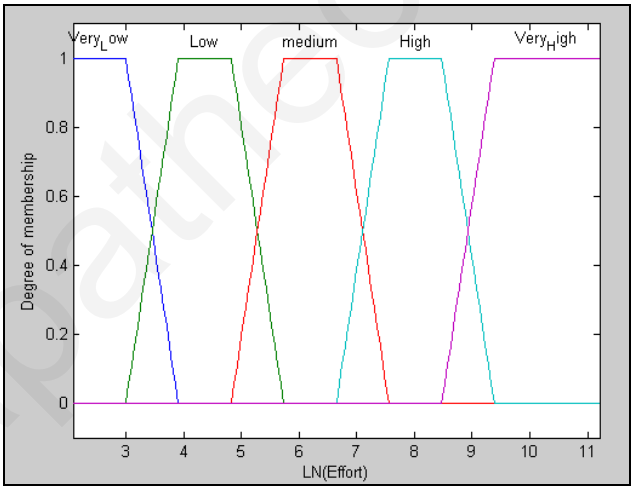


Figure 4.18: Membership function of attribute work effort for the project full-cycle *ln*(FCWEFF)

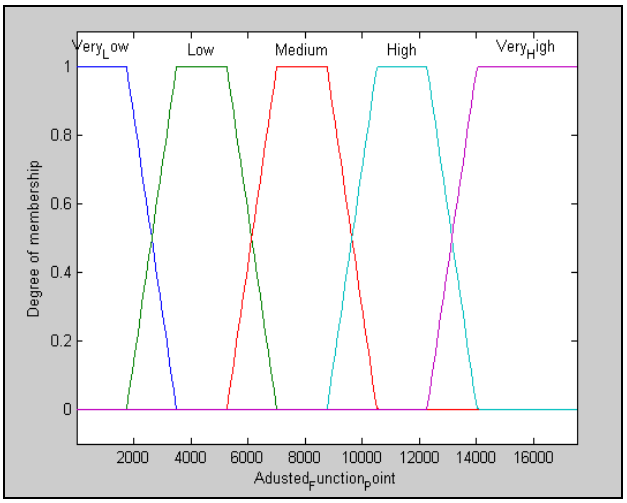


Figure 4.19: Membership function of attribute Adjusted Function Points (AFP)

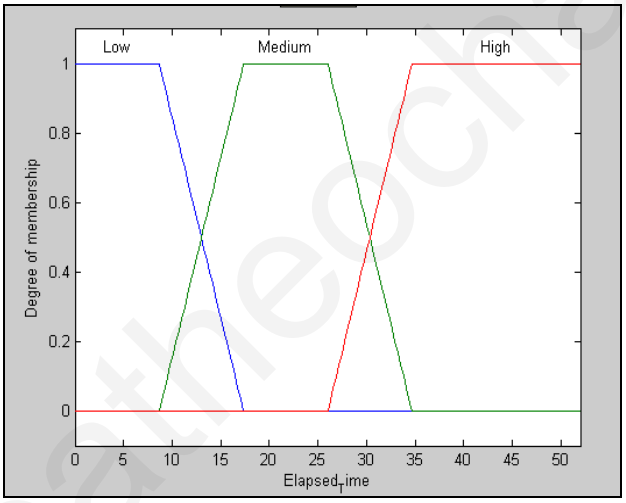


Figure 4.20: Membership function of attribute Project Elapsed Time (PET)

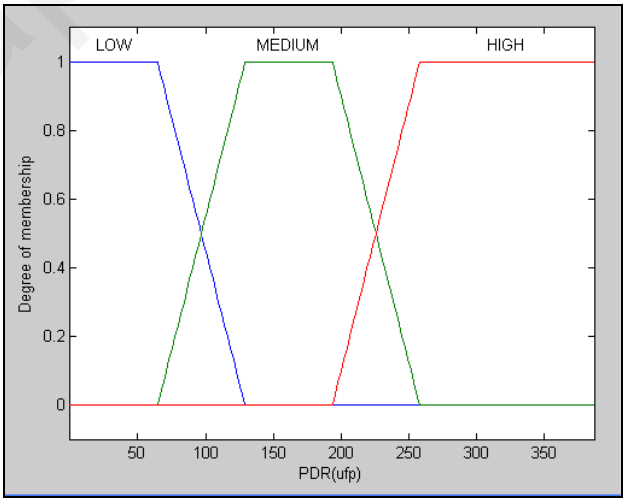


Figure 4.21: Membership function of attribute Project Delivery Rate (PROD)

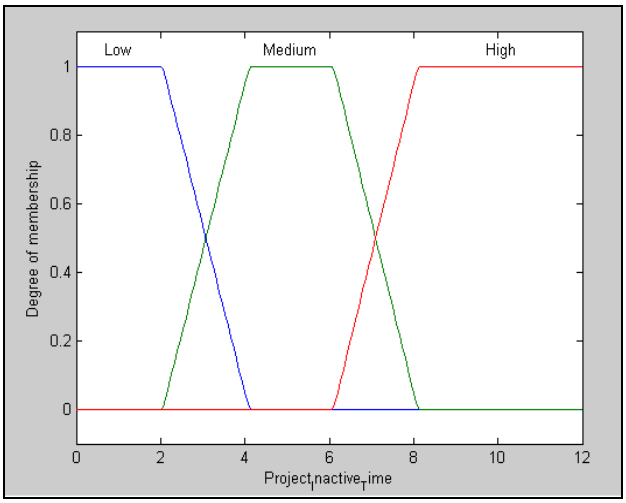


Figure 4.22: Membership function of attribute Project Inactive Time (PIT)

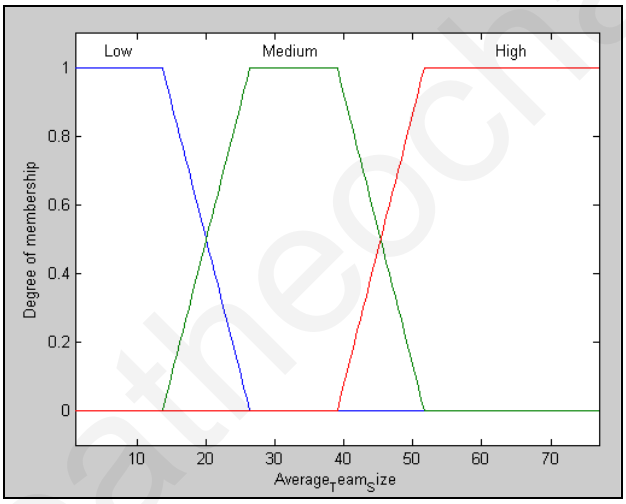


Figure 4.23: Membership function of attribute Average Team Size (ATS)

Table 4.35: Fuzzification values for the cost factors of the ISBSG R9-9.2 dataset

	FCWEFF	AFP	PET	PROD	PIT	ATS
m1	2.07	3	0	0.1	0	0.95
n1	2.99	1755	8.67	64.6	2.05	13.62
b1	3.90	3507	17.34	129.1	4.1	26.3
m2	3.90	3507	17.34	129.1	4.1	26.3
n2	4.81	5259	26	193.65	6.1	38.97
a2	2.99	1755	8.67	64.6	2.05	13.62
b2	5.73	7011	34.67	258.2	8.1	51.64
m3	5.73	7011	34.67	258.2	8.1	51.64
n3	6.64	8763	-	-	-	-
a3	4.81	5259	26	195.65	6.1	38.97
b3	7.55	10515	-	-	-	-
m4	7.55	10515	-	-	-	-
n4	8.47	12267	-	-	-	-
a4	6.64	8763	-	-	-	-
b4	9.38	14019	-	-	-	-
a5	8.47	12267	-	-	-	-

The subsequent experiments used the pre-processed data (i.e., after normalisation and transformation into fuzzy linguistic values was performed according to the ranges described above; see also steps (q) and (o) in Table 4.1). Before initiating the experimental process, random splitting was performed taking 70% of the data for the construction of the FDT (training phase) and the rest 30% for their evaluation (testing phase). The FDT were implemented using SPSS v.17.0 and both the CHAID and CART algorithms were applied. The trees generated numerous rules, starting from the parent node and reaching to a leaf, which were associated with a respective significance level. The rules achieving a significance level of over 65% were selected no matter the tree depth yielded by the corresponding algorithm and were included in the FIS. Additionally, the FDT were used to extract a hierarchy of the most significant project attributes appearing at the top-most nodes of the trees.

The fuzzy implication process expresses the rules using fuzzy sets represented by their membership function, which weighs appropriately the linguistic characteristics that are attributed to each set. Once a set of rules describing the relationship between cost factors and effort is available, we may calculate the degree to which each part of the antecedent (i.e., the conditions - the *IF* part of the rule) is satisfied for each rule; if the antecedent of a given rule has more than one part, a fuzzy operator is applied to combine the multiple degrees and obtain a single number that represents the result of the antecedent for that rule. This number is then applied to the output function (the *THEN* part of the rule). Rules combined with the *AND* operator are calculated using the *Minimum* value of the partial membership functions of the participating attributes, while the *Maximum* function is used for the corresponding *OR* combinations. The implication of the rules is performed with the *Minimum* function and the aggregation, i.e., the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set, were executed using the *Maximum* function.

The final step includes the defuzzification process. Defuzzification translates back the membership value of the corresponding fuzzy set into a single crisp value of the initial set of

real numerical values. Defuzzification follows a certain computational method to calculate the final estimation of a numerical attribute (in our case the development effort).

Figure 4.24 presents graphically an example of the whole process for the composite rule "*IF* (AFP = '*VERY LOW*' and (PET = '*MEDIUM*' or PET = '*HIGH*') *THEN* FCWEFF = '*HIGH*'". The rule is broken down into two parts, namely "*IF* (AFP = '*VERY LOW*' and PET = '*MEDIUM*') *THEN* FCWEFF = '*HIGH*'" and "*IF* (AFP = '*VERY LOW*' and PET = '*HIGH*') THEN FCWEFF = '*HIGH*'". Each rule is evaluated separately and then the two results are combined. For each input and for each attribute the degree of membership is calculated (according to the membership functions in Figure 4.19 and Figure 4.20). In the first rule AFP=1000 and $\mu_{1,AFP}$=1. Also, PET=30 and $\mu_{1,PET}$=0.5. Since the two factors are coupled with the *AND* operator the joint membership value is equal to their minimum, that is, $\mu_{1,FCWEFF}$=0.5. The same process is applied for the second rule yielding $\mu_{2,FCWEFF}$=0.5. The aggregation of the two rules, if we assume the LOM defuzzification method for demonstration purposes, evaluates to the largest of maximum, thus the effort value corresponds to the partial membership value μ_{FCWEFF}=0.5, which gives *ln*(FCWEFF)=8.84.

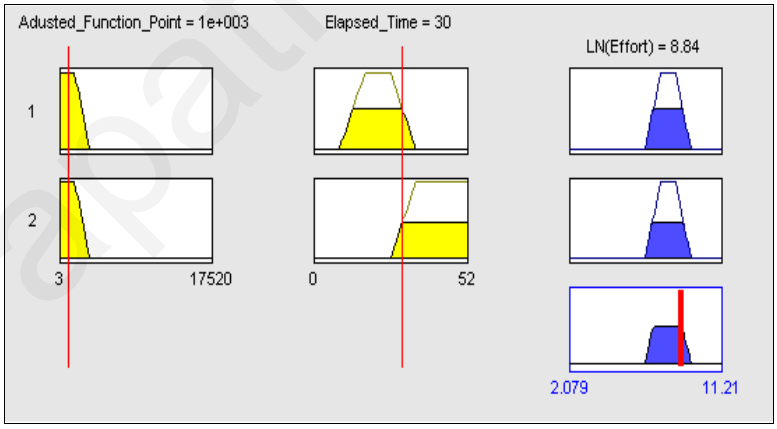


Figure 4.24: An example of rule aggregation and defuzzification.

The final part of the experimental process included the fuzzy cost estimation model implementation (carried out in Matlab R2007b) for each dataset and cost driver scheme. This included encoding the rules, ranges and membership functions for the participating attributes and producing six FIS instances, two for each dataset, as follows: The first (FIS_1) comprised the full spectrum of the available attributes, while the second (FIS_2) included only those

attributes that may be measured early in the development process. Finally, defuzzification was performed with the rules produced being aggregated and implicated to obtain numerical effort estimations from their fuzzy linguistic counterparts. An indicative set of rules obtained from the FDT is listed in Table 4.36.

Table 4.36: Indicative if-then rules obtained using FDT and the CHAID and CART algorithms

Dataset	If-then Rules	Significance
ISBSG R9-9.2	*IF* ((AFP = "LOW" *OR* AFP = "MEDIUM" *OR* AFP = "VERY HIGH" *OR* AFP = "HIGH") *AND* (PET = "" *OR* PET = "MEDIUM" *OR* PET = "HIGH")) *THEN* FCWEFF = "VERY HIGH"	79%
ISBSG R9-9.2	*IF* ((AFP = "VERY LOW") *AND* (PDRU = "LOW") *AND* (PET = "MEDIUM" *OR* PET = "HIGH")) *THEN* FCWEFF = "HIGH"	66%
COCOMO	*IF* ((DATA != "HIGH") *AND* (AEXP != "HIGH" *AND* AEXP != "LOW") *AND* (VIRT != "HIGH")) *THEN* EFF = "MEDIUM"	80%
COCOMO	*IF* ((DATA != "HIGH") *AND* (AEXP = "HIGH" *OR* AEXP = "LOW") *AND* (VEXP = "MEDIUM")) *THEN* EFF = "LOW"	71%
Desharnais	*IF* ((LAN != "HIGH") *AND* (LE != "LOW") *AND* (EN = "LOW" *OR* EN = "HIGH") *AND* (FPNA != "LOW")) *THEN* DEFF = "HIGH"	100%
Desharnais	*IF* ((LAN != "HIGH") *AND* (LE != "LOW") *AND* (EN = "LOW" *OR* EN = "HIGH")) *THEN* DEFF = "HIGH"	91%

The rules indicate a direct relationship between size and effort in all datasets; the former being expressed either in the form of functional aspects (FP), data model entities or database size. This relationship of software size and effort has been also observed in the results of most of the previously described SCE models (e.g., refer to the FSS models of ANN using ISA in section 4.2.2.1, to the attributes selected by the various FSS approaches examined in section 4.2.2.2 and to the GP equations obtained in section 4.2.3.3).

Less direct relationships are also observed such as: Applications experience (AEXP) as well as software and hardware complexity (VEXP) seem to play a decisive role in the COCOMO case and development language (LAN) in the Desharnais case. This suggests that both technical skills and experience, as expected, especially for projects with high complexity, guide expenses in human resources and hence drive the associated effort spent. Another important observation is that the cost attributes found significant are available (known) from the launch of a software project (e.g., skills and experience of the technical team experts). Nevertheless, this is a finding that must be interpreted cautiously as the results of the experiments with the datasets that contain categorical attributes, in our case COCOMO and Desharnais, do not agree as to which specific attributes representing skills and experience are

considered more significant. This supports the argument made earlier that the diversification in the number and type of factors measured in each dataset, as well as the different measurement processes followed and the variety of software markets and/or cultures of the development organisations providing the metrics, may affect the level of significance of the participating factors with respect to their explanatory power over effort. In other words, it seems that there are more complex relationships than one may expect between the factors measured, with complexity driving also the kind of relationship these factors may have with effort. Overall, one may also argue that the Desharnais and ISBSG R9-8 datasets follow the philosophy of measuring tangible aspects of the development process (i.e., the process itself), while COCOMO focuses mostly on characteristics that constrain the development process. This also enforces the argument that the datasets are quite diversified and that the measurement activities performed for each one possibly lead to highly uncertain (subjective) and, to a large extent, inconsistent values. Therefore, any differences observed regarding the attributes' significance between datasets may lead to deriving independent conclusions for each case.

Defuzzification was performed with more than one method, specifically, *Centroid* (*CEN*), *Bisector* (*BIS*), *Middle of Maximum* (*MOM*), meaning the average of the maximum value of the output set, *Largest of Maximum* (*LOM*) and *Smallest of Maximum* (*SOM*). The results of the hybrid FDT & FIS model are juxtaposed in Table 4.37.

Table 4.37: Performance results of the FDT & FIS hybrid SCE model

Defuzzification method	Error Metric	ISBSG R9-8		COCOMO		Desharnais	
		FIS_1	FIS_2	FIS_1	FIS_2	FIS_1	FIS_2
CEN	***MMRE***	0.12	0.18	0.29	0.62	0.19	0.19
	MBRE	0.12	0.20	-0.11	-0.55	-0.16	-0.16
	Pred(.25)	0.91	0.72	0.58	0.26	0.74	0.74
BIS	***MMRE***	0.12	0.18	0.29	0.63	0.19	0.19
	MBRE	0.12	0.20	-0.11	-0.56	-0.16	-0.16
	Pred(.25)	0.91	0.72	0.58	0.26	0.74	0.74
MOM	***MMRE***	0.14	0.18	0.29	0.63	0.20	0.20
	MBRE	0.14	0.21	-0.10	-0.56	-0.18	-0.18
	Pred(.25)	0.87	0.71	0.58	0.21	0.70	0.70
LOM	***MMRE***	0.11	0.16	0.37	0.93	0.27	0.27
	MBRE	0.05	0.13	-0.29	-0.90	-0.25	-0.25
	Pred(.25)	0.92	0.83	0.47	0.11	0.43	0.43
SOM	***MMRE***	0.18	0.22	0.33	0.51	0.15	0.15
	MBRE	0.23	0.29	0.20	-0.17	-0.11	-0.11
	Pred(.25)	0.79	0.61	0.53	0.21	0.87	0.87

A small superiority of the *LOM* defuzzification method is observed in the ISBSG case and of the *SOM* method for the Desharnais. The picture in the COCOMO dataset is clearly different, where the *MMRE* and *MBRE* are noticeably higher than in the other two datasets, while the *Pred(.25)* metric on average indicates also lower resemblance percentages under the threshold of 25%. The best prediction accuracy results in terms of *MMRE* obtained throughout the experiments were 0.11 for the ISBSG R9-9.2 dataset, 0.29 for the COCOMO and 0.15 for the Desharnais dataset, while the corresponding *MBRE* values were 0.05, -0.10 and -0.11 respectively.

Overall, the accuracy performance obtained from the hybrid FDT and FIS approach (or FDT & FIS model) described in this section, presents quite successful results that sometimes outperformed the results of previous work using normalised values for the estimation. The results are directly comparable for the Desharnais case with the previously described approaches of ISA combined with ANN since the same filtered dataset was used and the performance was measured on data transformed with the same manner. However, for the ISBSG since different filtered versions and subsets of project data were employed, many results presented thus far are not directly comparable to the hybrid FDT and FIS approach.

A more detailed comparison taking the above considerations is provided next. Particularly, the SCEs obtained with the Desharnais dataset in previous work (i.e., refer to the results in pg. 126 and pg. 133 in section 4.2.2.1) were better using the whole spectrum of attributes compared to the hybrid FDT & FIS approach described in this section. However, using the 'early' attributes in the FIS_2 of the hybrid model, i.e., those attributes that can be measured from the early project phases, the predictions outperform the majority of the effort predictions obtained with ANN (i.e., using the attributes selected by the ISA and the *Strict* (*S*), *Less Strict* (*LS*) and *Early* thresholds (refer to pg. 126)), while they are comparable to the results obtained with the *Strict* and *Relaxed* evaluation sets of ISA and ANN (refer to pg. 133).

As already mentioned, even though different datasets for the ISBSG were utilised in each experiment previously explained, the ISBSG dataset results are also compared here. Particularly, the results of the FDT & FIS approach are worse compared to the ANN results obtained with all the range of attributes and with the selected attributes from all types of ISA and thresholds (refer again to pg. 126 and pg. 133 of section 4.2.2.1) except the results obtained with the *Strict* evaluation set of attributes attained from the ISA on ANN.

Moreover, the results of the hybrid FDT & FIS model using the ISBSG dataset also outperform the results from the estimation of effort from the mean fuzzy range of the classified projects in the initial FDT constructed (refer to pg. 184 from section 4.2.3.4) in two out of the three set of attributes utilised. The third attribute case (i.e., ISBSG R9-9.2) achieved the same exact accuracy with the accuracy obtained with the hybrid FDT & FIS model. More detailed comparisons are provided in the next chapter. Ultimately, the approach presented in this section has therefore showed several improvements and that quite accurate cost predictions close to the actual development costs can be achieved. The following investigation relates to improving the type of prediction results obtained, to predictions based on intervals, even though predictions based on intervals have already been proposed (refer to CS & GA model in section 4.2.3.1, Fuzzy clustering and by-analogy estimation in section 4.2.3.2, GP estimations in the ISBSG case in section 4.2.3.3 as well as fuzzy interval estimation with FDT in section 4.2.3.4).

4.2.4 Predictive Intervals of Software Cost Estimation (PI-SCE)

The notion of Conformal Predictors (CP) is used to produce Predictive Intervals (PI), or regions, that satisfy a pre-determined level of confidence in the obtained estimations. The major advantage of the proposed method of Predictive Intervals in SCE (PI-SCE) is its flexibility to be applied or adopted in combination with any predictive technique. Additionally, a significant aspect is that in CP given a confidence level 1-ε and a computational method that produces a prediction of effort (such as Ordinary Least Regression

(OLS), Artificial Neural Networks (ANN)), these predictions typically will include the real effort values with probability (*1- ε*) (Papadopoulos et al., 2009). Therefore, in cases where the sample data is independent and identically distributed (i.i.d) the yielded predictive intervals of CP will also be well-calibrated, that is, they will contain the true effort value within a relative frequency of at least 1-*ε* (Nouretdinov et al., 2001).

Based on a sample of projects $\{(x_1,z_1), \ldots, (x_n,z_n)\}$, where $x_i \in \mathcal{R}^n$ is a vector consisting of the independent (input) variables and $z_i \in \mathcal{R}$ is the dependent variable (effort) for each project *i*, the prediction intervals are produced under the assumption that all pairs (x_i, z_i) are i.i.d. and a value a_i is assigned to each pair representing the 'strangeness' or non-conformity of the pair (x_i, z_i) to the rest of the pairs in the sample. This non-conformity value may be calculated using any machine learning algorithm. In this particular case the Ridge Regression (RR) algorithm has been selected and is already described in section 3.2.2.2.

Next, the non-conformity score a_n of the pair (x_n, z_n) is compared to the non-conformity scores of all other pairs to calculate the degree of non-conformity according to eq. (4.53).

$$p((x_1, z_1),...,(x_n, z_n)) = \frac{\#\{i = 1,...,n : a_i \geq a_n\}}{n} \tag{4.53}$$

If $p(z_n)$ is under some low threshold, e.g., 0.05, then it means that z_n is 'unlikely' as the probability of occurring is at most 5%. By calculating the *p*-value of every possible dependent variable using the above equation, all prediction variables under a very low threshold, or significance level *ε*, that have at most *ε* probability of being correct may be excluded.

Finally, CP for a significance level (*1-ε*) calculates the following set:

$$\{z_n : p(z_n) > \varepsilon\} \tag{4.54}$$

which represents the predictions that have a *p*-value greater than the specified significance level *ε*. Thus, instead of considering every possible prediction in $\mathcal{R}$ the technique uses eq. (4.54) (Nouretdinov et al., 2001). The algorithm is referred to as RR & CP model for SCE.

The datasets used for the experimentation were the COCOMO, the Desharnais and the ISBSG R9 dataset which went through the pre-processing activities described in Table 4.1 (a). Additionally, in the ISBSG R9 case steps (c), (f), (k)-(m) were executed that yielded the

ISBSG R9-4 summarised in Table 4.16. Thus, the COCOMO consisted of 63 projects, the Desharnais of 77 projects and the ISBSG R9-4 467 projects. The attributes of the COCOMO dataset are summarised in Table 4.32 while the attributes of the Desharnais are listed in the first part of Table 4.9.

The experiments conducted followed a 10 fold cross-validation process (Maimon and Rokach, 2005). Each data set was split randomly into 10 parts of almost equal size and the tests were repeated 10 times, each time using one of the 10 parts as the testing set and the remaining 9 as the training set. This procedure was repeated 100 times for each dataset and the results reported refer to the results obtained across all runs. The parameter γ of the RBF kernel, as well as the *ridge parameter* α, were typically determined by trial-and-error methods. Particularly for γ the values from 2 to 7 with increments of 0.5 were tried and for α values from the set {0.5, 0.1, 0.05, 0.01, 0.005, 0.001} were examined.

The best results obtained with the RR algorithm are reported in Table 4.38 whereas the application of the hybrid RR & CP yielded the results shown in Table 4.39 for 95%, 90% and 80% confidence levels to every project in each of the 10 parts of each dataset, using as training set the other 9 parts. The process was repeated 100 times and thus resulted in 100 x *N* predictive intervals for each confidence measure, where *N* represents the number of projects in each dataset. The *Median Width* (*MW*) and the *Median Relative Width* (*MRW*) for each of the three confidence levels (95%, 90% and 80%) was calculated. The MRW corresponds to the tightness of the *MW* and is calculated by dividing the predictive interval produced for each project with its actual effort value. The *Percentage of Errors* (*PE*) reports the reliability of the predictive intervals, that is, the percentage of projects for which the true effort value is not inside the interval outputted by the model. In effect *PE* checks empirically the validity of the predictive intervals produced.

Table 4.38: Best testing results of RR and the corresponding parameters

Dataset	γ	α	*MMRE*
COCOMO	5.5	0.001	0.422
Desharnais	5	0.05	0.345
ISBSG R9-4	3.5	1	0.596

Table 4.39: Tightness and reliability of the RR & CP

Dataset	Median Width (MW)			Median Relative Width (MRW)			Percentage of Errors (PE)		
	80%	90%	95%	80%	90%	95%	80%	90%	95%
COCOMO	170.8	278.8	383.4	1.35	2.20	3.11	19.11	8.68	3.68
Desharnais	4549	5579	6396	1.25	1.54	1.76	19.03	9.06	4.38
ISBSG R9-4	4451	6571	9126	1.68	2.46	3.37	20.01	10.02	5.01

The results obtained from the hybrid RR combined with CP (Papadopoulos et al., 2009) indicate lower *MW* for the COCOMO dataset in all three confidence levels compared to the Desharnais and ISBSG R9-4 cases. This means that the effort prediction for the projects enclosed in the latter two cases is harder. In addition, the *MRW* values indicate that promising tightness levels are obtained for 80% confidence level in all the datasets. The best levels of tightness were obtained for the Desharnais, then for the COCOMO and lastly the least promising levels were obtained in the ISBSG R9-4 case. A significant desirable improvement would be obtaining even lower widths of the predictive intervals for higher confidence values. In terms of tightness, however, the optimal results (i.e., the tighter widths) were obtained with the Desharnais dataset in all the three confidence levels (95%, 90% and 80%). Finally, satisfactory PE percentages were reported especially within the 95% confidence interval which indicates that the method yields provable results.

The hybrid RR & CP algorithm offers quite narrow predictive intervals (as shown for the 95% confidence level in the median widths for the COCOMO, Desharnais and the ISBSG datasets that correspond to 3.36%, 27.34% and 6.09% respectively). These were calculated on the whole range of efforts of the respective datasets.

The above observations show that the intervals produced are quite narrow and reliable to be useful in practice. The predictive intervals are also well-calibrated and provide considerably more informative results for a project manager instead of crisp counterpart predictions and can be more useful in practice for considering worst and best-case scenarios for SCE. The subsequent approach is the last SCE method proposed which provides quantitative results and is based on phased predictions calibrated to the time of the estimate.

4.2.5 Phased-Based Software Cost Estimations (PB-SCE)

The examination of the adjustment of work effort progressively through the project life-cycle (SDLC) and along the development phases using historical data of cost factors which are available at the time of the estimation is analysed in this section. This examination aims to identify patterns of development effort distribution in software projects and introduce some emerging opportunities that could improve project planning and controlling. ANN are employed to approximate estimations of effort as accurately as possible, using available project information and effort values of finished project phases. The SCE models developed for each phase are employed to yield Phased-Based effort estimates (PB-SCE) (Papatheocharous et al., 2012). Particularly, the models utilise data obtained from the 'early' project phases (i.e., before the actual implementation phase) and progressively produce estimates for each phase.

The following research questions are investigated: (i) How is the total development effort related with each development phase of the software development life-cycle?, (ii) Is there a correlation between the efforts distributed in the various development phases?, (iii) Updating the estimates during the software development process leads to significant improvement in the accuracy of effort estimates for the total effort?, and (iv) At a fixed point in time of the development process how accurately can effort be approximated for the next phase?

The investigation includes examining the relationship among the development effort of each project phase to each other and to the total work effort through correlation analysis on the pairs of phased-effort values and using ANN. ANN aim to identify patterns of development effort distribution in software projects and approximate estimations of effort as accurately as possible, using available project information and effort values of finished project phases.

The dataset used for the experimentation was the ISBSG R10 from which several attributes and the effort reported in the project phases was selected. Particularly, the pre-processing steps of Table 4.1 (a)-(c), (e)-(i), (k) and (p) were carried out, leading to only 65

projects in the dataset. The attributes formed a group of project attributes useful to carry out comparisons among different project types described by the Development Type (DT), Development Platform (DP), Language Type (LT) and Maximum Team Size (MTS). Also, any effort information available at the time of the estimate was used which involved phased-effort values and total effort reported for the 65 projects maintained in the dataset.

ANN were used to model the problem of cost estimation and predict the value of total effort using prior-phase information of effort. The prediction was carried out in the following four project phases: a) *Early*, using only the available project attributes at the project initiation (i.e., DT, DP, LT and MTS), b) *Post planning*, given that planning is complete and its effort is available, it is added with the aforementioned available project attributes to re-adjust the estimate, c) *Post specifications*, given that the planning and specifications phases are completed and their actual effort is available, then they are added to the input of the model to review the total effort estimate, and finally, d) *Post design*; given that the previous phases and design phase are complete and their actual effort values are known, the model is updated to re-adjust the total effort estimate. Evaluation of the estimation model was performed using 5 fold cross-validation (Maimon and Rokach, 2005). One round of cross-validation involved performing the analysis on one subset (containing 60% of the data samples), the validation on a second subset (containing 20% of the data samples) and the testing on a third subset (containing 20% of the data samples). In order to reduce variability multiple rounds of cross-validation were performed using different partitions.

A range of MLP ANN architectures were employed, each of which containing a number of neurons in the hidden layer equal to the number of attributes used as inputs in each experiment and the number of inputs was increased by 1 until the neurons doubled the size of the input attributes. All architectures were evaluated and the best performance was recorded, while each time the process was repeated, the layer's weights and momentum (or biases) were initialised with the Nguyen-Widrow initialisation method (Nguyen and Widrow, 1990). The weights were updated with the gradient descent with momentum weight/bias learning function. Training was performed with the scaled conjugate gradient method. This function

can train any network as long as its weight, net input and transfer functions have derivative functions. Training was repeated 10 times (as proposed by Prechelt (1994)) and also, validation was used as a pseudo-test to ensure that the ANN were indeed trained.

Moreover, to avoid overfitting, training was stopped when one of the following conditions occurred: 1) The maximum number of 1000 epochs was reached; 2) Performance was maximised with prediction error close to the goal, which was set to 0.0001; 3) The gradient fell below the minimum performance values, which was set to 0.000001; 4) The performance of validation had increased more than 5 times the maximum validation failures since the last time it decreased.

The *hyperbolic tangent sigmoid* transfer function was used (eq. (4.7), pg. 105) to calculate the hidden layer's output from its net input. The output from this function was bounded in the range [*-1, +1*] and it was selected because it does not require non-linearity and additional scaling. Finally, the hyperbolic tangent sigmoid function was used for the output layer. The Matlab R2009b was used to create, train, simulate and visualise the ANN performance.

The observations from the research questions examined are summarised below. The relationship between the development effort of each SDLC phase and the total work effort was examined using correlation analysis. Spearman's correlation coefficient non-parametric statistic was used as a measure of rank association between two variables (Spearman, 1904; Maritz, 1981; Myers and Arnold, 2003). The same analysis was used to examine the relation among the efforts reported for each development phase.

Figure 4.25 summarises the breakdown of effort for the Planning, Specify, Design, Building and Implementation phases in the selected projects from the ISBSG R10. A large effort proportion is spent on Building activities while Test and Implementation activities also demand a notable amount of the total effort. The high amount of effort required for design and testing and the even greater amount devoted for building shows the priority software engineers give on these activities in order to deliver qualitative software.

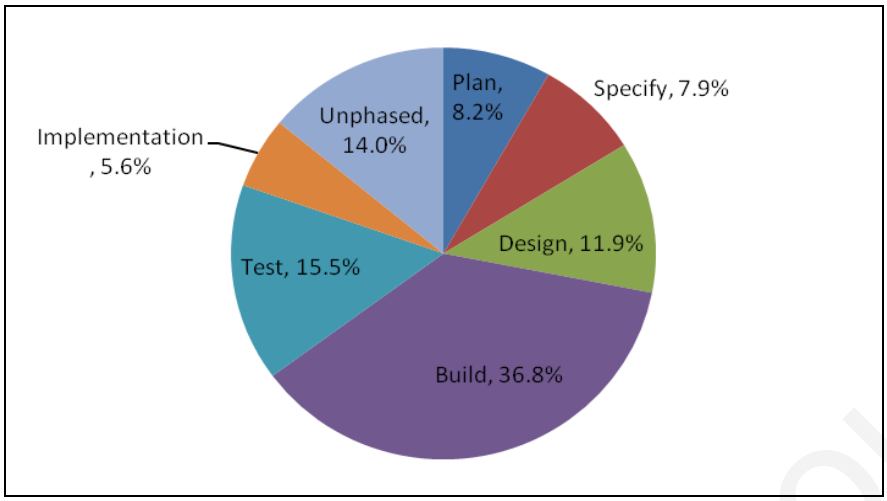


Figure 4.25: Effort average percentages distribution per phase (selected projects of ISBSG R10)

Initially, the effort reported for the completion of each phase was examined using distribution histograms (Papatheocharous et al., 2012). The histograms showed that for all the phases the distribution was negatively skewed (except for the Design and Build where it was normally distributed). The non-normal form of effort distributions revealed that for the analysis, Spearman's non-parametric test should be used. The raw values X_i (independent variable) and Y_i (dependent variable) were converted to the ranked x_i, y_i values and if there were no tied ranks, Spearman's ρ coefficient was calculated by eq. (4.60).

$$\rho = 1 - \frac{6\sum (x_i - y_i)^2}{n(n^2 - 1\_)} \tag{4.60}$$

The correlation analysis conducted on the phases and total effort (Table 4.40), as well as the pairs of phased effort values (Table 4.41) indicated that considerable correlations exist. The sign of Spearman's correlation (ρ) indicates the direction of association between the two variables examined and a zero correlation value indicates that there is no relation. Particularly Table 4.40 shows that very large correlation exists between the Plan, Design and Build efforts and the total effort, and moreover large correlation is found between Specify, Test and Implementation and the total effort.

Table 4.40: Spearman's two-tailed correlation coefficients (ρ) of the phased-effort values and total effort in ISBSG R10

Phase	ρ (*p*-value)
Plan	0.715 (0.000)
Specify	0.693 (0.000)
Design	0.827 (0.000)
Build	0.828 (0.000)
Test	0.633 (0.000)
Implement	0.603 (0.000)
Unphased	0.328 (0.008)

Moreover, Table 4.41 shows very large and significant positive correlations between the pairs of Specify - Design and Design - Build Effort (the *p*-value is lower than 0.001), while the pairs of Plan - Specify, Plan - Design, Plan - Build, Specify - Build, Specify - Implement, Design - Test, Design - Implement Effort also reveal high correlation.

Table 4.41: Spearman's two-tailed correlation coefficients ρ between the phased efforts

ρ (*p*-value)	Plan	Specify	Design	Build	Test	Implement	Unphased
Plan	1	0.600 (0.000)	0.542 (0.000)	0.625 (0.000)	0.421 (0.000)	0.481 (0.000)	0.276 (0.026)
Specify		1	0.764 (0.000)	0.564 (0.000)	0.428 (0.000)	0.583 (0.000)	-0.005 (0.970)
Design			1	0.751 (0.000)	0.581 (0.000)	0.577 (0.000)	0.103 (0.413)
Build				1	0.493 (0.000)	0.482 (0.000)	0.265 (0.033)
Test					1	0.482 (0.000)	-0.009 (0.944)
Implement						1	-0.023 (0.856)
Unphased							1

As already mentioned, the experiments were carried out for estimating the value of (i) the total effort and (ii) the subsequent in order phase was carried out in four progressive project phases, that is, a) *Early*, b) *Post planning*, c) *Post specifications* and d) *Post design* estimates. The mean results obtained with the ANN estimation models over 5 cross-validations are summarised in Table 4.42 for estimating the total effort and Table 4.43 for estimating the effort in the next phase according to the time the estimation is conducted (Papatheocharous et al., 2012).

Table 4.42: ANN performance results of total effort estimation in PB-SCE

Estimation Model	Architecture	*MMRE*	*CC*	*Pred(.25)*
Early	'12-17-1'	1.57	0.05	0.18
Post planning	'13-23-1'	1.14	0.48	0.28
Post specifications	'14-18-1'	1.03	0.59	0.26
Post design	'15-22-1'	0.94	0.75	0.34

Table 4.43: ANN performance results of the subsequent effort phases PB-SCE

Estimation Model	Architecture	*MMRE*	*CC*	*Pred(.25)*
Early	'12-19-1'	0.92	0.20	0.03
Post planning	'13-23-1'	0.92	0.05	0.03
Post specifications	'14-24-1'	0.85	0.31	0.05
Post design	'15-20-1'	1.01	0.43	0.08

The experiments showed that adding more information to the projects from the progressive completed effort phases (especially after the initial project phases of planning and specifications) and estimating total effort, results to accuracy increase in terms of *MMRE*, *CC* and *Pred(.25)*. Regarding the estimations of the subsequent project phase they are not overly impressive, although based on a small number of potentially diverse ISBSG projects, indicating that further analysis and experiments need to be conducted. If we attempt to compare the figures of effort estimation along the software product development phases (shown in Figure 2.3 pg. 68) and the actual effort estimated by the ANN PB-SCE model (Table 4.42), the main observation is that while Boehm's estimations start from a higher error they are considerably improved as the phases progress, whereas in our case the initial error is substantially lower compared to Boehm's estimations (at the 1/3) but is improved only until the Post planning phase where the error value remains for all the rest phases. This shows that for the ISBSG projects the estimation of the total effort until the initial phases is rather promising while later on estimation accuracy does not improve. This leads us to infer that apart from the project attributes considered by the models there is need for engaging in the SCE process other critical factors such as the developer team's communication, risk resolution, process maturity. The following section refers to Qualitative models developed for SCE.

4.3 Qualitative Software Cost Estimation Models

The process of SCE requires the consideration of the complex causal relationships among effort, productivity, personnel skills, communication and other factors. However and as already mentioned it is very hard even for experts and project managers to model and handle the complex interactions among cost factors. Therefore, two novel methodologies on software cost modelling and estimation are proposed based on Fuzzy Cognitive Maps (FCM) and Influence Diagrams (ID) respectively, that help to handle the relations between these essential components.

4.3.1 Fuzzy Cognitive Map for Software Cost Estimation (FCM-SCE)

Fuzzy Cognitive Maps (FCM) have been selected because they have shown promising results by modelling real world problems with success and indicating strong ability to capture the dynamics of complex environments. Particularly in SCE, some cost factors are purely quantitative (e.g., software size), and therefore easier to measure, but most cost factors are qualitative, subjective and hard to measure (e.g., team dynamics, cohesiveness, experience) and consequently their interactions with effort are very hard to understand and model. Therefore, in this section the investigation of the SCE process from a qualitative point of view is provided by trying to reach to an empirical influence matrix of the factors affecting effort.

The Certainty Neuron Fuzzy Cognitive Map (CNFCM) software cost model proposed in this section consists of 10 cognitive Concept Nodes (CN) representing cost factors and one additional node representing development effort. Figure 4.26, depicts graphically the CNFCM constructed that models the problem of SCE (Papatheocharous et al., 2008).

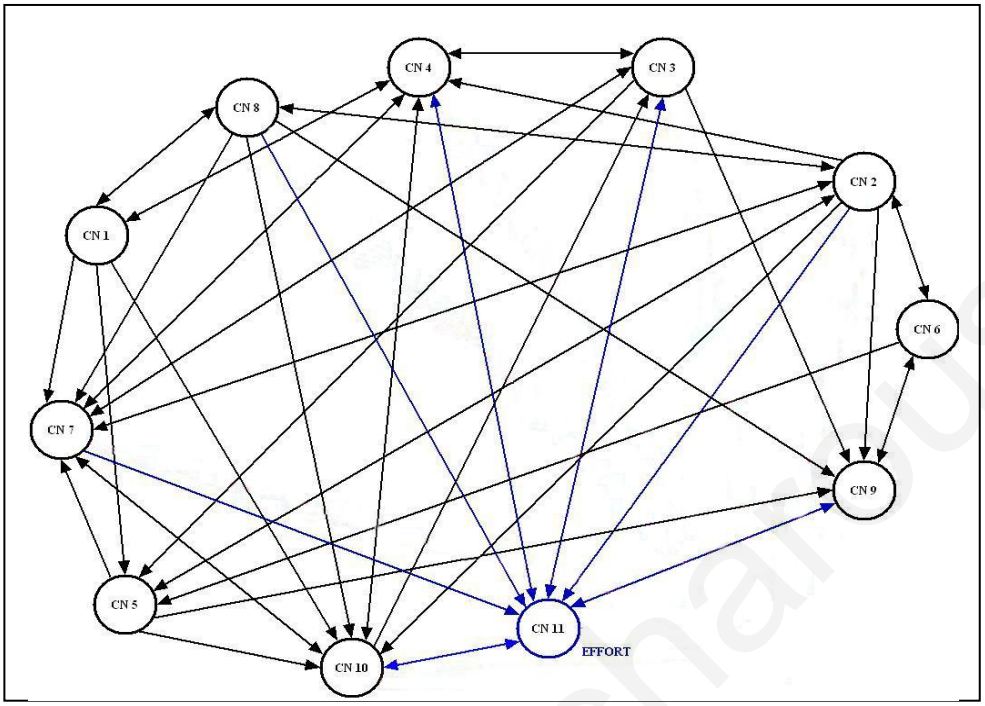


Figure 4.26: SCE Certainty Neuron Fuzzy Cognitive Map (Papatheocharous et al., 2008)

The nodes' description is provided below:

- **Application Domain Experience (CN1):** Represents the amount of knowledge and experience of the team in the domain of a certain application. A Very Low value implies no experience or knowledge in that specific domain while a Very High value implies great experience and knowledge. During project development, a low application domain experience may cause significant problems and some of them may be hard to solve due to lack of experience. This will result in an increase to the total cost and effort of the project.
- **Process Maturity (CN2):** Represents the company's maturity level. The Very Low and Very High values correspond to the lowest and highest level of the Capability Maturity Model (CMM) respectively. A company at a high CMM level will follow a more efficient development process than a company at a low level, handling errors more efficiently, with benefit gained on the total effort spent.
- **Project Size (CN3):** Represents the size of a project, in terms of LOC, FP, documentation etc. A Very Low value implies a very small project while a Very High value implies a very big project. Here, the assumption made is that if the size of a project is high then

more time will be needed in order to establish the project parameters and therefore the cost, effort and risks for the certain project will be increased. However, there is always the possibility of a small but highly innovative project requiring again more time, effort, etc. but this case is not considered in the scenarios modelled.

- **Project Complexity (CN4):** Represents the complexity of a project. A Very Low value implies a project of very low complexity while a Very High value implies a project of very high complexity. As with the project size the effect of project complexity on project cost is clear. A project with high complexity can cause a lot of time-consuming situations or hard to solve problems, thus the effort will be higher. The risk for such projects will also be higher, as it will be difficult for a project manager, to predict the possible problems the development team will face, and propose solutions to overcome them.
- **Technology Support (CN5):** Represents the available technological support tools, such as CASE tools, support configuration management tools etc. A Very Low value implies low and inadequate technological support while a Very High value implies very high technological support. The usage of supporting tools and technology increases the productivity of a project development team and therefore the required cost and effort is less than if the level of the technology used was lower.
- **Working Environment (CN6):** Reflects the developing company's working environment. A Very Low value implies a non-ideal working environment (noisy, overcrowded etc.) while a Very High value implies an ideal working environment. A good working environment is essential for a development team in order to be able to concentrate on their work without getting disturbed. An excellent working environment may increase the productivity of the team and decrease the total cost and effort of the project.
- **Architecture and Risk Analysis (CN7):** Represents the amount of architecture and risk analysis including planning for risk control and mitigation conducted by the company. A Very Low value implies an inadequate analysis, while a Very High value implies a detailed and extensive analysis. A good architecture and risk analysis is expected to cover

situations like unexpected managerial changes, reduction of the development team size, lack of sufficient technological support etc. If the architecture and risk analysis fails to map the possible risks and threats that might occur during development then the effort and the cost of the project will be more than expected.

- **Team Composition and Organisation (CN8):** Represents the quality of the composition and organisation of the developing team. A Very Low value implies a poorly composed team, with team members of limited ability and skills and ineffective hierarchical structure. A Very High value implies a team composed with highly skilled members and effective structure. It is obvious that a team composed by capable and experienced members who are organised effectively will be very productive, deliver work in less time and cost less than a team with weaker team composition.
- **Team Cohesiveness and Communication (CN9):** Reflects the quality of communication inside the team and the level of team cohesiveness during the development process. A Very Low value implies a team with poor communication and cohesiveness while a Very High value implies a team with strong communication and cohesiveness. Highly cohesive teams with high communication levels are expected to deliver work more efficiently, with more quality and will require less effort.
- **Project Quality (CN10):** Represents the quality of the project in general. A Very Low value implies a project requiring typical levels of quality (does not have any heavy restrictions as regards quality) while a Very High value implies a project of increased quality demands. If high quality is required for a project then more effort is required in order to achieve that level of quality. Consequently if the required level of quality for a certain project is low (quality restrictions are low i.e., the system is not mission critical), then the effort is expected to be less.
- **Effort (CN11):** Represents the project's overall cost, in terms of human effort, which is associated with money and time. The values assigned to this conceptual node are specific according to each company's definition of low or high cost. Thus, a Very Low value

implies a project that is expected to have minor cost for that specific company, while a Very High value implies a project that is expected to have a quite high cost.

The CN identification was established from investigating the relevant literature (Fenton and Pfleeger, 1997; Pressman, 2000; Schach, 2004; Sommerville, 2006) and also from discussion with experts. The CN were isolated and determined in terms of significance and interrelationships by conducting several sessions of interviews with experts coming from the software industry in three different European countries, namely Germany, Greece and Cyprus. The causal relationships (interactions) between the cost concepts were defined by the group of experts which included five active project managers in the software industry. The experts managed to reach to consensus regarding the type and structure of the cause-and-effect relationships among the identified critical cost factors as well as the strength of these connections specified by the values summarised in Table 4.44.

Table 4.44: Influence values (weights) between the Conceptual Nodes

	CN1	CN2	CN3	CN4	CN5	CN6	CN7	CN8	CN9	CN10	CN11
CN1	0	0	0	-0.1	0	0	0	0.3	0	0	0
CN2	0	0	0	0	0.6	0.3	0.3	0.5	0	0	0
CN3	0	0	0	+0.2	0	0	0.3	0	0	0.4	0.2
CN4	-0.3	-0.4	+0.3	0	0	0	-0.5	0	0	0.6	0.4
CN5	0.1	0.9	-0.5	0	0	0.1	0	0	0	0	0
CN6	0	0.2	0	0	0	0	0	0	0.3	0	0
CN7	-0.5	0.5	0.5	0.6	-0.4	0	0	0.1	0	0.7	0
CN8	+0.4	0.4	0	0	0	0	0	0	0	0	0
CN9	0	0.3	-0.5	0	0.1	0.5	0	0.7	0	0	-0.5
CN10	-0.3	0.7	0	0.4	0.5	0	0.5	0.3	0	0	0.5
CN11	0	-0.9	0.7	0.7	0	0	0.5	-0.6	-0.2	0.7	0

The causal relationships (weights) and the values of the activation levels of the participating concepts were characterised on a five scale fuzzification scheme listed in Table 4.45.

Table 4.45: Linguistic terms and corresponding numerical values for the influence between Conceptual Nodes and their initial Activation Level

Very Low	Low	Medium	High	Very High
-1.0 — -0.61	-0.6 — -0.21	-0.2 — 0.2	0.21 — 0.6	0.61 — 1.0

The experiments conducted were based on two hypothetical scenarios representing two extreme cases and aimed at testing the validity of the CNFCM model on situations where the output is considered known or expected (Papatheocharous et al., 2008). The target of the experimentation process was to reach to equilibrium when qualitative rather than quantitative measures of influences are available and try to obtain the real state of influences of Concept Nodes within a situation for a software development company.

The first scenario (Pessimistic Case) involved a case in which the developing company had Very Low maturity level, Medium levels of communication and discipline and had to produce a demanding software project. Under these circumstances the expected (estimated) value of effort was Medium. The rest initial linguistic values for the CN on the map are summarised in the third column of Table 4.46. The second scenario (Optimistic Case) reflected the opposite picture, where the company was mature, well-organised and had Very High communication quality, while the project undertaken had medium to low demands. Under this environment the expected (estimated) estimation of effort was High, whereas the rest initial linguistic values for the CN are defined in the fourth column of Table 4.46.

Table 4.46: Initial linguistic values for the scenarios executed with the FCM

CN	Conceptual Node	Scenario 1: Pessimistic	Scenario 2: Optimistic
CN1	Application Domain Experience	Very Low	Very High
CN2	Process Maturity	Very Low	Very High
CN3	Project Size and Complexity	Low	Medium
CN4	Technology Support	High	High
CN5	Working Environment	Low	Very High
CN6	Development Flexibility	Medium	Very High
CN7	Architecture and Risk Resolution	Very Low	High
CN8	Team Composition and Organisation	Low	Very High
CN9	Team Cohesiveness and Communication	Medium	Very High
CN10	Project Quality	High	Medium
CN11	*Effort*	*Estimated to be Medium*	*Estimated to be High*

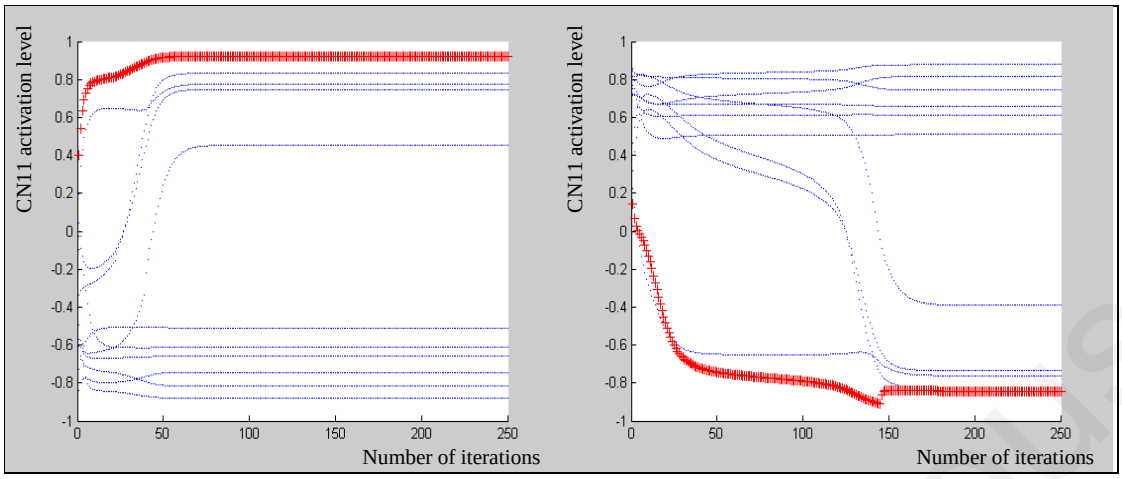
The corresponding numerical values for the hypothetical initial linguistic states of the FCM concepts are listed as initial conditions in the upper portion of Table 4.47. Particularly, in the first scenario, the map is executed taking these initial values and is then stabilised at equilibrium after 250 iterations as shown in Figure 4.27 (a). Analysing the results obtained (shown in Table 4.47, Scenario 1/Final) the cost model suggests that initially the required effort was rather underestimated (i.e., it was estimated to be of Medium intensity or having

the value CN11=0.20), and that under these circumstances it should have been anticipated to be Very High, which is the linguistic counterpart of the value CN11=0.92. Therefore, the model reacted successfully by recognising the dynamics of a 'negative' example where the poor capacity and ability of a company in conjunction with high demands and complexity of the project affect negatively the value estimated for effort.

Moreover, in the second scenario the final result of the model stabilised at equilibrium after 250 executions as shown in Figure 4.27(b). Analysing the results obtained (shown in the lower part of Table 4.47, Scenario 2/Final) the cost model suggests that initially the required effort was overestimated (it was estimated to be of High intensity or having the value CN11=0.30), and that under these particular circumstances it should have been anticipated to be Very Low, which is the linguistic counterpart of the value CN11=-0.84. Therefore, the model has successfully recognised the dynamics behind a 'positive'/ideal scenario where the company presents the necessary skills and qualifications to handle efficiently a project having the characteristics described. Therefore, the proposed model provides quantitative effort estimates in comparison to the initial set of effort estimations provided by project managers.

Table 4.47: Initial and final concepts activation values from executing two hypothetical scenarios

Scenario	Condition	CN1	CN2	CN3	CN4	CN5	CN6	CN7	CN8	CN9	CN10	CN11
1	**Initial**	-0.80	-0.90	0.10	-0.10	-0.50	0.10	-0.70	-0.50	0.20	-0.30	**0.20**
	Final	-0.51	-0.74	0.74	0.77	-0.81	-0.61	0.83	-0.65	-0.88	0.45	**0.92**
2	**Initial**	0.90	0.90	0.20	0.40	0.70	0.80	0.40	0.80	0.90	0.50	**0.30**
	Final	0.51	0.74	-0.74	-0.76	0.82	0.61	-0.83	0.66	0.88	-0.39	**-0.84**



(a) (b)

Figure 4.27: Experimental results of FCM-SCE for (a) Scenario 1: Pessimistic case and (b) Scenario 2: Optimistic case

The model produced quite successful quantitative results suggesting that in the first case the situation was underestimated in the initial estimation of effort and that this should have been estimated at much higher levels. The same successful performance was also observed by the model in the second case, where the results indicated an initial overestimation of the required effort and that this should have been estimated at much lower levels.

The hypothetical case studies aimed to validate the FCM cost model at the two extreme scenarios. The experiments conducted and the results obtained may be considered quite promising suggesting that further experimentation is needed with the model. This experimentation includes widening the spectrum of cases in which the model may capture the dynamics of real situations in software development, i.e., cases where the prevailing circumstances will not be clearly in favour of under- or overestimation. Also, the map may be also refined in terms of activation levels, i.e., specifying narrower ranges for the activation levels or restricting them to either increase or decrease in specific cases. Finally, the next investigation includes exploration of reaching to efficient and informed decisions for cost options in modern development environments, such as environments following Agile methods.

4.3.2 Agile Software Development and SCE (ASD-SCE)

The exploration of the Agile Paradigm in this thesis aims to answer two critical questions that software industry practitioners and cost estimators are very eager to answer. They relate with adopting agile and abandoning traditional plan-driven software development methodologies, something which in effect will lead to a radical changes in conventional project management. The decision diagrams namely Influence Diagrams (ID) are used for modelling the particular questions. The decision problems examined were the following: (i) "*Should an organisation switch from traditional development methods to agile or not?"*, and (ii) "*Will the software cost increase if an organisation switches from traditional to agile development methods or not?"* (Papatheocharous et al., 2011). The experiments conducted and described in this section involve SCE in Agile Software Development environments (ASD-SCE).

The agile software development, introduced in the Agile Manifesto (Beck et al., 2001), is a relatively new paradigm consisting of a group of methodologies created to deliver value to the customer. Even though it is hard to quantitatively assess the value delivered to the customer, it has a profound effect on the quality of the product delivered to the customer and the productivity of software developers. The added value from inserting flexibility and adaptability in the processes followed during software development is reported in one of the early surveys in agile methodologies (Johnson, 2003). In addition, companies using agile processes report lower or unchanged cost and better productivity, quality and business satisfaction. Value is considerably more useful to the customers as the streamlined development, in highly efficient ways, reduces time and delivers products that satisfy the real customer needs and achieve competitiveness in the market.

Nevertheless, SCE research in Agile and on the identification of the factors which affect agile development is scarce (e.g., Chandrasekaran et al., 2006; Laanti and Kettunen, 2006) and typically utilise expert-based estimations. Highsmith (2003) mentions that the nature of projects under the Agile paradigm often results in flexible project scope but fixed budgets and

schedule. On the contrary Ceschi et al. (2005) report that companies using agile methods usually result in "flexible contracts instead of fixed ones that pre-define functionalities, price, and time".

The ID created to answer the aforementioned questions are shown in Figure 4.28 and Figure 4.29 (Papatheocharous et al., 2011); each diagram consists of interrelated nodes whose description is provided below:

Customer: It represents the client's degree of participation in the development process. The options are: On-site and Away.

Manager Experience: It represents the project manager's experience in years using Agile methodologies.

Manager Confidence: Represents the project manager's confidence in the success of Agile based on the percentage of 'successful' agile projects within the organisation.

Manager Skills: Represents the project manager's skills in Agile methodologies.

Team Size: Represents the number of people in the IT/Systems/Development sector participating in the team.

Team Physical Environment: Represents the location of the team members. The two options are: Co-located and Far-located.

Team Experience: Represents the team's experience in years using Agile methodologies.

Team Skills: Represents the level of the team's skills in Agile methodologies.

Productivity: Represents the productivity degree in terms of product functionality delivered per time unit.

Effort: Represents the amount of effort required to deliver a system release.

Quality: Represents the quality level of the developed product and in consequence the ease of maintenance.

System Size: Represents the size of the system developed in terms of length or duration.

Evaluation: Represents the measure of desirability/reliability of the evaluation and is where the decision is quantified and the decision result obtained.

Agile or Traditional: Indicates the development method chosen to develop software. The two options are: Traditional or Agile.

Cost Increase: Indicates the possibility of cost increase. The two options are: Yes or No.

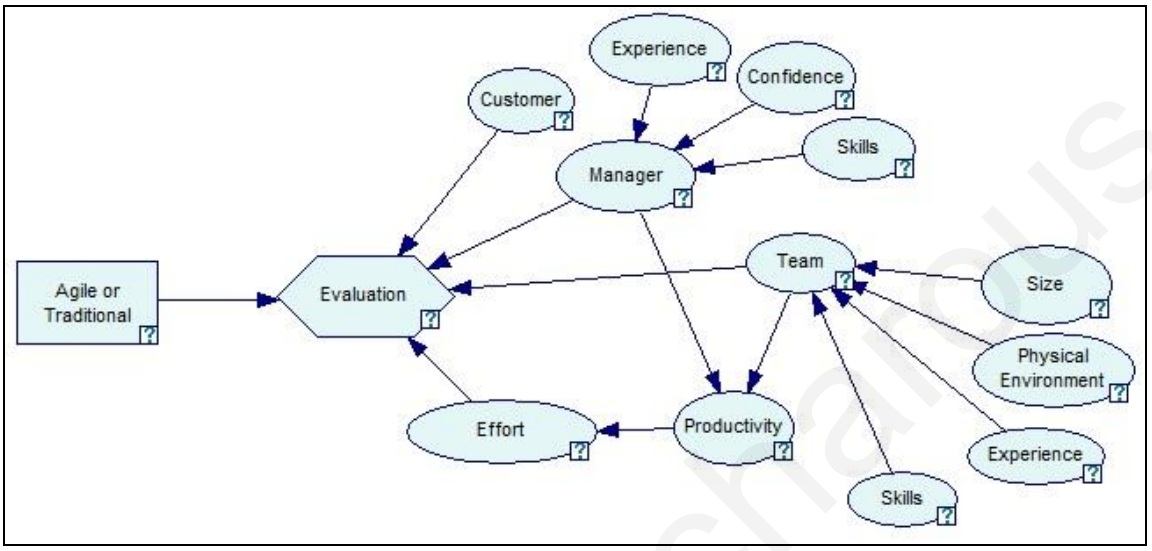


Figure 4.28: 'Follow Agile or Traditional development activities?' Influence Diagram

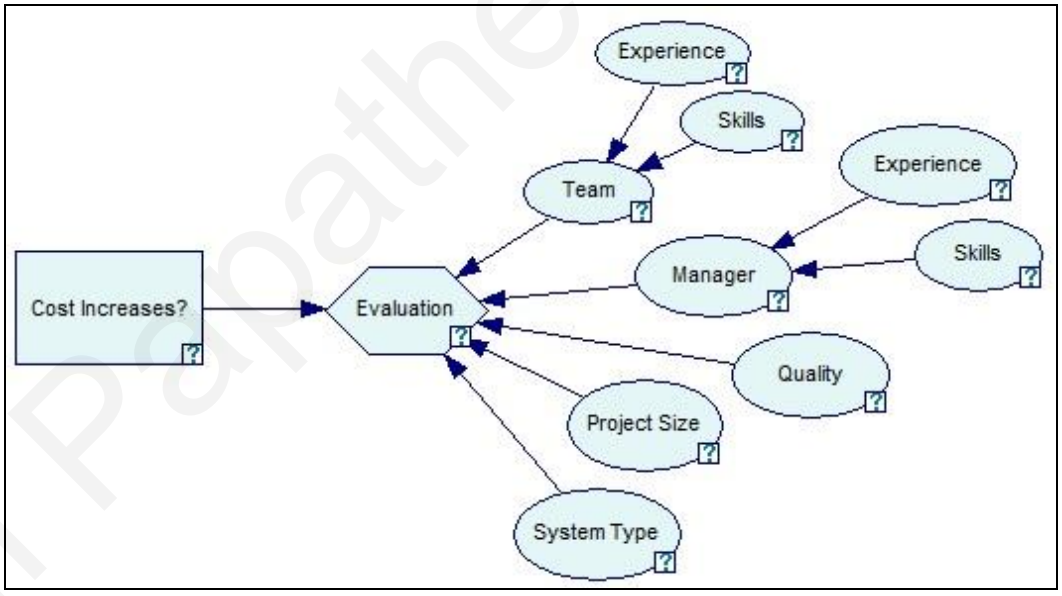


Figure 4.29: 'Will the cost increase if we follow the agile paradigm or not?' Influence Diagram

The GeNIe toolbox (Decision Systems Laboratory: GeNIE, 1998) was used to create the Influence Diagrams (ID) for the two models described and used in the experimentation. For answering the first question three different variations of the ID illustrated in Figure 4.28 were

built that differed in structure namely the *Simple* diagram, the *Deterministic* diagram and the *Advanced* diagram. In the *Simple* diagram all nodes can take any value in the range [*0, 1*]. In the *Deterministic* diagram all non-leaf nodes take deterministic values, i.e., can take any binary value from True (*1*) or False (*0*), and are represented by double oval shapes in the ID. In the *Advanced* diagram the nodes can take linguistic values, such as 'low', 'medium' or 'high'. The figures of the *Deterministic* and *Advanced* diagrams are not provided since there is no other difference than the double oval shapes for the non-leaf nodes of Figure 4.28. For answering the second question two different ID were created to estimate the change of cost; firstly in the case Agile and secondly in the case of Traditional development processes were followed. Figure 4.29 provides the diagram created for estimating the cost change in Agile environments. The diagram was slightly modified to assess cost estimation for Traditional software development by just adding one more node, i.e., the Documentation node.

For the first question the following three scenarios were examined:

- *Scenario 1 (Worst case)*: In the first scenario the case of a poor project manager and a weak team was assumed. This means the project manager has low experience in using Agile methodologies, low confidence and skills. The team is far-located, is large (includes many people) and thus communication and cooperation is harder, while the team members have low experience and skills. The customer is away during development.
- *Scenario 2 (Ideal case)*: In the second scenario the case of an ideal project manager and team was assumed. This means the project manager has high experience in using Agile methodologies, high confidence and skills. The team is co-located, team size is small and the team members have high experience and skills. The customer is on-site.
- *Scenario 3 (Real case)*: In the third scenario the values used were obtained from questionnaires reporting Information Technology practice (Ambler, 2008; Ambler, 2010). The analysis performed on the data produced sample bins of the answers provided, which showed that project managers using Agile were highly experienced and confident in the 'success of agile'. The success in using Agile was rated as interval percentage ranks in increments of 10 (i.e., 0-10%, 11-20%, etc.). Also, the data showed that a large percentage

of the teams had a small size but were highly experienced. Also, the case of an experienced project manager with high confidence and a small team with high experience and average skills was assumed. Finally, the members of the team were far and co-located and the customer was on-site.

Table 4.49 summarises the input values used in the experiments based on the linguistic terms specified for the factors of the *Simple*, *Deterministic* and *Advanced* diagrams. The range of values for each linguistic term and diagram are specified in Table 4.48. The values reflect the previously described scenarios where one can easily notice that the elicitation of values in the *Advanced* case leads to examining less strict, absolute or extreme (deterministic) scenarios.

Table 4.48: Linguistic terms and corresponding numerical values for the Influence Diagrams

Diagram Factor (Node)	Simple/Deterministic Diagram			Advanced Diagram		
	Range	Linguistic Values	Input Values	Range	Linguistic Values	Input Values
Manager Experience	[0,1]	Low	0 – 0.49	[0,1]	Low	0 - 0.3
	[0,1]	High	0.5 - 1	[0,1]	Medium	0.4 - 0.6
	[0,1]	-	-	[0,1]	High	0.7 – 1
Manager Confidence	[-1,1]	Low	0 – 0.49	[-1,1]	Low	0.2 - -1
	[-1,1]	High	0.5 - 1	[-1,1]	Medium	0.3 – 0.5
	[-1,1]	-		[-1,1]	High	0.6 - 1
Manager Skills	[0,1]	Low	0 – 0.49	[0,1]	Low	0 – 0.3
	[0,1]	High	0.5 - 1	[0,1]	Medium	0.4 – 0.6
	[0,1]	-		[0,1]	High	0.7 - 1
Team Physical Environment	[0,1]	Co-located	0.5 - 1	[0,1]	Co-located	0 – 0.5
	[0,1]	Far-located	0 – 0.49	[0,1]	Far-located	0.6 – 1
Team Size	[0,1]	Small	<10	[0,1]	Small	<10
	[0,1]	Medium	10 – 20	[0,1]	Medium	10 – 20
	[0,1]	Large	>20	[0,1]	Large	>20
Team Experience	[0,1]	Low	0 – 0.49	[0,1]	Low	0 - 0.3
	[0,1]	High	0.5 - 1	[0,1]	Medium	0.4 - 0.6
	[0,1]	-		[0,1]	High	0.7 – 1
Team Skills	[0,1]	Low	0 – 0.49	[0,1]	Low	0 – 0.3
	[0,1]	High	0.5 - 1	[0,1]	Medium	0.4 – 0.6
	[0,1]	-		[0,1]	High	0.7 - 1
Customer	[0,1]	On-site	0.5 - 1	[0,1]	On-site	0.5 - 1
	[0,1]	Away	0 – 0.49	[0,1]	Away	0 – 0.49
Quality	[0,1]	Low	0 – 0.49	-	-	-
	[0,1]	High	0.5 - 1	-	-	-
Project Size	[0,1]	Low	0 – 0.49	-	-	-
	[0,1]	High	0.5 - 1	-	-	-
System Type	[0,1]	New	0 – 0.49	-	-	-
	[0,1]	Customised	0.5 - 1	-	-	-
Documentation	[0,1]	Low	0 – 0.49	-	-	-
	[0,1]	High	0.5 - 1	-	-	-

Table 4.49: Input values for *Simple* and *Deterministic* diagrams in answering: *RQ1* Follow Agile or Traditional development activities?

Factor	Term	Scenario 1	Scenario 2	Scenario 3
Manager Experience	Low	0.2	0.8	0.885
	High	0.8	0.2	0.115
Manager Confidence	Low	0.2	0.8	0.909
	High	0.8	0.2	0.091
Manager Skills	Low	0.1	0.9	0.5
	High	0.9	0.1	0.5
Team Physical Environment	Co-located	0.2	0.8	0.5
	Far-located	0.8	0.2	0.5
Team Size	Small	0.1	0.8	0.612
	Medium	0.1	0.1	0.313
	Large	0.8	0.1	0.075
Team Experience	Low	0.6	0.4	0.222
	High	0.4	0.6	0.778
Team Skills	Low	0.8	0.2	0.537
	High	0.2	0.8	0.463
Customer	On-Site	0	1	1
	Away	1	0	0

Table 4.50: Input values for *Advanced* diagram in answering: *RQ1* Follow Agile or Traditional development activities?

Factor	Term	Scenario 1	Scenario 2	Scenario 3
Manager Experience	Low	0.7	0.2	0.115
	Medium	0.1	0.1	0.846
	High	0.2	0.7	0.039
Manager Confidence	Low	0.7	0.1	0.091
	Neutral	0.2	0.2	0.159
	High	0.1	0.7	0.75
Manager Skills	Low	0.7	0.1	0.33
	Medium	0.2	0.2	0.33
	High	0.1	0.7	0.34
Team Physical Environment	Co-located	0.2	0.8	0.5
	Far-located	0.8	0.2	0.5
Team Size	Small	0.1	0.7	0.612
	Medium	0.2	0.2	0.313
	Large	0.7	0.1	0.075
Team Experience	Low	0.1	0.7	0.222
	Medium	0.2	0.2	0.654
	High	0.7	0.1	0.125
Team Skills	Low	0.7	0.1	0.537
	Medium	0.2	0.2	0.336
	High	0.1	0.7	0.127
Customer	On-site	1	1	1
	Away	0	0	0

For the second question the following scenarios were examined:

- *Scenario 1 (Worst Case):* In the first scenario a weak team and a weak project manager, in terms of experience and skills, are assumed. The software quality is high, the project size is large, the system type is new software and the amount of documentation is high.

- *Scenario 2 (Ideal Case):* In the second scenario a strong team and project manager, in terms of experience and skills, are assumed. The software quality is medium, the project size is small, the system type is customisation software and the amount of documentation is low.
- *Scenario 3 (Ideal-Manager Case):* In the third scenario the dynamics between manager-team are investigated. Therefore, a weak team but a strong project manager (again, in terms of experience and skills) are assumed. The software quality is medium, the project size is large, the system type is customisation and the documentation is medium.
- *Scenario 4 (Ideal-Team Case):* In the final scenario the reverse dynamics between manager-team are investigated and thus the rest of the values are left unchanged. Thus, a strong team but a weak project manager in terms of experience and skills whereas the same conditions specified in Scenario 3 are assumed.

Table 4.51 summarises the values used for the factors of the two diagrams created, the first refers to answering the question for *Agile* and the second for *Traditional* projects.

Table 4.51: Input values for answering: *RQ2* Will the cost increase if we follow the agile paradigm or not?

Factor	Term	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Team Experience	Low	0.8	0.2	0.8	0.2
	High	0.2	0.8	0.2	0.8
Team Skills	Low	0.8	0.2	0.8	0.2
	High	0.2	0.8	0.2	0.8
Manager Experience	Low	0.8	0.2	0.2	0.8
	High	0.2	0.8	0.8	0.2
Manager Skills	Low	0.8	0.2	0.1	0.9
	High	0.2	0.8	0.9	0.1
Quality	Low	0.2	0.6	0.6	0.6
	High	0.8	0.4	0.4	0.4
Project Size	Small	0.2	0.8	0.2	0.2
	Large	0.8	0.2	0.8	0.8
System Type	New	0.9	0.1	0.2	0.2
	Customised	0.1	0.9	0.8	0.8
Documentation	Low	0.3	0.7	0.5	0.5
	High	0.7	0.3	0.5	0.5

Regarding the question of using Agile or Traditional development activities executing the Worst case scenario on the *Simple* diagram the decision was 0.072 for the Agile methods and 0.377 for the Traditional ones. The *Deterministic* diagram produced the value -0.441 for Agile

and 0.626 for the Traditional. Lastly, the *Advanced* diagram gave the value -0.231 for the Agile and 0.454 for the Traditional methods. Therefore, in the Worst case scenario all three diagrams agreed that traditional methods should be followed over agile.

Executing the Ideal case scenario the *Simple* diagram yielded the value 0.742 for the Agile and 0.625 for the Traditional methods. The *Deterministic* diagram gave the value of 0.747 for the Agile and -0.389 for the Traditional methods. Finally, the *Advanced* diagram produced the value of 0.740 for the Agile and -0.359 for the Traditional. Therefore, all three diagrams indicated that in the Ideal case scenario agile methodologies should be preferred over the traditional. The result was expected, as the Worst and Ideal cases are exact opposite situations and consequently the results matched those of the Worst case scenario in mirrored result values. The above results confirmed that in all the cases the diagrams created yield reasonable results.

Executing the Real case scenario with values drawn from questionnaires the *Simple* diagram yielded the value 0.620 for the Agile and 0.429 for the Traditional methods. The *Deterministic* diagram provided the value 0.542 for the Agile methods and -0.082 for the Traditional ones. Finally, the *Advanced* diagram provided the value 0.384 for the Agile and the value 0.007 for the Traditional methods. Therefore, in the Real case scenario all diagrams confirmed that Agile methods are favoured over the Traditional ones.

Regarding the question of cost increase in case the Agile paradigm is followed, executing the Worst case scenario the *Agile* diagram produced the value of -0.430 for no cost increase and the value 0.692 for cost increase. The *Traditional* diagram resulted to a value of -0.297 that cost will not increase and a value of 0.683 that cost will increase. As expected, considering the Worst case scenario software cost is expected to increase no matter which methods or activities are selected to follow, agile or traditional.

For the same question, executing the Ideal case scenario, the *Agile* diagram created showed that cost will not increase with a value of 0.718 while cost will increase with a value of -0.585. In the *Traditional* diagram the Ideal case showed that cost will not increase with a value of 0.543 and will increase with a value of -0.245. Therefore, the diagrams showed that

in the Ideal case cost will probably not increase in the Agile nor in the Traditional case, with the former having a stronger confidence.

The next two scenarios executed had the same conditions except the diversified experiences and skills of the team members and the project manager. Executing the Ideal-Manager the *Agile* diagram showed that with a value 0.005 cost will not increase and cost will increase with the value of 0.249. On the contrary, the *Traditional* diagram showed that cost will not increase with a value of 0.296 and will increase with 0.117. It is obvious that having a weak team, even with a strong project manager, in Agile methods software cost is more probable to increase, whereas in Traditional development the existence of a strong project manager counterweights the situation, and most probably cost will not increase. However, the decision in Traditional with a strong manager versus a weak team is not 'distinct' (clear) because the values produced were close.

Executing the Ideal-Team case scenario where a strong team supports the activities but the project manager is weak, in terms of experience and skills, the diagrams support a different decision. The *Agile* diagram yields that the cost will not increase with a value of 0.294 and it will increase with the value of 0.033. The *Traditional* diagram resulted that the cost will not increase with the value of 0.066 and it will increase with the value of 0.400. The experimental results showed that Agile methods with an ideal team will probably not lead to a cost increase (even though the project manager has low skills). On the contrary, even though there is a strong development team in the Traditional environment, due to the weakness of the manager, cost will most probably increase. Overall, the results obtained were very encouraging as they showed that the diagrams worked reasonably well, fully adopting the conditions in each of the paradigms followed. In cases where the organisation's conditions did not favour the Agile paradigm, all diagrams consent to following a Traditional method instead, as the use of Agile would have an increase in cost and should be avoided.

The following chapter provides a summary of the approaches explored in this thesis and includes an overall discussion on the targets achieved, problems and threats of each SCE model proposed and described the future research steps.

Chapter 5

Conclusions and Discussion

In this chapter the research goals of this diatribe are summarised and the main results and conclusions are discussed. Future work is also provided along with a list of open research issues and potential extensions of this work.

5.1 Summary

Recalling the modelling and Software Cost Estimation (SCE) methodologies explored in this diatribe we summarise: Two approaches were followed the Quantitative and the Qualitative to explore the subject from two different points of view.

The Quantitative models created followed a supervised type of learning based on empirical project data. The models were developed and calibrated using a part of the project samples and the rest samples (the 'unseen' part) were used to evaluate their prediction and generalisation performance. The techniques utilised by the models proposed originate from the Computational Intelligence (CI) area. They aimed to model and accurately estimate the development costs of real software projects. Each model utilised data from empirical software project datasets that are publicly available, such as the COCOMO, the Albrecht and Gaffney, the Kemerer, the Desharnais and the ISBSG datasets.

The exploration of the proposed modelling and cost estimation techniques carried out in this thesis began with Size-Based SCE (SB-SCE) models. The SB-SCE models investigated

the relation of software size and effort, as the former was expressed with Lines of Code (LOC) and/or Function Points (FP). The investigation included finding the optimum model for the relationship between size and development effort. The main conclusion of the SB-SCE models was that size expressed in FP presented considerable advantage in estimating effort compared to the corresponding size-related metric of LOC. The experiments showed that effort is driven by size-based factors but also other cost drivers exist that seem to affect the value of effort. This effect of other cost drivers, or software features, on effort was further analysed in subsequent investigations conducted in this thesis.

Moreover, the accuracy of the ANN models developed for SB-SCE was compared to regression which is one of the most popular approaches in SCE. The second observation made during the experimentation with SB-SCE models was that one of the critical factors affecting the performance of an ANN was the internal network architecture. Therefore, a Genetic Algorithm (GA) was developed and was responsible to optimise the patterns of data used as inputs and the internal ANN architecture, thus partially automating the ANN design. The patterns of inputs were specified to investigate the prediction ability of the ANN using data fed as patterns of size-effort from projects within the same company and from cross-organisations, i.e., international projects from many companies around the world. The experiments conducted showed that the hybrid model developed for SB-SCE provided optimised accuracy results in most dataset cases and explored a vast space of solutions. The main conclusion was that effort predictions were divergent over both the single-company and multi-company (ISBSG R9-1) datasets, i.e., it was easier to locate an improved accurate model in the single-company datasets than in the ISBSG dataset. However, apart from some ANN topologies and patterns of inputs found to optimise the prediction performance, it became evident that the SB-SCE based on ANN could not converge to optimal solutions for every single case (dataset or project data split examined). The above confirms the previous assumption that LOC and/or FP metrics alone cannot always produce accurate effort estimations, and an exploratory study of other factors affecting effort needs to be considered.

Therefore, the subsequent approaches investigated the evaluation and selection of appropriate inputs (or project features) and aimed to enhance our understanding of the cost drivers' effect in the SCE process. Also, the main outcome from selecting the most appropriate (or 'significant') features in SCE is that it may contribute to minimising the required effort and time to gather, process, store and maintain large quantities of project attributes for future SCE and simplify the overall estimation process.

Initially, the investigation of the utilisation of ANN for Feature Subset Selection (FSS) and evaluation was carried out. The methodologies developed aimed to improve effort predictions and lower the complexity of the model by reducing the input's dimension. This reduction was based on a process to extract the most influential cost drivers that describe best the effort devoted to development activities using the weights of the network connections. In addition, the cost drivers (or features) selected were assessed towards their effect on effort estimation performance. Three variants of FSS approaches investigated the ability of an ANN to predict effort, generalise the knowledge acquired during training and yield predictions with a minimal set of features, thus reducing the need to feed the network with large quantities of inputs. The hybrid approaches combined ANN and Input Sensitivity Analysis (ISA) by employing three different methodologies (although similar in principle) which utilised the network weights for obtaining the following: (i) the significance degree of inputs (through empirical thresholds), (ii) the relative input's strength (Azoff, 1994) and (iii) the overall relative importance (Garson, 1991). The FSS executed was successful in some datasets (e.g., the ISBSG R9-2) but less successful in other datasets (e.g., the Desharnais and the ISBSG R9-3). The common features selected were for the Desharnais dataset the number of transactions and software size (FP) and for the ISBSG the software functional size, number of software additions and external enquiries (reports). Also, the hybrid ANN & ISA techniques showed that the balance between the appropriate number and convenient type of inputs used in the model (i.e., a relatively small input size of attributes that can be measured from the initial project phases) need to be defined so that the level of accuracy performance will be within 'acceptable' - within explicit limits - for a project manager or a cost estimator. Although these

limits are considered speculative and subjective for each project manager, there is a considerable benefit from investigating the contribution of features in SCE.

In order to further explore this, nine dedicated FSS approaches were employed with Ridge Regression (RR) acting as the evaluator of each approach. The approaches aimed to automatically identify the 'significant' features and evaluate them separately (i.e., excluding the 'less significant' features from the original datasets of Desharnais and ISBSG R9-4) by carrying out SCE. Finally, a comparison of the initial predictions (using the full attributes set) and the final predictions (using the selected attributes set) is performed. In some cases, the prediction results were maintained at similar levels compared to the initial predictions obtained with the full feature set and, in addition, the number of 'optimal' features selected was reduced to more than half. The results also showed that the attributes selected for each dataset were related to concepts akin. Particularly, for the Desharnais dataset the promoted attributes were related with software size (in FP) and duration (in months). Other factors were the number of transactions, entities and scope of the project. For the ISBSG R9-4 the selected attributes related with the software size (in AFP), duration, recording method and team size. Other factors were the language type and development technique used. The argument made therefore in the former experimentations (with the SB-SCE models) that apart from size-related metrics there are other cost drivers that affect development effort was confirmed since they were selected by the FSS approaches. In addition, within the single-company dataset (the Desharnais) the features selected were not related with the team's characteristics, team size, domain experience etc., since in this case they were the same or quite similar, and therefore did not play a critical role in describing effort, as expected. In contrast, for the ISBSG case which is a multi-organisational dataset, the features selected were related with factors that are diversified between different organisations and development teams. This leads to the interesting observation that even though some cost drivers identified in the past to clearly affect the value of effort, there might be factors which are not mentioned/measured in single-company datasets, not because they do not affect effort, but because they are considered constant within the context of the company (e.g., domain experience and use of tools).

Moving on, the subsequent investigations of this thesis aimed to analyse from large and heterogeneous datasets (such as the ISBSG) factors of numerical, nominal and descriptive (including multi-categorical) nature. The factors of this nature required intensive pre-processing, filtering (vertical and horizontal), transformation and fuzzification, in order to be suitable for cost models. The activities elaborated subsequently aimed to improve the quality of the accuracy performance of SCE models from the CI-domain. Two approaches were followed for conducting Clustering and Classification for SCE (CC-SCE) using notions from the field of CI, i.e., evolutionary approaches, fuzzy logic, machine learning and fuzzy systems.

Initially, clustering was performed taking into consideration ranges of values instead of rules containing crisp attribute values. The novel technique proposed for SCE included a Genetic Algorithm (GA) for evolving value ranges of cost attributes based on the real project values within datasets and the evaluation of the value ranges was based on Conditional Sets (CS) theory. The aim was to address the problem of large value variances found in the available historical project data, especially in the ISBSG dataset case, used in SCE which resulted in inadequate fitting in some cases. Also and as already mentioned, meaningful software data is quite expensive to collect, manage and maintain. Therefore, in order to lower the need within the SCE process to gather accurate and homogenous data, the proposed technique considers replacing the actual data values with simulated or generated data ranges. The estimation of effort was performed using the mean effort of the actual projects that satisfied the value ranges included in equations expressed as Conditional Sets (CS). The hybrid technique created evolved CS through the application of a GA to satisfy as many similar projects and obtaining as narrow ranges in the CS as possible. The technique was found inadequate to produce particularly narrow ranges regarding the attributes investigated, but only satisfied a small number of projects, something which confirmed the existence of diversity in the values within the ISBSG R9 dataset for these attributes. The technique, even though was promising as a concept, produced adequate CS but failed to cover the whole spectrum, or at least a large spectrum, of projects in the dataset and thus, prediction performance was not impressive.

The next clustering technique employed aimed to address the multi-dimensionality and uncertainty included in fuzzified datasets and obtain predictive intervals of effort. The predictions were based again on the mean and standard deviation of the samples with strong membership within a fuzzy cluster. The fuzzy clustering algorithm of Entropy-based Fuzzy *k*-modes was applied and yielded relatively invariable clusters of projects within the ISBSG R9-8 dataset and with the estimations lying within the calculated interval in a large number of cases. The results obtained confirmed the need for applying clustering in such multi-dimensional datasets and the existence of highly divergent measurements in the dataset. Also, the above observations triggered the need for investigation of classification rules within the context of SCE that may facilitate the optimisation of the clustered (grouped) projects which will be later on selected and used for effort approximations

Therefore, the exploration of a Genetic Programming (GP-based) approach for SCE was performed which investigated algorithmic SCE and created automatic large quantities of rule-based equations. These equations included arithmetic and logical expressions for the numerical software attributes (such as LOC and programming language experience in years) and for the nominal software features (such as organisation type) respectively. The optimal rules were selected by the GP through an evolutionary process and were used to directly estimate effort by solving the arithmetic expressions and by classifying projects through the logical expressions and predicting effort as a range within the mean effort and the standard deviation of the projects satisfying the expression. Even though the model was quite promising the prediction results obtained were not very successful, considering the acceptable rate for software prediction models set by Conte et al. (1986), i.e., ≤ 0.25. From the rules obtained, however, the participating attributes were analysed in terms of having a higher influence/presence for each dataset utilised. Therefore, the attributes having higher presence in the GP rules were: for the COCOMO dataset the attributes of project size, duration, complexity and analyst experience, for the Desharnais dataset the project duration, scope, number of transactions and function points (adjusted) and for the ISBSG R9-4 dataset the attributes of organisation type, development technique, database system used and the

maximum team size. The main advantage of the GP tool developed was that it automatically explored a vast set of potential solutions in an adaptive search technique. The main limitation was that it was too data-dependent.

Subsequently, the following assumption was made: if we proposed a methodology to acquire classification rules that included fuzziness it would enhance the CC-SCE techniques proposed thus far, into Fuzzy CC-FCE, since they would not be crisp-value-dependent and thus taking better into account the uncertainty of the data values. The methodology developed for classification and prediction used the techniques of Fuzzy Logic (FL) and Decision Trees (DT) to obtain fuzzy rules. These rules were used to obtain estimations based on: (i) the mean effort values of the fuzzy range, (ii) the mean value and standard deviation of effort from the classified projects (classification was based on the aforementioned fuzzy rules), and (iii) Fuzzy Implication Systems (FIS) that aggregate the aforementioned fuzzy rules to perform effort implication (suggestion). Thus, the predictions were based on the production of rules obtained from Fuzzy Decision Trees (FDT) and their utilisation as classifiers and predictors. The main advantage was the automation in the construction of fuzzy rules which were overly simple, described in an intuitive form and may be considered to enhance the knowledge and understanding of the project manager and team regarding the factors in software development driving effort. The methodologies yielded quite accurate predictions and more importantly provided comprehensible linguistic rules that may increase the acceptability by individuals since they are expressed in a way closer to the way humans think. Comparing the prediction results of the hybrid FDT & FIS technique with previous work the main observation was that the accuracy was not improved for the cases of Desharnais and ISBSG R9-9 compared with the ANN technique. However, using a set of features that may be measured from the 'early' project phases the hybrid FDT & FIS algorithm outperformed the predictions of ANN.

Before moving on and concluding this summary, another approach was utilised for improving SCE in the sense of producing Predictive Interval estimations (PI-SCE) through the novel technique in the area of SCE of Conformal Predictors (CP) and RR. The hybrid approach yielded narrow and reliable intervals according to specified confidence levels, which

may be regarded acceptable in practice by project managers. Finally, the last Quantitative model proposed in SCE was the Phased-Based SCE (PB-SCE) model. Empirical investigations were carried out on the adjustment of work effort progressively throughout the whole software project life-cycle and along the development phases. Particularly, estimates were progressively updated and total effort, as well as at fixed point in time of the development process, the effort of the next phase were assessed. The experimental results obtained with only a small portion of the newer version of the ISBSG (R10) were at a preliminary stage and further investigations need to be conducted.

The above Quantitative investigations led to the conclusion that even through exploration of a variety of techniques, based on different concepts and enhancers/optimisers (i.e., feature subset selection approaches, genetic-based evolutionary optimisations, methods of fuzzification, clustering and classification) the SCE prediction can be enhanced, but it seems that a single optimal (panacea) method cannot be used as a solution for all project cases. Therefore, under specific requirements the estimator might prefer one technique over another. The above indicates that there is need for structuring the decision process of deciding which SCE technique to use. Taking this into consideration, for each dataset and with respect to the experiments conducted in this work the following is proposed: The model achieving the lowest *MMRE* in estimating effort and should be selected for SCE for the COCOMO dataset was Ridge Regression (RR) utilising all features (with *MMRE* equal to 0.422), for the Albrecht and Gaffney dataset was Ordinary Least Regression (OLS) utilising only the feature Function Points (FP) (with *MMRE* equal to 0.248), for the Kemerer dataset was OLS utilising only the feature Adjusted Function Points (AFP) (with *MMRE* equal to 0.196), for the Desharnais dataset was Artificial Neural Networks (ANN) utilising only the feature of AFP and also using Input Sensitivity Analysis (ISA) (with *MMRE* equal to 0.051 and 0.293 respectively) and finally for the ISBSG dataset again the was ANN and also combined with ISA (with *MMRE* equal to 0.199 and 0.191 respectively). This short comparison per dataset was made taking into consideration the method performance metrics were calculated in each case so that comparison of "apples with apples" is ensured.

Finally, concluding this research work Qualitative models were developed for simulating and analysing the dynamics of various factors in the development environment which cannot be quantitatively assessed (such as team composition and organisation). A model based on Fuzzy Cognitive Maps (FCM) for SCE was evaluated via two real-case scenarios taking into consideration the opinions and expertise of three project managers from three European countries. The FCM-SCE model showed that it captured the dynamics of the situations tested. Finally, the qualitative investigation in SCE included the influence of the Agile paradigm which is a relatively popular approach followed during the last 10 years in software development. The Agile Influence Diagrams (ID) proposed for SCE (ASD-SCE) examined the benefits of switching from traditional software development to agile methods and how software cost might change if agile methodologies are adopted by an organisation.

5.2 Goals Achieved and Significance

The main contribution of this work is the exploration of the effectiveness and applicability in specific project cases of CI techniques and models for obtaining accurate and reliable approximations of the human effort required to develop software systems. This exploration included the analysis and understanding of the relationships holding among the various cost drivers and the development effort required to produce software. The analysis concluded also provided the identification of the most significant and appropriate cost drivers for particular project cases, i.e., within single-company or multi-company international empirical software datasets.

In addition, the relationships within the development environments were effectively modelled through various CI techniques. The models were distinguished into Quantitative and Qualitative according to the aspect of SCE targeted in each respective case, i.e., quantitative or scenario-based and decision analysis effort approximations. Specifically, the Quantitative explorations initially confirmed the importance of the size metric in SCE but the difficulty to approximate effort accurately for every single project case led to the conclusion that other

factors need to be investigated. The SCE models described in this thesis' quantitative approach were considered successful and novel compared to other studies since they investigated beyond accuracy, which is the dominant approach in SCE studies, the significance of cost drivers through Feature Subset Selection (FSS) approaches. Regarding accuracy of the SCE performed in this work, the figures obtained showed improved prediction accuracy for all the datasets examined compared to other related research work (e.g., some recent examples include Huang et al. (2008), Azzeh et al. (2008), Li et al. (2009a), Li et al. (2009b), Oliveira et al. (2010) and Azzeh et al. (2010)). The FSS-SCE also provided intuitive support for understanding the strength of inputs and the sensitivity of the relationships between cost driver parameters and model prediction results. The implementation of the algorithms, techniques and models for SCE explored in this thesis, originating from the domain of CI, established reliable, accurate and comprehensible (in the sense of practical) effort estimations and particularly identified the strong relationships among factors contributing with a relatively higher degree (than the rest factors) to the accurate prediction of effort. The abovementioned practicality also included addressing the particular needs of people that are involved in project resource management and in SCE, including software managers, system and subsystem engineers, stakeholders and cost estimators. However, this practicality needs to be further investigated through application of the CI-based approaches in a real industrial setting (i.e., collaboration with a software company).

This thesis increased the comprehensibility of both the CI techniques for SCE and the results obtained by the models proposed. This level of understanding was increased by the clear definition of experiments, selection of parameters and by brightening up the internal workings of models such as the ANN for example, that are usually considered 'black-box', by utilising fuzzy rules, linguistic terms and intervals to express a result (a rule, a prediction or a confidence level) and reach to a more informed decision. In addition, some of the models developed were more robust than others (e.g., FDT compared to ANN) that is the solutions are less sensitive to the change of parameters.

This research thesis has addressed key factors that contributed in obtaining improved effort estimates and were generally missing from many other SCE attempts. The key factors taken into consideration for this thesis have never been embraced into one exploration, and have addressed the following issues: the inherent software data uncertainty/heterogeneity, the existence of null (missing), multi-categorical, nominal and numerical values in the datasets, outliers (in the sample and in the predictors), causality (analysis of parameters' change and inputs' contribution), feature (inputs) selection and new development paradigms. In addition, optimisation of the software data quality was produced through clustering, fuzzy clustering, genetically evolved clustering and classifications, fuzzy classifications and predictions through intervals.

Qualitative explorations were conducted to represent and realise the complex environment of software development and particularly the degree of causality among a concise set of factors causing alterations to software effort. The models were composed of factors found in the relevant literature to affect development processes in general and in the particular cases of development methods with the Agile paradigm. The effectiveness of such models depended on the experts' understanding of the influences among concepts and the techniques used to perform the calculations for the model result. The knowledge of a group of experts, from three different European countries, was used and several hypothetical scenarios were used for estimating effort in particular cases and in the Agile environment.

The CI-based models proposed in this thesis increased our understanding of the SCE problem and proposed viable alternatives for accurate and reliable estimations. The models assisted in reducing the complexity of the problem by automating the estimation process and disengaging it from the need of an expert or consultant (at least after a model is efficiently constructed and trained). Moreover, the significance of the models proposed is the creation and utilisation of novel forms of Computational Intelligent models and hybrid models, combining more than one technique for the challenging task of SCE. Finally, the aim of this dissertation to override some of the main disadvantages, limitations and problems occurring

with the examined CI methods was achieved through the detailed experimental form which furthermore, addressed the overall complexity and uncertainty of the problem of SCE.

5.3 Discussion of the Threats to Validity

The main threads to validity of the models proposed in this thesis included:

- **Availability of software metrics.** Both Quantitative and Qualitative models rely on the existence of prior knowledge regarding several project attributes and characteristics of the organisation. Therefore, one possible threat is that some of this information is impossible to know beforehand, i.e., before the actual development of the software, or even is considered subjective to measure. Such attributes are considered complex and difficult to be objectively calculated. However, this threat is not prohibitive in including such software metrics within the models proposed since they are considered to represent values that are believed or anticipated to be valid for the organisation and software under development. Therefore, project managers may for example approximate or use information from prior completed projects, knowledge and past experience to reach to estimates for these software metrics.
- **Availability of size metrics.** The models based on size metrics (all proposed models in this thesis including software size as a factor and especially the SB-SCE models) require the availability of the software size, before the project begins or at least at the initial project phases where the SCE process is more practical. This threat is recognised as a threat in almost all data-driven software cost models in the relevant literature (Fenton and Pfleeger, 1997). However, in this work this threat has been addressed in two ways: Firstly, size metrics are not considered to represent the actual code length in Lines of Code (LOC), number of Function Points (FP), etc. delivered, but represent values anticipated (estimated) by project managers. Secondly, size metrics are used from past completed projects as available measures to construct

(and/or train) a model and then the model is utilised for yielding estimates for the new (similar) project case(s).

- **Small dataset size.** Several datasets (e.g., the COCOMO, Albrecht and Gaffney, Kemerer and Desharnais dataset) included a small size of samples. This on one hand raises doubts as regards proper model training versus overfitting on such small datasets and on the other hand, limits the significance of the findings since the results may be considered valid for only the small number of projects examined. Nevertheless, the threat of overfitting was handled by using holdout samples and testing the generalisation ability of the model on 'unseen' samples. This of course raises the threat discussed in the next point. Moreover, regarding overfitting in the training process of the ANN models (which are generally blamed for obtaining too accurate results), it was also avoided by stopping training in any of the following cases: if a maximum amount of epochs or time was reached, if performance reached the goal set and if validation performance was increased more than 5 times of maximum validation failures since the last time it decreased. The latter threat of the significance of the results was addressed by utilising larger in size datasets, like the ISBSG R9.
- **Lucky or local successful performance.** This threat is also related to the generalisability of the models when samples change which is discussed in the next point. Some of the models proposed were constructed based on a set of samples (training) which were obtained from separating the datasets. Therefore, one may argue that the results obtained from these models (e.g., the SB-SCE models of multiple hidden layer ANN, the genetically evolved CS for classification and SCE and the Entropy-based fuzzy *k*-modes clustering algorithm and SCE) were dependent to the random division of training and testing set of samples. However, this threat was treated in the experimentation with the models proposed by: (i) repeating the process (including the data separation) a number of times and reporting the best accuracy results from all the models constructed (e.g., refer to the rest SB-SCE of ANN and the

GP application for CC-SCE), (ii) repeating the process (including the data separation) a number of times and reporting means of performances over a number of executions (e.g., the hybrid ANN with the first two ISA approaches for FSS-SCE), and (iii) performing *k*-fold cross-validations (e.g., the ANN and the third ISA approach and RR in the FSS-SCE, FDT in the CC-SCE, CP with RR in the PI-SCE and the ANN in the PB-SCE models).

- **Model generalisability.** As already mentioned above, some models developed and trained will not necessarily work sufficiently well when conditions (e.g., data samples, environments) change. Having in mind that the proposed models are empirical investigations based on available project data datasets, it is clear that when new data, projects or organisations emerge to utilise the models, they might fail to generalise. Especially in software development environments that are frequently characterised by rapid change in the technologies used, the people involved and the software constructed, it is hardly the case that within different conditions the same accuracy results in terms of performance errors will be obtained. In the investigations carried out in this thesis and for the specific datasets, the trained models seem to have worked sufficiently well, using the 'holdout' samples (and during validation – when validation was used). The same models, though, in light of a large number of completely different projects may or may not work that well. In such case, the cost estimator might need to repeat the process of training until the models restore their ability to generalise with the new data (if this is possible to achieve).
- **Data degree of variability/heterogeneity**: The data samples included in the experiments vary considerably in terms of characteristics. This was observed especially in the cases of the multi-organisational and international datasets such as the ISBSG R9 and R10. Even though the same data collection method was used for these datasets the measurements yielded are considered highly subjective or the projects are extremely diversified. Through the investigations of this thesis it was observed that effort distributions in relation to project features, such as functional

size, development type, language type and team size, did not follow the normal form and therefore huge variations existed. The lack of homogeneity may be addressed if project samples were carefully collected, under the same conditions i.e., similar processes, technologies, environments, people and requirements, and as long as consistent counting methods are used (Leung, 2002). Some of these conditions might be less variable in cases were projects belonged to a single-company and therefore such datasets were utilised in this work (e.g., COCOMO, Albrecht and Gaffney, Kemerer and Desharnais datasets). In addition, to eliminate the inherent heterogeneity, improve the quality and consistency of the datasets filtering and pre-processing activities were carried out in this dissertation (refer to Table 4.1 and also to the methodologies including fuzzification, clustering and classification).

- **Subjectiveness of data pre-processing.** The pre-processing activities carried out on the available project data were different in each approach proposed and analysed in this diatribe. Thus, the trained models might not work in different contexts and if for example insufficient pre-processing is carried out. Thus, the performance figures obtained from the various models may not be always directly comparable. This is a common threat found in most SCE studies as discussed by Mair and Shepperd (2005). However, this threat was partially treated by performing common necessary pre-processing activities to obtain a more homogeneous dataset appropriate for each SCE approach and in some cases the exact filtered datasets were obtained (e.g., Desharnais, COCOMO, ISBSG R9-4). Moreover, the variables selected and included in the SCE models proposed were those considered more appropriate to describe development effort. This selection was purely empirical and was not based on any scientific evidence apart from relevant studies describing attempts and experiences with other models utilising specific variables. The variables selected involved in some cases highly subjective measures, such as team experience, project size and complexity, whose effect on effort was harder to be captured or explained by traditional SCE models. The target of this work, though, was not to assess the subjectivity of the

measurements but to produce successful effort estimations with the use of a limited set of variables from the specific datasets, something that was finally achieved.

- **Data representativeness and quality.** Another threat is that the results of both Quantitative and Qualitative methodologies depend on the quality of the data utilised. There are cases where the form of data, the presence of collinearity, heteroscedasticity or even outliers within the samples jeopardise the outcome of estimation processes that rely on learning by examples (like the ANN models included in this thesis). In addition, projects listed within datasets usually do not constitute the industry norm (ISBSG, 2007a; ISBSG, 2007b). This is mainly observed because organisations are usually reluctant to submit 'unsuccessful' projects and the 'failed' projects will usually not be 'advertised' (i.e., submitted to the ISBSG organisation). The representativeness of the projects within the datasets unfortunately was not tackled in this thesis but the quality and consistency issues are believed to be resolved by considering the recommendations of the ISBSG's quality reviewers that assessed the quality of the submitted data values. In addition, the pre-processing steps (Table 4.1) and filtering activities (clustering, classification and feature selection) carried out in this thesis enhanced the quality of the data samples utilised.
- **Fuzziness/Defuzzification.** Another threat is related to the way fuzzification and defuzzification were performed which might have affected the credibility of the results.
  - *Fuzzification:* The selection of the membership functions was clearly empirical. The shape was decided based on the general picture observed using histograms for each cost attribute which indicated that the respective function used was a good candidate for approximating the shape of the fuzzy set to which the crisp values were transformed. Whether this is actually a threat to the proposed approach can only be determined by looking at the results of each approach. If for example the selected membership function was appropriate then this is reflected in the accuracy of the final estimations. If the

datasets change then obviously the same empirical process must be followed once again for defining the appropriate shape and the experiments must be repeated on a trial-and-error basis. Once, though, the shape is correct the cost model may be considered reliable under the assumption that the dataset used for creating the rules reflects also the characteristics of new projects.

- *Defuzzification:* The process of defuzzification may also constitute a potential threat to the estimation process as it may also affect the accuracy of the estimates. Different methods for transforming the linguistic values of fuzzy variables into crisp numerical approximations were utilised, each resulting in possible loss of 'detailed' information included in the part of the original value. This value is sort of filtered out by the fuzzification step described above and consequently by the reverse action of defuzzification. Again it is also a trial-and-error situation where one has to test various approaches for defuzzifying the result of the cost model to see which method suits best the available datasets. Once more, if this proves successful then accurate estimates may be produced by the model.

- **Fitness function.** A possible threat which might have led to overly time-consuming explorations of the search space solutions in genetically evolved approaches was the use of conflicting factors within the fitness functions. Such conflicting objectives were employed in approaches like the hybrid ANN with the Genetic Algorithm (GA) in SB-SCE and the genetically evolved CS for the CC-SCE models. This threat could be treated by a multi-objective GA which is one of the future plans described in the following section.

Closing this section, we may argue that the validity of the proposed SCE and modelling techniques does not suffer from serious threats or threats that were not addressed or planned to be addressed in some manner. Having the aforementioned threats in mind, their potential influence on the accuracy of the estimations for the SCE models proposed may be minimised or required the models to be re-calibrated to the current environment to minimise their effect.

5.4 Future Work

The research conducted has provided an insight into using advanced CI approaches and paved the path for further research on software cost modelling and estimation. Some of the future research plans are summarised in this section. Future work will include two types of investigations based on extensions of the proposed models and developing new approaches pointing to new areas of research.

Initially, the extensions or improvements of some aspects of the Quantitative and Qualitative SCE approaches application are summarised. First of all, a practical extension includes the application of the described SCE models in a real industrial setting, i.e., by a software company. The only requirement is that functional size measurement and work effort (in person-hours or person-months) are measured by the industrial partner. Also, other extensions include the development and investigation of more ISA approaches for the neurons of trained ANN, an issue which has not been used in previous SCE research nor has received attention to date. The ISA methods presented in this thesis for ANN are considered promising, since they are simple, enhance and elucidate the knowledge acquired within the neurons and may contribute in significantly increasing the acceptability of the ANN technique as a method for SCE by project managers and other researchers. Therefore, continuing the investigation towards this area is reasonable and beneficiary.

Another possible extension of this work is to apply the hybrid genetically evolved ANN to investigate non-SB-SCE since the accuracy results obtained with only size-related features were quite promising. Another significant investigation regarding the application of ANN in the SCE research area is to explore the statistical robustness of the technique by comparing the performance obtained in terms of training ability and generalisation ability.

Moreover, another general concern that needs to be quantified through the future SCE experiments carried out is the complexity of a method selected to be applied, compared to the accuracy increase of the predictions obtained with the respective method. Therefore, the balance between underfitting and overfitting needs to be found in order to achieve the best (or

optimal) model. This issue, although taken into consideration by some of the techniques presented in this work (e.g., ANN), it was not however quantified. Another related significant issue raised in the genetically evolved approaches of this dissertation (e.g., the hybrids ANN & GA and the CS & GA) which needs to be addressed in future work is the appropriate selection of fitness function. In the approaches developed and presented in this thesis, some of the goals set were rather conflicting (i.e., performance vs. complexity, narrow ranges vs. clustering performance), and thus the consideration to develop a Multi-Objective evolutionary algorithm to handle this issue might be a novel step forward.

Moreover, another future extension of this work is the coupling of the approach of Conformal Prediction (CP) with any other Computational Intelligent technique (such as ANN, FDT) to provide further positive contribution in the area of Predictive Intervals for SCE (PI-SCE). This extension is particularly worthy of investigation in future research since it takes into consideration the variability, volatility and uncertainty of several factors in the SCE process. An interesting addition to the investigation will be the discovery of appropriate (or acceptable) predictive intervals in industry from interaction with, and feedback from, industrial partners.

The incorporation of Fuzzy Logic theory was especially useful in several of the techniques developed in this research work (i.e., fuzzy clustering, FDT, FCM) and has showed great potential in addressing the uncertainty of the SCE process and in the values of the available datasets. Therefore, in the future, Fuzzy Logic might also be included in other approaches (like ANN, GP, ID) and thus enhance their properties and merits.

In addition, future plans include the expansion of the work conducted related with Phase-Based SCE (PB-SCE) and further investigation of the temporal dimension of project data to improve the accuracy already obtained through the preliminary experiments described in this dissertation.

Also, an interesting step of future work might include combinations of the approaches described in this work that have not been examined; for example fuzzy clustering or CS might

be employed to obtain ‘similar’ projects as a ‘filtering’ step and then the GP technique might be applied for obtaining more robust rules and predictions in SCE.

Other future plans of research include expanding the Qualitative models presented in this diatribe which provided quite promising conclusions regarding the development environment and SCE. Flexibility and adaptability in these models is a key notion. Particularly, further experimentation with the FCM-SCE models will aim at refining its parameters, widening the spectrum of real-cases and capturing the dynamics of a real situation in a software development environment. In future work also, some Concept Nodes of the FCM might be re-defined so that their value will be deterministic, restricted or biased. For example, the Process Maturity (CMM level) may be determined to a specific and unchanged value (e.g., CMM=3), the Application Domain Experience may only increase (e.g., from low to medium), and the Project Quality may only increase up to a certain point (e.g., from low up to average quality) with respect to the specific scenario executed. Moreover, in the FCM-SCE it might be interesting to observe the activation value of each concept and how the value is altered based on the specific scenario realised and investigate whether this change is acceptable or not based on the particular project and conditions. Other future possible extensions include locating the optimal weight matrices by applying a GA instead of consulting a group of experts. Such automated intelligent solutions may pave the way for simpler, future decision-support tools offering project managers a significant aid in SCE.

Future extensions of this work will also include utilisation of the already developed models and enhancing them towards different targets. For example, the SB-SCE models developed may be particularly useful in analysing Open Source software. Open source software today has become an overarching software development paradigm which can be benefited by ‘post-mortem’ analysis of the LOC of the source code produced.

Other future new approaches in SCE include the utilisation of Self-Organising Maps (SOM) for visualising the heterogeneity of vast datasets (like the ISBSG) and addressing the problem of multi-dimensionality. SOM may facilitate advancing our research efforts of utilising a multi-dimensional dataset in SCE which contains a plethora of software projects

and a large number of categorical and multi-valued attributes. Visualisation and analysis methods will enable increasing our understanding regarding such datasets and might contribute in taking decisions on how to reduce the dimensionality of the dataset, thus significantly minimising the complexity and time required to reach to an estimation using a particular SCE technique.

Also, as already mentioned, the proposed SCE models need to be investigated in real software development environments, where our future research activities might target software re-use, component-based software development, open source software, software size estimations utilising user requirements, use cases or other software artefacts, estimating duration (scheduling) and maintenance effort. SCE approaches need to be directed towards fields that have been adapted and adopted today by software engineers (e.g., object-orientation, very high level languages of development, rapid and agile development processes).

References

(Abdel-Hamid and Madnik, 1983) Abdel-Hamid, T.K., and Madnik, S.E. 1983. The Dynamics of Software Project Scheduling. *Communications of the ACM* 26 (5), 340-346.

(Abran and Robillard, 1994) Abran, A., and Robillard, P.N. Function Points: a Study of their Measurement Processes and Scale Transformations. *Journal of Systems and Software* 25, 171-184.

(Adamopoulos et al., 1998) Adamopoulos, A.V., Likothanassis, S.D., and Georgopoulos, E.F. 1998. A Feature Extractor of Seismic Data Using Genetic Algorithms, Signal Processing IX: Theories and Applications. In *Proceedings of the 9th European Signal Processing Conference* (*EUSIPCO*), Vol. 2, Island of Rhodes, Greece, 2429-2432.

(Aggarwal et el., 2005a) Aggarwal, K.K., Singh, Y., Chandra, P., Puri, M. 2005. Evaluation of various Training Algorithms in a Neural Network Model for Software Engineering Applications. *ACM SIGSOFT Software Engineering Notes* 30 (4), 1-4.

(Aggarwal et el., 2005b) Aggarwal, K.K., Singh, Y., Chandra, P., Puri, M. 2005. Bayesian Regularization in a Neural Network Model to Estimate Lines of Code Using Function Points. *Journal of Computer Sciences* 1 (4), 505-509.

(Aggarwal et el., 2005c) Aggarwal, K.K., Singh, Y., Chandra, P., Puri, M. 2005. An Expert Committee Model to Estimate Lines of Code. *ACM SIGSOFT Software Engineering Notes* 30 (5), 1-4.

(Ahmed et al., 2005) Ahmed, M.A., Saliu, M.O., and AlGhamdi, J. 2005. Adaptive Fuzzy logic-based Framework for Software Development Effort Prediction. *Information and Software Technology* 47, 31-48.

(Albrecht and Gaffney, 1983) Albrecht, A.J., and Gaffney, J.R. 1983. Software Function Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering* 9 (6), 639-648, ISSN: 0098-5589.

(Albrecht, 1979) Albrecht, A.J. 1979. Measuring Application Development. In *Proceedings of IBM Applications Development Joint SHARE/GUIDE Symposium*, Monterey, CA, USA, 83-92.

(Ambler, 2005) Ambler, S.W. 2005. A Manager's Introduction to the Rational Unified Process (RUP), December 2005 (online). Available at: http://www.ambysoft.com/downloads/managersIntroToRUP.pdf.

(Ambler, 2008**)** Ambler, S.W. 2008. Agile Adoption Rate 2008 (online). Available at: http://www.ambysoft.com/surveys/agileFebruary2008.html. Accessed: December 2011.

(Ambler, 2010**)** Ambler, S.W. 2010. Agile Project Success Rates 2010 (online). Available at: http://www.ambysoft.com/surveys/agileSuccess2010.html. Accessed: December 2011.

(Andreou and Papatheocharous, 2008a) Andreou, S.A., and Papatheocharous E. 2008. Computational Intelligence in Software Cost Estimation: Evolving conditional sets of effort value ranges. In *Tools in Artificial Intelligence*, I-Tech Education and Publication KG, Vienna, 1-20.

(Andreou and Papatheocharous, 2008b) Andreou, S.A., and Papatheocharous, E. 2008. Software Cost Estimation using Fuzzy Decision Trees. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering* (*ASE*), L'Aquila, Italy, 371-374.

(Andreou et al., 2004) Andreou, A.S., Mateou, N.H., and Zombanakis, G.A. 2004. Optimization in Genetically Evolved Fuzzy Cognitive Maps Supporting Decision-Making: The Limit Cycle Case. In *Proceedings of International Conference on Information and Communication Technologies: From Theory to Applications* (*ICCTA*), Damascus, Syria, 377-378.

(Andreou et al., 2007) Andreou, S.A., Papatheocharous E., and Skouroumounis C. 2007. Evolving Conditional Value Sets of Cost Factors for Estimating Software Development Effort. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI*), Vol. 1, Patras, Greece, IEEE Computer Society, 165-172.

(Angelis and Stamelos, 2000) Angelis, L., and Stamelos, I. 2000. A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Engineering 5(1), 35-68.

(Angelis et al., 2001) Angelis, L., Stamelos, I., Morisio, M. 2001. Building A Software Cost Estimation Model Based On Categorical Data. In *Proceedings of the 7th International Symposium on Software Metrics* (*METRICS*), London, United Kingdom, IEEE Computer Society, 4-15.

(Antoniol et al., 2003) Antoniol, G., Fiutem, R., and Lokan, C. 2003. Object-Oriented Function Points: An Empirical Validation. *Empirical Software Engineering* 8(3), 225-254.

(Aroba et al., 2008) Aroba, J., Cuadrado-Gallego, J.J., Sicilia, M., Ramos, I., and García-Barriocanal, E. 2008. Segmented Software Cost Estimation Models based on Fuzzy Clustering. *Journal of Systems and Software* 81, 1944-1950.

(Attarzadeh and Ow, 2010) Attarzadeh, I. and Ow, S.H. 2010. Improving the Accuracy of Software Cost Estimation Model Based on a New Fuzzy Logic Model. *World Applied Sciences Journal* 8 (2), 177-184.

(Axelrod, 1976) Axelrod, R. 1976. *Structure of Decision: The Cognitive Maps of Political Elites*. Princeton University Press, Princeton, NJ.

(Azoff, 1994) Azoff, E.M. 1994. *Neural Network Time Series Forecasting of Financial Markets*. John Wiley & Sons, NY.

(Azzeh et al., 2008) Azzeh, M., Neagu, D., Cowling P. 2008. Improving Analogy Software Effort Estimation using Fuzzy Feature Subset Selection Algorithm. In *Proceedings of 4th InternationalWorkshop on Predictor Models in Software Engineering* (*PROMISE*), Leipzig, Germany, 71–78.

(Azzeh et al., 2010) Azzeh, M., Neagu, D., and Cowling, P.I. 2010. Fuzzy Grey Relational Analysis for Software Effort Estimation. *Empirical Software Engineering* 15 (1), 60-90.

(Bailey and Basili, 1981) Bailey, J.J., and Basili, V.R. 1981. A Meta-model for Software Development Resource Expenditures. In *Proceedings of the 5th International Conference Software Engineering* (*ICSE*), San Diego, CA, USA, IEEE/ACM/NBS, 107-116.

(Barry et al., 2002) Barry, E.J., Mukhopadhyay, T., and Slaughter, S.A. 2002. Software Project Duration and Effort: An Empirical Study. *Information Technology and Management* 3(1-2), 113-136.

(Beale et al., 2011) Beale, M.H., Hagam, M.T., and Demuth, H.B. 2011. Neural Network Toolbox User's Guide R2011b, The MathWorks, Natick, MA. Available at: http://www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf. Accessed: December 2011.

(Beck et al., 2001) Beck, K., Grenning, J., Martin, C.R., Beedle, M., Highsmith, J., Mellor, S., Bennekum, van A., Hunt, A., Schwaber, K., Cockburn, A., Jeffries, R., Sutherland, J., Cunningham, W., Kern, J., Thomas, D., Fowler, M., and Marick, B. 2001. Manifesto for Agile Software Development. Agile Alliance (online). Available at: http://agilemanifesto.org/. Accessed: April 2010.

(Belue and Bauer, 1995) Belue, L.M., and Bauer, K.W. 1995. Determining Input Features for Multilayer Perceptrons. *Neurocomputing* 7, 111-121.

(Benediktsson and Dalcher, 2003) Benediktsson, O., and Dalcher, D. 2003. Effort Estimation in Incremental Software Development. *IEE Proceedings Software Engineering* 150 (6), 351-357.

(Benington, 1956) Benington, H.D. 1956. Production of Large Computer Programs. In *Proceedings of the ONR Symposium on Advanced Program Methods for Digital Computers*, Washington, DC, USA, 15-27. (Also available in the *Annals of the History of Computing*, October 1983, 350-361.)

(Betteridge, 1992) Betteridge, R. 1992. Successful Experience of Using Function Points to Estimate Project Costs Early in the Life-Cycle. *Information and Software Technology* 34 (10), 655-658.

(Bezdek, 1980) Bezdek, J.C. 1980. A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1), 1-8, ISSN 0162-8828.

(Bhatnagar et al., 2010) Bhatnagar, R., Bhattacharjee, V., and Ghose, M.K. 2010. A Proposed Novel Framework for Early Effort Estimation using Fuzzy Logic Techniques. *Global Journal of Computer Science and Technology* 10 (14), 66-72.

(Bibi et al., 2008) Bibi, M, Stamelos, I., and Angelis, L. Combining Probabilistic Models for Explanatory Productivity Estimation. *Information and Software Technology* 50, 656-669.

(Biggs et al., 1991) Biggs, D., de Ville, B, Suen, E. 1991. A Method for Choosing Multiway Partitions for Classification and Decision Trees. *Journal of Applied Statistics* 18 (1), 49-62.

(Boehm and Sullivan, 1999) Boehm, B.W., and Sullivan, K. 1999. Software Economics: Status and Prospects. *Information and Software Technology* 41 (14), 937–946.

(Boehm and Valerdi, 2008) Boehm, B.W., and Valerdi, R. 2008. Achievements and Challenges in COCOMO-Based Software Resource Estimation. *IEEE Software* 25 (5), 74-83.

(Boehm et al., 2000a) Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D. J., and Steece, B. 2000. *Software Cost Estimation with COCOMO II*. Prentice-Hall, Upper-Saddle River, NJ.

(Boehm et al., 2000b) Boehm, B.W., Abts, C., and Chulani, S. 2000. Software Development Cost Estimation Approaches—A survey. *Annals of Software Engineering* 10 (1), 177-205.

(Boehm, 1981) Boehm, B.W. 1981. *Software Engineering Economics*. Prentice-Hall Inc., Englewood Cliffs, NJ, ISBN: 0130266922.

(Boehm, 1984) Boehm, B.W. 1984. Software Engineering Economics. *IEEE Transactions on Software Engineering* 10 (1), 4-21.

(Boehm, 1988) Boehm, B.W. 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21 (5), 61-72.

(Braz and Vergilio, 2004) Braz, M.R., and Vergilio, S.R. 2004. Using Fuzzy Theory for Effort Estimation of Object-Oriented Software. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI*), Boca Raton, FL, USA, 196-201.

(Breiman et al., 1984) Breiman, L., Friedman, J., Oshlen, R., and Stone, C. 1984. *Classification and Regression Trees*, Wadsworth International Group, 1984.

(Briand and Wieczorek, 2000) Briand, L.C., and Wieczorek, I. 2000. Resource Estimation in Software Engineering. International Software Engineering Research Network, Technical Report ISERN-00-05, Fraunhofer Institute for Experimental Software Engineering, Germany.

(Brillinger, 2002) Brillinger, D.R. 2002. John W. Tukey: His life and professional contributions. *The Annals of Statistics* 30 (6), 1535-1575.

(Brooks, 1995) Brooks, Jr, F.P. 1995. *The Mythical Man-month (anniversary edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

(Burgess and Lefley, 2001) Burgess, C.J., and Lefley, M. 2001. Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation. *Information and Software Technology* 43, 863-873.

(Ceschi et al., 2005) Ceschi, M., Sillitti, A., Succi, G., and De Panfilis, S. 2005. Project Management in Plan-Based and Agile Companies. *IEEE Software* 22, 21-25.

(Chandrasekaran et al., 2006) Chandrasekaran, S, Lavanya R., and Kanchana V. 2006. Multi-criteria Approach for Agile Software Cost Estimation Model. In *Proceedings of the International Conference Global Manufacturing and Innovation* (*GMI*), Coimbatore, India.

(Chiu and Huang, 2007) Chiu, N., and Huang, S. 2007. The Adjusted Analogy-based Software Effort Estimation based on Similarity Distances. *Journal of Systems and Software* 80, 628-640.

(Conte et al., 1986) Conte, S.D., Dunsmore, H.E., and Shen, V.Y. 1986. *Software Engineering Metrics and Models*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA.

(Cooper, 1988) Cooper, G.F. 1988. A method for using Belief Networks as Influence Diagrams. In *Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence* (*UAI*), Minneapolis, MN, USA, 55-63.

(Decision Systems Laboratory: GeNIE, 1998) Decision Systems Laboratory, University of Pittsburgh. Graphical Network Interface (GeNIE). 1998 (online). Available at: http://genie.sis.pitt.edu/. Accessed: October, 2010.

(DeMarco and Lister, 1999) DeMarco, T., and Lister, T. 1999. *Peopleware: Productive Projects and Teams* (2nd edition). Dorset House Publishing Co., Inc., New York, NY.

(DeMarco, 1982) DeMarco, T. 1982. *Controlling Software Projects*. Prentice-Hall, Englewood Cliffs, NJ.

(Demuth and Beale, 2000) Demuth, H., and Beale, M. 2000. Neural Network Toolbox (for use with Matlab). MathWorks, Natick, MA.

(Desharnais, 1989) Desharnais, J.M. 1988. Analyse Statistique de la Productivite des Projects de Development en Informatique a Partir de la Technique de Points de Fonction. MSc. Thesis, Université du Québec, Montréal, Canada.

(Dircks, 1981) Dircks, H.F. 1981. SOFCOST: Grumman's Software Cost Eliminating Model. In *Proceedings of the IEEE National Aerospace and Electronics Conference* (*NAECON*), Dayton, OH, USA.

(Draper and Smith, 1998) Draper, N.R. and Smith, H. 1998. Applied Regression Analysis. Wiley-Interscience, Hoboken, NJ, 307–312.

(Engelbrecht , 2007) Engelbrecht, A.P. 2007. *Computational Intelligence: An Introduction* (2nd edition). John Wiley & Sons Ltd, West Sussex, England.

(Fenton and Pfleeger, 1997) Fenton, N.E., and Pfleeger, S.L. 1997. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA.

(Fenton, 2000) Fenton, N.E., and Neil, M. 2000. Software Metrics: Roadmap. In *Proceedings of the 22nd International Conference of Software Engineering* (*ICSE*), Future of Software Engineering Track, Limerick, Ireland. ACM, 357–370.

(Finnie et al., 1997) Finnie, G.R., Wittig, G.E., and Desharnais, J-M. 1997. A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. *Journal of Systems and Software* 39 (3), 281-289.

(Foss et al., 2003) Foss, T., Stensrud, E., Kitchenham, B.A., and Myrtveit, I. 2003. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering* 29, 985-995.

(Fowler et al., 2000) Fowler, M., Parsons, R., and MacKenzie, J. 2000 (Talk): Coined the term POJO in a 2000 Conference (online). Available at: http://www.martinfowler.com/bliki/POJO.html. Accessed: December 2011.

(Freiman and Park, 1979) Freiman, F.R., and Park, R.D. 1979. PRICE Software Model-Version 3: An Overview. In *Proceedings of the IEEE-PINY Workshop on Quantitative Software* Models, IEEE Cat. TH0067-9, 32-41.

(Garson, 1991) Garson, G.D. 1991. Interpreting Neural-Network Connection Weights. *AI Expert* 6, 46-51.

(Genuchten and Koolen, 1991) Genuchten, M.V., and Koolen, H. 1991. On the Use of Software Cost Models. *Information and Management* 21 (1), 37-44.

(Ghezzi et al., 2003) Ghezzi, C., Jazayeri M., and Mandrioli, D. 2003. *Fundamentals of Software Engineering* (2nd edition). Pearson Prentice Hall, Upper Saddle River, NJ.

(Gilb, 1976) Gilb T. 1976. *Software Metrics*, Chartwell-Bratt, Cambridge MA.

(Glorfeld, 1996) Glorfeld, L.W. 1996. A Methodology for Simplification and Interpretation of Backpropagation-Based Neural Network Models. *Expert Systems with Applications* 10, 37-54.

(Gray and MacDonell, 1997a) Gray, A.S., and MacDonell, S.G. 1997. A Comparison of Techniques for Developing Predictive Models of Software Metrics. *Information and Software Technology* 39, 425–437.

(Gray and MacDonell, 1997b) Gray, A.S., and MacDonell, S.G. 1997. Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation. In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society* (*NAFIPS*), Syracuse, NY, USA, 394-399.

(Grimstad et al., 2006) Grimstad, S., Jørgensen, M., and Moløkken-Østvold, K. 2006. Software Effort Estimation Terminology: The Tower of Babel. *Information and Software Technology* 48 (4), 302-310.

(Gruschke and Jørgensen, 2008) Gruschke, T.M., and Jørgensen, M. 2008. The role of Outcome Feedback in Improving the Uncertainty Assessment of Software Development Effort Estimates. *ACM Transactions of Software Engineering Methodology* 17, 1–35.

(Hamming, 1950) Hamming, R.W. 1950. Error Detecting and Error Correcting Codes. *Bell System Technical Journal* 29 (2), 147–160.

(Han et al., 2006) Han, J., Kamber, M., and Pei, J. 2006. *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, (3rd edition), Morgan Kaufmann, Waltham, MA.

(Haykin, 1999) Haykin, S. 1999. Neural Networks: A Comprehensive Foundation (2nd edition). Prentice Hall, Upper Saddle River, NJ.

(Heemstra and Kusters, 1991) Heemstra, F.J., and Kusters, R.J. 1991. Function Point Analysis: Evaluation of a Software Cost Estimation Model. *European Journal of Information Systems* 1(4), 223-237.

(Heiat, 2002) Heiat, A. 2002. Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort. *Information and Software Technology* 44 (15), 911-922.

(Herd et al., 1977) Herd, J.R., Postak, J.N., Russell, W.E., and Steward, K.R. 1977. Software Cost Estimation Study-Study Results, Final Technical Report (RADC-TR-77-220) Vol. 1, Doty Associates, Inc., Rockville, MD.

(Highsmith, 2003) Highsmith, J. 2003, *Agile Project Management: Principles and Tools*. Cutter Consortium 4, 1-37.

(Hill et al., 2000) Hill, J., Thomas, L.C., and Allen, D.E. 2000. Experts' Estimates of Task Durations in Software Development Projects. *International Journal of Project Management* 18(1), 13-21.

(Holland, 1992) Holland, J.H. 1992. Genetic Algorithms. *Scientific American* 267 (1), NY, 66–72.

(Horgan et al., 1998) Horgan, G., Khaddaj, S., and Forte, P. 1998. Construction of an FPA-Type Metric for Early Lifecycle Estimation. *Information and Software Technology* 40 (8), 409-415.

(Höst and Wohlin, 1997) Höst, M., and Wohlin, C. 1997. A Subjective Effort Estimation Experiment. *Information and Software Technology* 39 (11), 755-762.

(Howard and Matheson, 1984) Howard, R.A., and Matheson, J.E. 1984. Influence diagrams. *Readings on the Princinples and Applications of Decision Analysis II*, Howard, R. A., and Matheson, J. E. eds., Strategic Decision Group, Menlo Park, CA, 719–762. (Reprinted: Howard, R.A., and Matheson, J.E. 2005. Influence diagrams. Decision Analysis 2 (3), 127–143).

(Huang and Chiu, 2006) Huang, S., and Chiu, N. 2006. Optimization of Analogy Weights by Genetic Algorithm for Software Effort Estimation. *Information and Software Technology* 48, 1034-1045.

(Huang et al., 2003) Huang, X.L., Capretz, J.R., and Ho, D. 2003. A Neuro-Fuzzy Model for Software Cost Estimation. In *Proceedings of the 3rd International Conference on Quality Software* (*QSIC*), Dallas, TX, USA, 126-133.

(Huang et al., 2006) Huang, S.-J., Lin, C.-Y., and Chiu, N.-H. 2006. Fuzzy Decision Tree Approach for Embedding Risk Assessment Information into Software Cost Estimation Model. *Software Engineering and Software* 22, 297-313.

(Huang et al., 2008) Huang, S.J., Chiu, N.H., and Chen, L.W. 2008. Integration of the Grey Relational Analysis with Genetic Algorithm for Software Effort Estimation. *European Journal of Operational Research* 188 (3), 898-909.

(Huang, 1998) Huang, Z. 1998. Extensions to the *k*-Means Algorithm for Clustering Large Datasets with Categorical Values. *Data Mining and Knowledge Discovery 2* (3), 283-304.

(Huang, 1999) Huang, Z., and Ng, M.K. 1999. A Fuzzy k-Modes Algorithm for Clustering Categorical Data. *IEEE Transactions on Fuzzy Systems* 7 (4), 446-452.

(Hughes 1997) Hughes, R.T. 1997. An Evaluation of Machine Learning Techniques for Software Effort Estimation. PhD Thesis, Department of Computing, University of Brighton, United Kingdom.

(Hughes, 1996) Hughes, R.T. 1996. Expert Judgement as an Estimating Method. *Information and Software Technology* 38(2), 67-75.

(Idri and Abran, 2001) Idri, A., and Abran, A. 2001. A Fuzzy Logic Based Set of Measures for Software Project Similarity: Validation and Possible Improvements. In *Proceedings of the 7th International Software Metrics Symposium* (*METRICS*), London, United Kingdom, 86-96.

(Idri et al., 2000) Idri, A., Abran, A., and Krishna, L. 2000. COCOMO Cost Model using Fuzzy Logic. In *Proceedings of the 7th International Conference on Fuzzy Theory and Techniques*, Atlantic City, NJ, USA, 219-223.

(Idri et al., 2004) Idri, A., Mbarki, S., Abran, A. 2004. Validating and Understanding Software Cost Estimation Models based on Neural Networks. In *Proceedings of the 1st International Conference on Information and Communication Technologies: From Theory to Applications* (*ICTTA*), Damascus, Syria, 433-434.

(IEEE standard 610.12, 1990) IEEE Standard Glossary of Software Engineering Terminology, IEEE standard 610.12-1990, 1990.

(ISBSG, 2007a) ISBSG. Special Analysis Report: Early Lifecycle Software Estimation. Report. Available at: http://www.isbsg.org/.

(ISBSG, 2007b) ISBSG. Guidelines for use of the ISBSG data. Report. Available at: http://www.isbsg.org/.

(ISBSG, Repository Data Release 10) International Software Benchmarking Standards Group (ISBSG). 2008. The Benchmark Release 10. Available at: http://www.isbsg.org.

(ISBSG, Repository Data Release 9) International Software Benchmarking Standards Group (ISBSG). 2005. Estimating, Benchmarking & Research Suite Release 9, Victoria. Available at: http://www.isbsg.org/.

(ISO/IEC 20926, 2003) IFPUG. 2003. IFPUG 4.1 Unadjusted Functional Size Measurement Method – Counting Practices Manual, International Organization for Standardization, ISO/IEC 20926:2003.

(Jensen, 1983) Jensen, R.W. 1983. An Improved Macrolevel Software Development Resource Estimation Model. In *Proceedings of the 5th International Society of Parametric Analysts* (*ISPA*) *Conference*, 88-92.

(Johnson, 2003) Johnson, M. 2003. Agile methodologies: Survey results. Victoria, Australia: Shine Technologies (online). Available at: http://www.shinetech.com/attachments/104_ShineTechAgileSurvey2003-01-17.pdf. Accessed: Accessed: April 2010.

(Jones, 2007) Jones, C. 2007. *Estimating Software Costs: Bringing Realism to Estimating* (2nd edition), McGraw-Hill Osborne Media, NY.

(Jørgensen and Moløkken, 2002) Jørgensen, M., and Moløkken, K. 2002. Combination of Software Development Effort Prediction Intervals: Why, When and How? In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering* (*SEKE*), Ischia, Italy, 425–428.

(Jørgensen and Shepperd, 2007) Jørgensen, M., and Shepperd, M. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering* 33 (1), 33-53.

(Jørgensen et al., 2004) Jørgensen, M., Teigen, K.H., and Moløkken, K. 2004. Better sure than safe? Over-confidence in judgment based software development effort prediction intervals. *Journal of Systems and Software* 70, 79-93.

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

(Jørgensen, 2004a) Jørgensen, M. 2004. A Review of Studies on Expert Estimation of Software Development Effort. *Journal of Systems and Software* 70, 37-60.

(Jørgensen, 2004b) Jørgensen, M., Realism in Assessment of Effort Estimation Uncertainty: It Matters How You Ask. *IEEE Transactions on Software Engineering* 30 (4).

(Jørgensen, 2004c) Jørgensen, M. 2004. Top-Down and Bottom-Up Expert Estimation of Software Development Effort. *Information and Software Technology* 46 (1), 3-16.

(Jørgensen, 2007) Jørgensen, M. 2007. Forecasting of Software Development Work Effort: Evidence on Expert Judgement and Formal Models. *International Journal Forecast* 23(3), 449–462.

(Kan, 2003) Kan, S.H. 2003. *Metrics and Models in Software Quality Engineering* (2nd edition). Addison-Wesley Longman Publishing Co., Inc., Upper Saddle River, NJ.

(Karray and Silva, 2004) Karray, F.O., and Silva, C.W.D. 2004. *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*. Addison-Wesley, London, UK.

(Kartalopoulos, 1996) Kartalopoulos, S.V. 1996. *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications*. IEEE Press, NY, ISBN.0-7803-1128-0.

(Kass, 1990) Kass, G.V. 1990. An Exploratory Technique for Investigating Large Quantities of Categorical Data. Applied Statistics 20 (2), 119-127.

(Kaur et al., 2010) Kaur, J., Singh, S., Kahlon, K.S., and Bassi, P. 2010. Neural Network – A Novel Technique for Software Effort Estimation. *International Journal of Computer Theory and Engineering* 2 (1) 1793-8201, 17-19.

(Kemerer, 1987) Kemerer, C.F. 1987. An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM* 30 (5), 416-429, ISSN:0001-0782.

(Kemerer, 1993) Kemerer, C. 1993. Reliability Function Points Measurement: a Field Experiment. *Communications of the ACM* 36(2), 85-97.

(Keung et al., 2008) Keung, J.W., Kitchenham, B.A., and Jeffery, D.R. 2008. Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation. *IEEE Transactions on Software Engineering* 2008 34(4), 471–484.

(Keung et al., 2012) Keung, J., Kocaguneli, E., and Menzies, T. 2012. A Ranking Stability Indicator for Selecting the Best Effort Estimator in Software Cost Estimation. *Journal of Automated Software Engineering*, Under review.

(Kim et al., 2004) Kim, D.-W., Lee, K.H., and Lee, D. 2004. Fuzzy Clustering of Categorical Data Using Fuzzy Centroids. *Pattern Recognition Letters* 25 (11), 1263-1271, ISSN 0167-8655.

(Kirsopp and Shepperd, 2002) Kirsopp, C., and Shepperd, M. 2002. Case and Feature Subset Selection in Case-Based Software Project Effort Prediction. In *Proceedings of the 22nd SGAI International Conference of Knowledge-Based Systems and Applied Artificial Intelligence*, Cambridge, United Kingdom.

(Kitchenham and Linkman, 1997) Kitchenham, B.A., and Linkman, S. 1997. Estimates, Uncertainty, and Risk. *IEEE Software* 14 (3), 69-74.

(Kitchenham and Mendes, 2009) Kitchenham, B.A., and Mendes, E. 2009. Why Comparative Effort Prediction Studies may be Invalid. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering* (*PROMISE*), Vancouver, BC, Canada, ACM, 1-5.

(Kitchenham and Taylor, 1985) Kitchenham, B.A., and Taylor, N.R. 1985. Software Project Development Cost Estimation. *Journal of Systems and Software* 5, 267-278.

(Kitchenham et al., 1997) Kitchenham, B.A., Linkman, S.G., Law, D.: DESMET: A methodology for Evaluating Software Engineering Methods and Tools. *IEE Computing and Control Journal*, 120-126.

(Kitchenham et al., 2001) Kitchenham, B.A., MacDonell, S.G., Pickard, L., Shepperd, M. 2001. What Accuracy Statistics Really Measure. *IEE Proceedings of Software Engineering* 148(3), 81-85.

(Kitchenham et al., 2002) Kitchenham, B.A., Pfleeger, S.L., McColl, B., and Eagan, S. 2002. An Empirical Study of Maintenance and Development Estimation Accuracy. *Journal of Systems and Software* 64 (1), 57–77.

(Kitchenham et al., 2003) Kitchenham, B.A., Pickard, L.M., Linkman, S., and Jones, P.W. 2003. Modeling Software Bidding Risks. *IEEE Transactions on Software Engineering* 29 (6), 542-554.

(Kitchenham et al., 2004) Kitchenham, B.A., Dyba, T., and Jørgensen, M. 2004. Evidence-Based Software Engineering. In *Proceedings of the 26th International Conference on Software Engineering* (*ICSE*), Edinburg, Scotland, 273- 281.

(Kitchenham, 1990) Kitchenham, B.A. 1990. Measuring Software Development. In *Software Reliability Handbook*, Elsevier, Amsterdam, 303-332.

(Koivo, 2008) Koivo, H.N. 2008. Neural Networks: Basics using Matlab Neural Network Toolbox (online). Available at: http://pis.unicauca.edu.co/moodle/file.php/458/2010b/clase_29/AS-74_3115_neural_networks_-_basics.pdf. Accessed: December 2011.

(Kosko, 1986) Kosko, B. 1986. Fuzzy Cognitive Maps. International Journal of Man-Machine Studies 24, 65-75.

(Kosko, 1995) Kosko, B. 1995. *Fuzzy Thinking, the New Science of Fuzzy Logic* (2nd edition). Harper Collins, London, UK.

(Koza, 1992) Koza, J.R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Massachusetts.

(Kumar et al., 2008) Kumar, K.V., Ravi, V., Carr, M., and Kiran, N.R. 2008. Software Development Cost Estimation using Wavelet Neural Networks. *Journal of Systems and Software* 81, 1853-1867.

(Laanti and Kettunen, 2006) Laanti, M., and Kettunen P., Cost Modeling Agile Software Development. *International Transactions on Systems Science and Applications* 1 (2), 175-179.

(Laird and Brennan, 2006) Laird, L.M., and Brennan, M.C. 2006. *Software Measurement and Estimation: A Practical Approach*, Quantitative Software Engineering Series. Wiley-IEEE Computer Society Press, Hoboken, NJ.

(Lefley and Shepperd, 2003) Lefley, M., and Shepperd, M.J. 2003. Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets. *In Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO*), Chicago, IL, USA, 2477-2487.

(Leung, 2002) Leung, H., and Fan, Z. 2002. Software Cost Estimation, *Handbook of Software Engineering and Knowledge Engineering* 2, S.K. Chang, Ed., World Scientific, River Edge, NJ, Available at: ftp://cs.pitt.edu/chang/handbook/42b.pdf, Accessed: December 2011.

(Levenberg, 1944) Levenberg, K. 1944. A Method for the Solution of Certain Problems in Least Squares. *Quarterly Applied Mathematics* 2, 164-168.

(Li and Ruhe, 2008) Li, J.Z., and Ruhe, G. 2008. Analysis of Attribute Weighting Heuristics for Analogy-Based Software Effort Estimation Method AQUA+. *Empirical Software Engineering* 13 (1), 63-96.

(Li et al., 2007) Li, J.Z., Ruhe, G., Al-Emran, A., and Richter, M.M. 2007. A Flexible Method for Effort Estimation by Analogy. *Empirical Software Engineering* 12 (1), 65-106.

(Li et al., 2009a) Li, Y.F., Xie, M., and Goh, T.N. 2009. A Study of Mutual Information Based Feature Selection for Case Based Reasoning in Software Cost Estimation. *Expert Systems with Applications* 36 (3), 5921-5931.

(Li et al., 2009b) Li, Y.F., Xie, M., and Goh, T.N. 2009. A Study of Project Selection and Feature Weighting for Analogy Based Software Cost Estimation, *Journal of Systems and Software* 82 (2), 241-252.

(Li et al., 2010) Li, Y.F., Xie, M., and Goh, T.N. 2010. Adaptive Ridge Regression System for Software Cost Estimating on Multi-Collinear Datasets. *Journal of Systems and Software* 83 (11), 351-363.

(Liu et al., 2008) Liu, Q., Qin, W.Z., Mintram, R., and Ross, M. 2008. Evaluation of Preliminary Data Analysis Framework in Software Cost Estimation based on ISBSG R9 Data. *Software Quality Journal* 16 (3), 411-458.

(MacDonell and Gray, 1997) MacDonell, S.G., and Gray, A.R. 1997. A Comparison of Modeling Techniques for Software Development Effort Prediction. In *Proceedings of the International Conference on Neural Information Processing and Intelligent Information Systems* (*ICNIP*), Dunedin, New Zealand, Springer-Verlag, 869-872.

(MacDonell and Shepperd, 2003a) MacDonell, S.G., and Shepperd, M.J. 2003. Combining Techniques to Optimize Effort Predictions in Software Project Management. *Journal of Systems and Software* 66 (2), 91-98.

(MacDonell and Shepperd, 2003b) MacDonell, S.G., and Shepperd, M.J. 2003. Using Prior-Phase Effort Records for Re-estimation During Software Projects. In *Proceedings of the 9th IEEE International Software Metrics Symposium* (*METRICS*), Sydney, Australia, IEEE Computer Society, 1-13.

(Maimon and Rokach, 2005) Maimon, O.Z., and Rokach, L. 2005. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York Inc., NY.

(Mair and Shepperd, 2005) Mair, C., and Shepperd, M. 2005. The Consistency of Empirical Comparisons of Regression and Analogy-based Software Project Cost Prediction. In *Proceedings of the International Symposium on Empirical Software Engineering* (*ISESE*), Noosa Heads, Australia, 509-518.

(Mair et al., 2000) Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., and Webster, S. 2000. An Investigation of Machine Learning Based Prediction Systems. *Journal of Systems and Software* 53, 23-29.

(Mamdani, 1977) Mamdani, E.H. 1977. Applications of Fuzzy Set Theory to Control Systems: A Survey. In *Fuzzy Automata and Decision Processes*, Gupta, M.M., Saridis, G.N., and Gaines B.R., Eds. North-Holland, New York, 1-13.

(Mann and Whitney, 1947) Mann, H.B., and Whitney, D.R. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics* 18 (1), 50–60.

(Maritz, 1981) Maritz, J.S. 1981. *Distribution-Free Statistical Methods*. Chapman & Hall, London, UK.

(Marquardt, 1963) Marquardt, D. 1963. An Algorithm for Least-Squares Estimation of Non-linear Parameters. *SIAM Journal of Applied Mathematics* 11, 431-441.

(Martin, 2002) Martin, R.C. 2002. *Agile Software Development, Principles, Patterns, and Practices* (1st edition), Prentice Hall.

(Mateou et al., 2005) Mateou, N.H., Hadjiprokopis, A.P., and Andreou, A.S 2005. Fuzzy Influence Diagrams: An Alternative Approach to Decision Making Under Uncertainty. In *Proceedings of International Conference on Computational Intelligence for Modelling* (*CIMCA*), Control and Automation, Vienna, Austria, IEEE Computer Society, Vol. 1, 58-64.

(MathWorks, 2009) The MathWorks. 2009. Genetic Algorithm and Direct Search Toolbox User's Guide R2009b, Version 2.4.2, The MathWorks Inc., Natick, MA.

(Matson et al., 1994) Matson, J.E., Barrett, B.E., and Mellichamp, J.M. 1994. Software Development Cost Estimation using Function Points. *IEEE Transactions on Software Engineering* 20, 275-287.

(McCulloch and Pitts, 1943) McCulloch, W.S., and Pitts, W. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology* 5 (4), 115-133.

(Menzies et al., 2006) Menzies, T., Chen, Z., Hihn, J., and Lum, K. 2006. Selecting Best Practices for Effort Estimation. *IEEE Transactions on Software Engineering* 32, 883-895.

(Menzies et al., 2010) Menzies, T., Jalali, O., Hihn, J., Baker, D., and Lum, K. 2010. Stable Rankings for Different Effort Models. *Automated Software Engineering* 17 (4), 409-437.

(Meyer and Packard, 1992) Meyer, T.P., and Packard, N.H. 1992. Local Forecasting of High-dimensional Chaotic Dynamics, Non-linear Modeling and Forecasting. Addison-Wesley.

(Michalewicz, 1994) Michalewicz, Z. 1994. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin.

(Mills and O'Neil, 1980) Mills, H.D., and O'Neil, D. 1980. The Management of Software Engineering. *IBM Systems Journal* 24 (2), 414-477.

(Mittal et al., 2010) Mittal, A., Parkash, K., and Mittal, H. 2010. Software Cost Estimation Using Fuzzy Logic. *ACM SIGSOFT Software Engineering Notes* 35 (1), 1-7.

(Mittas et al., 2010) Mittas, N., Kosti, M.V., Argyropoulou, V., and Angelis, L. 2010. Modeling the Relationship between Software Effort and Size Using Deming Regression. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (*PROMISE*), Timisoara Romania, ISBN: 978-1-4503-0404-7.

(Miyazaki et al., 1994) Miyazaki, Y., Terakado, M., Ozaki, K., and Nozaki, H. 1994. Robust Regression for Developing Software Estimation Models. *Journal of Systems and Software* 27 (1), 3-16, ISSN: 0164-1212.

(Moløkken and Jørgensen, 2003) Moløkken, K., and Jørgensen, M. 2003. A Review of Software Surveys on Software Effort Estimation. In *Proceedings of the International Symposium on Empirical Software Engineering* (*ISESE*), Rome, Italy, 223-230.

(Musilek et al., 2000) Musilek, P., Pedrycz, W., Succi, G., and Reformat, M. 2000. Software Cost Estimation with Fuzzy Models. *Applied Computing Review* 8 (2), 24-29.

(Myers and Arnold, 2003) Myers, J.L., and Arnold, D.W. 2003. *Research Design and Statistical Analysis* (2nd edition). Lawrence Erlbaum Associates Inc., NJ.

(Myrtveit et al., 2005) Myrtveit, I., Stensrud, E., and Shepperd, M. 2005. Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Transactions on Software Engineering* 31 (5), 380-391.

(Naumann and Jenkins, 1982) Naumann, J.D., and Jenkins, A.M. 1982. Prototyping: The Bew Paradigm for Systems Development. *MIS Quarterly* 6 (3), 29-44.

(Naur and Randell, 1968) Naur, P., and Randell, B., Eds. 1969, Software Engineering: Report on *NATO Conference*, Garmisch, Germany, October 7–10, 1968.

(Nelson, 1966) Nelson, E.A. 1966. Management Handbook for the Estimation of Computer Programming Costs, Systems Development Corporation, Santa Monica, CA. NTIS Report No. AD-A 648750. Available from National Technical Information Service (NTIS), Springfield, Virginia. Available at: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=AD0648750. Accessed: December 2011.

(Nguyen and Widrow, 1990) Nguyen, D., and Widrow, B. 1990. Improving the Learning Speed of 2-layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In *Proceedings of the International Joint Conference on Neural Networks* (*ICJNN*), Vol. 3, Wasington, DC, USA, 21–26.

(Nguyen et al., 2008) Nguyen, V., Steece, B., and Boehm, B.W. 2008. A Constrained Regression Technique for COCOMO Calibration. In *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (*ESEM*), Kaiserslautern, Germany, 213-222.

(Norden, 1958) Norden, P.V. 1958. Curve Fitting for a Model of Applied Research and Development Scheduling. *IBM Journal Research and Development* 2 (3), 232-248.

(Nouretdinov et al., 2001) Nouretdinov, I., Vovk, V., Vyugin, M.V., and Gammerman, A. 2001. Pattern Recognition and Density Estimation under the General i.i.d. Assumption. In *Proceedings of the 14th Annual Conference on Computational Learning Theory* (*COLT*) and *5th European Conference on Computational Learning Theory* (*EuroCOLT*), Amsterdam, Netherlands. Lecture Notes in Computer Science 2111, Springer, 337–353.

(Olden and Jackson, 2002) Olden, J.D., and Jackson, D.A. 2002. Illuminating the "Black Box": a Randomization Approach for Understanding Variable Contributions in Artificial Neural Networks. *Ecological Modelling* 154, 135-150.

(Oliveira et al., 2010) Oliveira, A.L.I., Braga, P.L., Lima, R.M.F., and Cornélio M. 2010. GA-based Method for Feature Selection and Parameters Optimization for Machine Learning Regression Applied to Software Effort Estimation. *Information and Software Technology* 52 (11), 1155-1166.

(Packard, 1990) Packard, N.H. 1990. A Genetic Learning Algorithm for the Analysis of Complex Data. *Complex Systems* 4 (5), 543-572.

(Papadopoulos et al., 2009) Papadopoulos, H., Papatheocharous, E., and Andreou, S.A. 2009. Reliable Confidence Intervals for Software Effort Estimation. In *Proceedings of the 2nd Artificial Intelligence Techniques in Software Engineering Workshop* (*AISEW*), 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI), Thessaloniki, Greece, 211-220. Available at: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-475/AISEW2009/22-pp-211-220-208.pdf.

(Papatheocharous and Andreou, 2007) Papatheocharous, E., and Andreou, S.A. 2007. Software Cost Estimation using Artificial Neural Networks with Inputs Selection. In *Proceedings of the 9th International Conference on Enterprise Information Systems* (*ICEIS*), Madeira, Portugal, Vol. Databases and Information Systems Integration (DISI), 398-407.

(Papatheocharous and Andreou, 2008) Papatheocharous, E., and Andreou, S.A. 2008. Size and Effort-based Computational Models for Software Cost Prediction. In *Proceedings of the 10th International Conference on Enterprise Information Systems* (*ICEIS*), Barcelona, Spain, Vol. Databases and Information Systems Integration (DISI), Databases and Information Systems Integration, 57-64.

(Papatheocharous and Andreou, 2009a) Papatheocharous, E., and Andreou, S.A. 2009 Approaching Software Cost Estimation Using an Entropy-based Fuzzy *k*-modes Clustering Algorithm. In *Proceedings of the 2nd Artificial Intelligence Techniques in*

Software Engineering Workshop (*AISEW*), 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI), Thessaloniki, Greece, 231-241. Available at: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-475/AISEW2009/24-pp-231-241-211.pdf.

(Papatheocharous and Andreou, 2009b) Papatheocharous, E., and Andreou, S.A. 2009. Classification and Prediction of Software Cost Through Fuzzy Decision Trees. *Lecture Notes in Business Information Processing* (*LNBIP*) 24, Enterprise Information Systems, 11th International Conference, ICEIS 2009, Milan, Italy, May 2009 Proceedings, Filipe J., and Cordeiro, J., Eds. Springer-Verlag, Berlin Heidelberg, 234-247.

(Papatheocharous and Andreou, 2009c) Papatheocharous, E., and Andreou, S.A. 2009. Hybrid Computational Models for Software Cost Prediction: An Approach Using Artificial Neural Networks and Genetic Algorithms. *Lecture Notes in Business Information Processing* (*LNBIP*) 19, Enterprise Information Systems, 10th International Conference ICEIS 2008, Barcelona, Spain, June 2008, Revised Selected Papers, Filipe J., and Cordeiro, J., Eds. Springer-Verlag, Berlin Heidelberg, 87-100.

(Papatheocharous and Andreou, 2010) Papatheocharous, E., and Andreou, A.S. 2010. On the Problem of Attribute Selection for Software Cost Estimation: Input Backward Elimination Using Artificial Neural Networks. In *Artificial Intelligence Applications and Innovations* (AIAI), Papadopoulos, H., Andreou, A.S., and Bramer, M., Eds. Vol. 339, Springer Berlin Heidelberg, 287-294. Available at: http://www.springerlink.com/content/7607363884w45248/.

(Papatheocharous and Andreou, 2011) Papatheocharous, E., and Andreou, S.A. 2011. Size-based Software Cost Modelling with Artificial Neural Networks and Genetic Algorithms. *Artificial Neural Networks Application*, In-Tech Open Access Publisher, 168-188. Available at: http://www.intechopen.com/source/pdfs/14908/InTech-Size_based_software_cost_modelling_with_artificial_neural_networks_and_genetic_algorithms.pdf.

(Papatheocharous and Andreou, 2012a) Papatheocharous, E., and Andreou, S.A. 2012. A Hybrid Software Cost Estimation Approach Utilizing Decision Trees and Fuzzy Logic. To appear in: *Journal of Software Engineering and Knowledge Engineering*.

(Papatheocharous and Andreou, 2012b) Papatheocharous, E., and Andreou, S.A. 2012. Software Cost Modelling and Estimation Using Artificial Neural Networks Enhanced by Input Sensitivity Analysis. To appear: *Journal of Universal Computer Science*.

(Papatheocharous et al., 2008) Papatheocharous, E., Rossides, G., and Andreou, S.A. 2008 Qualitative Software Cost Estimation Using Fuzzy Cognitive Maps. In *Proceedings of the Artificial Intelligence Techniques in Software Engineering Workshop* (*AISEW*), 18th European Conference on Artificial Intelligence (ECAI), Patras, Greece, 1-5.

(Papatheocharous et al., 2010a) Papatheocharous, E., Iasonos, A., and Andreou, S.A. 2010. A Genetic Programming Approach to Cost Modeling and Estimation. In *Proceedings of the 12th International Conference on Enterprise Information Systems* (*ICEIS*), Funchal, Portugal, Vol. 1 Databases and Information Systems Integration (DISI), 281-287.

(Papatheocharous et al., 2010b) Papatheocharous, E., Papadopoulos, H., and Andreou, S.A. 2010 Feature Selection Techniques for Software Cost Modelling and Estimation. *Engineering Intelligent Systems* 18 (3/4), September/December, CRL Publishing, 233-246.

(Papatheocharous et al., 2010c) Papatheocharous, E., Papadopoulos, H., and Andreou, S.A. 2010. Software Effort Estimation with Ridge Regression and Evolutionary Attribute Selection. In *Proceedings of the 3rd Artificial Intelligence Techniques in Software Engineering Workshop* (*AISEW*), 6th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI), Larnaca, Cyprus. Available at: http://arxiv.org/abs/1012.5754.

(Papatheocharous et al., 2011) Papatheocharous, E., Trikomitou, D., Yiasemis, P.S., and Andreou, S.A. 2011. Cost Modeling and Estimation in Agile Software Development Environments Using Influence Diagrams. *In Proceedings of the 13th International*

Conference on Enterprise Information Systems (*ICEIS*), Beijing, China, Vol. 3 Information Systems Analysis and Specification, 117-127.

(Papatheocharous et al., 2012) Papatheocharous, E., Bibi, S., Stamelos, I., and Andreou A.S. 2012. Investigating Empirically Effort Distribution among Development Phases: A Four Stage Progressive Software Effort Estimation Model. Under Preparation.

(Papatheocharous, 2004) Papatheocharous, E. 2004. Software Effort Modeling and Forecasting using Computational Intelligent Methods. BSc Thesis, Department of Computer Science, University of Cyprus, Cyprus.

(Park and Baek, 2008) Park, H., and Baek, S. 2008. An Empirical Validation of a Neural Network Model for Software Effort Estimation. *Expert Systems with Applications* 35, 929-937.

(Park, 1988) Park, R. 1988. The Central Equations of the PRICE Software Cost Model. In *Proceedings of the 4th COCOMO User's Group Meeting*, Los Angeles, CA, USA.

(Parkinson, 1957) Parkinson, G.N. 1957. *Parkinson's Law and Other Studies in Administration*. Houghton-Miffin, Boston, MA.

(Parsa et al., 2008) Parsa, S., Vahidi-Asl, M., and Naree, S.A. 2008. Finding Causes of Software Failure Using Ridge Regression and Association Rule Generation Methods. In *Proceedings of the 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing* (*SNPD*), Phuket, Thailand, 873-878.

(Pendharkar et al., 2005) Pendharkar, P.C., Subramanian, G.H., Rodger, J.A., A Probabilistic Model for Predicting Software Development Effort. *IEEE Transactions on Software Engineering* 31 (7), 615-624.

(Poels, 1996) Poels, G. 1996. Why Function Points do not Work in Search of New Software Measurement Strategies. *Guide Share Europe Journal* 1 (2), 9-26.

(Prasad Reddy, 2010) Prasad Reddy, P.V.G.D. 2010. Particle Swarm Optimization in the Fine-Tuning of Fuzzy Software Cost Estimation Models. *International Journal of Software Engineering* (*IJSE*) 1 (2). CSC Journals, 12-23.

(Prechelt, 1994) Prechelt, L. 1994. PROBEN1 - a set of Neural Network Benchmark Problems and Benchmarking Rules.Technical Report 21/94, Faculty of Informatics, Universität Karlsruhe, 1-38.

(Pressman, 2000) Pressman, R.S. 2000. *Software Engineering: a Practitioner's Approach* (5th edition). McGraw-Hill, NY.

(Putnam and Myers, 1992) Putnam, L.H., and Myers, W. 1992. *Measures for Excellence: Reliable Software on Time, Within Budget (Yourdon Press Computing Series)*. Prentice Hall, Englewood Cliffs, NJ.

(Putnam, 1978) Putnam, L.H. 1978. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering* 4 (4), 345-361.

(Quinlan, 1986) Quinlan, J.R. 1986. Induction of Decision Trees. *Machine Learning* 1, 81-106.

(Quinlan, 1993) Quinlan, J.R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA.

(Rao et al., 2009) Rao, B.T., Sameet, B., Swathi, G.K., Gupta, K.V., Teja, C.R., and Sumana, S. 2009. A Novel Neural Network Approach for Software Cost Estimation using Functional Link Artificial Neural Network (FLANN). *International Journal of Computer Science and Network Security* 9 (6), 126-131.

(Reddy and Raju, 2009) Reddy, C.S., and Raju, K. 2009. A Concise Neural Network Model for Estimating Software Effort. *International Journal of Recent Trends in Engineering* 1 (1), 188-193.

(Refenes et al., 1995) Refenes, A.N., Kollias, C., and Zarpanis, A. 1995. External Security Determinants of Greek Military Expenditure: An Empirical Investigation Using Neural Networks. *Defence and Peace Economics* 6, 27-41.

(Rosenblatt, 1957) Rosenblatt, F. 1957, The Perceptron - a Perceiving and Recognizing Automaton. Report 85-460-1. Cornell Aeronautical Laboratory.

(Rosencrance, 2007) Rosencrance, L. 2007. Poor Communication Causes Most IT Project Failures. Inadequate Resource Planning, Unrealistic Deadlines also cited in CompTIA study. Computerworld (online). Available at: http://www.computerworld.com/s/article/9012758/Survey_Poor_communication_causes_most_IT_project_failures.

(Royce, 1970) Royce, W.W. 1970. Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the IEEE Western Electronic Show and Convention* (*WESTCON*), Los Angeles, CA, USA. Technical Papers Vol. 14, 1-9.

(Ruhe et al., 2003) Ruhe, M., Jeffery, R., and Wieczorek, I. 2003. Cost estimation for web application. In *Proceedings of 25th International Conference on Software Engineering* (*ICSE*), Portland, OR, USA, 285-294.

(Rumelhart et al., 1986) Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning Internal Representations by Error Propagation. *In Parallel Distributed Processing: Explorations in the Microstmctures of Cognition* 1. MIT Press, Cambridge, MA, 318-362.

(Rush and Roy, 2001) Rush, C., and Roy, R. 2001. Expert Judgement in Cost Estimating: Modelling the Reasoning Process. *Concurrent Engineering: Research and Applications* 9 (4), 271-284.

(Samson et al., 1997) Samson, B., Ellison, D., and Dugard, P. 1997. Software Cost Estimation using an Albus Perceptron (CMAC). *Information and Software Technology* 39, 55-60.

(Satizábal and Pérez-Uribe, 2007) Satizábal, H.M., and Pérez-Uribe, A. 2007. Relevance Metrics to Reduce Input Dimensions in Artificial Neural Networks. In *Proceedings of International Conference on Artificial Neural Networks* (*ICANN*), Porto, Portugal, 39-48, Springer Berlin/Heidelberg.

(Saunders et al., 1999) Saunders, C., Gammerman, A., and Vovk, V. 1999. Transduction with Confidence and Credibility. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (*IJCAI*), Stockholm, Sweden. Vol. 2, 722–726. Morgan Kaufmann, Los Altos, CA.

(Schach, 2004) (Schach, 2005) Schach, S.R. 2005. *Object-Oriented and Classical Software Engineering* (6th edition). McGraw-Hill Publishing Co., New York, NY.

(Selby, 2007) Selby, R.W. 2007. *Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research*. IEEE Computer Society/Wiley Partnership, John Wiley & Sons Inc., Hoboken, NJ.

(Serluca, 1995) Serluca, C. 1995. An Investigation into Software Effort Estimation using a Back-Propogation Neural Network. MSc. Thesis, Bournemouth University, United Kingdom.

(Shachter and Peot, 1992) Shachter, R.D., and Peot, M.A. 1992. Decision Making Using Probabilistic Inference Methods. In *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence* (*UAI*), Stanford, CA, USA, 276–283.

(Shachter, 1988) Shachter, R.D. 1988. Probabilistic Inference and Influence Diagrams. *Operations Research* 36, 589-605.

(Shepperd and Kadoda, 2001) Shepperd, M., and Kadoda, G. 2001. Comparing Software Prediction Techniques Using Simulation. *IEEE Transactions on Software Engineering* 27 (11), 1014-1022.

(Shepperd and Schofield, 1997) Shepperd, M., and Schofield, C. 1997. Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering* 23, 736-743.

(Shukla and Misra, 2008) Shukla, R., and Misra, A.K. 2008. Estimating Software Maintenance Effort - A Neural Network Approach. In *Proceedings of the 1st India Software Engineering Conference* (*ISEC*), Hyderabad, India. ACM Digital Library, 107-112.

(Silva and Almeida, 2003) Silva, S., and Almeida, J. 2003. Dynamic Maximum Tree Depth - a Simple Technique for Avoiding Bloat in Tree-Based GP, In *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO*), Chicago, IL, USA, 1776-1787.

(Silva and Costa, 2005) Silva, S., Costa, E. 2005. Resource-Limited Genetic Programming: The Dynamic Approach. *In Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO*), Washington, DC, USA. ACM Press, 1673-1680.

(Silva, 2007) Silva, S. 2007. GPLAB A Genetic Programming Toolbox for Matlab, Version 3, April 1007. Evolutionary and Complex Systems Group (ECOS), University of Coimbra, Portugal. Available at: http://klobouk.fsv.cvut.cz/~leps/teaching/mmo/data/gplab.manual.3.pdf

(Sommerville, 2006) Sommerville, I. 2006. *Software Engineering* (8th edition), International Computer Science. Pearson Education Limited, Essex, United Kingdom.

(Song and Shepperd, 2011) Song, Q., and Shepperd, M. 2011. Predicting Software Project Effort: A Grey Relational Analysis Based Method. *Expert Systems with Applications* 38 (6), 7302-7316.

(Song et al., 2005) Song, Q., Shepperd, M., and Mair, C. 2005. Using Grey Relational Analysis to Predict Software Effort with Small Data Sets. In *Proceedings of the 11th IEEE International Software Metrics Symposium* (*METRICS*), Como, Italy, 35-45.

(Spearman, 1904) Spearman, C. 1904. The Proof and Measurement of Association between Two Things. *American Journal of Psychology* 15, 72-101.

(Srinivasan and Fisher, 1995) Srinivasan, K., and Fisher, D. 1995. Machine Learning Approaches to Estimating Software Development Effort. *IEEE Transactions on Software Engineering* 21 (2), 126-137.

(Stutzke, 2006) Stutzke, R.D. 2006. Software Project Estimation: An Overview. In *Software Management* (7th edition), Reifer, D.J., Eds. IEEE Computer Society, 189-202. John Wiley & Sons Inc., Hoboken, NJ.

(Stylianou and Andreou, 2007) Stylianou, C., and Andreou, A.S. 2007. A Hybrid Software Component Clustering and Retrieval Scheme Using an Entropy-Based Fuzzy *k*-Modes Algorithm. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI*), Vol. 1., Washington, DC, USA. IEEE Computer Society, 202-209. DOI=10.1109/ICTAI.2007.16.

(Taber, 1987) Taber, W.R., and Siegel, M. 1987. Estimation of Expert Weights and Fuzzy Cognitive Maps. In *Proceedings of the 1st IEEE International Conference on Neural Networks*, Vol. 2, San Diego, CA, USA, 319-325.

(Tausworthe, 1981) Tausworthe, R.C. 1981. Deep Space Network Software Cost Estimation Model, Jey Propulsion Laboratory Publication 81-7, Pasadena, CA, 67-78.

(The Standish Group, 1994) The Standish Group. 1994. The Chaos Report. Standish Group Internal Report. Available at: http://www.standishgroup.com/.

(The Standish Group, 1995) The Standish Group. 1995. CHAOS Chronicles. Standish Group Internal Report. Available at: http://www.standishgroup.com/.

(The Standish Group, 2007) The Standish Group. 2007. Chaos Report. Standish Group International Internal Report. Available at: http://www.standishgroup.com/.

(Tronto et al., 2008) Tronto, I.F.D.B., Silva, J.D.S.D., and Sant'Anna, N. 2008. An Investigation of Artificial Neural Networks based Prediction Systems in Software Project Management. *Journal of Systems and Software* 81, 356-367.

(Tsadiras and Margaritis, 1996) Tsadiras, A.K., and Margaritis, K.G. 1996. Using Certainty Neurons in Fuzzy Cognitive Maps. *Neural Network World* 6, 719-728.

(Tsekouras et al., 2005) Tsekouras, G.E., Papageorgiou, D., Kotsiantis, S., Kalloniatis, C., and Pintelas, P. 2005. Fuzzy Clustering of Categorical Attributes and its Use in Analyzing Cultural Data. *International Journal of Computing Intelligence* 1 (2), 123-127, ISSN 1304-2386.

(Walston and Felix, 1977) Walston, C.E., and Felix, C.P. 1977. A Method of Programming Measurement and Estimation. *IBM Systems Journal* 16 (1), 54-73.

(Ward System Group, 2008) Ward System Group, Inc. 2008. NeuroShell 2 User's Manual. Available at: http://www.wardsystems.com/manuals/neuroshell2/index.html. Accessed: December 2011.

(Weiss and Kulikowski, 1991) Weiss, S., and Kulikowski, C.A. 1991. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine*

Learning, and Expert Systems, Machine Learning Series, Morgan Kaufmann Puclishers, Inc., San Mateo, CA.

(Werbos, 1974) Werbos, P.J. 1974. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, Cambridge, MA, USA.

(Widrow and Hoff, 1960) Widrow, B., and Hoff, Jr.M. 1960. *Adaptive Switching Circuits, IRE WESCON Convention Record at the Western Electronic Show and Convention*, Los Angeles, CA, 96-104.

(Wittig and Finnie, 1997) Wittig, G.E., and Finnie, G.R. 1997. Estimating Software Development Effort with Connectionist Models. *Information and Software Technology* 39 (7), 469-476.

(Wolverton, 1974) Wolverton, R.W. 1974. The Cost of Developing Large-Scale Software. *IEEE Transactions on Computers*, 615-636.

(Xia et al., 2008) Xia, W., Capretz, L.F., Ho, D., and Ahmed, F. 2008. A new calibration for Function Point complexity weights. *Information and Software Technology* 50, 670-683,

(Xu and Khoshgoftaar, 2004) Xu, Z., and Khoshgoftaar, T.M. 2004. *Identification of Fuzzy Models of Software Cost Estimation. Fuzzy Sets and Systems* 145 (1). Elsevier, New York, 141-163.

(Yao et al., 2000) Yao, J., Dash, M., Tan, S.T., and Liu, H. 2000. Entropy-based Fuzzy Clustering and Fuzzy Modeling. *Fuzzy Sets and Systems* 113 (3), 381-388.

(Zadeh, 1965) Zadeh, L.A. 1965. Fuzzy sets. *Information and Control* 8 (3), 338–353, ISSN 0019-9958.

(Zhang and Zhang, 2009) Zhang, B., and Zhang, R. 2009. Evaluation Model of Software Cost Estimation Methods Based on Fuzzy-Grey Theory. In *Proceeding of the International Conference on Internet Computing for Science and Engineering*, Harbin, China, 53-55.

(Zivkovic et al., 2005) Zivkovic, A., Rozman, I., and Hericko, M. 2005. Automated software size estimation based on function points using UML models. *Information and Software Technology* 47, 881-890.

Appendix A

Statistical Profile of Datasets

In this Appendix the statistical profile and the description of attributes of the datasets utilised in this thesis are presented. The datasets include the COCOMO, the Albrecht and Gaffney, the Kemerer, the Desharnais, the ISBSG R9 and the ISBSG R10.

A.1 The COCOMO Dataset

The COCOMO dataset contains 63 projects developed at TRW Inc. Aerospace. The projects range in size from 2,000 to 100,000 Lines Of Code (LOC) and the programming languages range from assembly to PL/I. These projects were based on the Waterfall model of software development which was the prevalent software development process in 1981.

Table A. 1: COCOMO dataset software cost attributes definitions

Code	Attribute name	Levels-Definition
EFF	*Development Effort*	*Development effort (in person-months) - Continuous*
LOC	Lines of Code	Project size – Continuous
RELY	Required Reliability	Continuous
DATA	Database Size	Continuous
CPLX	Product Complexity	Continuous
TIME	Execution Time Constraint	Continuous
STOR	Main Storage Constraint	Continuous
VIRT	Virtual Machine Volatility	Continuous
TURN	Computer Turnaround Time	Continuous
ACAP	Analyst Capability	Continuous
AEXP	Applications Experience	Continuous
PCAP	Programmer Capability	Continuous
VEXP	Virtual Machine Experience	Continuous
LEXP	Programming Language Experience	Continuous
MODP	Modern Programming Practices	Continuous
TOOL	Use of Software Tools	Continuous
SCED	Required Development Schedule	Continuous

Table A. 2: COCOMO dataset cost attributes descriptives

Code	Mean	Standard Error	Median	Mode	Standard Deviation	Sample Variance	Kurtosis	Skewness	Range	Min	Max
EFF	683.32	229.50	98.00	8.00	1821.58	3318162.25	21.87	4.47	11394.10	5.90	11400
LOC	77.21	21.23	25.00	23.00	168.51	28395.41	26.96	4.71	1148.02	1.98	1150
RELY	1.04	0.02	1.00	0.88	0.19	0.04	-0.54	0.60	0.65	0.75	1.40
DATA	1.00	0.01	1.00	0.94	0.07	0.01	-0.18	0.95	0.22	0.94	1.16
CPLX	1.09	0.03	1.07	1.30	0.20	0.04	-0.33	-0.08	0.95	0.70	1.65
TIME	1.11	0.02	1.06	1.00	0.16	0.03	2.89	1.77	0.66	1.00	1.66
STOR	1.14	0.02	1.06	1.00	0.18	0.03	1.13	1.47	0.56	1.00	1.56
VIRT	1.01	0.02	1.00	1.00	0.12	0.01	-0.27	0.61	0.43	0.87	1.30
TURN	0.97	0.01	1.00	1.00	0.08	0.01	-1.26	-0.09	0.28	0.87	1.15
ACAP	0.91	0.02	0.86	0.86	0.15	0.02	1.84	1.02	0.75	0.71	1.46
AEXP	0.95	0.02	1.00	1.00	0.12	0.01	1.24	0.94	0.47	0.82	1.29
PCAP	0.94	0.02	0.86	0.86	0.17	0.03	1.64	1.00	0.72	0.70	1.42
VEXP	1.01	0.01	1.00	0.90	0.09	0.01	-0.93	0.33	0.31	0.90	1.21
LEXP	1.00	0.01	1.00	1.00	0.05	0.00	0.51	1.01	0.19	0.95	1.14
MODP	1.00	0.02	1.00	0.91	0.13	0.02	-0.75	0.47	0.42	0.82	1.24
TOOL	1.02	0.01	1.00	1.00	0.09	0.01	0.49	0.56	0.41	0.83	1.24
SCED	1.05	0.01	1.00	1.00	0.08	0.01	1.65	1.69	0.23	1.00	1.23

A.2 The Albrecht and Gaffney Dataset

The Albrecht and Gaffney dataset contains information 24 projects developed by the IBM DP service organisation.

Table A. 3: Albrecht and Gaffney dataset software cost attributes definitions

Code	Attribute name	Levels-Definition
EFF	*Development Effort*	*Development effort (in K-hours) - Continuous*
FP	Function Points	Project functional size - Continuous
SLOC	Thousand Lines of Code	Project size - Continuous
LAN	Language	1=COBOL 2=PL/I 3= DMS

Table A. 4: Albrecht and Gaffney dataset cost attributes descriptives

Code	EFF	FP	SLOC
Mean	21.88	647.63	61.08
Standard Error	5.80	99.61	13.00
Median	11.45	506.00	41.00
Mode	n/a	512.00	24.00
Standard Deviation	28.42	488.00	63.68
Sample Variance	807.58	238139.38	4055.56
Kurtosis	4.67	1.56	11.73
Skewness	2.30	1.54	3.09
Range	104.70	1703.00	315.00
Min	0.50	199.00	3.00
Max	105.20	1902.00	318.00
Sum	525.00	15543.00	1466.00
Count	24.00	24.00	24.00

A.3 The Kemerer Dataset

The Kemerer dataset contains 15 projects from an organisation in the USA mainly written in COBOL.

Table A. 5: Kemerer dataset software cost attributes definitions

Code	Attribute name	Levels-Definition
EFF	*Development Effort*	*Development effort - Continuous*
FP	Function Points (unadjusted)	Project functional size - Continuous
AFP	Adjusted Function Points	Adjusted project functional size - Continuous
KSLOC	Thousand Source Lines of Code	Project size - Continuous
DU	Duration	Project duration (months) - Continuous

Table A. 6: Kemerer dataset cost attributes descriptives

Code	EFF	FP	AFP	KSLOC	DU
Mean	219.25	993.87	999.14	186.57	14.27
Standard Error	67.92	154.25	152.23	35.33	1.95
Median	130.30	976.00	993.00	164.80	14.00
Mode	n/a	n/a	n/a	n/a	5.00
Standard Deviation	263.06	597.43	589.59	136.82	7.54
Sample Variance	69198.16	356917.98	347618.82	18719.01	56.92
Kurtosis	10.59	-0.05	0.22	-0.09	0.43
Skewness	3.07	0.49	0.50	0.78	0.72
Range	1084.11	2187.00	2206.90	411.00	26.00
Min	23.20	97.00	99.90	39.00	5.00
Max	1107.31	2284.00	2306.80	450.00	31.00
Sum	3288.71	14908.00	14987.10	2798.60	214.00
Count	15.00	15.00	15.00	15.00	15.00

A.4 The Desharnais Dataset

The Desharnais dataset includes observations for 81 systems developed by a Canadian software development house.

Table A. 7: Desharnais dataset software cost attributes definitions

Code	Attribute name	Levels-Definition
EFF	*Development Effort*	*Development effort (in person-hours) - Continuous*
TE	Team Experience (years)	Team's experience (in years) - Continuous
ME	Manager Experience (years)	Project Manager's experience (in years) - Continuous
DU	Duration (months)	Project duration (in months) - Continuous
TR	Transactions	Continuous
EN	Entities	Continuous
FPA	Function Points Adjusted	Continuous
SC	Scale of project	Continuous
FPNA	Function Points Non-Adjusted	Continuous
LAN	Language	1=Environment 1 2=Environment 2 3=Environment 3

Table A. 8: Desharnais dataset cost attributes descriptives

Code	EFF	TE	ME	DU	TR	EN	FPA	SC	FPNA
Mean	5046.31	2.27	2.67	11.72	179.90	122.33	302.23	27.63	287.05
Standard Error	490.97	0.15	0.17	0.82	15.92	9.43	19.96	1.18	20.57
Median	3647.00	2.00	3.00	10.00	139.00	99.00	260.00	28.00	253.00
Mode	n/a	4.00	4.00	12.00	97.00	52.00	100.00	34.00	192.00
Standard Deviation	4418.77	1.34	1.52	7.40	143.31	84.88	179.68	10.59	185.11
Sample Variance	19525503.82	1.79	2.30	54.76	20539.17	7204.98	32283.76	112.19	34265.02
Kurtosis	4.72	-1.26	0.07	3.12	7.73	1.48	4.94	-0.28	4.16
Skewness	2.01	-0.04	0.20	1.60	2.36	1.34	1.78	-0.11	1.66
Range	23394.00	4.00	7.00	38.00	877.00	380.00	1054.00	47.00	1054.00
Min	546.00	0.00	0.00	1.00	9.00	7.00	73.00	5.00	62.00
Max	23940.00	4.00	7.00	39.00	886.00	387.00	1127.00	52.00	1116.00
Sum	408751.00	179.00	208.00	949.00	14572.00	9909.00	24481.00	2238.00	23251.00
Count	81.00	79.00	78.00	81.00	81.00	81.00	81.00	81.00	81.00

A.5 The ISBSG R9 Dataset

The filtered versions of the ISBSG R9 dataset utilised throughout the experiments of this thesis are summarised in this section. Initially, in Table A. 9 the summary statistics for the ISBSG R9-1 and the only two attributes used in the SB-SCE experiments of section 4.2.1 are described (i.e., Summary Work Effort (SWE) and Adjusted Function Points (AFP)).

Table A. 9: ISBSG R9-1 dataset descriptive of attributes SWE and AFP

Mean	Standard Error	Median	Mode	Standard Deviation	Sample Variance	Kurtosis	Skewness	Range	Min	Max
4762.75	272.82	1740.00	384.00	8457.54	71530049.00	19.86	3.98	73912.00	8.00	73920.00
478.05	33.94	196.00	101.00	1052.05	1106804.56	131.29	9.58	17515.00	3.00	17518.00

Table A. 10: ISBSG R9-2 dataset cost attributes descriptives

Code	Mean	Standard Error	Median	Mode	Standard Deviation	Sample Variance	Kurtosis	Skewness	Min	Max
SWE	4162.94	264.96	2047.00	0.00	7198.02	51811470.57	168.31	9.93	0	138883.00
FS	447.93	20.78	249.00	119.00	564.51	318675.40	10.26	2.79	4.00	4326.00
AFP	455.26	21.45	252.50	128.00	582.78	339627.11	12.26	2.96	4.00	4932.00
PDRA	14.74	0.90	7.90	3.30	24.53	601.84	90.47	7.66	0	387.10
PDRU	15.73	1.24	8.00	3.30	33.76	1139.78	182.68	11.51	0	640.00
NPDRA	15.96	1.00	8.30	0.00	27.10	734.21	82.07	7.44	0	387.10
NPDRU	16.95	1.31	8.35	0.00	35.65	1271.28	152.60	10.50	0	640.00
PET	6.26	0.21	4.70	3.00	5.76	33.13	49.63	4.91	0	84.00
PIT	0.24	0.06	0.00	0.00	1.76	3.11	431.63	18.91	0	42.00
RL	1.34	0.03	1.00	1.00	0.84	0.71	4.93	2.51	1	4.00
INC	104.25	6.66	44.00	0.00	180.90	32724.80	24.79	4.14	0	1935.00
OC	104.16	5.56	48.50	0.00	151.04	22812.43	13.26	3.11	0	1337.00
EC	78.25	4.89	28.00	0.00	132.97	17681.53	19.08	3.68	0	1306.00
FC	109.14	5.77	49.50	0.00	156.69	24552.90	10.22	2.82	0	1252.00
IFC	52.44	4.32	17.00	0.00	117.44	13792.53	33.17	5.18	0	1097.00
AC	341.17	19.81	148.50	0.00	538.06	289506.28	12.73	3.10	0	4326.00
CC	103.57	10.89	0.00	0.00	295.91	87565.58	46.42	5.88	0	3622.00
DC	3.49	1.35	0.00	0.00	36.68	1345.20	325.15	17.04	0	780.00

Table A. 11: ISBSG R9-3 dataset cost attributes descriptives

Code	Mean	Standard Error	Median	Mode	Standard Deviation	Sample Variance	Kurtosis	Skewness	Range	Min	Max
SWE	4674.91	626.47	1974.00	1238.00	6659.42	44347932.67	8.43	2.75	35906.00	140	3155
FS	436.27	49.18	252.00	244.00	522.80	273320.38	7.72	2.59	3113.00	42	3471
AFP	455.27	52.57	264.00	56.00	558.85	312316.79	8.45	2.65	3432.00	39	84
PET	9.50	0.84	8.00	7.00	8.89	79.03	44.01	5.53	83.00	1	42
PIT	1.34	0.41	0.00	0.00	4.31	18.55	72.25	7.86	42.00	0	4
RL	2.00	0.13	1.00	1.00	1.34	1.79	-1.36	0.73	3.00	1	65
MTS	6.05	0.73	4.00	0.00	7.75	60.06	29.39	4.32	65.00	0	1327
INC	141.70	20.63	66.00	21.00	219.33	48106.82	9.54	2.90	1327.00	0	620
OC	87.73	9.88	57.00	4.00	105.03	11031.36	9.20	2.73	620.00	0	534
EC	63.13	8.34	30.00	0.00	88.70	7867.35	11.10	2.96	534.00	0	995
FC	111.08	15.47	56.00	7.00	164.46	27047.18	11.42	3.11	995.00	0	329
IFC	32.62	5.79	10.00	0.00	61.60	3794.36	10.71	3.19	329.00	0	3155
AC	395.28	49.21	213.00	0.00	523.12	273654.37	8.18	2.63	3155.00	0	844
CC	39.41	10.33	0.00	0.00	109.80	12056.42	26.96	4.54	844.00	0	128
DC	1.58	1.17	0.00	0.00	12.41	153.89	98.80	9.74	128.00	0	36046

Table A. 12: ISBSG R9-4 dataset software cost attributes definitions

Abbreviation	Attribute name	Levels-Definition
CA1-4	Count Approach	1=IFPUG 2=Mark II 3=Feature Points 4=NESMA
AFP	Adjusted Function Points	Continuous
PET	Project Elapsed Time	Continuous
IY	Implementation Year	Nominal
DTY1-4	Development Type	1=New Development 2=Re-development 3=Enhancement 4=New Utility
OT1-12	Organization Type	1=Aerospace / Automotive 2=Banking 3=Communications 4=Electricity, Gas, Water 5=Financial, Property & Business Services 6=Government 7=Insurance 8=Manufacturing 9=Public Administration 10=Transport & Storage 11=Wholesale & Retail Trade 12=OTHER
DT1-15	Development Technique	1=Data Modelling 2=Prototyping 3=Process Modelling 4=Joint Application Development 5=Multifunctional Teams 6=Object Oriented Design 7=Rapid Application Development 8=Regression Testing 9=Event Modelling 10=Business Area Modelling 11=Object Oriented Analysis 12=Timeboxing 13=Standards 14=Waterfall 15=OTHER

Abbreviation	Attribute name	Levels-Definition
FST1-3	Functional Sizing Technique	1=Using a tool 2=Manually 3=UNKNOWN
DP1-5	Development Platform	1=Multi 2=MF 3=PC 4=MR 5=UNKNOWN
LT1-6	Language Type	1=2GL 2=3GL 3=4GL 4=ApG 5=5GL 6=UNKNOWN
PPL1-11	Primary Programming Language	1=ACCESS 2=C/C++/C# 3=PL/I 4=SQL 5=TELON 6=VB 7=WEBDEV 8=COBOL 9=JAVA 10=NATURAL 11=OTHER
DBS1-9	Database System	1=DB2 2=IMS 3=ORACLE 4=ADABAS 5=VSAM 6=SYBASE 7=ACCESS 8=SQL SERVER 9=OTHER
RM1-7	Recording Method	1=Recorded Productivity 2=Total hours Weekly/Daily 3=Recorderd Staff Hours 4=Derived Staff Hours 5=Task 6=Hours Weekly/Daily 7=Combinations
RL	Resource Level	Ordinal
MTS	Max Team Size	Continuous
ATS	Average Team Size	Continuous

Table A. 13: ISBSG R9-5 dataset cost attributes descriptives

Code	FCWEFF	AFP	EC	FC	AC	CC
Mean	5110.70	628.08	90.58	136.61	478.05	117.43
Standard Error	428.00	60.85	10.25	13.00	48.02	20.92
Median	2149.00	289.50	28.00	60.00	199.50	0.00
Mode	1950.00	44.00	0.00	0.00	0.00	0.00
Standard Deviation	8408.84	1195.53	201.43	255.38	943.41	411.04
Sample Variance	70708539.81	1429289.49	40572.19	65217.27	890025.59	168956.39
Kurtosis	18.11	106.75	100.46	44.78	97.80	78.64
Skewness	3.79	8.47	8.24	5.46	7.87	7.88
Range	66538.00	17511.00	2886.00	2955.00	13580.00	5193.00
Min	62.00	7.00	0.00	0.00	0.00	0.00
Max	66600.00	17518.00	2886.00	2955.00	13580.00	5193.00
Sum	1972729.00	242440.00	34962.00	52730.00	184529.00	45329.00
Count	386.00	386.00	386.00	386.00	386.00	386.00

Table A. 14: ISBSG R9-6 dataset cost attributes descriptives

Code	FCWEFF	NAFP	EC	FC	AC
Mean	4988.53	12.99	101.79	133.64	487.00
Standard Error	398.17	0.76	6.59	7.75	28.45
Median	2647.50	7.95	47.00	71.00	267.50
Mode	0.00	1.40	6.00	14.00	129.00
Standard Deviation	8521.23	16.17	141.11	165.94	608.83
Sample Variance	72611368.75	261.35	19910.98	27537.60	370670.37
Kurtosis	134.43	28.37	9.73	8.96	9.22
Skewness	9.28	4.33	2.85	2.62	2.66
Range	138883.00	159.40	919.00	1245.00	4322.00
Min	0.00	0.10	3.00	7.00	4.00
Max	138883.00	159.50	922.00	1252.00	4326.00
Sum	2284746.00	5947.30	46620.00	61209.00	223046.00
Count	458.00	458.00	458.00	458.00	458.00

Table A. 15: ISBSG R9-7 dataset cost attributes descriptives

Code	FCWEFF	AFP	PDRU	PET	RL	ATS
Mean	6756.81	676.51	13.41	9.50	1.41	7.63
Standard Error	630.20	67.67	1.19	0.39	0.05	0.54
Median	2350.00	344.00	7.40	7.30	1.00	4.50
Mode	1788.00	118.00	3.80	6.00	1.00	3.00
Standard Deviation	11500.13	1234.83	21.74	7.15	0.99	9.80
Sample Variance	132252962.63	1524812.98	472.73	51.19	0.98	95.94
Kurtosis	11.67	83.06	94.33	7.52	2.79	18.05
Skewness	3.25	7.70	8.05	2.25	2.15	3.80
Range	73750.00	16131.00	300.00	51.00	3.00	76.05
Min	170.00	17.00	0.30	1.00	1.00	0.95
Max	73920.00	16148.00	300.30	52.00	4.00	77.00
Sum	2250018.00	225279.00	4467.10	3163.80	468.00	2541.50
Count	333.00	333.00	333.00	333.00	333.00	333.00

Table A. 16: ISBSG R9-8 dataset software cost attributes definitions

Abbreviation	Attribute name	Levels-Definition
CA	Count Approach	Nominal
AFP	Adjusted Function Points	Ordinal
PET	Project Elapsed Time	Ordinal
IY	Implementation Year	Nominal
DT	Development Type	Nominal
OT1-12	Organization Type	1=Aerospace / Automotive 2=Banking 3=Communications 4=Electricity, Gas, Water 5=Financial, Property & Business Services 6=Government 7=Insurance 8=Manufacturing 9=Public Administration 10=Transport & Storage 11=Wholesale & Retail Trade 12=OTHER

Abbreviation	Attribute name	Levels-Definition
DT1-15	Development Technique	1=Data Modelling 2=Prototyping 3=Process Modelling 4=Joint Application Development 5=Multifunctional Teams 6=Object Oriented Design 7=Rapid Application Development 8=Regression Testing 9=Event Modelling 10=Business Area Modelling 11=Object Oriented Analysis 12=Timeboxing 13=Standards 14=Waterfall 15=OTHER
FST	Functional Sizing Technique	Nominal
DP	Development Platform	Nominal
LT	Language Type	Nominal
PPL	Primary Programming Language	Nominal
DBS1-9	Database System	1=DB2 2=IMS 3=ORACLE 4=ADABAS 5=VSAM 6=SYBASE 7=ACCESS 8=SQL SERVER 9=OTHER
RM	Recording Method	Nominal
RL	Resource Level	Ordinal
MTS	Max Team Size	Ordinal
ATS	Average Team Size	Ordinal

Table A. 17: ISBSG R9-8.5 Dataset Summary Work Effort (SWE) descriptives (outlier-free)

Mean	Standard Error	Median	Mode	Standard Deviation	Sample Variance	Kurtosis	Skewness	Range	Min	Max
4762.75	3107.40	160.19	1935.00	1238.00	3093.74	9571232.23	1.74	1.50	14113.00	97.00

Table A. 18: ISBSG R9-9 dataset software cost attributes definitions

Abbreviation	Attribute name	Levels-Definition (original/after fuzzification)
EFF	*Full Cycle Work Effort*	*Continuous/Ordinal*
PET	Project Elapsed Time	Continuous/Ordinal
PIT	Project Inactive Time	Continuous/Ordinal
PDRU	Project PDR (ufp)	Continuous/Ordinal
AFP	Adjusted Function Points	Continuous/Ordinal
ATS	Average Team Size	Continuous/Ordinal
DT	Development Type	1=New Development 2=Re-development 3=Enhancement 4=New Utility
AT	Application Type	Nominal
DP	Development Platform	1=Multi 2=MF 3=PC 4=MR 5=UNKNOWN

Abbreviation	Attribute name	Levels-Definition (original/after fuzzification)
LT	Language Type	1=2GL 2=3GL 3=4GL 4=ApG 5=5GL 6=UNKNOWN
RL	Resource Level	Ordinal
PPL	Primary Programming Language	1=ACCESS 2=C/C++/C# 3=PL/I 4=SQL 5=TELON 6=VB 7=WEBDEV 8=COBOL 9=JAVA 10=NATURAL 11=OTHER

A.6 The ISBSG R10 Dataset

The ISBSG R10 dataset utilised in the experiments contained 65 projects for which the breakdown of effort for the Planning, Specify, Design, Building and Implementation phases was available.

Table A. 19: ISBSG r10 dataset software cost attributes definitions

Code	Attribute name	Levels-Definition
SWE	*Summary Work Effort*	*Total actual effort (in person-hours) - Continuous*
PLAN	Plan effort	Continuous
SPEC	Specify effort	Continuous
DESN	Design effort	Continuous
BLD	Build effort	Continuous
TEST	Test effort	Continuous
IMPL	Implement effort	Continuous
DT	Development Type	1=New Developement 2=Re-development 3=Enhancement
DP	Development Platform	1=Multi 2=MF 3=PC 4=MR
LT	Language Type	1=2GL 2=3GL 3=4GL 4=ApG
MTS	Max Team Size	Continuous

Table A. 20: ISBSSG R10 dataset effort breakdown in phases descriptives

Code	SWE	PLAN	SPEC	DESN	BLD	TEST	IMPL
Mean	6999.71	575.63	556.23	831.77	2577.68	1085.55	390.38
Standard Error	1048.22	180.67	127.36	127.11	581.32	169.28	97.25
Median	4294.00	158.00	250.00	400.00	1300.00	626.00	168.00
Mode	3138.00	30.00	138.00	120.00	420.00	228.00	50.00
Standard Deviation	8451.03	1456.60	1026.80	1024.79	4686.74	1364.81	784.02
Sample Variance	71419927.65	2121697.36	1054308.18	1050194.96	21965510.38	1862711.50	614680.62
Kurtosis	10.07	39.18	23.30	8.19	22.54	8.26	26.89
Skewness	2.90	5.84	4.60	2.50	4.51	2.71	4.74
Range	47080.00	10796.00	6496.00	5790.00	29930.00	6926.00	5391.00
Minimum	172.00	4.00	4.00	17.00	70.00	40.00	2.00
Maximum	47252.00	10800.00	6500.00	5807.00	30000.00	6966.00	5393.00
Sum	454981.00	37416.00	36155.00	54065.00	167549.00	70561.00	25375.00
Count	65.00	65.00	65.00	65.00	65.00	65.00	65.00

Appendix B

Complete Experimental Results

This Appendix includes the complete experimental results obtained from the SCE models created and described in Chapter 4. The results report the prediction errors obtained during the training and testing phases. The training phase conducted is goal-oriented, i.e., it calibrates the particular model or technique employed. Effective training will ultimately reduce some error or satisfy some pre-defined requirements. Thus, the target of the training process is to find the set of appropriate settings (conditions) that yield successful results, in terms of either prediction, generalisation or other specific requirements set. This evaluation process, also referred to as testing phase, assesses the ability of the model or technique to match the actual target values or requirements as closely as possible. The best models obtained appear in bold.

B.1 Complete Results of MLP ANN for SB-SCE

The results summarised below refer to the single hidden layer MLP ANN created during repetition of the experiments described in section 4.2.1.1 (pg. 103) for Size-Based SCE.

Table B. 1: Results from Single hidden layer MLP ANN for SB-SCE for the COCOMO dataset

INPUT	ANN TOPOLOGY	TRAINING PHASE				TESTING PHASE			
		MMRE	*CC*	*NRMSE*	*Pred(.25)*	*MMRE*	*CC*	*NRMSE*	*Pred(.25)*
SLOC	1-2-1	3.340	0.663	0.736	0.100	3.701	0.987	0.448	0.077
	1-3-1	1.026	0.837	0.538	0.200	1.664	0.711	0.750	0.154
	1-4-1	1.196	0.677	0.730	0.067	1.910	0.645	0.742	0.154
	1-2-1	1.601	0.831	0.573	0.167	1.257	0.796	0.966	0.308
	1-3-1	1.334	0.670	0.730	0.167	1.449	0.933	1.227	0.077
	1-4-1	**0.888**	**0.998**	**0.063**	**0.333**	**1.629**	**0.597**	**2.890**	**0.154**
	1-2-1	1.772	0.821	0.561	0.233	1.010	0.765	0.637	0.077
	1-3-1	1.554	0.865	0.512	0.267	2.480	0.501	1.488	0.154
	1-4-1	1.647	0.934	0.584	0.167	1.029	0.995	0.327	0.154

Table B. 2: Experimental results from Single hidden layer MLP ANN for SB-SCE for the Kemerer dataset

INPUT	ANN TOPOLOGY	TRAINING PHASE				TESTING PHASE			
		MMRE	*CC*	*NRMSE*	*Pred(.25)*	*MMRE*	*CC*	*NRMSE*	*Pred(.25)*
AFP	1-2-1	0.286	0.988	0.145	0.625	0.380	0.574	0.822	0.000
	1-3-1	**0.204**	**0.966**	**0.248**	**0.625**	**0.282**	**0.943**	**0.614**	**0.333**
	1-4-1	0.074	0.976	0.211	1.000	0.652	-0.995	1.167	0.000
	1-2-1	0.253	0.977	0.231	0.500	0.976	0.952	0.665	0.333
	1-3-1	0.187	0.986	0.158	0.750	0.604	0.998	0.873	0.667
	1-4-1	0.071	0.997	0.084	0.875	0.318	0.626	0.910	0.667
	1-2-1	0.235	0.892	0.474	0.625	0.826	0.798	0.681	0.000
	1-3-1	0.228	0.850	0.575	0.750	0.386	0.995	1.357	0.333
	1-4-1	0.003	1.000	0.002	1.000	1.726	0.033	4.836	0.000
SLOC	1-2-1	0.183	0.993	0.111	0.625	0.503	0.026	0.917	0.333
	1-3-1	**0.258**	**0.861**	**0.476**	**0.750**	**0.257**	**0.792**	**0.527**	**0.333**
	1-4-1	0.067	0.967	0.238	0.875	0.272	0.936	0.828	0.667
	1-2-1	0.222	0.990	0.133	0.500	0.931	0.920	0.527	0.333
	1-3-1	0.193	0.980	0.483	0.750	0.449	0.455	1.040	0.000
	1-4-1	0.258	0.987	0.411	0.250	0.674	-0.327	1.105	0.333
	1-2-1	0.134	0.976	0.251	0.750	0.907	0.603	0.762	0.333
	1-3-1	0.377	0.983	0.174	0.625	0.304	0.763	0.535	0.667
	1-4-1	0.235	0.809	0.552	0.625	1.221	0.698	4.288	0.000

Table B. 3: Experimental results from Single hidden layer MLP ANN for SB-SCE for the Albrecht and Gaffney dataset

INPUT	TOPO-LOGY	TRAINING PHASE				TESTING PHASE			
		MMRE	*CC*	*NRMSE*	*Pred(.25)*	*MMRE*	*CC*	*NRMSE*	*Pred(.25)*
FP	1-2-1	0.141	0.995	0.108	0.818	0.713	0.300	0.885	0.200
	1-3-1	0.264	0.990	0.141	0.636	0.501	0.889	0.787	0.400
	1-4-1	0.165	0.998	0.069	0.636	0.561	0.999	0.543	0.000
	1-2-1	**0.304**	**0.978**	**0.213**	**0.545**	**0.324**	**0.987**	**0.149**	**0.600**
	1-3-1	0.303	0.980	0.192	0.455	0.449	0.892	0.474	0.200
	1-4-1	0.240	0.994	0.133	0.636	2.804	0.923	0.702	0.600
	1-2-1	0.253	0.976	0.209	0.545	1.408	0.998	5.356	0.400
	1-3-1	0.183	0.991	0.125	0.545	0.603	0.684	0.850	0.200
	1-4-1	0.135	0.998	0.061	0.909	3.646	0.830	0.785	0.400
SLOC	1-2-1	0.228	0.989	0.139	0.545	1.466	0.995	0.825	0.200
	1-3-1	0.378	0.996	0.090	0.636	0.463	0.840	0.524	0.200
	1-4-1	**0.301**	**0.991**	**0.130**	**0.364**	**0.469**	**0.985**	**0.691**	**0.200**
	1-2-1	0.318	0.976	0.218	0.455	1.051	0.876	0.617	0.000
	1-3-1	0.279	0.988	0.150	0.636	0.746	0.732	0.844	0.000
	1-4-1	0.327	0.967	0.278	0.455	1.534	0.854	0.497	0.600
	1-2-1	0.453	0.910	0.395	0.545	1.051	0.998	2.885	0.400
	1-3-1	0.351	0.965	0.252	0.727	0.704	0.882	2.410	0.200
	1-4-1	0.363	0.975	0.222	0.364	2.961	0.773	0.801	0.200

Table B. 4: Experimental results from Single hidden layer MLP ANN for SB-SCE for the Desharnais dataset

INPUT	TOPO-LOGY	TRAINING PHASE				TESTING PHASE			
		MMRE	*CC*	*NRMSE*	*Pred(.25)*	*MMRE*	*CC*	*NRMSE*	*Pred(.25)*
AFP	1-2-1	0.609	0.680	0.744	0.447	1.155	0.640	0.769	0.267
	1-3-1	0.471	0.713	0.722	0.474	0.853	0.377	0.906	0.600
	1-4-1	0.555	0.618	0.785	0.316	0.459	0.733	0.681	0.533
	1-2-1	0.483	0.672	0.730	0.474	0.648	-0.509	1.273	0.333
	1-3-1	**0.487**	**0.697**	**0.715**	**0.474**	**0.348**	**0.712**	**0.696**	**0.400**
	1-4-1	0.502	0.793	0.627	0.526	0.350	0.231	1.026	0.467
	1-2-1	0.539	0.803	0.696	0.395	0.624	0.523	1.214	0.267
	1-3-1	0.453	0.769	0.647	0.474	0.496	-0.407	1.374	0.400
	1-4-1	0.515	0.691	0.718	0.421	0.544	-0.364	1.226	0.400

The results summarised below refer to the Regression models created for comparison purposes of the single hidden layer MLP ANN experiments described above for Size-Based SCE.

Table B. 5: Experimental results from Regression for SB-SCE for the COCOMO dataset

INPUT	TRAINING PHASE				TESTING PHASE			
	MMRE	*CC*	*NRMSE*	*Pred(.25)*	*MMRE*	*CC*	*NRMSE*	*Pred(.25)*
SLOC	1.078	0.573	0.858	0.167	0.778	0.973	0.273	0.308
	0.832	0.821	0.619	0.233	1.273	0.961	0.268	0.231
	0.802	0.647	0.759	0.200	2.010	0.628	0.767	0.154
	0.978	0.807	0.622	0.200	1.107	0.785	1.194	0.231
	0.932	0.646	0.802	0.133	1.212	0.901	0.690	0.231
	0.608	**0.834**	**0.579**	**0.233**	**1.022**	**0.639**	**1.045**	**0.154**
	0.991	0.802	0.609	0.133	0.578	0.734	0.746	0.462
	0.887	0.989	0.238	0.267	0.863	0.336	1.032	0.077
	0.884	0.627	0.776	0.200	0.562	0.982	0.741	0.231

Table B. 6: Experimental results from Regression for SB-SCE for the Kemerer dataset

INPUT	TRAINING PHASE				TESTING PHASE			
	MMRE	*CC*	*NRMSE*	*Pred(.25)*	*MMRE*	*CC*	*NRMSE*	*Pred(.25)*
AFP	0.480	0.943	0.539	0.375	0.541	0.573	0.765	0.000
	0.424	**0.764**	**0.663**	**0.500**	**0.196**	**0.974**	**0.289**	**0.667**
	0.411	0.790	0.645	0.375	0.457	0.973	0.864	0.333
	0.348	0.935	0.506	0.375	0.125	1.000	0.059	0.667
	0.461	0.824	0.661	0.500	0.522	0.967	0.827	0.333
	0.408	0.795	0.673	0.750	0.517	0.626	0.827	0.000
	0.244	0.887	0.457	0.750	0.919	0.805	0.654	0.000
	0.240	0.882	0.446	0.750	0.474	0.973	0.789	0.000
	0.212	0.965	0.405	0.625	1.784	0.052	5.086	0.000
SLOC	0.419	0.772	0.747	0.500	0.688	0.147	1.030	0.333
	0.330	**0.776**	**0.627**	**0.500**	**0.234**	**0.825**	**0.566**	**0.667**
	0.337	0.706	0.686	0.375	0.489	0.931	0.933	0.000
	0.417	0.802	0.680	0.625	0.778	0.994	0.289	0.333
	0.441	0.802	0.689	0.375	0.739	-0.012	1.053	0.667
	0.344	0.894	0.593	0.500	0.877	-0.064	1.281	0.667
	0.178	0.938	0.335	0.750	0.630	0.767	0.781	0.333
	0.485	0.878	0.636	0.375	0.446	0.876	0.786	0.667
	0.397	0.679	0.744	0.625	1.055	0.664	3.730	0.000

Table B. 7: Experimental results from Regression for SB-SCE for the Albrecht and Gaffney dataset

INPUT	TRAINING PHASE				TESTING PHASE			
	MMRE	***CC***	***NRMSE***	***Pred(.25)***	***MMRE***	***CC***	***NRMSE***	***Pred(.25)***
FP	0.164	0.959	0.336	0.818	0.828	0.249	0.914	0.200
	0.276	0.969	0.258	0.636	0.392	0.994	0.612	0.200
	0.283	0.981	0.359	0.364	0.415	0.987	0.244	0.400
	0.354	**0.930**	**0.472**	**0.364**	**0.248**	**0.969**	**0.453**	**0.600**
	0.326	0.961	0.329	0.455	0.285	0.992	0.223	0.400
	0.351	0.948	0.458	0.364	1.748	0.920	0.752	0.400
	0.318	0.930	0.452	0.545	0.343	0.998	0.340	0.200
	0.357	0.955	0.330	0.364	0.388	0.951	0.931	0.200
	0.186	0.978	0.256	0.818	1.117	0.880	0.590	0.200
SLOC	0.273	0.868	0.478	0.545	0.847	0.962	0.449	0.200
	0.384	0.848	0.600	0.455	0.472	0.802	0.571	0.200
	0.408	**0.837**	**0.530**	**0.364**	**0.397**	**0.970**	**0.556**	**0.200**
	0.245	0.982	0.193	0.545	0.887	0.869	0.741	0.000
	0.344	0.981	0.250	0.455	0.457	0.967	0.493	0.400
	0.379	0.885	0.498	0.545	0.538	0.909	1.236	0.200
	0.411	0.859	0.556	0.455	0.354	0.999	0.239	0.600
	0.421	0.951	0.489	0.455	0.405	0.860	1.056	0.200
	0.564	0.962	0.355	0.182	0.411	0.958	0.481	0.400

Table B. 8: Experimental results from Regression for SB-SCE for the Desharnais dataset

INPUT	TRAINING PHASE				TESTING PHASE			
	MMRE	***CC***	***NRMSE***	***Pred(.25)***	***MMRE***	***CC***	***NRMSE***	***Pred(.25)***
AFP	0.393	0.718	0.701	0.447	0.759	0.691	0.737	0.467
	0.478	0.714	0.714	0.474	0.871	0.379	0.908	0.600
	0.561	0.491	0.883	0.447	0.463	0.620	0.787	0.333
	0.481	0.555	0.837	0.395	0.501	0.905	0.695	0.400
	0.514	**0.543**	**0.848**	**0.395**	**0.311**	**0.708**	**0.819**	**0.467**
	0.552	0.672	0.752	0.500	0.351	0.347	0.951	0.467
	0.519	0.866	0.508	0.342	0.676	0.495	1.576	0.267
	0.525	0.626	0.790	0.500	0.391	0.902	0.659	0.333
	0.533	0.532	0.856	0.395	0.461	0.870	0.668	0.467

The results summarised below refer to the hybrid model developed for Size-Based SCE combining ANN with GA to optimise the neural network structure using various Input Output Method (IOM) as described in section 4.2.1.3 (pg. 113).

The *Sign Predictor* (*Sign(p)*) metric appearing in Table B. 9 assesses if there is a positive or a negative transition of the actual and predicted effort trace in the projects used only during the evaluation of the models with the sliding-window technique on unknown test data. The practical usage of this measure is that it does not take into account the exact prediction values obtained but considers the degree of correct prediction tendency, i.e., if the real value compared to the next value and the predicted value compared to the next predicted value have the same tendency (upwards or downwards). This is expressed in eqs (B.1) and (B.2).

$$Sign(p) = \frac{\sum_{i=1}^{n} z_i}{n} \tag{B.1}$$

$$where\ z_i = \begin{cases} 1\ if\ ((x_{t+1}^{pred} - x_t^{pred}) * (x_{t+1}^{act} - x_t^{act})) > 0 \\ 0 \qquad\qquad otherwise. \end{cases} \tag{B.2}$$

Table B. 9: Experimental results obtained from the hybrid genetically evolved multiple hidden layer MLP ANN SB-SCE (hybrid ANN&GA) for a constant sliding-window size

DATASET	METHOD*	ANN ARCHITECTURE	TRAINING PHASE			TESTING PHASE				
			MMRE	*CC*	*NRMSE*	*MMRE*	*CC*	*NRMSE*	*Sign(p)*	*Sign(p)%*
COCOMO	**IOM1**	**1-9-17-10-1**	**0.004**	**1.000**	**0.014**	**0.003**	**1.000**	**0.014**	**24/24**	**100**
	IOM3	2-20-18-3-1	0.092	0.963	0.270	0.075	0.961	0.278	13/18	72.22
	IOM5	3-19-20-4-1	0.043	0.990	0.149	0.044	0.981	0.199	14/24	58.33
Kemerer	**IOM1**	**1-17-13-16-1**	**0.008**	**1.000**	**0.015**	**0.009**	**1.000**	**0.019**	**4/4**	**100**
	IOM3	2-18-14-18-1	0.246	0.825	0.539	0.211	0.822	0.550	1/3	33.33
	IOM5	3-19-15-20-1	0.004	1.000	0.006	0.028	0.997	0.081	3/3	100
	IOM2	1-17-20-11-1	0.006	1.000	0.005	0.009	1.000	0.006	4/4	100
	IOM4	2-19-1520-1	0.041	0.998	0.074	0.062	0.993	0.122	3/3	100
	IOM6	3-19-9-16-1	0.029	0.999	0.045	0.031	0.999	0.051	3/3	100
Albrecht and Gaffney	IOM1	1-13-20-6-1	0.002	1.000	0.008	0.005	1.000	0.024	6/6	100
	IOM3	2-19-20-8-1	0.087	0.990	0.136	0.163	0.977	0.210	5/6	83.33
	IOM5	3-19-11-10-1	0.089	0.990	0.139	0.173	0.975	0.218	3/6	50.00
	IOM2	**1-9-17-10-1**	**0.011**	**1.000**	**0.018**	**0.014**	**1.000**	**0.018**	**6/6**	**100**
	IOM4	2-18-15-11-1	0.092	0.989	0.145	0.112	0.985	0.171	5/6	83.33
	IOM6	3-20-19-10-1	0.088	0.983	0.179	0.084	0.984	0.177	5/6	83.33
Desharnais	**IOM2**	**1-3-18-20-1**	**0.013**	**0.999**	**0.038**	**0.016**	**0.998**	**0.075**	**22/22**	**100**
	IOM4	2-20-19-20-1	0.912	0.495	1.107	0.589	0.437	1.022	13/22	59.09
	IOM6	3-20-20-19-1	0.354	0.878	0.480	0.381	0.674	0.750	21/22	95.45
ISBSG R9-1	**IOM2**	**1-16-18-11-1**	**0.004**	**0.998**	**0.068**	**0.004**	**0.998**	**0.073**	**288/288**	**100**
	IOM4	2-19-14-20	0.329	0.174	0.985	0.952	0.030	1.141	62/288	21.53
	IOM6	3-19-15-26-1	0.164	0.728	0.686	1.312	0.705	0.742	235/288	81.60

*sliding-window size *i* was equal to 1 (refer to Table 4.5).

Table B. 10: Experimental results of multiple hidden layer MLP ANN hybrid model coupled with GA (ANN&GA) with varying sliding-window size for SB-SCE of the Desharnais dataset

METHOD	WINDOW SIZE *i*	ANN ARCHITECTURE	TRAINING PHASE			TESTING PHASE				
			MMRE	*CC*	*NRMSE*	*MMRE*	*CC*	*NRMSE*	*Sign(p)*	*Sign(p)%*
IOM4	**1**	**2-20-19-19-1**	**0.230**	**0.960**	**0.279**	**0.285**	**0.899**	**0.443**	**15/22**	**68.18**
IOM2	1	4-19-20-17-1	0.623	0.892	0.961	0.566	0.658	0.755	11/22	50.00
IOM2	5	5-20-19-20-1	0.305	0.885	0.464	0.493	0.461	1.058	13/21	61.90
IOM2	3	3-20-18-20-1	0.327	0.881	0.470	0.386	0.701	0.763	9/22	40.91
IOM4	3	6-17-15-20-1	0.705	0.869	0.979	0.327	0.835	0.550	10/21	47.62
IOM2	7	7-20-19-18-1	0.136	0.967	0.254	0.707	0.579	0.936	13/21	61.90

Table B. 11: Experimental results of multiple hidden layer MLP ANN hybrid model coupled with GA (ANN&GA) with varying sliding-window size for SB-SCE of the ISBSG R9-1 dataset

METHOD	WINDOW SIZE *i*	ANN ARCHITECTURE	TRAINING PHASE			TESTING PHASE				
			MMRE	*CC*	*NRMSE*	*MMRE*	*CC*	*NRMSE*	*Sign(p)*	*Sign(p)%*
IOM4	1	2-18-3-19-1	0.089	0.175	1.100	0.073	0.045	1.012	80/288	27.78
IOM2	1	4-19-20-20-1	1.980	0.288	0.957	1.201	0.005	1.738	110/286	38.46
IOM2	5	5-19-20-19-1	1.980	0.123	0.992	0.892	-0.053	1.648	94/286	32.87
IOM6	**1**	**3-19-9-3-1**	**0.085**	**0.274**	**0.962**	**0.070**	**0.241**	**1.000**	**143/286**	**50.00**
IOM6	2	6-19-20-19-1	1.851	0.183	0.983	1.244	-0.031	1.845	98/286	34.27
IOM2	7	7-3-12-15-1	0.070	0.100	0.994	0.071	0.039	1.013	163/286	56.99

B.2 Complete Results of ANN and ISA for FSS-SCE

The following results were obtained with the ISBSG R9-3 and with Input Sensitivity Analysis (ISA) performed on Artificial Neural Networks (ANN) as described in the backward elimination methodology of section 4.2.2.1 (pg. 136-137).

Table B. 12: Random sampling and first four attributes removed from the Desharnais dataset using backward elimination using the *Relative Importance* (*RI*) of inputs using ISA on ANN

Exp.Id	Order of Attributes Removed	ANN Training Phase				ANN Testing Phase			
		Initial *MMRE*	Initial *Pred(.25)*	Final *MMRE*	Final *Pred(.25)*	Initial *MMRE*	Initial *Pred(.25)*	Final *MMRE*	Final *Pred(.25)*
1	TE,DU,SC,ME	0.384	0.936	0.559	0.936	0.536	0.867	0.600	0.867
2	ME,DU,TR,TE	0.280	0.979	0.343	1.000	0.409	0.933	0.487	0.933
3	SC,EN,TE,DU	0.557	0.936	0.583	0.936	0.198	1.000	0.335	1.000
4	TE,TR,PA,SC	0.474	0.915	0.485	0.894	0.364	1.000	0.387	1.000
5	ME,PA,TE,DU	0.361	0.957	0.360	0.979	1.264	0.867	1.060	0.867
6	TE,SC,DU,ME	0.512	0.915	0.784	0.915	0.386	0.933	0.346	0.933
7	ME,SC,PNA,PA	0.472	0.957	0.572	0.957	0.569	0.800	0.512	0.800
8	TE,ME,SC,EN	0.509	0.957	0.572	0.957	0.351	0.800	0.512	0.800
9	TE,ME,EN,DU	0.358	0.936	0.356	0.936	0.507	0.933	0.578	0.933
10	TE,SC,EN,DU	0.482	0.894	0.768	0.894	0.312	0.933	0.293	0.933
	Mean	**0.439**	**0.938**	**0.538**	**0.940**	**0.490**	**0.907**	**0.511**	**0.907**

Table B. 13: Random sampling-first seven attributes removed from the ISBSG R9-3 dataset using backward elimination using the *Relative Importance* (*RI*) of inputs using ISA on ANN

Exp.Id	Order of Attributes Removed	ANN Training Phase				ANN Testing Phase			
		Initial *MMRE*	Initial *Pred(.25)*	Final *MMRE*	Final *Pred(.25)*	Initial *MMRE*	Initial *Pred(.25)*	Final *MMRE*	Final *Pred(.25)*
1	CC,AFP,MTS,PET,OC,PIT,EC	0.223	0.957	0.329	0.928	0.358	1.000	0.578	1.000
2	DC,CC,PET,RL,PIT,INC,EC	0.280	0.986	0.352	0.971	0.418	0.955	0.575	0.955
3	INC,CC,DC,RL,OC,AC,FC	0.337	1.000	0.404	0.986	0.255	0.955	0.265	0.909
4	MTS,CC,RL,INC,FC,FS,AC	0.350	0.957	0.493	0.957	0.199	0.955	0.191	0.955
5	FS,RL,FC,EC,IC,INC,CC	0.367	0.971	0.411	0.986	0.202	0.955	0.298	0.955
6	RL,DC,CC,INC,OC,EC,FS	0.362	1.000	0.303	1.000	0.346	0.909	0.385	0.864
7	CC,OC,RL,FC,INC,AFP,IC	0.210	1.000	0.271	0.986	0.377	0.909	0.257	0.909
8	FC,OC,AC,AFP,RL,IC,FS	0.247	0.957	0.344	0.957	0.188	1.000	0.424	1.000
9	FC,IC,RL,AFP,AC,EC,PIT	0.261	0.957	0.305	0.971	0.662	1.000	0.500	1.000
10	FS,IC,PIT,OC,AFP,RL,FC	0.338	0.986	0.279	0.986	0.249	0.955	0.270	0.955
	Mean	**0.297**	**0.977**	**0.349**	**0.972**	**0.325**	**0.959**	**0.374**	**0.950**

B.3 Complete Results from Ridge Regression (RR) and FSS-SCE

The complete results obtained using the nine different Feature Subset Selection (FSS) approaches during the phases of training and testing are presented below. Table B. 14 and Table B. 15 report the best prediction result obtained with each FSS (MIN) and the mean across all the 10 fold cross-validation experiments conducted (AVG).

Table B. 14: SCE across various FSS with RR on the Desharnais dataset

FSS	RESULTS	TRAINING PHASE				TESTING PHASE				#Features
		INITIAL		FINAL		INITIAL		FINAL		
		MMRE	*Pred(.25)*	*MMRE*	*Pred(.25)*	*MMRE*	*Pred(.25)*	*MMRE*	*Pred(.25)*	
BFE	MIN	0.565	0.339	0.566	0.323	0.334	0.400	0.337	0.333	4
	MAX	0.482	0.419	0.489	0.435	0.815	0.267	0.916	0.267	5
	AVG	**0.522**	**0.379**	**0.528**	**0.389**	**0.582**	**0.387**	**0.622**	**0.347**	**4**
FFS	MIN	0.565	0.339	0.566	0.323	0.334	0.400	0.337	0.333	4
	MAX	0.482	0.419	0.487	0.435	0.815	0.267	0.903	0.267	6
	AVG	**0.522**	**0.379**	**0.528**	**0.395**	**0.582**	**0.387**	**0.622**	**0.340**	**5**
BSWF	MIN	0.565	0.339	0.584	0.435	0.334	0.400	0.345	0.400	4
	MAX	0.474	0.435	0.516	0.435	0.779	0.333	0.859	0.333	2
	AVG	**0.522**	**0.379**	**0.552**	**0.427**	**0.582**	**0.387**	**0.623**	**0.413**	**4**
FSWF	MIN	0.556	0.323	0.623	0.387	0.387	0.467	0.416	0.400	2
	MAX	0.482	0.419	0.537	0.403	0.815	0.267	0.911	0.533	3
	AVG	**0.522**	**0.379**	**0.577**	**0.427**	**0.582**	**0.387**	**0.653**	**0.413**	**3**
LSBFE	MIN	0.559	0.306	0.564	0.290	0.389	0.400	0.377	0.400	4
	MAX	0.448	0.435	0.462	0.403	1.264	0.067	1.170	0.200	3
	AVG	**0.516**	**0.342**	**0.540**	**0.353**	**0.648**	**0.293**	**0.699**	**0.320**	**3**
LSFFS	MIN	0.554	0.274	0.587	0.306	0.339	0.400	0.388	0.333	2
	MAX	0.448	0.435	0.479	0.403	1.264	0.067	1.096	0.267	3
	AVG	**0.516**	**0.342**	**0.538**	**0.348**	**0.648**	**0.293**	**0.639**	**0.333**	**3**
LSGA	MIN	0.556	0.323	0.623	0.387	0.387	0.467	0.416	0.400	2
	MAX	0.474	0.436	0.510	0.452	0.779	0.333	0.867	0.267	3
	AVG	**0.522**	**0.379**	**0.573**	**0.416**	**0.582**	**0.387**	**0.626**	**0.320**	**3**
GA	MIN	0.566	0.339	0.566	0.323	0.334	0.400	0.337	0.333	4
	MAX	0.474	0.436	0.474	0.500	0.779	0.333	0.836	0.400	5
	AVG	**0.522**	**0.379**	**0.530**	**0.377**	**0.582**	**0.387**	**0.614**	**0.327**	**4**
BANN	MIN	0.565	0.339	0.605	0.468	0.334	0.400	0.382	0.333	4
	MAX	0.474	0.435	0.533	0.387	0.779	0.333	0.792	0.200	4
	AVG	**0.522**	**0.379**	**0.573**	**0.387**	**0.582**	**0.387**	**0.606**	**0.447**	**4**

Table B. 15: SCE across various FSS with RR on the ISBSG R9-4 dataset

FSS	RESULTS	TRAINING PHASE				TESTING PHASE				#Features
		INITIAL		FINAL		INITIAL		FINAL		
		MMRE	*Pred(.25)*	*MMRE*	*Pred(.25)*	*MMRE*	*Pred(.25)*	*MMRE*	*Pred(.25)*	
BFE	MIN	0.355	0.520	0.444	0.436	0.434	0.398	0.456	0.390	34
	MAX	0.337	0.532	0.409	0.433	0.657	0.415	0.851	0.423	36
	AVG	**0.330**	**0.536**	**0.407**	**0.467**	**0.585**	**0.349**	**0.592**	**0.377**	**32**
FFS	MIN	0.355	0.520	0.451	0.453	0.434	0.398	0.439	0.455	33
	MAX	0.337	0.532	0.418	0.456	0.657	0.415	0.771	0.463	29
	AVG	**0.330**	**0.536**	**0.410**	**0.471**	**0.585**	**0.349**	**0.577**	**0.382**	**29**
BSWF	MIN	0.355	0.520	0.465	0.419	0.434	0.398	0.434	0.447	32
	MAX	0.313	0.523	0.414	0.436	0.763	0.309	0.880	0.407	30
	AVG	**0.330**	**0.536**	**0.451**	**0.434**	**0.585**	**0.349**	**0.618**	**0.386**	**27**
FSWF	MIN	0.355	0.520	0.551	0.390	0.434	0.398	0.425	0.447	11
	MAX	0.313	0.523	0.482	0.401	0.763	0.309	0.817	0.382	11
	AVG	**0.330**	**0.536**	**0.518**	**0.387**	**0.585**	**0.349**	**0.600**	**0.373**	**12**
LSBFE	MIN	0.355	0.520	0.462	0.387	0.434	0.398	0.399	0.439	39
	MAX	0.312	0.564	0.415	0.477	0.815	0.317	0.869	0.301	27
	AVG	**0.330**	**0.536**	**0.431**	**0.446**	**0.585**	**0.349**	**0.592**	**0.372**	**32**
LSFFS	MIN	0.355	0.520	0.461	0.448	0.434	0.398	0.447	0.407	37
	MAX	0.312	0.564	0.403	0.477	0.815	0.317	0.812	0.350	31
	AVG	**0.330**	**0.536**	**0.442**	**0.435**	**0.585**	**0.349**	**0.596**	**0.363**	**34**
LSGA	MIN	0.355	0.520	0.457	0.410	0.434	0.398	0.432	0.407	36
	MAX	0.314	0.523	0.422	0.462	0.763	0.309	0.867	0.374	29
	AVG	**0.330**	**0.536**	**0.434**	**0.436**	**0.585**	**0.349**	**0.607**	**0.355**	**31**
GA	MIN	0.355	0.520	0.424	0.422	0.434	0.398	0.445	0.415	35
	MAX	0.312	0.564	0.374	0.491	0.815	0.317	0.748	0.382	30
	AVG	**0.331**	**0.536**	**0.401**	**0.472**	**0.585**	**0.349**	**0.570**	**0.376**	**33**
BANN	MIN	0.355	0.520	0.566	0.384	0.434	0.398	0.478	0.325	41
	MAX	0.312	0.564	0.465	0.401	0.815	0.317	0.954	0.317	41
	AVG	**0.330**	**0.536**	**0.489**	**0.402**	**0.585**	**0.349**	**0.686**	**0.320**	**41**

B.4 Complete Results of Fuzzy Clustering in CC-SCE

In the fuzzy clustering algorithm utilised for Clustering and Classification Software Cost Estimations (CC-SCE) (Papatheocharous and Andreou, 2009a), namely the Entropy-based fuzzy *k*-modes algorithm the following experiments and experimental subsets were used: The experimental dataset ISBSG R9-8.1 included all available project characteristics plus the effort; all project characteristics excluding effort constituted ISBSG R9-8.2; removing the outliers from ISBSG R9-8.1 and ISBSG R9-8.2 based on the Box Plots of the effort sample values resulted datasets ISBSG R9-8.3 and ISBSG R9-8.4 respectively; finally, using ISBSG R9-8.3 and adjusting the weight of the effort variable to reach the dominant significance level of 51% in the clustering process compared to the rest of the attributes, produced dataset

ISBSG R9-8.5. Similarity Table B. 16 summarises the best results obtained with respect to the width of the estimation (or prediction) interval and the Hit Ratio (*HR*).

Table B. 16: SCE results obtained with the fuzzy *k*-modes algorithm and various ISBSG R9-8 subsets

Dataset	β	α	k	φ	OS (%)	ClS (%)	*HR* (%)	mean effort ($\bar{e}$)	std effort (σ)	width
ISBSG R9-8.1	0.55	1.5	42	0.75	7.80	49.09	38.68	12727.77	5843.59	11687.19
ISBSG R9-8.2	0.7	1.2	95	0.75	10.16	49.39	50.94	7885.80	7614.47	15228.94
ISBSG R9-8.3	0.4	1.4	6	0.85	2.26	67.68	36.17	1711.81	1695.34	3390.68
ISBSG R9-8.4	0.3	1.8	3	0.85	2.39	67.45	28.72	1931.39	1788.68	3577.36
ISBSG R9-8.5	0.8	1.7	25	0.75	1.60	37.92	76.60	2030.93	1198.76	2397.53

The results reported indicate relatively large prediction intervals in most of the cases, except in the datasets ISBSG R9-8.1 and ISBSG R9-8.5, with standard deviations being lower than the means in all cases. Moreover, a mediocre Hit Ratio (*HR*) performance is observed, which amounts to approximately 30-40% hits for ISBSG R9-8.1, ISBSG R9-8.3 and ISBSG R9-8.4 and slightly over 50% hits for the ISBSG R9-8.2. The accuracy of the predicted effort values is significantly improved in the ISBSG R9-8.5 case; the *HR* is quite high suggesting that estimations produced lay within the calculated width in nearly 77% of the cases. It is worth noticing that when the effort attribute participates in a dataset performance is improved (cases ISBSG R9-8.1 and ISBSG R9-8.3 in comparison with ISBSG R9-8.2 and ISBSG R9-8.4 respectively). This outcome suggests that the effect of previous values for the attribute being estimated leads to forming better clusters. One may argue that the participation of effort samples in the clustering process may bias results, but this is not true; past effort values are treated by the algorithm as descriptors of the behaviour of effort in relation with the rest of the participating factors. Hence, what effort samples offer is essentially a way to map cost factors onto the effort attribute and thus form knowledge regarding how effort is affected. Additionally, the narrower widths obtained with ISBSG R9-8.3 and ISBSG R9-8.4 confirm that when extreme values are removed from the datasets the estimation performance is again improved. Overall, ISBSG R9-8.5 yielded the most promising results.

B.5 Complete Results for GP in CC-SCE

The complete results using the Genetic Programming tool to examine all possible algorithmic cost estimations and predict the effort value using the indicative regression equations listed in Table B. 17 are summarised in Table B. 18 and Table B. 19.

Table B. 17: Best cost functions obtained using GP and including arithmetic and logical operators (i.e., numeric and categorical attributes) across datasets

Dataset	Id	Algorithmic Expressions
COCOMO	C_1	(((LOC*SCED)^(TIME/TOOL))+((LOC^VEXP)^(TIME*MODP)))
	C_2	(((LOC^TIME)^MODP)+((LOC*STOR)^RELY))
	C_3	((((LOC^MODP)+(STOR+STOR))^TIME)+(((SCED+LOC)^RELY)*(STOR^RELY)))
	C_4	(((LOC^(STOR*MODP))+((TIME+TIME)^(VEXP+STOR)))+((LOC^(TIME*DATA))*AEXP))
Desharnais	D_1	((((DU* DU)+ SC)+((ME+FPA)+(TR+ME)))*((FPNA-(TR+ DU))))
	D_2	(((FPA+FPA)+SC)*((TR-FPNA)))
	D_3	((((FPNA+SC)+SC)+EN)*(((EN-((DU*DU)+(DU+DU)))*DU)))
	D_4	(((((((*log*(EN)*DU)+(EN*(*log*(*log*(FPNA))*DU)))*log(SC))-(FPA*DU)))+((DU)))*FPA)
ISBSG R9-4	I_1	((INS='0') ||((((ODT='0') NAND(EGW='0')) NORIF ((IMS='0') THEN ((DMODEL='0')) ELSE ((BANK='0')))) NOR(((SYBASE='0') ||(MTEAM='0')) &&((DMODEL='1') NOR(ORACLE='0')))))
	I_2	((OT6='0') ||(OT7='0'))
	I_3	((((DBS5='0') NAND(OT5='0')) &&((IY='2000') NOR(0T12='0'))) NAND(((OT11='1') NAND(DBS1='1'))))
	I_4	((((DT13='0')) XOR((DBS2='0') XOR(OT12='0'))) ||(((OT8='0') ||(OT12='0')) ||((OT11='1'))))

Table B. 18: Software cost estimation performance of the GP arithmetic cost functions execution

Dataset	Id	Tree Depth	No. of Nodes	TRAINING			TESTING		
				MMRE	*CC*	*NRMSE*	*MMRE*	*CC*	*NRMSE*
COCOMO	**C_1**	**4**	**-**	**0.459**	**0.985**	**0.184**	**0.469**	**0.945**	**0.322**
	C_2	4	-	0.450	0.986	0.183	0.497	0.962	0.349
	C_3	5	-	0.465	0.986	0.179	0.497	0.962	0.354
	C_4	5	-	0.485	0.991	0.138	0.494	0.979	0.243
Desharnais	**D_1**	**5**	**-**	**0.474**	**0.836**	**0.562**	**0.485**	**0.768**	**0.722**
	D_2	4	-	0.533	0.787	0.628	0.541	0.733	0.702
	D_3	-	20	0.520	0.799	0.606	0.521	0.731	0.684
	D_4	-	28	0.454	0.851	0.546	0.574	0.744	0.710

Table B. 19: Software cost estimation performance of the GP logical cost functions classification and execution (calculations were based on eq. (4.25) on classified projects)

Dataset	Id	Tree Depth	TRAINING			TESTING	
			mean effort ($\bar{e}$)	std effort (σ)	*HR*	*HR*	*HR(%)*
ISBSG R9-4	**I_1**	**5**	**6983.43**	**13806.51**	**374/374**	**89/93**	**95.70**
	I_2	2	6726.99	13511.18	374/374	87/93	93.55
	I_3	4	7068.52	13827.94	374/374	90/93	96.77
	I_4	4	6983.43	13806.51	374/374	89/93	95.70

A second series of experiments was conducted after identifying and isolating the attributes that appeared more frequently in the best performing equations from all the experiments conducted, i.e., the frequent attributes were considered the most 'important'. The datasets

used thus consisted of only those attributes selected and are indicated by the Imp subscript in Table B. 21. Particularly, the following subsets were created: for the COCOMO dataset, i.e., the $COCOMO_{Imp}$ dataset included the attributes project size (LOC), execution time constraint (TIME), product complexity (CPLX) and applications experience (AEXP) and for the Desharnais dataset, i.e., $Desharnais_{Imp}$ dataset included the attributes project duration (DU), scope (SC), number of transactions (TR) and function points (FPA). Also, the series of experiments was conducted on the 'complementing' datasets of the 'important' attributes, thus including the 'non-important' attributes, i.e., the $COCOMO_{Imp'}$ and $Desharnais_{Imp'}$ respectively which excluded the abovementioned 'important' attributes. The equations obtained and the experimental results using the important and the non-important attributes are summarised in Table B. 20 and Table B. 21 respectively.

Table B. 20: Best cost functions obtained using GP and including/excluding 'important' attributes across datasets

Dataset	Id	Algorithmic Expressions
$COCOMO_{Imp}$	C_5	(((LOC^AEXP)+(LOC+LOC))^(TIME^AEXP))
	C_6	(((LOC^AEXP)+(LOC/TIME))*((LOC*LOC)^log(TIME)))
$COCOMO_{Imp'}$	C_7	(MODP^(STOR/log10(TURN)))
	C_8	((TOOL*((STOR+TOOL)^(DATA+DATA)))^(((SCED*STOR)+ (DATA*DATA))^((DATA*DATA)*DATA)))
$Desharnais_{Imp}$	D_5	((*log*(FPA)*FPA)+(*log2*(FPA)*(SC*DU)))
	D_6	(*log2*(((DU*DU)*SC))*((SC+FPA)+(SC+(DU+SC))))
$Desharnais_{Imp'}$	D_7	(((EN+FPNA)+(TE+ FPNA))**log*((FPNA+FPNA)))
	D_8	((FPNA+*log2*(FPNA))**log2*((EN*EN)))

Table B. 21: Performance results including/excluding 'important' attributes with GP

Dataset	Id	Tree Depth	TRAINING			TESTING		
			MMRE	*CC*	*NRMSE*	*MMRE*	*CC*	*NRMSE*
$COCOMO_{Imp}$	**C_5**	**4**	**0.575**	**0.942**	**0.347**	**0.520**	**0.931**	**0.681**
	C_6	4	0.545	0.968	0.260	0.608	0.788	0.546
$COCOMO_{Imp'}$	C_7	4	0.856	0.829	0.613	1.127	0.527	0.901
	C_8	5	0.897	0.852	0.545	0.989	0.456	0.967
$Desharnais_{Imp}$	**D_5**	**4**	**0.597**	**0.751**	**0.659**	**0.290**	**0.820**	**0.601**
	D_6	4	0.549	0.747	0.665	0.441	0.905	0.523
$Desharnais_{Imp'}$	D_7	4	0.659	0.731	0.684	0.325	0.803	0.646
	D_8	5	0.668	0.726	0.685	0.373	0.809	0.649

The results obtained from the experiments utilising the full attributes, only the important attributes and excluding the important attributes (non-important attributes) show that effort prediction quality and research suffers from the effect of insignificant attributes participating

in the prediction and that researchers should turn to identifying the most important cost factors to gain accuracy improvements in the SCE approaches utilised.

B.6 Complete Results from FDT for in CC-SCE

The complete experimental results obtained using the enhanced FDT approach are presented below.

Table B. 22: Classification results using enhanced FDT approach for SCE on ISBSGR9-9

DS	Algorithm	SL	TRAINING			TESTING			PM		RM	
			n(673)	$\widehat{eff}$	σ_n	n(288)	$\widehat{eff}$	σ_n	*HR*	*HR(%)*	*HR*	*HR(%)*
All	CHAID	0.939	155	7.332	1.075	66	7.099	1.194	42/66	63.63	42/66	63.64
All	CHAID	0.974	418	7.118	1.279	168	6.985	1.383	108/168	64.28	107/168	63.69
All	**CART**	**0.953**	**632**	**7.386**	**1.252**	**262**	**7.265**	**1.354**	**166/262**	**63.35**	**170/262**	**64.89**
All	**CART**	**0.959**	**630**	**7.380**	**1.248**	**261**	**7.255**	**1.348**	**166/261**	**63.60**	**169/261**	**64.75**
All	exCHAID	0.909	10	8.353	1.359	4	8.330	0.634	3/4	75.00	3/4	75.00
All	**exCHAID**	**0.896**	**43**	**7.831**	**1.079**	**20**	**7.669**	**1.232**	**11/20**	**55.00**	**12/20**	**60.00**
Cat	CHAID	0.971	269	7.008	1.373	106	6.942	1.489	68/106	64.15	68/106	64.15
Cat	CHAID	0.969	405	7.109	1.298	153	6.899	1.396	97/153	63.39	99/153	64.71
Cat	**CART**	**0.873**	**96**	**8.013**	**1.007**	**41**	**6.711**	**1.169**	**21/41**	**51.21**	**21/41**	**51.22**
Cat	CART	1.000	92	6.766	1.101	33	5.217	1.143	22/33	66.66	21/33	63.64
Erl	**CHAID**	**0.973**	**403**	**7.097**	**1.291**	**152**	**6.904**	**1.399**	**96/152**	**63.15**	**98/152**	**64.47**
Erl	CHAID	0.921	210	7.931	1.009	91	7.749	1.155	52/91	57.14	51/91	56.04
Erl	exCHAID	0.991	110	7.058	1.005	43	6.867	1.165	28/43	65.11	27/43	62.79
Erl	**exCHAID**	**1.000**	**60**	**7.859**	**1.014**	**22**	**7.420**	**1.198**	**10/22**	**45.45**	**11/22**	**50.00**